ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Η πιστοποίηση γνώσεων και δεξιοτήτων σε αντικείμενα που αφορούν είτε την καθημερινή ζωή (π.χ. γνώση κανόνων κώδικα οδικής κυκλοφορίας) είτε γενικά ή ειδικά επαγγελματικά προσόντα (π.χ. ξένες γλώσσες, χρήση βασικών εφαρμογών ηλεκτρονικού υπολογιστή, εξειδικευμένες γνώσεις σε σχεδιασμό και υλοποίηση δικτύων ηλεκτρονικών υπολογιστών), απαιτεί την ύπαρξη οργανωμένων συστημάτων διενέργειας εξετάσεων και έκδοσης αποτελεσμάτων. Τα συστήματα αυτά, ανεξάρτητα των ιδιομορφιών που σχετίζονται με το αντικείμενο για το οποίο καλούνται να αξιολογήσουν γνώσεις και δεξιότητες, πρέπει να πληρούν μερικά βασικά χαρακτηριστικά: Πρέπει να είναι γρήγορα τόσο στην εξέταση και στην έκδοση αποτελεσμάτων, πρέπει να είναι οικονομικά στην χρήση τους και πρέπει να είναι αδιάβλητα. Σε ορισμένα συστήματα τα οποία αφορούν ευαίσθητες κοινωνικά εξετάσεις (π.χ. εξετάσεις διορισμού στο δημόσιο, διπλώματα οδήγησης, κ.λ.π.) το τελευταίο χαρακτηριστικό είναι και το πιο σημαντικό.

Πλήθος τέτοιων συστημάτων αξιολογούν τη γνώση ή τη δεξιότητα χρησιμοποιώντας διαδικασίες εξέτασης οι οποίες στο σύνολο τους, ή σε ένα μεγάλο μέρος τους, αποτελούνται από ερωτήσεις οι απαντήσεις στις οποίες είναι εύκολο να κριθούν ως σωστές ή λάθος. Το πιο συχνά χρησιμοποιούμενο τέτοιο είδος εξέτασης είναι τα τεστ πολλαπλής επιλογής (multiple choice tests). Τα τεστ πολλαπλής επιλογής, παρόλο που δεν μπορούν να χρησιμοποιηθούν για την αξιολόγηση γνώσεων που χρειάζονται ανάπτυξη λόγου ή πολύπλοκων δεξιοτήτων, χρησιμοποιούνται όπου είναι δυνατό διότι παρουσιάζουν σειρά πλεονεκτημάτων: Έχουν διάφορες μορφές και μπορούν να χρησιμοποιηθούν σε πολλά είδη εξέτασης, είναι εύκολο να παράσχουν ακριβή και αντικειμενική βαθμολόγηση σύμφωνα με τους κανόνες της εξέτασης και είναι εύκολο ή διαδικασία εξέτασης ή / και η βαθμολόγηση των απαντήσεων να γίνει με τη βοήθεια ηλεκτρονικού εξοπλισμού.

Στο παρελθόν έχει αναπτυχθεί αριθμός ηλεκτρονικών συστημάτων τα οποία βοηθούν με διάφορους τρόπους την διαδικασία εξέτασης και αξιολόγησης γνώσεων. Ειδικά για τα τεστ πολλαπλής επιλογής έχουν αναπτυχθεί, από συστήματα αυτόματης βαθμολόγησης απαντήσεων που έχουν καταγραφεί σε ειδικά έντυπα / φόρμες και τα οποία λειτουργούν ως σαρωτές, ως ολοκληρωμένα πληροφοριακά συστήματα μέσω των οποίων διεκπεραιώνεται το μεγαλύτερο μέρος

της συνολικής διαδικασίας αξιολόγησης γνώσεων, δηλαδή η παρουσίαση ερωτήσεων η καταγραφή απαντήσεων η βαθμολόγηση του εξεταζόμενου και η έκδοση επίσημων αποτελεσμάτων.

Το πιο γνωστό σύστημα της κατηγορίας αυτής είναι το “Μηχανογραφικό Σύστημα Θεωρητικής Εξέτασης Υποψηφίων Οδηγών” (Μ.Σ.Θ.Ε.Υ.Ο.). Το Μ.Σ.Θ.Ε.Υ.Ο. αναπτύχθηκε από το Τμήμα Πληροφορικής του ΤΕΙ Αθηνών για το Υπουργείο Μεταφορών και Επικοινωνιών, αρχικά, για την διεξαγωγή των θεωρητικών εξετάσεων των υποψήφιων οδηγών των τεσσάρων βασικών κατηγοριών διπλωμάτων οδήγησης: Κώδικας (αυτοκίνητα), μοτοσυκλέτες, φορτηγά οχήματα και λεωφορεία. Αργότερα επεκτάθηκε και για άλλες κατηγορίες εξέτασης που σχετίζονται με το οικείο υπουργείο (το σημερινό Υπουργείο Υποδομών, Μεταφορών και Δικτύων). Οι εξετάσεις για τα διπλώματα οδήγησης είναι ένα ευαίσθητο κοινωνικά θέμα, όχι μόνο γιατί αφορά το μεγαλύτερο μέρος του πληθυσμού της χώρας και άρα οι διαδικασίες του πρέπει να είναι οικονομικές και γρήγορες, αλλά κυρίως διότι έχουν συνδεθεί με υποθέσεις οικονομικής διαφθοράς σε συνδυασμό με τη λεπτή φύση τους που σχετίζεται με την ανθρώπινη ζωή και ασφάλεια.

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός και η μερική ανάπτυξη ενός μηχανογραφικού συστήματος διενέργειας εξετάσεων και έκδοσης αποτελεσμάτων για γενική χρήση. Αφορμή για την ανάληψη του θέματος αυτού στάθηκε η απασχόληση του γράφοντα στη δημιουργία του Μ.Σ.Θ.Ε.Υ.Ο., ως υπεύθυνου ανάπτυξης, κάτω από τις οδηγίες του καθηγητή Ιωάννη Κεκλίκογλου ο οποίος είναι ο υπεύθυνος του έργου. Η εργασία βασίζεται, κύρια, στην εμπειρία που αποκτήθηκε από την ανάπτυξη αλλά κυρίως από την εφαρμογή του συστήματος στις υπηρεσίες μεταφορών και επικοινωνιών της χώρας. Η εμπειρία αυτή είναι εξαιρετικά χρήσιμη στην ανάπτυξη ενός τέτοιου συστήματος διότι δεν αφορά μόνο τεχνικά θέματα. Αντίθετα πηγάζει από την συμπεριφορά τριών κατηγοριών χρηστών: Τους αποδέκτες της στρατηγικής πληροφορίας (κεντρική υπηρεσία Υ.Μ.Ε.), τους υπεύθυνους διεξαγωγής της διαδικασίας εξέτασης (υπάλληλοι υπηρεσιών) και τους εξεταζόμενους (πολίτες). Είναι προφανές ότι στον πραγματικό κόσμο τα συμφέροντα των τριών αυτών κατηγοριών χρηστών δεν είναι πάντα τα ίδια. Ένα καλό μηχανογραφικό σύστημα διεξαγωγής εξετάσεων πρέπει να ικανοποιεί όλες τις κατηγορίες χρηστών του.

Η εργασία αποτελείται από 7 κεφάλαια και 4 παραρτήματα. Ακολουθεί σύντομη αναφορά στα περιεχόμενα των επόμενων κεφαλαίων της εργασίας.

- Στο δεύτερο κεφάλαιο παρουσιάζεται το Μ.Σ.Θ.Ε.Υ.Ο.. Γίνεται μία αναφορά στην ιστορία του, στις βασικές προδιαγραφές του και στον τρόπο υλοποίησής του. Παρουσιάζονται οι ελλείψεις και οι αδυναμίες του.

- Στο τρίτο κεφάλαιο αναλύονται οι απαιτήσεις του νέου προτεινόμενου συστήματος διεξαγωγής εξετάσεων, στο εξής: “Atlas”. Καταγράφονται οι λειτουργικές προδιαγραφές και οι όροι που πρέπει να πληρούνται ώστε να

είναι πετυχημένο ένα σύστημα διεξαγωγής εξετάσεων σύμφωνα πάντα και με το είδος της εξέτασης για την οποία προορίζεται.

- Στο τέταρτο κεφάλαιο αναφέρονται οι τεχνολογίες υλοποίησης του Atlas. Ιδιαίτερη βαρύτητα δίδεται στην ιδιότητα του συστήματος να είναι συμβατό με ποικίλο hardware και διάφορα Λειτουργικά Συστήματα.

- Στο πέμπτο κεφάλαιο αναπτύσσεται ο σχεδιασμός και η μοντελοποίηση του νέου συστήματος. Γίνεται αναλυτική περιγραφή της λύσης και των επιμέρους λειτουργιών. Ιδιαίτερη βαρύτητα δίδεται στην παρουσίαση του μοντέλου ασφαλείας του νέου συστήματος.

- Στο έκτο κεφάλαιο παρουσιάζεται η εφαρμογή που αναπτύχθηκε. Η εφαρμογή καλύπτει ένα μεγάλο μέρος των προδιαγραφών του συστήματος και αποδεικνύει την ευχρηστία των σχεδιαστικών επιλογών του.

- Στο έβδομο κεφάλαιο γίνεται μία ανακεφαλαίωση και παρουσιάζονται τα συμπεράσματα της εργασίας.

- Στα παραρτήματα περιέχεται ο πηγαίος κώδικας που αναπτύχθηκε για τον εξυπηρετητή του συστήματος, ο πηγαίος κώδικας των προγραμμάτων τα οποία τρέχουν στους σταθμούς εξέτασης και η δομή της Βάσης Δεδομένων.

Θα ήθελα να ευχαριστήσω το σύνολο του εκπαιδευτικού προσωπικού το οποίο δίδαξε στο Μ.Π.Σ. για τις γνώσεις που μας πρόσφερε και ιδιαίτερα την επίκουρη καθηγήτρια του Τμήματος κ. Βέρα Αλεξάνδρα Σταυρουλάκη τόσο για την ανάθεση της παρούσας εργασίας και την συνεργασία στην εκπόνησή της, όσο και για τις ευχάριστες ώρες εκπαίδευσης τις οποίες μας πρόσφερε διδάσκοντας μας, σε διάφορες μορφές, προγραμματισμό ηλεκτρονικών υπολογιστών.

ΚΕΦΑΛΑΙΟ 2

ΤΟ ΜΗΧΑΝΟΓΡΑΦΙΚΟ ΣΥΣΤΗΜΑ ΘΕΩΡΗΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ ΥΠΟΨΗΦΙΩΝ ΟΔΗΓΩΝ (Μ.Σ.Θ.Ε.Υ.Ο.)

ΛΙΓΗ ΙΣΤΟΡΙΑ

Το 2000 το "Υπουργείο Μεταφορών και Επικοινωνιών", το οποίο μετεξελίχθηκε σε "Υπουργείο Υποδομών, Μεταφορών και Δικτύων"[1], ανάθεσε στο Τμήμα Πληροφορικής του ΤΕΙ Αθηνών τη δημιουργία μίας εφαρμογής η οποία θα αυτοματοποιούσε τη διαδικασία εξέτασης των θεωρητικών εξετάσεων για τις βασικές κατηγορίες των διπλωμάτων οδήγησης. Η εφαρμογή θα είχε τη δυνατότητα να εξετάζει τους υποψήφιους οδηγούς στο Βασικό Ερωτηματολόγιο (κώδικας), το οποίο είναι απαραίτητο για την έκδοση πρώτου διπλώματος για αυτοκίνητο ή μοτοσυκλέτα, στο Ερωτηματολόγιο των Μοτοσυκλετών, στο ερωτηματολόγιο των Φορτηγών και στο ερωτηματολόγιο των Λεωφορείων, ανάλογα με το είδος του διπλώματος που θέλει να εκδώσει ή να επανεκδώσει ο υποψήφιος.

Η ανάπτυξη της πρώτης έκδοσης του συστήματος ολοκληρώθηκε την ίδια χρονιά. Το Μ.Σ.Θ.Ε.Υ.Ο. εγκαταστάθηκε αρχικά στη Διεύθυνση Μεταφορών και Επικοινωνιών της Νομαρχίας Αργολίδας (Ναύπλιο) όπου και λειτούργησε πιλοτικά και με απόλυτη επιτυχία, για μεγάλο χρονικό διάστημα.

Κατά τη διάρκεια της πιλοτικής εφαρμογής του Μ.Σ.Θ.Ε.Υ.Ο. στην Νομαρχία Αργολίδας, αναπτύχθηκε η δεύτερη έκδοση του λογισμικού του συστήματος, η οποία περιελάμβανε την εκφώνηση των ερωτήσεων και των απαντήσεων όλων των ερωτηματολογίων (Αυτοκινήτων, Μοτοσυκλετών, Φορτηγών και Λεωφορείων), με αποτέλεσμα να εφαρμόζεται το σύστημα αυτό και σε μια ακόμη ομάδα υποψηφίων οδηγών: τους αναλφάβητους.

1 Στο εξής θα αναφέρεται ως Υ.Μ.Ε.. Τα αρχικά αυτά χρησιμοποιούνται ακόμα και σήμερα. Για παράδειγμα ο ιστότοπος του υπουργείου βρίσκεται στην ηλεκτρονική διεύθυνση "http://www.yme.gov.gr/".

Μετά την επιτυχή αξιολόγηση της πιλοτικής λειτουργίας του Μ.Σ.Θ.Ε.Υ.Ο. στην Νομαρχία Αργολίδας, η εφαρμογή του συστήματος αυτού επεκτάθηκε, από τον Ιούλιο του 2001, στην ευρύτερη περιφέρεια της χώρας και ειδικότερα σε πέντε Υπηρεσίες Μεταφορών και Επικοινωνιών: στις Νομαρχίες Δράμας, Καρδίτσας, Μαγνησίας (Βόλος) και Ηλείας (Πύργος και Αμαλιάδα).

Τον επόμενο χρόνο (2002) υλοποιήθηκε η τρίτη έκδοση του συστήματος η οποία επέκτεινε το Μ.Σ.Θ.Ε.Υ.Ο., ώστε αυτό να περιλαμβάνει την δυνατότητα εξέτασης ξενόγλωσσων υποψηφίων οδηγών, πέραν της ελληνικής, στην αγγλική, ρωσική και αλβανική γλώσσα και σε όλες τις κατηγορίες εξετάσεων.

Το 2005 ακολούθησε η εγκατάσταση του συστήματος και η εφαρμογή του σε όλες τις Νομαρχιακές Αυτοδιοικήσεις της χώρας. Το Μ.Σ.Θ.Ε.Υ.Ο. καθιερώθηκε ως ο αποκλειστικός τρόπος διενέργειας θεωρητικών εξετάσεων υποψηφίων οδηγών σε όλη την επικράτεια. Σήμερα λειτουργούν 96 εξεταστικά κέντρα σε όλη την Ελλάδα. Η καθυστέρηση της εφαρμογής του συστήματος σε όλη τη χώρα οφείλεται σε γραφειοκρατικούς λόγους που σχετίζονται με την προμήθεια και την εγκατάσταση του υλικού και όχι σε προβλήματα του λογισμικού του συστήματος.

Το 2009 στο Μ.Σ.Θ.Ε.Υ.Ο. προστέθηκαν απλοποιημένα ερωτηματολόγια, ως νέες κατηγορίες εξέτασης, για τις κατηγορίες του κώδικα και των μοτοσυκλετών. Τα ερωτηματολόγια αυτά απευθυνόντουσαν στους Αθίγγανους και γενικά στα άτομα τα οποία έχουν πρόβλημα με τη χρήση της ελληνικής γλώσσας (αναλφάβητοι). Η χρήση των ειδικών αυτών ερωτηματολογίων καταργήθηκε μετά από μικρό χρονικό διάστημα για λόγους που δεν σχετίζονται με το τεχνικό μέρος του Μ.Σ.Θ.Ε.Υ.Ο.

Το 2011 το Μ.Σ.Θ.Ε.Υ.Ο. επεκτάθηκε ώστε να καλύπτει τις εξετάσεις για τα Πιστοποιητικά Επαγγελματικής Ικανότητας (Π.Ε.Ι.) οδηγών Φορτηγών και Λεωφορείων.

Τέλος, το 2012, το Μ.Σ.Θ.Ε.Υ.Ο. τροποποιήθηκε ώστε να καλύπτει και άλλου τύπου εξετάσεις, πέραν των υποψηφίων οδηγών, που αφορούν το Υ.Μ.Ε. και συγκεκριμένα τις εξετάσεις για την πιστοποίηση των ραδιοερασιτεχνών.

Από τη πρώτη εφαρμογή του συστήματος ως και σήμερα το λογισμικό ενημερώθηκε αρκετές ως προς το περιεχόμενο της βάσης δεδομένων. Σειρά ερωτήσεων τροποποιήθηκε λόγω αλλαγών στις διατάξεις του κώδικα οδικής κυκλοφορίας και εφαρμογής νέων πολιτικών από το Υ.Μ.Ε., όπως για παράδειγμα η οικολογική οδήγηση. Αξίζει να σημειωθεί ότι όλες οι τροποποιήσεις του συστήματος, από την πιλοτική εφαρμογή του ως και σήμερα, αφορούν προσαρμογές του σε νέες πραγματικότητες και όχι διορθώσεις προβλημάτων.

ΒΑΣΙΚΕΣ ΠΡΟΔΙΑΓΡΑΦΕΣ

Οι βασικές προδιαγραφές του Μ.Σ.Θ.Ε.Υ.Ο. δεν είναι καταγραμμένες με κάποιον τυπικό, για έργα πληροφορικής, τρόπο (π.χ. διαγράμματα UML). Αντίθετα, οι βασικές προδιαγραφές του έργου καταγράφηκαν, σε γενικές γραμμές, στην σύμβαση ανάθεσης του έργου και διαμορφώθηκαν στην τελική μορφή τους κατά την ανάπτυξή του (prototyping). Πολλές από τις λειτουργικές προδιαγραφές του Μ.Σ.Θ.Ε.Υ.Ο., αποτελούν τυπικό παράδειγμα προδιαγραφών ενός συστήματος διενέργειας εξετάσεων τύπου “τεστ πολλαπλών επιλογών” για ένα κρίσιμο κοινωνικά δημόσιο φορέα ή για ένα οργανισμό ο οποίος εκδίδει πιστοποιητικά τα οποία αποτελούν σημαντικό επαγγελματικό προσόν. Μερικές από αυτές είχαν ως στόχο την δημιουργία ενός συστήματος το οποίο θα είναι προσαρμοσμένο στην... πραγματικότητα του ελληνικού δημόσιου τομέα. Για τους λόγους αυτούς οι προδιαγραφές του Μ.Σ.Θ.Ε.Υ.Ο. καταγράφονται συνοπτικά στις παραγράφους που ακολουθούν.

Μορφή ερωτήσεων / απαντήσεων

Οι ερωτήσεις των ερωτηματολογίων του Μ.Σ.Θ.Ε.Υ.Ο. αποτελούνται από το λεκτικό της ερώτησης, τα λεκτικά των πιθανών απαντήσεων και μία εικόνα η οποία σχετίζεται με την ερώτηση. Δεν έχουν όλες οι ερωτήσεις εικόνα. Σε κάθε ερώτηση υπάρχει μία και μόνο μία σωστή απάντηση.

Μορφή ερωτηματολογίου

Κάθε ερωτηματολόγιο, ανάλογα με την κατηγορία του, αποτελείται από συγκεκριμένο αριθμό ερωτήσεων. Για παράδειγμα τα ερωτηματολόγια του κώδικα (αυτοκίνητα) αποτελούνται από 30 ερωτήσεις και τα ερωτηματολόγια των λεωφορείων από 10. Οι ερωτήσεις χωρίζονται σε εσωτερικές κατηγορίες (σήματα, προτεραιότητες, κ.λ.π.). Κάθε ερωτηματολόγιο περιέχει συγκεκριμένο αριθμό ερωτήσεων από κάθε εσωτερική κατηγορία. Στην εφαρμογή διατηρείται αριθμός ερωτήσεων, για κάθε κατηγορία εξέτασης και εσωτερική κατηγορία, ο οποίος είναι αρκετά μεγαλύτερος από τον απαιτούμενο για τη δημιουργία ενός ερωτηματολογίου. Η επιλογή των ερωτήσεων, ανά εσωτερική κατηγορία, γίνεται με τυχαίο τρόπο. Με τον τρόπο αυτό είναι πρακτικά αδύνατο να δημιουργηθούν δύο ίδια ερωτηματολόγια. Η σειρά των απαντήσεων μίας ερώτησης, επίσης, κληρώνονται κατά τη δημιουργία των ερωτηματολογίων. Με αυτό τον τρόπο αποφεύγεται η αποστήθιση της σειράς της σωστής απάντησης εις βάρος της ίδιας της απάντησης.

Ο εξεταζόμενος έχει συγκεκριμένο χρόνο να ολοκληρώσει το ερωτηματολόγιο. Ο διαθέσιμος χρόνος εξαρτάται από την κατηγορία εξέτασης και από το αν ο υποψήφιος είναι αναλφάβητος ή όχι.

Ειδικές κατηγορίες εξεταζομένων

Στην Ελλάδα δικαίωμα απόκτησης διπλώματος οδήγησης, εκτός από τους Έλληνες πολίτες, έχουν και οι αλλοδαποί ο οποίοι διαμένουν νόμιμα στην χώρα. Μεγάλο μέρος των αλλοδαπών αυτών δεν γνωρίζουν την ελληνική γλώσσα σε τέτοιο βαθμό ώστε να μπορούν να αξιολογηθούν με γραπτές εξετάσεις. Επίσης, δυσκολίες με το γραπτό λόγο αντιμετωπίζει και μέρος του πληθυσμού το οποίο είναι αναλφάβητο, ανήκει σε μειονότητες, κλπ. Προς διευκόλυνση των προηγούμενων κατηγοριών υποψηφίων, το Μ.Σ.Θ.Ε.Υ.Ο. πληροί δυο ακόμα λειτουργικές προδιαγραφές: Τα ερωτηματολόγια έχουν μεταφραστεί σε ξένες γλώσσες (αγγλικά, ρωσικά, αλβανικά) και σε περίπτωση που οριστεί, το σύστημα υπαγορεύει στον εξεταζόμενο τις ερωτήσεις και τις απαντήσεις. Οι αναλφάβητοι οι οποίοι εξετάζονται με το σύστημα της υπαγόρευσης έχουν στη διάθεσή τους 50% περισσότερο χρόνο για να ολοκληρώσουν τη διαδικασία.

Ευχρηστία Συστήματος

Το Μ.Σ.Θ.Ε.Υ.Ο. το χειρίζονται, κυρίως, δύο βασικές κατηγορίες χρηστών: οι πολίτες οι οποίοι εξετάζονται για την απόκτηση διπλώματος και οι εξεταστές οι οποίοι δεν είναι άλλοι από το προσωπικό της οικείας υπηρεσίας μεταφορών και επικοινωνιών. Το μεγαλύτερο μέρος του πληθυσμού και των δύο κατηγοριών έχει μικρή ή και μηδενική εμπειρία από χρήση ηλεκτρονικού υπολογιστή. Για το λόγο αυτό το Μ.Σ.Θ.Ε.Υ.Ο. πρέπει να είναι όσο το δυνατό πιο εύκολο στη χρήση του. Ειδικά το πρόγραμμα που τρέχει στους σταθμούς εξέτασης πρέπει να είναι τόσο εύκολο στη χρήση του, όσο και η σημείωση της σωστής απάντησης σε ένα φύλλο χαρτί. Επίσης, οι σχετικές οθόνες πρέπει να είναι φιλικές και να μην δημιουργούν άγχος στον εξεταζόμενο.

Τοπικός Έλεγχος Δεδομένων

Σύμφωνα με την ελληνική νομοθεσία, αρμόδια αρχή για την έκδοση διπλωμάτων οδήγησης και την τήρηση των σχετικών αρχείων είναι η δευτεροβάθμια τοπική αυτοδιοίκηση (νομαρχιακές υπηρεσίες μεταφορών και επικοινωνιών). Το Μ.Σ.Θ.Ε.Υ.Ο. πρέπει να υλοποιηθεί με τέτοιο τρόπο ώστε να μην παραβιάζεται ο προηγούμενος κανόνας. Η πρόσβαση στα δεδομένα του συστήματος πρέπει να γίνεται μόνο από το προσωπικό της αρμόδιας υπηρεσίας ή από προσωπικό εξουσιοδοτημένο από αυτή.

Ασφάλεια και Αξιοπιστία

Η έκδοση διπλωμάτων οδήγησης, στην συνείδηση μεγάλου μέρους των πολιτών της χώρας, είναι μία διαδικασία η οποία απέχει πολύ από το να χαρακτηριστεί ως “αδιάβλητη”. Η διαβλητότητα της διαδικασίας δεν αφορά μόνο το πρακτικό μέρος της εξέτασης, αλλά και το θεωρητικό. Προκειμένου να διασφαλιστεί η διαφάνεια στο σύστημα εξέτασης και η αντικειμενικότητα των αποτελεσμάτων, το Μ.Σ.Θ.Ε.Υ.Ο. πληροί τις ακόλουθες προδιαγραφές ασφαλείας:

- Η βάση δεδομένων που περιέχει τα στοιχεία των εξεταζομένων και τα αποτελέσματα των εξετάσεων είναι κλειδωμένα ώστε να μην μπορούν να τροποποιηθούν. Το ίδιο συμβαίνει και με τα αρχεία που αντιστοιχούν στα φύλλα εξέτασης του κάθε υποψηφίου.

- Ο εξυπηρετητής του Μ.Σ.Θ.Ε.Υ.Ο. δεν είναι συνδεδεμένος στο διαδίκτυο ώστε να μην μπορεί να παραβιαστεί από μακρυά. Πρόσβαση στον εξυπηρετητή έχει μόνο το προσωπικό το οποίο έχει φυσική πρόσβαση στο χώρο των εξετάσεων.

- Σε κάθε υπηρεσία της χώρας τρέχει ένα διακριτό αντίγραφο του Μ.Σ.Θ.Ε.Υ.Ο.. Με τον τρόπο αυτό διασφαλίζεται ότι σε περίπτωση μη εξουσιοδοτημένης χρήσης του συστήματος, εκτός υπηρεσίας, θα γίνει άμεσα γνωστή η πηγή της διαρροής. Επίσης, με τον τρόπο αυτό, η παραβίαση του συστήματος μίας υπηρεσίας δεν θα συνεπάγεται την παραβίαση του Μ.Σ.Θ.Ε.Υ.Ο. συνολικά.

- Τα εκτυπωμένα πρακτικά εξέτασης, τα οποία τοποθετούνται στον φάκελο του υποψηφίου, φέρουν ψηφιακή υπογραφή η οποία πιστοποιεί τα στοιχεία του υποψηφίου και το αποτέλεσμα της εξέτασης που αναγράφεται σε αυτό. Με τον τρόπο αυτό διασφαλίζεται ότι δεν θα δημιουργούνται πρακτικά εξέτασης με απλή επεξεργασία κειμένου.

- Τα ερωτηματολόγια στα οποία εξετάζονται οι υποψήφιοι δημιουργούνται μετά την έναρξη της διαδικασίας εξέτασης, και όχι όταν προγραμματίζονται οι ομάδες, ώστε να μην υπάρχει χρόνος να παραβιαστούν τα ερωτηματολόγια.

- Οι θέσεις στις οποίες θα καθίσουν οι υποψήφιοι κληρώνονται κατά την έναρξη της διαδικασίας ώστε να αποφεύγεται να κάθονται σε διπλανές θέσεις γνωστοί, μαθητές της ίδιας σχολής, κλπ.

- Το σύστημα πρέπει να είναι, κατά το δυνατό, ανθεκτικό σε λάθη. Για παράδειγμα, δεν θα πρέπει να σταματά η διαδικασία εξέτασης αν κολλήσει ένας σταθμός εξέτασης ή αν δημιουργηθεί πρόβλημα στον εξυπηρετητή. Επίσης η πτώση του δικτύου, από αστοχία υλικού ή ανθρώπινο λάθος (ξήλωμα καλωδίου), δεν θα πρέπει να καταστρέφουν τη διαδικασία.

Οι παραπάνω προδιαγραφές εφαρμόστηκαν στο σύνολό τους. Το σύστημα που δημιουργήθηκε από αυτές, παρόλο που έχει αρχίσει να... δείχνει τα χρόνια του, λειτουργεί χωρίς σοβαρά προβλήματα και κυρίως με την αποδοχή όλων των εμπλεκόμενων πλευρών.

ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ Μ.Σ.Θ.Ε.Υ.Ο.

Η εφαρμογή του συστήματος αυτοματοποίησης της διαδικασίας θεωρητικής εξέτασης υποψηφίων οδηγών καλύπτει το ακόλουθο λειτουργικό σενάριο, που αποτελείται από τρία στάδια διαδικασιών. Τα στάδια αυτά περιλαμβάνουν την προετοιμασία της εξέτασης (φάση Α), την διενέργεια της εξέτασης (φάση Β) και τέλος την ολοκλήρωση της όλης διαδικασίας (φάση C). Οι τρεις αυτές φάσεις, με τις όποιες αναγκαίες διαφοροποιήσεις απαιτούνται, είναι ίδιες ή παρόμοιες στα διάφορα πληροφοριακά συστήματα διεξαγωγής εξετάσεων. Για το λόγο αυτό είναι απαραίτητη η αναλυτική παρουσίαση τους.

Προκαταρκτικά / Προετοιμασία εξέτασης

Α1: Σε κάθε εξεταστικό κέντρο δημιουργείται για κάθε ομάδα εξέτασης (αίθουσα) το αρχείο με τα στοιχεία των εξεταζομένων για κάθε εξέταση.

Α2: Το αρχείο φορτώνεται στον εξυπηρετητή του Εξεταστικού Κέντρου και με την έναρξη της διαδικασίας εξέτασης, το σύστημα επιλέγει τυχαία τη θέση / τερματικό του κάθε υποψήφιου, δημιουργεί τα ηλεκτρονικά φύλλα εξέτασης με τις τυχαία επιλεγμένες ερωτήσεις και τα μοιράζει στους σταθμούς εξέτασης, ενώ παράλληλα εκτυπώνει για τον υπεύθυνο του εξεταστικού κέντρου ένα κατάλογο με τα ονόματα και τις θέσεις των υποψηφίων.

Α3: Ακολουθεί έλεγχος ταυτοπροσωπίας, αφενός βάσει του ονόματος και των άλλων στοιχείων του υποψηφίου, που εμφανίζονται στην οθόνη και αφετέρου του Δελτίου Εκπαίδευσης και Εξέτασης Υποψηφίου Οδηγού (Δ.Ε.Ε.) και του δελτίου ταυτότητας, που πρέπει να φέρει μαζί του κάθε υποψήφιος. Στον εξυπηρετητή καταγράφονται οι απόντες.

Α4: Ο υποψήφιος επιβεβαιώνει τα εμφανιζόμενα στην οθόνη στοιχεία του και ενημερώνεται από τον υπεύθυνο του Εξεταστικού Κέντρου για τη διαδικασία της εξέτασης και το τρόπο απάντησης των ερωτήσεων.

Διενέργεια εξέτασης

Β1: Οι υποψήφιοι ξεκινούν ταυτόχρονα την εξέταση. Σε κάθε υποψήφιο το σύστημα παρουσιάζει το δημιουργημένο ερωτηματολόγιο. Σε περίπτωση που ο υποψήφιος εξετάζεται και σε δεύτερη κατηγορία, οι ερωτήσεις της δεύτερης εξέτασης εμφανίζονται, αφού ολοκληρωθεί η διαδικασία της εξέτασης της πρώτης κατηγορίας. Οι σταθμοί εξέτασης οι οποίοι είχαν ανατεθεί σε απόντες υποψήφιους κλειδώνονται ηλεκτρονικά.

Β2: Ο χρόνος εξέτασης είναι περιορισμένος για κάθε εξέταση, καθορίζεται από τις κείμενες διατάξεις και δεν υπολογίζεται αθροιστικά στην περίπτωση που ο υποψήφιος εξετάζεται και σε δεύτερη κατηγορία.

Β3: Το σύστημα επιτρέπει στον εξεταζόμενο να προχωρήσει στην επόμενη ερώτηση, χωρίς να απαντήσει αυτή την οποία επεξεργάζεται εκείνη τη στιγμή. Αφού φθάσει ο εξεταζόμενος στην τελευταία ερώτηση, το σύστημα επιστρέφει αυτόματα στη πρώτη μη απαντημένη ερώτηση και συνεχίζει κατά τον ίδιο τρόπο μέχρι να απαντηθούν όλες οι ερωτήσεις ή εξαντληθεί ο χρόνος που αντιστοιχεί στην κατηγορία που εξετάζεται.

Β4: Μόλις ο υποψήφιος ολοκληρώσει το ερωτηματολόγιο του, ο σταθμός εξέτασης “κλειδώνει”, ώστε να μην έχει πρόσβαση σε αυτόν ο εξεταζόμενος.

Β5: Η φάση της διενέργειας εξέτασης ολοκληρώνεται όταν όλοι οι υποψήφιοι ολοκληρώσουν την διαδικασία.

Ολοκλήρωση διαδικασίας

C1: Το σύστημα συγκεντρώνει στον εξυπηρετητή τα συμπληρωμένα φύλλα εξέτασης και τα βαθμολογεί.

C2: Για κάθε εξεταζόμενο το σύστημα εκτυπώνει, πρακτικά εξέτασης με τα στοιχεία του υποψηφίου, τις ερωτήσεις του ερωτηματολογίου στο οποίο εξετάσθηκε, τις απαντήσεις που αυτός έδωσε (χρησιμοποιώντας ως αναφορά τα επίσημα εκπαιδευτικά βιβλία του Υπουργείου Μεταφορών), την ορθότητα η όχι αυτών, καθώς και τα τελικά αποτελέσματα της πραγματοποιηθείσας εξέτασης.

C3: Το σύστημα εκδίδει συνολικό κατάλογο με τα στοιχεία των εξετασθέντων και τα τελικά αποτελέσματα.

C4: Τα πρακτικά εξέτασης υπογράφονται από τους εξεταστές, ενημερώνονται οι υποψήφιοι για το αποτέλεσμα της εξέτασης και ακολουθούνται οι όποιες διαδικασίες προβλέπονται από τον κανονισμό (π.χ. αρχειοθέτηση αποτελεσμάτων).

ΑΡΧΙΤΕΚΤΟΝΙΚΗ Μ.Σ.Θ.Ε.Υ.Ο.

Υλικό και Λογισμικό

Το Μ.Σ.Θ.Ε.Υ.Ο., ως σύστημα, αποτελείται από τις ακόλουθες συνιστώσες:

- Τον εξυπηρετητή.
- Τους σταθμούς εξέτασης.
- Τον εξοπλισμό δικτύωσης.
- Τον εκτυπωτή.

Σε μία τυπική αίθουσα εξετάσεων υπάρχουν, επίσης, UPS και ένα modem το οποίο χρησιμοποιείται για την DIALUP σύνδεση με την κεντρική υπηρεσία του Υ.Μ.Ε. για λόγους τηλεϋποστήριξης.

Ο εξυπηρετητής του Μ.Σ.Θ.Ε.Υ.Ο. είναι ένα σύστημα στο οποίο, σήμερα, τρέχει κάτω από κάποια έκδοση του λειτουργικού συστήματος Microsoft Windows XP[2]. Το υλικό του εξυπηρετητή δεν παρουσιάζει ιδιαίτερο ενδιαφέρον. Πρόκειται για απλό PC, όχι εξειδικευμένο server, το οποίο συνήθως περιέχει δύο σκληρούς δίσκους οργανωμένους σε συστοιχία RAID 1 για ασφάλεια των δεδομένων και CD-Recorder για δημιουργία εφεδρικών αντιγράφων των δεδομένων αυτών. Στον εξυπηρετητή τρέχει το πρόγραμμα “Server” μέσω του οποίου το προσωπικό εκτελεί τις βασικές εργασίες του συστήματος. Οι βασικές λειτουργίες του συστήματος είναι:

- Εισαγωγή ομάδας εξέτασης. Τα στοιχεία των εξεταζόμενων μίας ομάδας εξέτασης εισάγονται στο σύστημα μέσω ενός φύλλου του Microsoft Excel συγκεκριμένης γραμμογράφησης. Με τον τρόπο αυτό η προετοιμασία της ομάδας μπορεί να γίνει στον υπολογιστή του γραφείου του υπεύθυνου κάθε φορά υπαλλήλου. Το Microsoft Excel είναι ένα πρόγραμμα το οποίο βρίσκεται εγκατεστημένο στους περισσότερους υπολογιστές των υπαλλήλων του Υ.Μ.Ε. και οι περισσότεροι υπάλληλοι έχουν κάποια εξοικείωση με αυτό.

- Διενέργεια Εξέτασης. Αποτελεί τη βασική λειτουργία του συστήματος. Μέσω απλών στην χρήση οθονών, ο χειριστής ο οποίος είναι ένας από τους εξεταστές, επιλέγει την προς εξέταση ομάδα (παρουσιάζονται μόνο αυτές που έχουν προγραμματιστεί για την συγκεκριμένη ημερομηνία), καταγράφει τα ονόματα των εξεταστών για να τυπωθούν στα πρακτικά (τα επιλέγει από λίστα), και καταγράφει τους απόντες. Στην συνέχεια ξεκινά η εξέταση. Μόλις τελειώσουν όλοι οι εξεταζόμενοι το πρόγραμμα μαζεύει τα ερωτηματολόγια και τα βαθμολογεί. Στο τέλος τυπώνει τα αποτελέσματα. Πρέπει να τονιστεί ότι από την έναρξη της εξέτασης και μέχρι το τέλος της συνολικής διαδικασίας (εκτύπωση αποτελεσμάτων), το πρόγραμμα λειτουργεί αυτοματοποιημένα χωρίς να χρειάζεται ανθρώπινη παρέμβαση (γεγονός το οποίο, μεταξύ άλλων, προσφέρει και μία αίσθηση διαφάνειας στους εξεταζόμενους).

- Αναζήτηση και εκτύπωση πρακτικών περασμένων εξετάσεων

- Διαχειριστικές Λειτουργίες. Με την βοήθεια απλών στη χρήση διεπαφών ο υπεύθυνος της αίθουσας μπορεί να εισάγει ή τροποποιήσει τα στοιχεία των σταθμών εξέτασης (δικτυακό όνομα, κατάσταση, κλπ), των εξεταστών, κλπ.

2 Αξίζει να σημειωθεί ότι το σύστημα σχεδιάστηκε για να λειτουργήσει σε οικονομικό εξοπλισμό με Microsoft Windows 98 και Microsoft Windows ME. Το στοιχείο είναι χρήσιμο προκειμένου να γίνουν κατανοητές οι επιλογές της ομάδας ανάπτυξης για την αρχιτεκτονική του συστήματος, ιδιαίτερα στα σημεία εκείνα που σχετίζονται με τη δικτύωση.

- Τερματισμός σταθμών εξέτασης. Η λειτουργία αυτή τερματίζει (shutdown) τους σταθμούς εξέτασης χωρίς να χρειάζεται κάποιος υπάλληλος να ενεργοποιήσει την αντίστοιχη λειτουργία σε κάθε ένα από αυτούς. Υπάρχουν υπηρεσίες οι οποίες διαθέτουν πάνω από 20 σταθμούς εξέτασης στην αίθουσά τους.

Οι σταθμοί εξέτασης είναι, επίσης, απλά PCs τα οποία τρέχουν κάτω από το λειτουργικό σύστημα Microsoft Windows XP. Είναι εφοδιασμένοι με οθόνες αφής (touch screens) και μπορούν να χρησιμοποιηθούν εύκολα και από ανθρώπους οι οποίοι δεν έχουν γνώση χρήσης Η.Υ. Ο εξεταζόμενος απλά ακουμπάει το δάκτυλό του στην απάντηση την οποία θεωρεί σωστή και την κατοχυρώνει πατώντας με το δάκτυλο το κουμπί “ΕΠΙΒΕΒΑΙΩΣΗ ΑΠΑΝΤΗΣΗΣ”. Η κεντρική μονάδα, το πληκτρολόγιο και το ποντίκι των σταθμών εξέτασης δεν είναι προσπελάσιμα από του εξεταζόμενους. Οι σταθμοί εξέτασης διαθέτουν και ατομικά ακουστικά ώστε οι αναλφάβητοι υποψήφιοι να μπορούν να ακούν την εκφώνηση των ερωτήσεων και των απαντήσεων. Στον σταθμό εξέτασης τρέχουν δύο προγράμματα:

- Το πρόγραμμα “Client”. Το Client παρουσιάζει στον εξεταζόμενο την κάθε ερώτηση, την εικόνα (αν υπάρχει), τις πιθανές απαντήσεις και καταγράφει την επιλογή του. Παρουσιάζει, επίσης, με γραφικό τρόπο τον διαθέσιμο χρόνο που έχει στη διάθεσή του ο εξεταζόμενος για να ολοκληρώσει τη διαδικασία. Το πρόγραμμα τρέχει στην σταθερή ανάλυση των 1024 X 768 pixels με large fonts (120 dpi). Αιτία για αυτή τη φαινομενικά εκκεντρική επιλογή είναι ότι το σύστημα σχεδιάστηκε για να δουλέψει σε οθόνες 14 ιντσών και παρόλα αυτά να είναι ευανάγνωστη και από εξεταζόμενους με μειωμένες δυνατότητες όρασης όπως τα άτομα προχωρημένης ηλικίας.

- Το πρόγραμμα Daemon. Το πρόγραμμα αυτό είναι Service το οποίο ενεργοποιείται με την εκκίνηση του συστήματος. Η δουλειά του είναι να “ακούει” το Server και μόλις λάβει το... σύνθημα να ενεργοποιήσει το Client.

Ο εξοπλισμός δικτύου αποτελείται μόνο από ένα απλό switch (και την αντίστοιχη καλωδίωση). Για την λειτουργία του Μ.Σ.Θ.Ε.Υ.Ο. δεν απαιτούνται δρομολογητές ή άλλος ειδικός δικτυακός εξοπλισμός.

Ο εκτυπωτής του συστήματος είναι συνδεδεμένος στον εξυπηρετητή. Δεν διαμοιράζεται στο δίκτυο. Ο εκτυπωτής που εξυπηρετεί μεγάλες, σε πλήθος εξεταζομένων υπηρεσίες, πρέπει να είναι γρήγορος προκειμένου να μην καθυστερεί η έκδοση των αποτελεσμάτων.

Εσωτερική Λειτουργία Μ.Σ.Θ.Ε.Υ.Ο.

Παρόλο που το Μ.Σ.Θ.Ε.Υ.Ο. αποτελείται από τον εξυπηρετητή και τους σταθμούς εξέτασης, δεν είναι ένα τυπικό client – server σύστημα. Το επίπεδο της αξιοπιστίας των υπηρεσιών δικτύου του λειτουργικού συστήματος Windows ΜΕ σε συνδυασμό

με την ανάγκη για ένα απλό σύστημα το οποίο θα τρέχει σε υπολογιστές χαμηλών απαιτήσεων, δεν επέτρεψαν την εφαρμογή μίας τέτοιας αρχιτεκτονικής. Έτσι, οι σταθμοί εξέτασης, κατά την διαδικασία της εξέτασης, δεν καταναλώνουν υπηρεσίες του εξυπηρετητή ούτε βασίζονται στο δίκτυο για να διεκπεραιώσουν τις εργασίες τους.[3] Αντίθετα, το δίκτυο χρησιμοποιείται μόνο για μεταφορά αρχείων. Το δίκτυο χρησιμοποιεί αρχιτεκτονική διαμοιραζόμενων φακέλων (shared folders) του Windows.

Κάθε φορά που ξεκινάει η εξέταση μίας ομάδας, ο εξυπηρετητής κληρώνει τις θέσεις στις οποίες θα καθίσει ο κάθε εξεταζόμενος. Για κάθε εξεταζόμενο δημιουργεί και ένα φύλλο εξέτασης, δηλαδή ένα Microsoft Access αρχείο, το οποίο περιέχει τον πίνακα με τα στοιχεία του εξεταζομένου και της εξέτασης, τον πίνακα με τις ερωτήσεις και τον πίνακα με τις πιθανές απαντήσεις. Στον τελευταίο καταγράφονται και οι απαντήσεις του εξεταζόμενου. Τα φύλλα μεταφέρονται από τον εξυπηρετητή στους σταθμούς εξέτασης με απλή αντιγραφή αρχείων.

Όταν ολοκληρωθεί η μεταφορά των αρχείων σε όλους τους σταθμούς εξέτασης, ο εξυπηρετητής στέλνει εντολή έναρξης του προγράμματος Client. Οι σταθμοί εξέτασης ανοίγουν το φύλλο εξέτασης και παρουσιάζουν τα στοιχεία του εξεταζόμενου. Αφού γίνει ο έλεγχος ταυτοπροσωπίας σε όλους τους σταθμούς και γίνει καταγραφή των πιθανών απουσιών, ενεργοποιείται η διαδικασία εξέτασης. Το πρόγραμμα παρουσιάζει τις ερωτήσεις και τις πιθανές απαντήσεις και καταγράφει τις επιλογές του υποψηφίου. Οι εικόνες των ερωτήσεων και τα αρχεία ήχου για την εκφώνηση των ερωτήσεων και των απαντήσεων βρίσκονται στον σκληρό δίσκο όλων των σταθμών εξέτασης και δεν χρειάζεται να μεταφέρονται μέσω του δικτύου.

Μόλις ολοκληρωθεί η διαδικασία εξέτασης σε όλους τους σταθμούς, ο εξυπηρετητής αντιγράφει τα φύλλα από τους σταθμούς εξέτασης στον σκληρό του δίσκο. Στην συνέχεια τα βαθμολογεί και τυπώνει τα αποτελέσματα.

Η μεταφορά εντολών και η καταγραφή τρέχουσας κατάστασης γίνεται με την εγγραφή ενός ακέραιου αριθμού σε ένα αρχείο. Το αρχείο αυτό βρίσκεται σε κάθε σταθμό εξέτασης. Όταν ο εξυπηρετητής θέλει να στείλει μία εντολή στον σταθμό εξέτασης γράφει τον κωδικό της στο αρχείο αυτό. Ο σταθμός εξέτασης ενημερώνει τον εξυπηρετητή για την αλλαγή της κατάστασής του (π.χ. ο υποψήφιος ολοκλήρωσε την διαδικασία εξέτασης) γράφοντας τον αντίστοιχο κωδικό στο ίδιο αρχείο.

Το πρόγραμμα Daemon εκτελείται αυτόματα με την εκκίνηση του λειτουργικού συστήματος. Από τεχνικής πλευράς, δεν είναι υπηρεσία (service), είναι ένα απλό πρόγραμμα το οποίο δεν έχει διεπαφή με το χρήστη. Εξετάζει σε τακτά διαστήματα τα περιεχόμενα του αρχείου με το οποίο επικοινωνεί ο εξυπηρετητής με τον κάθε

3 Σύμφωνα με την ομάδα ανάπτυξης, το Μ.Σ.Θ.Ε.Υ.Ο. είναι ένα on-line σύστημα το οποίο λειτουργεί ακόμα και αν πέσει το δίκτυο.

σταθμό εξέτασης και όταν διαβάσει την κατάλληλη εντολή ενεργοποιεί το πρόγραμμα Client.

Η απλή αρχιτεκτονική του Μ.Σ.Θ.Ε.Υ.Ο., όπως αυτή παρουσιάστηκε συνοπτικά, έχει μεταξύ άλλων το ακόλουθο πλεονέκτημα: Η διαδικασία μπορεί να ολοκληρωθεί χωρίς να δημιουργηθούν σοβαρά προβλήματα ακόμα και αν κατά τη διάρκεια της εξέτασης αστοχήσει το δίκτυο ή ακόμα και αν δημιουργηθεί πρόβλημα στον εξυπηρετητή. Επίσης, αν κατά τη διάρκεια της εξέτασης παρουσιάσει πρόβλημα κάποιος σταθμός εξέτασης, η διαδικασία ολοκληρώνεται για τους υποψήφιους οι οποίο εξετάζονται σε άλλους σταθμούς, δημιουργώντας έτσι την ελάχιστη δυνατή αναστάτωση. Το χαρακτηριστικό αυτό είναι εξαιρετικά σημαντικό για ένα πληροφοριακό σύστημα το οποίο υποστηρίζει μία διαδικασία του δημόσιου τομέα η οποία δεν είναι ιδιαίτερα... δημοφιλής στους πολίτες.

ΟΙ ΑΔΥΝΑΜΙΕΣ ΤΟΥ Μ.Σ.Θ.Ε.Υ.Ο.

Κατά γενική ομολογία το Μ.Σ.Θ.Ε.Υ.Ο. είναι ένα πετυχημένο πληροφοριακό σύστημα διότι: τόσο η αρχική του ανάπτυξη όσο και η ανάπτυξη των επεκτάσεων του ολοκληρώθηκαν στο προϋπολογισμένο χρόνο, το κόστος ανάπτυξης και εφαρμογής του δεν ξεπέρασε τον αρχικό προϋπολογισμό και κάλυψε απόλυτα τις προδιαγραφές του. Παρόλα αυτά, η εφαρμογή του συστήματος έκανε εμφανή ορισμένα μειονεκτήματα του συστήματος. Ακολουθεί μία σύντομη αναφορά των μειονεκτημάτων αυτών.

Δυσκολίες ενημέρωσης και αναβάθμισης

Όπως αναφέρθηκε σε προηγούμενες παραγράφους, το Μ.Σ.Θ.Ε.Υ.Ο. λειτουργεί αυτόνομα σε κάθε υπηρεσία χωρίς να είναι συνδεδεμένο στο διαδίκτυο ή άλλο δίκτυο του Υ.Μ.Ε.. Επίσης, κάθε υπηρεσία διαθέτει μία διακριτή έκδοση του λογισμικού η οποία διαφοροποιείται από τις υπόλοιπες στα κλειδώματα των βάσεων δεδομένων. Αυτό σημαίνει πως αν απαιτηθεί να ενημερωθούν τα δεδομένα στη βάση δεδομένων του συστήματος (π.χ. προσθαφαίρεση μερικών ερωτήσεων) τότε θα πρέπει να πραγματοποιηθούν τα ακόλουθα βήματα:

- Οι απαιτούμενες αλλαγές πραγματοποιούνται στην πρωτότυπη βάση δεδομένων.
- Επαναδημιουργούνται τα διακριτά αντίγραφα της βάσης δεδομένων, ένα για κάθε μία υπηρεσία μεταφορών και επικοινωνιών της χώρας.
- Τα αρχεία αποστέλλονται στις οικίες υπηρεσίες και αντικαθιστούν τα αντίστοιχα υπάρχοντα.

Η αναβάθμιση του συστήματος, για παράδειγμα η εισαγωγή μίας νέας κατηγορίας εξέτασης, απαιτεί περαιτέρω εργασία. Εκτός από την τροποποίηση της βάσης δεδομένων, συνήθως απαιτείται τροποποίηση του κώδικα, όχι μόνο του εξυπηρετητή, αλλά και των σταθμών εξέτασης, του κάθε σταθμού ξεχωριστά. Επιπλέον απαιτείται ενημέρωση των δεδομένων που βρίσκονται στους σταθμούς εξέτασης (εικόνες και ήχοι). Οι εγκατάσταση των ενημερώσεων αυτών απαιτεί σχετικά εξειδικευμένο προσωπικό το οποίο δεν είναι πάντα διαθέσιμο, ειδικά στις υπηρεσίες της επαρχίας.

Υψηλό κόστος δημιουργίας και λειτουργίας αίθουσας εξετάσεων

Η δημιουργία μιας νέας αίθουσας εξετάσεων έχει υψηλό κόστος. Το πλήθος των υποψηφίων οι οποίοι θα πρέπει να μπορούν να εξετάζονται ταυτόχρονα (ομάδα εξέτασης) εξαρτάται από τις συνθήκες του τόπου στον οποίο εδρεύει και τον οποίο εξυπηρετεί η υπηρεσία (τοπικός πληθυσμός, δυνατότητα της υπηρεσίας να προγραμματίζει καθημερινά εξετάσεις, κ.λ.π.). Εκτός από τον εξυπηρετητή, τον εκτυπωτή και το δικτυακό εξοπλισμό, για κάθε ένα υποψήφιο θα πρέπει να υπάρχει ένας προσωπικός υπολογιστής με οθόνη αφής. Σε υπηρεσίες με μεγάλο πλήθος εξεταζομένων, θα πρέπει να υπάρχουν εφεδρικοί υπολογιστές, εγκατεστημένοι και έτοιμοι για λειτουργία. Σε αντίθετη περίπτωση και σε περίπτωση βλάβης σταθμού εξέτασης, υποψήφιοι οι οποίοι χρειάστηκε να περιμένουν αρκετές μέρες μέχρι να έρθει η ώρα τους να εξεταστούν και πιθανόν ανέβαλαν επαγγελματικές υποχρεώσεις προκειμένου να παραβρεθούν σε αυτές, δεν θα μπορέσουν να λάβουν μέρος στις εξετάσεις και θα πρέπει να περιμένουν και αρκετές μέρες μέχρι την νέα ημερομηνία εξέτασης. Οι αντιδράσεις σε αυτές τις περιπτώσεις δεν είναι ευχάριστες.

Οι σταθμοί εξέτασης πρέπει να είναι τοποθετημένοι σε ειδικά έπιπλα. Τα έπιπλα αυτά, πέρα από την εργονομία, θα πρέπει να διασφαλίζουν δύο πράγματα: την κατά το δυνατό απομόνωση κάθε υποψηφίου από τους διπλανούς του και την προστασία του εξοπλισμού από τον υποψήφιο. Ο υποψήφιος θα πρέπει να μπορεί να έχει πρόσβαση στην οθόνη αφής και, αν πρέπει, στα ατομικά ακουστικά, αλλά όχι στο ποντίκι και στο πληκτρολόγιο. Σε καμία, βέβαια, περίπτωση ο υποψήφιος δεν πρέπει να έχει πρόσβαση στο κουμπί ενεργοποίησης / απενεργοποίησης του υπολογιστή, στο κουμπί reset, στο τροφοδοτικό του, στις θήρες usb, κ.λ.π.

Η τροφοδοσία του ηλεκτρονικού εξοπλισμού της αίθουσας εξέτασης θα πρέπει να υποστηρίζεται από μονάδες αδιάλειπτης παροχής ενέργειας (UPS), ικανές να κρατήσουν ζωντανό τον εξοπλισμό για αρκετή ώρα, τουλάχιστον για την ονομαστική χρονική διάρκεια της απαιτητικότερης κατηγορίας εξέτασης.

Η αίθουσα, προκειμένου να μην κινδυνεύουν άνθρωποι και εξοπλισμός, θα πρέπει να κλιματίζεται.

Πέρα από τη δημιουργία της αίθουσας εξετάσεων, το Μ.Σ.Θ.Ε.Υ.Ο. απαιτεί πόρους της υπηρεσίας προκειμένου να συντηρηθεί. Εκτός από τα αναλώσιμα (χαρτί, toner, κ.λ.π.) και την δαπάνη για ηλεκτρικό ρεύμα, το σύστημα κοστίζει στην υπηρεσία σε απαραίτητα συμβόλαια συντήρησης και άμεσης επισκευής / αντικατάστασης εξοπλισμού, συχνή αντικατάσταση μπαταριών για τα UPSs, κλπ.

Σε όλα τα προηγούμενα θα πρέπει να υπολογιστεί και το κόστος της ίδιας της αίθουσας, ως χώρου. Η αίθουσα των εξετάσεων του Μ.Σ.Θ.Ε.Υ.Ο., τόσο λόγω της διαμόρφωσης της, όσο και για λόγους ασφαλείας θα πρέπει να χρησιμοποιείται αποκλειστικά για εξετάσεις και δεν μπορεί να καλύψει άλλες ανάγκες της υπηρεσίας.

Έλλειψη Συνδεσιμότητας με άλλα συστήματα

Η αρχική απαίτηση του Υ.Μ.Ε. της μη συνδεσιμότητας του Μ.Σ.Θ.Ε.Υ.Ο. με άλλα συστήματα, εξυπηρετούσε τους ακόλουθους σκοπούς:

- Να κρατηθεί το σύστημα απλό στη χρήση του.
- Να μην αλλάξουν ριζικά τις διαδικασίες εξέτασης, αλλά να διατηρηθούν, κατά το δυνατό, όπως τις είχαν συνηθίσει τα εμπλεκόμενα μέρη (να αλλάξει το μέσο, όχι ο τρόπος εξέτασης).
- Να αναπτυχθεί το σύστημα γρήγορα και με μικρό κόστος.
- Να μπορεί να λειτουργήσει χωρίς την ύπαρξη ειδικευμένου, σε πληροφοριακά συστήματα, προσωπικού.

Η προδιαγραφή τηρήθηκε και οι σκοποί επιτεύχθηκαν. Η αυτόνομη λειτουργία του Μ.Σ.Θ.Ε.Υ.Ο., όμως, του περιορίζει τη λειτουργικότητα. Ακολουθεί σύντομη αναφορά στους βασικούς περιορισμούς.

- Τα δεδομένα των ομάδων εξέτασης δεν εισάγονται από κάποιο άλλο σύστημα. Αντίθετα, παρόλο που υπάρχουν διαθέσιμα σε ηλεκτρονική μορφή σε άλλα συστήματα, επαναπληκτρολογούνται στην μορφή που τα απαιτεί το σύστημα (φύλλο του Microsoft Excel) σε άλλους υπολογιστές και μεταφέρονται στο Μ.Σ.Θ.Ε.Υ.Ο. με τη χρήση περιφερειακών αποθηκευτικών μέσων.
- Τα αποτελέσματα των εξετάσεων είναι διαθέσιμα μόνο σε έντυπη μορφή. Για να ενημερωθούν τα άλλα πληροφορικά συστήματα του υπουργείου ή να ανακοινωθούν ηλεκτρονικά στο κοινό (π.χ. μέσω του portal του Υ.Μ.Ε.) πρέπει να εισαχθούν σε αυτά με πληκτρολόγηση.
- Το υπουργείο δεν μπορεί να αντλήσει στρατηγική πληροφορία από τα αποτελέσματα των εξετάσεων (στατιστικά για τα λάθη, ποσοστά επιτυχίας ανά περιοχή, κ.λ.π.)

- Τα αποτελέσματα των εξετάσεων δεν είναι διαθέσιμα στους ελεγκτικούς μηχανισμούς του κράτους. Για παράδειγμα είναι πολύ χρονοβόρο, αν όχι και δύσκολο, να ελεγχθεί η εγκυρότητα ενός διπλώματος μέσω του αρχείου θεωρητικών εξετάσεων.

ΘΕΜΑΤΑ ΑΣΦΑΛΕΙΑΣ

Η ασφάλεια των πληροφοριακών συστημάτων στηρίζεται σε τρεις βασικές ιδέες:

- Στην ακεραιότητα των δεδομένων, δηλαδή τη διατήρηση των δεδομένων σε μια γνωστή κατάσταση χωρίς ανεπιθύμητες τροποποιήσεις, αφαιρέσεις ή προσθήκες από μη εξουσιοδοτημένα άτομα.
- Στην διαθεσιμότητα των δεδομένων, δηλαδή την εξασφάλιση ότι οι υπολογιστές, τα δίκτυα και τα δεδομένα θα είναι στη διάθεση των χρηστών όποτε απαιτείται η χρήση τους.
- Στην εμπιστευτικότητα των δεδομένων, δηλαδή την εξασφάλιση ότι ευαίσθητες πληροφορίες δεν θα πρέπει να αποκαλύπτονται σε μη εξουσιοδοτημένα άτομα.

Ακολουθεί μία σύντομη περιγραφή της εφαρμογής των ιδεών αυτών στο Μ.Σ.Θ.Ε.Υ.Ο.

Η διασφάλιση της εμπιστευτικότητας των δεδομένων βασίζεται στην φυσική προστασία του συστήματος από μη εξουσιοδοτημένη πρόσβαση. Το σύστημα για λόγους απλότητας χρήσης και διαχείρισης δεν υποστηρίζει αυθεντικοποίηση των χρηστών, ούτε σε επίπεδο εφαρμογής ούτε σε επίπεδο λειτουργικού συστήματος. Η μόνη προστασία του συστήματος, σε ό,τι αφορά την εμπιστευτικότητα των δεδομένων, είναι μία... κλειδωμένη πόρτα. Το μέτρο, πάντως, κρίνεται επαρκές λαμβάνοντας υπόψη πόσο χαμηλή είναι η πιθανότητα να προσπαθήσει κάποιος να διαρρήξει τα γραφεία μίας υπηρεσίας μεταφορών και επικοινωνιών μόνο και μόνο για να αποκτήσει την σχετική πληροφορία.

Η εξασφάλιση της διαθεσιμότητας των δεδομένων ενός συστήματος όπως το Μ.Σ.Θ.Ε.Υ.Ο. εξαρτάται, αποκλειστικά, από τη λειτουργία του υλικού του εξυπηρετητή. Όσο το σύστημα λειτουργεί, τα δεδομένα είναι διαθέσιμα. Η διαθεσιμότητα δεν κινδυνεύει από δικτυακές ή άλλες επιθέσεις λόγω της αρχιτεκτονικής του. Κινδυνεύει μόνο από βλάβες υλικού ή φυσικές καταστροφές (πλημμύρες, πυρκαϊές, κ.λ.π.). Το αντίμετρο στον κίνδυνο αυτό είναι η συχνή δημιουργία εφεδρικών αντιγράφων της βάσης δεδομένων που βρίσκεται στον εξυπηρετητή. Το πρωτόκολλο λειτουργίας του συστήματος, όπως αυτό περιγράφεται σε σχετικές εγκυκλίους, ορίζει ότι πρέπει να δημιουργούνται συχνά εφεδρικά αντίγραφα του συστήματος. Δεν είναι γνωστό το κατά πόσο εφαρμόζονται

οι διαδικασίες δημιουργίας εφεδρικών αντιγράφων, ειδικά στις περιφερειακές υπηρεσίες όπου η έλλειψη ειδικευμένου, στη χρήση ηλεκτρονικών υπολογιστών, προσωπικού είναι έντονη.

Η ακεραιότητα των δεδομένων είναι πολύ σημαντική για ένα σύστημα εξετάσεων όπως το Μ.Σ.Θ.Ε.Υ.Ο. διότι, σε μεγάλο βαθμό, πάνω σε αυτή βασίζεται η αξιοπιστία του συνολικού συστήματος. Στην συγκεκριμένη περίπτωση ακεραιότητα δεδομένων σημαίνει:

- Τα στοιχεία του εξεταζομένου μίας εξέτασης να μην τροποποιούνται.
- Τα περιεχόμενα του ερωτηματολογίου, οι απαντήσεις του εξεταζομένου και κατά συνέπεια το αποτέλεσμα της εξέτασης να μην αλλοιώνονται.
- Τα πρακτικά που τυπώνονται να παρουσιάζουν την ακριβή εικόνα της εξέτασης ειδικά σε ό,τι αφορά τα στοιχεία του εξεταζόμενου και το αποτέλεσμα της εξέτασης.
- Τα γενικά δεδομένα τα οποία αφορούν την εξέταση και καταγράφονται στο σύστημα (π.χ. ημερομηνία και ώρα έναρξης και λήξης της εξέτασης) πρέπει να είναι τα πραγματικά.

Η δυσκολία στο να κρατηθεί ένα σύστημα σαν το Μ.Σ.Θ.Ε.Υ.Ο. ασφαλές είναι ότι οι κίνδυνοι δεν προέρχεται από κάποιες ξένες στο σύστημα οντότητες, π.χ. Crackers οι οποίοι προσπαθούν να αποκτήσουν πρόσβαση στο σύστημα μέσω δικτύου. Αντίθετα, το σύστημα κινδυνεύει από τους ίδιους τους χρήστες του και ιδιαίτερα την κατηγορία χρηστών η οποία είναι υπεύθυνη για την εύρυθμη λειτουργία του: το προσωπικό των υπηρεσιών μεταφορών και επικοινωνιών στις οποίες λειτουργεί. Η φυσική πρόσβαση που έχουν οι χρήστες αυτοί στο σύστημα τους δίνει τη δυνατότητα με μικρό ή μεγάλο κόπο και την βοήθεια “ειδικών” να το παραβιάσουν.

Για παράδειγμα μπορούν:

- Να ξεκλειδώσουν τη βάση δεδομένων του συστήματος[4] αποτελεσμάτων και να τροποποιήσουν τα στοιχεία που βρίσκεται σε αυτή.
- Να τροποποιήσουν τα φύλλα εξέτασης.
- Να αντιγράψουν το λογισμικό του Μ.Σ.Θ.Ε.Υ.Ο. (image) και να εγκαταστήσουν ένα πλήρως λειτουργικό αντίγραφό του σε χώρο εκτός υπηρεσίας προκειμένου να δημιουργούνται παράνομα αλλά έγκυρα πρακτικά εξέτασης.

Πρέπει να τονιστεί ότι λόγω της φύσης του συστήματος καμία μέθοδος κρυπτογράφησης ή ψηφιακής υπογραφής δεν μπορεί να κρατήσει τα δεδομένα ασφαλή.

4 Η ασφάλεια των Microsoft Access αρχείων είναι εύκολο να παρακαμφθεί.

Βέβαια, το Μ.Σ.Θ.Ε.Υ.Ο. ως πληροφοριακό σύστημα, δεν αποτελείται μόνο από υλικό και λογισμικό, αλλά και από ανθρώπινο δυναμικό (peopleware) και από διαδικασίες. Ο ευκολότερος, ίσως, τρόπος να παραβιαστεί η ακεραιότητα της εξεταστικής διαδικασίας είναι η μη τήρηση των διαδικασιών που διασφαλίζουν την ακεραιότητα αυτή. Για παράδειγμα αν δεν γίνεται σωστά ο έλεγχος ταυτοπροσωπίας του εξεταζόμενου ή αν δεν επιτηρείται σωστά η αίθουσα κατά την διάρκεια της εξέτασης, μπορούν να εκδοθούν αποτελέσματα τα οποία δεν κατοπτρίζουν τις γνώσεις του εξεταζόμενου στον οποίο αναφέρονται τα αντίστοιχα πρακτικά.

Κεφάλαιο 3

ΑΠΑΙΤΗΣΕΙΣ ΝΕΟΥ ΣΥΣΤΗΜΑΤΟΣ

Όπως αναλύθηκε στο προηγούμενο κεφάλαιο, παρόλο που το Μ.Σ.Θ.Ε.Υ.Ο. είναι ένα πετυχημένο σύστημα διεξαγωγής εξετάσεων, παρουσιάζει ορισμένες αδυναμίες οι οποίες γίνονται περισσότερο φανερές με το πέρασμα του χρόνου και την προσπάθεια του Υ.Μ.Ε. να το χρησιμοποιήσει σε διαδικασίες εξέτασης οι οποίες δεν είχαν προβλεφθεί στον αρχικό του σχεδιασμό. Οι αδυναμίες αυτές οδηγούν στην αναγκαιότητα σχεδιασμού και ανάπτυξης ενός νέου συστήματος διεξαγωγής εξετάσεων και αξιολογήσεων γνώσεων γενικής χρήσης το οποίο θα μπορούσε να αντικαταστήσει υπάρχοντα συστήματα, όπως το Μ.Σ.Θ.Ε.Υ.Ο., αλλά και να εφαρμοστεί στη θέση χειρόγραφων ή εν μέρει αυτοματοποιημένων διαδικασιών εξέτασης.

Το νέο σύστημα, το σύστημα “Atlas”, βασίζεται στις λειτουργικές απαιτήσεις του πετυχημένου Μ.Σ.Θ.Ε.Υ.Ο.· πληροί όμως και νέες προδιαγραφές, ενώ κάποιες από τις προδιαγραφές του παλαιού συστήματος μεταφέρονται τροποποιημένες. Με το τρόπο αυτό το σύστημα Atlas καλύπτει τις σύγχρονες ανάγκες και μπορεί να εφαρμοστεί με επιτυχία σε πολλά περιβάλλοντα.

Στις παραγράφους οι οποίες ακολουθούν, περιγράφονται οι νέες ή οι τροποποιημένες απαιτήσεις του Atlas και οι προδιαγραφές που θα πρέπει να πληροί το σύστημα ώστε οι απαιτήσεις αυτές να ικανοποιηθούν.

ΜΕΙΩΣΗ ΚΟΣΤΟΥΣ ΛΕΙΤΟΥΡΓΙΑΣ

Το κόστος δημιουργίας και συντήρησης μίας αίθουσας εξετάσεων του Μ.Σ.Θ.Ε.Υ.Ο., στο οποίο συμπεριλαμβάνεται ο εξοπλισμός (ηλεκτρονικός ή μη) και ο χώρος, είναι ιδιαίτερα υψηλό και σε περιόδους μεγάλης οικονομικής δυσχέρειας το κόστος αυτό είναι απαγορευτικό. Επίσης, τις περισσότερες φορές, οι εξετάσεις πραγματοποιούνται μία φορά την εβδομάδα ή και σπανιότερα, που σημαίνει ότι η αίθουσα υποχρησιμοποιείται και η απαιτούμενη επένδυση, ακόμα και όταν είναι απαραίτητη, συνιστά σπατάλη πόρων.

Για το λόγο αυτό το σύστημα Atlas θα πρέπει:

- Να μπορεί να χρησιμοποιηθεί για την διεξαγωγή, κατά το δυνατό, όλων των κατηγοριών εξέτασης ενός οργανισμού ή φορέα.
- Να μπορεί να καλύψει τις ανάγκες διαφόρων οργανισμών ώστε να μοιραστεί το κόστος εγκατάστασης του συστήματος. Εναλλακτικά, κάποιος οργανισμός μπορεί να δημιουργήσει ένα εξεταστικό κέντρο και να παρέχει υπηρεσίες διενέργειας εξετάσεων σε τρίτους.
- Οι σταθμοί εξέτασης θα πρέπει να είναι κατά το δυνατό φθηνοί και να πιάνουν το λιγότερο δυνατό χώρο, ώστε σε μία αίθουσα να μπορούν να εξετάζονται ταυτόχρονα περισσότεροι υποψήφιοι.
- Η αρχιτεκτονική του συστήματος θα πρέπει να είναι τέτοια ώστε ο εξοπλισμός να μην χρειάζεται να είναι μόνιμα εγκατεστημένος σε ένα χώρο. Αντίθετα, θα πρέπει να μπορεί να εγκατασταθεί εύκολα σε χώρους οι οποίοι πληρούν στοιχειώδεις προδιαγραφές αίθουσας εξετάσεων. Με το πέρας των εξετάσεων, ο εξοπλισμός θα πρέπει να είναι εύκολο να αποθηκευτεί με ασφάλεια σε μικρό χώρο.

Οι παραπάνω απαιτήσεις μπορούν να καλυφθούν αν οι σταθμοί εξέτασης είναι tablets. Με αυτό τον τρόπο:

- Το κόστος των σταθμών εξέτασης είναι χαμηλό. Ένα android tablet, καλής ποιότητας, είναι πολύ φθηνότερο από ένα PC με οθόνη αφής.
- Πολλοί εξεταζόμενοι μπορούν να εξετάζονται ταυτόχρονα. Σε ένα τυπικό αμφιθέατρο 100 θέσεων είναι εφικτό να εξεταστούν 50 άτομα σε μία εξέταση χωρίς να απειληθεί, από τη συμπεριφορά των εξεταζομένων, η ομαλή διεξαγωγή της διαδικασίας.
- Με την χρήση ασύρματων συσκευών δεν απαιτείται η ύπαρξη εγκατεστημένου ενσύρματου δικτύου. Ένας φορητός υπολογιστής ως εξυπηρετητής, ένας ασύρματος δρομολογητής, οι σταθμοί εξέτασης και ένας εκτυπωτής, αποτελούν ικανό βασικό εξοπλισμό για την διεξαγωγή εξετάσεων.

Ένας οργανισμός, για παράδειγμα ένα Ανώτατο Εκπαιδευτικό Ίδρυμα, μπορεί να αγοράσει το απαιτούμενο εξοπλισμό για να οργανώσει ένα εξεταστικό κέντρο. Οι εξετάσεις, όποιων μαθημάτων μπορούν να διεξαχθούν στο Atlas, θα πραγματοποιούνται σε κάποιο αμφιθέατρο το οποίο με το πέρας την εξεταστικής διαδικασίας θα επιστρέφει στην κανονική χρήση του. Επίσης, το ΑΕΙ θα μπορεί να νοικιάζει το σύστημα σε φορείς οι οποίοι θέλουν να πραγματοποιήσουν εξετάσεις και δεν διαθέτουν δικό τους σύστημα. Για παράδειγμα, το Υ.Μ.Ε. θα μπορούσε να πραγματοποιήσει τις εξετάσεις για το δίπλωμα των ραδιοερασιτεχνών, οι οποίες

πραγματοποιούνται δύο φορές το χρόνο, μίας περιφερειακής ενότητας (νομαρχίας), στους χώρους και με τον εξοπλισμό του οικείου πανεπιστημίου.

ΔΥΝΑΤΟΤΗΤΕΣ ΤΡΟΠΟΠΟΙΗΣΗΣ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ

Το σύστημα Atlas πρέπει να δίνει στους χρήστες του δυνατότητα τροποποίησης του λογισμικού του. Οι τροποποιήσεις αυτές αφορούν:

- Την ενημέρωση (update) του λογισμικού, δηλαδή την διόρθωση πιθανών προβλημάτων του κώδικα και τις απαιτούμενες αλλαγές στη βάση δεδομένων των ερωτήσεων και των απαντήσεων
- Την αναβάθμιση (upgrade) του συστήματος, δηλαδή την προσθήκη μονάδων (modules) ώστε το Atlas να μπορεί να υποστηρίζει νέες κατηγορίες εξέτασης και νέες διαδικασίες εξέτασης.

Οι διαδικασίες τροποποίησης πρέπει να είναι αυτοματοποιημένες και να μπορούν να εκτελεστούν από μη εξειδικευμένο προσωπικό. Προκειμένου τα παραπάνω να είναι εφικτά, το σύστημα πρέπει:

- Να έχει σύνδεση στο διαδίκτυο.
- Να είναι σχεδιασμένο και υλοποιημένο με τέτοιο τρόπο ώστε οι κατηγορίες εξέτασης να υλοποιούνται με ανεξάρτητες μονάδες (modules).
- Τόσο στον εξυπηρετητή όσο και στους σταθμούς εξέτασης, θα πρέπει να τρέχει ειδική υπηρεσία με την μορφή δαίμονα (daemon) η οποία θα εκτελεί την ενημέρωση ή την αναβάθμιση του λογισμικού στον κατάλληλο χρόνο.

Οι διαδικασίες αναβάθμισης του συστήματος πρέπει να υπόκεινται σε αυστηρά πρωτόκολλα ασφαλείας. Τα πρωτόκολλα αυτά θα αναλυθούν σε επόμενες παραγράφους.

ΣΥΝΔΕΣΙΜΟΤΗΤΑ ΜΕ ΑΛΛΑ ΣΥΣΤΗΜΑΤΑ

Το Atlas θα πρέπει να μπορεί να συνδέεται με άλλα συστήματα προκειμένου να μπορεί να ανταλλάσσει δεδομένα με αυτά. Η ανταλλαγή δεδομένων είναι απαραίτητη κυρίως σε εφαρμογές του συστήματος που αφορούν μεγάλους οργανισμούς και που τα δεδομένα χρησιμοποιούνται και από άλλα συστήματα. Παράδειγμα τέτοιας εφαρμογής του Atlas θα μπορούσε να είναι το σύστημα εξετάσεων του Υ.Μ.Ε.. Η συνδεσιμότητα με εξωτερικά συστήματα θα μπορούσε να αφορά:

- Την είσοδο των ομάδων εξέτασης από το σύστημα στο οποίο καταγράφεται ο προγραμματισμός της υπηρεσίας.
- Την αποστολή των αποτελεσμάτων στο portal του υπουργείου ώστε αυτά να είναι διαθέσιμα στους πολίτες.
- Την εξαγωγή στατιστικών στοιχείων και άλλης στρατηγικής πληροφορίας προς τους εξυπηρετητές του Υπουργείου.

Σε περίπτωση που το σύστημα εφαρμοστεί από ένα μικρό οργανισμό ή εξυπηρετεί πολλούς διαφορετικού πελάτες, τα παραπάνω δεν αποτελούν αναγκαιότητα· τουλάχιστον όχι για όλες τις κατηγορίες εξέτασης. Για το λόγο αυτό, η συνδεσιμότητα πρέπει να αποτελεί ξεχωριστό module του συστήματος.

ΑΣΦΑΛΕΙΑ

Όπως προαναφέρθηκε, η ασφάλεια, σε όλες τις μορφές της, είναι ένας κρίσιμος παράγοντας για ένα σύστημα εξετάσεων, ειδικά όταν το σύστημα αυτό είναι οικονομικά και κοινωνικά ευαίσθητο. Προκειμένου το σύστημα Atlas να είναι, κατά το δυνατό, ασφαλές πρέπει να πληροί μία σειρά από προδιαγραφές που σχετίζονται με τις διάφορες πτυχές της έννοιας “ασφάλεια”.

Φυσική ασφάλεια

Για να διατηρηθεί φυσικά ασφαλές το Atlas, δεν χρειάζεται ιδιαίτερος κόπος. Αρκεί να προστατεύεται ο εξοπλισμός από κλοπές ή φυσικές καταστροφές. Ένας εφεδρικός εξυπηρετητής, μερικά έξτρα tablets ως εφεδρικοί σταθμοί εξέτασης, και μία εξωτερική συσκευή αποθήκευσης αρχείων είναι αρκετά για να προστατέψουν τη διαδικασία εξέτασης από τεχνικά προβλήματα.

Σε περίπτωση που η εξετάσεις πραγματοποιούνται σε αίθουσα αποκλειστικής, από το σύστημα, χρήσης με προσωπικούς υπολογιστές, είναι απαραίτητη η ύπαρξη συστήματος αδιάλειπτης παροχής ενέργειας το οποίο θα καλύπτει το σύνολο του εξοπλισμού, και σύστημα κλιματισμού το οποίο θα κρατά τη θερμοκρασία της αίθουσας μέσα στα όρια λειτουργίας του εξοπλισμού[5].

Ασφάλεια επικοινωνιών

Εφόσον ο οργανισμός ο οποίος χρησιμοποιεί το Atlas για την εξυπηρέτηση των αναγκών του σε εξετάσεις, αποφασίσει ότι το θέλει συνδεδεμένο με κάποια από τα άλλα πληροφοριακά συστήματά του, σύμφωνα με τα αναγραφόμενα στην παράγραφο “Συνδεσιμότητα με άλλα συστήματα”, θα πρέπει να ενεργοποιηθούν

5 Ο μέσος υγιής άνθρωπος μπορεί να… λειτουργήσει σε ευρύτερα όρια θερμοκρασίας από ένα προσωπικό υπολογιστή.

κατάλληλα πρωτόκολλα ασφαλείας τα οποία θα διασφαλίζουν το σύστημα από δικτυακές επιθέσεις. Ακολουθεί σύντομη περιγραφή των τεχνικών δικτυακής ασφάλειας που θα πρέπει να χρησιμοποιηθούν.

- *Αυθεντικοποίηση χρηστών*. Το σύστημα επιτρέπει πρόσβαση στις δικτυακές του υπηρεσίες μόνο σε εγκεκριμένους χρήστες οι οποίοι μπορούν να αποδείξουν την ταυτότητά τους μέσω κάποιου συνθηματικού. Ο όρος "χρήστες" δεν αναφέρεται μόνο σε φυσικά πρόσωπα αλλά και σε άλλα προγράμματα τα οποία επικοινωνούν αυτόματα με το Atlas. Για κρίσιμες εργασίες, όπως για παράδειγμα η αναβάθμιση του συστήματος, σύνδεση στο σύστημα θα εγκρίνεται μόνο από συγκεκριμένες IP διευθύνσεις.

- *Κρυπτογράφηση δεδομένων*. Οι πληροφορίες που μεταφέρονται από και προς τον εξυπηρετητή του Atlas θα πρέπει να είναι κρυπτογραφημένες. Εφόσον το περιβάλλον λειτουργίας του συστήματος το επιτρέπει, οι επικοινωνίες πρέπει να πραγματοποιούνται μέσα σε εικονικό ιδιωτικό δίκτυο (virtual private network – vpn). Κρυπτογράφηση, επίσης, απαιτούν οι επικοινωνίες του εξυπηρετητή με τους σταθμούς εξέτασης εφόσον αυτές πραγματοποιούνται μέσω ασύρματου δικτύου.

- *Ασφαλές λειτουργικό σύστημα*. Είναι επιθυμητό ο εξυπηρετητής να τρέχει σε περιβάλλον το οποίο δεν θα προσθέτει τα δικά του προβλήματα ασφαλείας στο σύστημα. Συστήνεται το λειτουργικό σύστημα Linux. Αν αυτό δεν είναι δυνατό, το λειτουργικό σύστημα στο οποίο θα τρέχει ο εξυπηρετητής θα πρέπει να είναι όσο το δυνατό καλύτερα ρυθμισμένο σε θέματα δικτυακής ασφάλειας. Θα πρέπει, επίσης, να υποστηρίζεται από εξειδικευμένο λογισμικό ασφαλείας.

- *Προστασία του ασύρματου δικτύου*. Κατά τη διαδικασία της εξέτασης, στον ασύρματο δρομολογητή θα πρέπει να συνδέονται μόνο οι εγκεκριμένοι σταθμοί εξέτασης. Ο ασύρματος δρομολογητής, θα πρέπει να υποστηρίζει πρωτόκολλα MAC Filtering δηλαδή να επιτρέπει την σύνδεση μόνο σε συσκευές των οποίων η MAC Address έχει καταχωρηθεί σε αυτόν.

Προστασία από εσωτερικές επιθέσεις.

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, ο πραγματικός κίνδυνος για το πληροφοριακό σύστημα και κατά συνέπεια για την αξιοπιστία της διαδικασίας εξέτασης (και του οργανισμού που την πραγματοποιεί), προέρχεται από το προσωπικό το οποίο είναι υπεύθυνο για τη λειτουργία του. Το προσωπικό αυτό έχει φυσική πρόσβαση στο σύστημα. Επειδή είναι αδύνατο να προστατευτεί το λογισμικό και τα δεδομένα ενός συστήματος από κάποιον ο οποίος έχει φυσική πρόσβαση σε αυτό[6], το Atlas θα πρέπει να ακολουθεί τις ακόλουθες πολιτικές /

6 Κάθε σύστημα στο οποίο ο επιτιθέμενος έχει φυσική πρόσβαση, μπορεί να παραβιαστεί με την μέθοδο του reverse engineering. Για παράδειγμα αν η βάση δεδομένων είναι κλειδωμένη και κρυπτογραφημένη, το

προδιαγραφές ασφαλείας ώστε να δυσκολεύει τις προσπάθειες παραβίασής του σε τέτοιο βαθμό ώστε να τις κάνει ασύμφορες.

- Παρόλο που η εφαρμογή δεν απαιτεί την αυθεντικοποίηση των χρηστών που τη χρησιμοποιούν, το περιβάλλον στο οποίο εκτελείται (λειτουργικό σύστημα) πρέπει να είναι προστατευμένο από μη εξουσιοδοτημένες χρήσεις. Για κάθε εγκατάσταση θα πρέπει να υπάρχει μικρός αριθμός ατόμων ο οποίος είναι υπεύθυνος για την διαχείριση του συστήματος. Σε κρίσιμα συστήματα τα άτομα αυτά θα πρέπει να μην ανήκουν στο προσωπικό το οποίο διενεργεί εξετάσεις, αλλά να είναι μέλη αρχής η οποία θα φέρει ευθύνη για την ασφάλεια του συστήματος. Η αρχή αυτή θα είναι υπεύθυνη για την αρχική εγκατάσταση του συστήματος και τις όποιες αλλαγές απαιτηθούν σε αυτό κατά τη διάρκεια της λειτουργίας του.

- Το σύστημα αρχείων το οποίο περιέχει τον κώδικα και τα δεδομένα του συστήματος θα πρέπει να είναι κρυπτογραφημένο ώστε να μην μπορεί να διαβαστεί από άλλο υπολογιστή αν αφαιρεθεί από τον εξυπηρετητή της υπηρεσίας ή δημιουργηθεί κλώνος του. Ομοίως, η βάση δεδομένων του συστήματος πρέπει να είναι κρυπτογραφημένη. Τα κλειδιά της κρυπτογράφησης θα πρέπει να είναι γνωστά μόνο σε άτομα τα οποία δεν έχουν αυτόνομη φυσική πρόσβαση στο hardware του συστήματος.

- Το λογισμικό θα πρέπει να αναγνωρίζει το υλικό στο οποίο τρέχει (MAC address, Bios serial number, κ.λ.π.). Σε περίπτωση που το υλικό δεν είναι το αναμενόμενο, το λογισμικό πρέπει να κλειδώνει το σύστημα και να στέλνει, μέσω του διαδικτύου, όσες περισσότερες πληροφορίες μπορεί για το περιβάλλον στο οποίο τρέχει σε κάποια υπεύθυνη αρχή. Σε κρίσιμα συστήματα εξετάσεων, το Atlas δεν θα ξεκινά τη λειτουργία του αν πρώτα δεν διασφαλίσει ότι υπάρχει ενεργή δικτυακή σύνδεση με τις αρχές αυτές.

- Το ρολόι του εξυπηρετητή θα ρυθμίζεται αυτόματα από ασφαλή NTP Server. Σε περίπτωση που διαπιστωθεί απόπειρα χειροκίνητης αλλαγής της ημερομηνίας και της ώρας του συστήματος, θα πρέπει να αναφέρεται. Με τον τρόπο αυτό διασφαλίζεται, μεταξύ άλλων, ότι δεν θα πραγματοποιούνται εξετάσεις σε... περίεργες μέρες ή ώρες.

- Το λογισμικό θα πρέπει να εξετάζει το εαυτό του για πιθανές τροποποιήσεις. Σε περίπτωση που διαπιστωθούν τροποποιήσεις στο λογισμικό ή στα δεδομένα, στο σύστημα θα κλειδώνει και θα αναφέρει το πρόβλημα δικτυακά σε υπεύθυνη αρχή.

- Η κρυπτογράφηση των δεδομένων θα πραγματοποιείται με κλειδιά που αλλάζουν συχνά, αλλά και με αλγόριθμους οι οποίοι δεν θα ανακοινώνονται

πρόγραμμα που τη χρησιμοποιεί συνδέεται σε αυτή με κάποιο συνδυασμό ονόματος χρήστη και συνθηματικού. Ο συνδυασμός αυτός περιέχεται, σε κάποια μορφή, στο πρόγραμμα. Η ανάλυση του προγράμματος θα αποκαλύψει τον συνδυασμό.

και θα αλλάζουν, στο μέτρο του δυνατού, σε τακτά χρονικά διαστήματα. Τα κλειδιά θα πρέπει να δημιουργούνται από τον κώδικα με πολύπλοκο τρόπο.

- Ο κώδικας ο οποίος είναι υπεύθυνος για την ασφάλεια του συστήματος (δημιουργία κλειδιών, κρυπτογράφηση αρχείων, κλπ) πρέπει να είναι γραμμένος σε γλώσσα χαμηλού επιπέδου και με τέτοιο τρόπο ώστε να δυσκολεύεται η ανάγνωσή του με τεχνικές reverse engineering. Σε καμία περίπτωση δεν θα πρέπει να χρησιμοποιηθούν για το σκοπό αυτό γλώσσες για τις οποίες είναι εύκολο να επαναδημιουργηθεί ο πηγαίος κώδικας από τον αντικειμενικό (π.χ. Java).
- Το σύστημα θα πρέπει να ελέγχει για περίεργα φαινόμενα κατά τη διάρκεια της διαδικασίας εξέτασης και να τα αναφέρει δικτυακά. Παραδείγματα τέτοιων φαινομένων είναι: η σύνδεση στο δίκτυο άγνωστου τερματικού, η τροποποίηση του φύλλου εξέτασης σε σημεία διαφορετικά από τις θέσεις καταγραφής των απαντήσεων του εξεταζόμενου, η καθυστέρηση ολοκλήρωσης της διαδικασίας εξέτασης, κ.α.

Οι προδιαγραφές που περιγράφηκαν στο κεφάλαιο αυτό συνδυαζόμενες με τις αρχικές προδιαγραφές του Μ.Σ.Θ.Ε.Υ.Ο., δημιουργούν ένα σύστημα το οποίο είναι εύκολο στη χρήση του για όλα τα εμπλεκόμενα μέρη, οικονομικό, και ασφαλές. Δημιουργούν ένα σύστημα το οποίο μπορεί να εξοικονομήσει χρόνο και χρήμα ενώ ταυτόχρονα διασφαλίζει τη διαφάνεια της διαδικασίας εξέτασης και προσδίδει κύρος στον φορέα που το χρησιμοποιεί.

Κεφάλαιο 4

Τεχνολογίες Υλοποίησης

Ένα σύστημα σαν το Atlas, πέρα από το υλικό στο οποίο τρέχει, συνδυάζει αρκετές τεχνολογίες και προϊόντα λογισμικού τόσο για την ανάπτυξή του όσο και για την λειτουργία του. Είναι προφανές ότι το σύστημα που προδιαγράφηκε στα προηγούμενα κεφάλαια μπορεί να υλοποιηθεί, αν όχι με πολλούς διαφορετικούς τρόπους, με πολλές διαφορετικές τεχνολογίες. Προκειμένου, όμως, το σύστημα να είναι εύκολα υλοποιήσιμο και προσαρμόσιμο σύμφωνα με τις ανάγκες διάφορων οργανισμών, οι τεχνολογίες και τα προϊόντα που θα χρησιμοποιηθούν σε αυτό, πρέπει να πληρούν ορισμένους βασικούς όρους:

- Το λογισμικό να είναι οικονομικό και να τρέχει σε οικονομικό εξοπλισμό.
- Οι τεχνολογίες ανάπτυξης να είναι ευρέως γνωστές ώστε να είναι εύκολο να συντηρηθεί το σύστημα μετά την ανάπτυξή του.
- Το βασικό σύστημα (δημιουργία ερωτηματολογίων – προβολή στην οθόνη του χρήστη – αξιολόγηση απαντήσεων) θα πρέπει να μην περιέχει από πλευράς λογισμικού τίποτα “εξωτικό”, ώστε να είναι εύκολο να τροποποιηθεί προκειμένου να παραχθεί λογισμικό προσομοίωσης της διαδικασίας (εκπαιδευτικό λογισμικό) το οποίο θα εγκαθίσταται εύκολα στον εξοπλισμό που διαθέτει ο μέσος χρήστης σπίτι του, χωρίς ή με ελάχιστο κόστος προαπαιτούμενου λογισμικού και θα προσομοιώνει με ακρίβεια την διαδικασία εξέτασης στο κομμάτι που αφορά τον εξεταζόμενο.

Από τα παραπάνω προκύπτει εύκολα ότι το μεγαλύτερο μέρος του συστήματος θα πρέπει να βασίζεται σε ελεύθερο λογισμικό αναγνωρισμένης ποιότητας και αξίας. Στις παραγράφους που ακολουθούν γίνεται σύντομη περιγραφή των προτεινόμενων τεχνολογιών και προϊόντων λογισμικού στα οποία βασιστεί η υλοποίηση του συστήματος Atlas.

ANDROID ΦΟΡΗΤΕΣ ΣΥΣΚΕΥΕΣ

Στο προηγούμενο κεφάλαιο αναφέρθηκαν οι λόγοι για τους οποίους είναι βολικό, οι σταθμοί εξέτασης να μετατραπούν από PCs σε Tablets. Υπάρχουν δύο κύριες τεχνολογίες τέτοιων συσκευών: Η σειρά Ipad της εταιρίας Apple και τα tablets διάφορων κατασκευαστών οι οποίες βασίζονται στο λειτουργικό σύστημα Android της Google. Από τις δύο αυτές τεχνολογίες επιλέγεται η δεύτερη διότι:

- Είναι ευκολότερο τεχνικά και φθηνότερο οικονομικά να αναπτυχθεί λογισμικό για Android συσκευές. Το δεύτερο οφείλεται και σε περιορισμούς ή απαιτήσεις που θέτει η Apple στους Developers.
- Υπάρχει μεγάλη ποικιλία Android συσκευών τόσο σε τεχνικά χαρακτηριστικά, όσο και σε τιμές. Ανάλογα με το είδος των εξετάσεων που διεξάγονται σε ένα φορέα (π.χ. αν χρειάζεται να απεικονιστεί video ή όχι) , μπορούν να επιλεγούν συσκευές με αντίστοιχα τεχνικά χαρακτηριστικά και μειωμένο κόστος. Κατά τεκμήριο, οι Android συσκευές είναι οικονομικότερες από τις αντίστοιχων δυνατοτήτων Apple συσκευές.

Πολλοί θεωρούν την ποικιλία τεχνικών προδιαγραφών των συσκευών στις οποίες τρέχει το Android, ένα από τα μειονεκτήματά του, διότι ο κώδικας του πρέπει να προσαρμόζεται σε αυτές. Το επιχείρημα είναι αμφισβητήσιμο γενικά. Ειδικά σε ό,τι αφορά το σύστημα Atlas, το επιχείρημα μπορεί να απορριφθεί εύκολα. Για το συγκεκριμένο λογισμικό δεν παίζει ρόλο αν το tablet διαθέτει π.χ. πυξίδα ή όχι. Η μόνη διαφορά η οποία θα μπορούσε να δημιουργήσει προβλήματα είναι η ανάλυση ή κάποιο άλλο από τα βασικά τεχνικά χαρακτηριστικά της οθόνης. Το Android είναι κτισμένο πάνω σε αυτή την ποικιλομορφία και προσαρμόζεται πάρα πολύ καλά στις διάφορες οθόνες χωρίς ιδιαίτερο κόπο από την πλευρά του προγραμματιστή.

ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ MySQL.

Τόσο το Μ.Σ.Θ.Ε.Υ.Ο., όσο και το Atlas, δεν είναι συστήματα τα οποία θα οποία θα έφερναν ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) στα όρια του. Στη βάση δεδομένων του συστήματος, σε συντομία, καταγράφονται:

- Οι Ερωτήσεις και οι απαντήσεις όλων των κατηγοριών εξέτασης, οι οποίες, εκτός από τα λεκτικά, περιέχουν και τις υπόλοιπες σχετικές πληροφορίες (σωστή απάντηση, εσωτερική υποκατηγορία, συσχετιζόμενη εικόνα, κ.λ.π.).
- Τα στοιχεία των ομάδων εξέτασης (ημερομηνία και ώρα διεξαγωγής, υπεύθυνοι εξεταστές, κ.λ.π.) και των εξεταζόμενων σε αυτές (στοιχεία ταυτότητας, κατηγορία εξέτασης, αποτέλεσμα, κ.λ.π.).

- Πληροφορίες οι οποίες αφορούν τους σταθμούς εξέτασης και το δίκτυο συνολικά.
- Πληροφορίες επικοινωνίας με τον έξω κόσμο (portal υπηρεσίας, εξυπηρετητές Υ.Μ.Ε).
- Ημερολόγιο ενεργειών και συμβάντων (log files).
- Πληροφορίες που αφορούν την ασφάλεια του συστήματος.

Λόγω της αρχιτεκτονικής του συστήματος, όπως αυτή αναφέρθηκε στο 2ο κεφάλαιο για το Μ.Σ.Θ.Ε.Υ.Ο. και θα αναπτυχθεί στο 5ο κεφάλαιο για το Atlas, δεν υπάρχει συνεχής πρόσβαση στη βάση δεδομένων κατά την κρίσιμη ώρα λειτουργίας του συστήματος: την ώρα διεξαγωγής της εξέτασης. Αντίθετα, πρόσβαση στην βάση απαιτείται κατά την πρώτη φάση της διαδικασίας όπου δημιουργούνται τα ερωτηματολόγια και κατά την τρίτη φάση, όπου γίνεται η αξιολόγηση των απαντήσεων και η καταγραφή των αποτελεσμάτων. Καμία από τις δύο αυτές φάσεις δεν είναι χρονικά κρίσιμη, με την έννοια ότι μία μικρή καθυστέρηση στην απόκριση θα δημιουργούσε πρόβλημα στην διαδικασία. Παράλληλα, ο όγκος της διακινούμενης πληροφορίας και το πλήθος των ταυτόχρονων συνδέσεων στη βάση δεδομένων είναι από μικρά ως αμελητέα για ένα σύγχρονο RDBMS. Η πρόσβαση στη βάση δεδομένων εκτός διαδικασίας εξέτασης (προγραμματισμός ομάδων εξέτασης, εξαγωγή στατιστικών), επίσης, απέχει πολύ από το να φέρει ένα σύγχρονο RDBMS στα όριά του.

Το σύστημα διαχείρισης βάσεων δεδομένων “MySQL”[7], προτείνεται ως το RDBMS που θα εξυπηρετεί το σύστημα Atlas, για τους ακόλουθους λόγους:

- Πρόκειται για ελεύθερο λογισμικό για την εγκατάσταση και χρήση του οποίου δεν απαιτείται η καταβολή χρηματικού αντιτίμου. Για παράδειγμα, το MySQL, θα μπορούσε να εγκατασταθεί σε όλες τις υπηρεσίες μεταφορών και επικοινωνιών της χώρας χωρίς να ξοδευτεί ούτε ένα ευρώ, σε αντίθεση με την Microsoft Access που είναι εγκατεστημένη, ως μέρος του Office, σε κάθε εξυπηρετητή του Μ.Σ.Θ.Ε.Υ.Ο. και κοστίζει στον Έλληνα φορολογούμενο ένα διόλου ευκαταφρόνητο ποσό.
- Είναι προϊόν αναγνωρισμένης ποιότητας και αξιοπιστίας[8] το οποίο με βεβαιότητα καλύπτει τις ανάγκες του συστήματος. Σήμερα ανήκει, συντηρείται και επεκτείνεται από την εταιρεία Oracle η οποία είναι κορυφαία εταιρεία στα προϊόντα βάσεων δεδομένων.
- Είναι προϊόν το οποίο χρησιμοποιείται από πλήθος επαγγελματιών, και όχι μόνο, πληροφορικής με αποτέλεσμα να υπάρχει πληθώρα ανθρώπινου δυναμικού με μεγάλη τεχνογνωσία σε αυτό.

7 Αναφέρεται στην ελεύθερη έκδοση του συστήματος (community server)

8 Δεν είναι τυχαίο ότι το συγκεκριμένο σύστημα χρησιμοποιείται περισσότερο από οποιοδήποτε άλλο στους διακομιστές του διαδικτύου και ότι το προϊόν έχει πελάτες του μεγέθους της NASA.

- Είναι συμβατό με τα πιο γνωστά λειτουργικά συστήματα. Αυτό σημαίνει ότι, με την χρήση του, μπορεί να αναπτυχθεί ένα cross platform σύστημα χωρίς να απαιτηθούν ιδιαίτερες προσαρμογές (και τα αντίστοιχα κόστη) ανάλογα με το περιβάλλον το οποίο προτιμά ο κάθε πελάτης του.
- Υπάρχει API[9] για όλες τις βασικές γλώσσες προγραμματισμού (C, C++, Java, PHP) το οποίο συντηρείται από τον επίσημο κατασκευαστή ή από μεγάλους οργανισμούς του χώρου και αποτελεί ελεύθερο λογισμικό.
- Είναι συμβατό με τα παγκόσμια πρότυπα για τις σχεσιακές βάσεις δεδομένων. Σε περίπτωση που για κάποιο λόγο το σύστημα πρέπει να αντικατασταθεί από άλλο RDBMS, θα χρειαστεί να γίνουν τροποποιήσεις μόνο στα τμήματα του κώδικα τα οποία σχετίζονται με το API του MySQL.
- Το σύστημα διαθέτει πληθώρα ελεύθερων βοηθητικών εργαλείων χρήσης και διαχείρισής του (π.χ. phpMyAdmin), τα οποία βοηθάνε τόσο στην χρήση του, κατά τη διαδικασία ανάπτυξης λογισμικού το οποίο θα χρησιμοποιεί το συγκεκριμένο RDBMS, όσο και την διαχείρισή του κατά την λειτουργία του τελικού συστήματος. Τα εργαλεία αυτά μπορεί να φανούν ως και αναγκαία σε περιπτώσεις που την διαχείριση του RDBMS αναλάβει προσωπικό το οποίο δεν έχει μεγάλη σχετική πείρα.

Τα παραπάνω πλεονεκτήματα οδηγούν στην επιλογή του MySQL ως κατάλληλου RDBMS για το Atlas, χωρίς να αποκλείουν άλλα αντίστοιχα συστήματα όπως το Postgresql.

Η χρήση του MySQL ή κάποιου άλλου αντίστοιχου RDBMS κάνει το Atlas να μειονεκτεί σε ένα σημείο σε σχέση με τον πρόγονό του. Απαιτεί την ξεχωριστή εγκατάσταση και διαχείριση ενός σε RDBMS, ενώ το Μ.Σ.Θ.Ε.Υ.Ο. διατηρεί τα δεδομένα του σε μία ενσωματωμένη (self-contained, embeddable) βάση δεδομένων όπως η Access και η sqlite.

ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVA.

Στη σημερινή εποχή σπάνια ένα πολύπλοκο σύστημα λογισμικού είναι γραμμένο σε μία και μόνο γλώσσα προγραμματισμού ή αποτελείται από ένα stand alone πρόγραμμα. Συνήθως εξαρτάται από διάφορα APIs, βιβλιοθήκες, έτοιμα modules κ.λ.π. τα οποία αναπτύσσονται σε ποικίλες γλώσσες και εργαλεία. Το Atlas, στο μεγαλύτερο μέρος του (εξυπηρετητής εξετάσεων και λογισμικό σταθμών εξέτασης), είναι ένα σύστημα το οποίο βασίζεται σε αυτόνομες (standalone) εφαρμογές[10]. Για την ανάπτυξή του επιλέγεται η γλώσσα προγραμματισμού Java. Η επιλογή αυτή

9 API: Application Programming Interface - Διεπαφή Προγραμματισμού Εφαρμογών

10 Σε αντίθεση με τις Web εφαρμογές οι οποίες τρέχουν σε browser. Ο όρος “standalone εφαρμογή” δεν αποκλείει ότι η εφαρμογή θα είναι δικτυοκεντρική.

βασίζεται στην συνισταμένη των χαρακτηριστικών και των πλεονεκτημάτων που παρουσιάζονται στις επόμενες παραγράφους· δεν την καθιστούν όμως μοναδική επιλογή. Απλά, της δίνουν πλεονέκτημα σε σχέση με άλλες γλώσσες, όπως η C και η C++.

Η Java είναι γλώσσα πολλαπλής πλατφόρμας (cross platform). Ο κώδικας ο οποίος παράγεται μπορεί να τρέξει κανονικά σε διάφορα λειτουργικά συστήματα, εφόσον είναι εγκατεστημένο το απαιτούμενο λογισμικό (runtime environment), χωρίς να χρειάζεται προσαρμογή και νέα μεταγλώττιση. Οι απαιτούμενες βιβλιοθήκες ενσωματώνονται στο εκτελέσιμο προϊόν (.jar αρχείο). Ο αντικειμενικός κώδικας που παράγεται (bytecode) έχει πολύ μικρότερες ανάγκες εγκατάστασης από τον εκτελέσιμο κώδικα ο οποίος παράγεται από τις κλασικές γλώσσες προγραμματισμού.

Οι εφαρμογές οι οποίες προορίζονται να τρέξουν σε Android συσκευές αναπτύσσονται υποχρεωτικά στη γλώσσα προγραμματισμού Java. Με εξαίρεση την διαχείριση της οθόνης, ο κώδικας ο οποίος θα γραφεί για τους Android σταθμούς εξέτασης, μπορεί να χρησιμοποιηθεί και σε PCs. Θα χρειαστεί μόνο να προσαρμοστεί ο κώδικας για τον χειρισμό της οθόνης. Ο προσαρμοσμένος κώδικας, φυσικά, θα μπορεί να τρέξει – ως έχει – και σε Linux και σε Windows σταθμούς εξέτασης.

Ως αντικειμενοστραφής (object oriented) και αρθρωτή (modular) γλώσσα προγραμματισμού η Java δίνει στον προγραμματιστή του Atlas, μεταξύ άλλων, τη δυνατότητα να κτίσει το σύστημα με τέτοιο τρόπο ώστε να είναι εύκολη η ενημέρωση και η αναβάθμισή του. Με απλές, δηλαδή, προγραμματιστικές διαδικασίες το πρόγραμμα μπορεί να ενημερώνει, από το διαδίκτυο, τον εαυτό του. Μπορεί επίσης, με απλή προσθήκη ".jar" αρχείων, να υποστηρίζει νέες κατηγορίες ή και μορφές εξέτασης, να επεκτείνει τις λειτουργίες του, κ.λ.π.

Η Java επιτρέπει γρηγορότερη (και άρα οικονομικότερη) ανάπτυξη λογισμικού σε σχέση με άλλες γλώσσες μεσαίου και υψηλού επιπέδου. Αυτό συμβαίνει λόγω της ενσωμάτωσης στις βιβλιοθήκες της λειτουργιών οι οποίες κανονικά σχετίζονται με το API του λειτουργικού συστήματος (νήματα, δικτύωση, διαχείριση πολυμέσων, κ.λ.π), αλλά και των έλεγχο που ασκεί πάνω στη μνήμη (δείκτες, δέσμευση μνήμης, κ.α.). Το τίμημα σε ταχύτητα και πόρους (μνήμη) είναι αμελητέο, τουλάχιστον σε ό,τι αφορά το Atlas.

Οι κλάσεις που θα δημιουργηθούν για τον εξυπηρετητή και τους σταθμούς εξέτασης είναι εύκολο να επαναχρησιμοποιηθούν για την δημιουργία εκπαιδευτικού λογισμικού το οποίο θα προσομοιώνει με ακρίβεια την διαδικασία εξέτασης. Το λογισμικό αυτό θα μπορεί να διανεμηθεί στους υποψήφιους οι οποίοι θα το εγκαταστήσουν με την βοήθεια απλών οδηγιών χωρίς να χρειάζεται να έχουν εξειδικευμένες γνώσεις. Και φυσικά θα μπορεί εύκολα να απευθύνεται σε χρήστες όλων των βασικών λειτουργικών συστημάτων. Επίσης, οι Java κλάσεις που αφορούν

διαδικασίες όπως η δημιουργία ερωτηματολογίου, η αξιολόγηση των απαντήσεων, κ.λ.π. μπορούν να επαναχρησιμοποιηθούν ως τμήμα εκπαιδευτικής ή άλλου τύπου Web εφαρμογής.

Η χρήση της γλώσσας Java παρουσιάζει και ένα μειονέκτημα. Ο αντικειμενικός της κώδικας (bytecode) μπορεί εύκολα να μετατραπεί σε πηγαίο. Οποιοσδήποτε έχει φυσική πρόσβαση στον εξυπηρετητή ή στους σταθμούς εξέτασης μπορεί να "κλέψει" την εφαρμογή, να τη διαβάσει και να βρει πως να παραβιάσει τα πρωτόκολλα ασφαλείας του συστήματος. Όπως έχει ήδη αναφερθεί ο κίνδυνος αυτός υπάρχει έτσι και αλλιώς, με την χρήση της Java όμως τα πράγματα γίνονται εύκολα για τον επιτιθέμενο στην ασφάλεια του συστήματος.

Μία λύση στο πρόβλημα είναι η χρήση του Java Native Interface (JNI). Το JNI είναι ένα framework το οποίο επιτρέπει σε Java προγράμματα να καλούν προγράμματα τα οποία είναι γραμμένα σε άλλες γλώσσες, καθώς και να καλούνται από αυτά. Με τον τρόπο αυτό, οτιδήποτε έχει σχέση με κρυπτογράφηση ή την ασφάλεια του συστήματος γενικά, μπορεί να υλοποιηθεί σε γλώσσα C και να ενσωματωθεί στο σύστημα[11].

EXTENSIBLE MARKUP LANGUAGE

Η Extensible Markup Language (XML) είναι μία γλώσσα σήμανσης η οποία χρησιμοποιείται για την κωδικοποίηση πληροφοριών σε μορφή κειμένου. Σε γενικές γραμμές, ένα XML κείμενο, αποτελείται από την πληροφορία που καταγράφεται σε αυτό και τη σήμανσή της (ετικέτες, χαρακτηριστικά, κ.λ.π.).

Η χρήση της XML είναι υποχρεωτική στην ανάπτυξη Android εφαρμογών. Μέσω αυτής περιγράφονται η διεπαφές με το χρήστη, καταγράφονται τα δομικά στοιχεία της εφαρμογής και τα χαρακτηριστικά τους (manifest), εισάγονται πόροι (εικόνες, κείμενα), κ.α.

Πέρα από την χρήση της στην ανάπτυξη της εφαρμογής στους Android σταθμούς εξέτασης, η XML χρησιμοποιείται για την περιγραφή των δεδομένων που ανταλλάσσονται μεταξύ του Atlas και του "έξω κόσμου". Τα δεδομένα που αποστέλλονται στο Atlas από άλλο πληροφοριακό σύστημα του οργανισμού και αφορούν μία ομάδα εξέτασης, είναι κωδικοποιημένα σε XML μορφή. Τα αποτελέσματα των εξετάσεων ή τα στατιστικά στοιχεία που ζητούνται από το σύστημα και αποστέλλονται σε άλλα πληροφοριακά συστήματα είναι, επίσης, XML αρχεία.

Παρόλο που με βάση τις γενικά ακολουθούμενες πρακτικές και τα ερωτηματολόγια των εξετάσεων θα έπρεπε να είναι XML αρχεία, η συγκεκριμένη τεχνολογία δεν επιλέχτηκε για τον σκοπό αυτό. Τα ερωτηματολόγια, ως αντικείμενα κλάσης,

11 Για κάθε λειτουργικό σύστημα πρέπει να δημιουργηθεί ξεχωριστή έκδοση του υποσυστήματος ασφαλείας.

μετατρέπονται σε αρχεία μέσω serialization[12] και αποθηκεύονται ή αποστέλλονται σε κάποιο παραλήπτη. Με την αντίστροφη διαδικασία το αρχείο μετατρέπεται σε δεδομένα τα οποία, ως αντικείμενο, φυλάσσονται στην RAM του υπολογιστή. Για να μπορεί να δουλέψει η μέθοδος αυτή, όπου χρησιμοποιούνται αυτά τα αρχεία, θα πρέπει η κλάση που παριστά το ερωτηματολόγιο να περιγράφεται με τον ίδιο ακριβώς τρόπο. Αυτό είναι εύκολο, ειδικά με την χρήση της Java, για δύο ή περισσότερα προγράμματα τα οποία αναπτύσσονται στα πλαίσια ενός ενιαίου συστήματος, αλλά πρακτικά αδύνατο να συμβεί σε ετερογενή συστήματα που αναπτύσσονται ξεχωριστά· σε αυτές τις περιπτώσεις χρησιμοποιείται η XML.

Η χρήση του serialization για την αποθήκευση και τη μεταφορά ερωτηματολογίων γίνεται για τους ακόλουθους λόγους οικονομίας:

- Η αποθήκευση των αρχείων αυτών απαιτεί μικρότερο αποθηκευτικό χώρο στον εξυπηρετητή και η αποστολή τους μέσω δικτύου γίνεται γρηγορότερα.
- Δεν απαιτείται, όπως στα XML αρχεία, η ειδική επεξεργασία της συντακτικής ανάλυσης (parsing), προκειμένου να αντληθεί η πραγματική πληροφορία. Η διαδικασία parsing ενός XML αρχείου σε μία Android συσκευή είναι ιδιαίτερα δαπανηρή σε χρόνο επεξεργασίας και ενέργεια[13].

XML αρχεία χρησιμοποιούνται και σε άλλα υποσυστήματα του Atlas, όπως για παράδειγμα στο υποσύστημα αυτόματης ενημέρωσης και αναβάθμισης του λογισμικού του.

QT FRAMEWORK

Ένα από τα προβλήματα της γλώσσας προγραμματισμού Java είναι ότι, προκειμένου να τρέχει σε πλήθος διαφορετικών λειτουργικών συστημάτων με τον ίδιο αντικειμενικό κώδικα, τα περιβάλλοντα γραφικής διεπαφής με το χρήστη που υποστηρίζει (Awt, Swing) παρουσιάζουν μερικούς περιορισμούς / προβλήματα:

- Υποστηρίζουν μόνο τις λειτουργίες που είναι κοινές σε όλα τα λειτουργικά συστήματα και δεν κάνουν χρήση των ειδικών χαρακτηριστικών της μητρικής διεπαφής χρήστη (native user interface)[14].

12 Οι ελληνικοί όροι “σειριοποίηση” και “σειριακοποίηση” είναι μάλλον αδόκιμοι. Ο όρος “δραματοποίηση”, προφανώς, δεν αφορά την πληροφορική.

13 Η σπατάλη υπολογιστικής ισχύος είναι ο λόγος που οδήγησε την Google στην απόφασή της να μην υποστηρίζει το περιβάλλον Android Web Services τα οποία βασίζονται στην τεχνολογία SOAP η οποία, με τη σειρά της, βασίζεται στη μεταφορά και επεξεργασία δυσανάλογα μεγάλων, σχετικά με την πραγματική πληροφορία που μεταφέρεται, XML αρχείων. Αντίθετα, υποστηρίζεται η τεχνολογία REST. Η τεχνολογία REST εξαιτίας της... οικονομίας της τείνει να αντικαταστήσει την τεχνολογία SOAP στα Web Services σε όλες τις εφαρμογές και όχι μόνο σε αυτές που απευθύνονται σε φορητές ή άλλες... αδύναμες συσκευές.

14 Ο όρος “μητρική διεπαφή χρήστη” είναι η προτεινόμενη από την υπηρεσία “Google Translate” απόδοση στα ελληνικά του αποδεκτού τεχνικού όρου “native user interface”.

- Είναι πολύ αργά στην εκτέλεση λειτουργιών.
- Είναι λιγότερο ευπαρουσίαστα και λειτουργικά από τα αντίστοιχα των μητρικών διεπαφών. Ένα παράθυρο το οποίο δημιουργήθηκε από κώδικα γραμμένο σε Java Swing, ξεχωρίζει αμέσως...

Η λύση στο προηγούμενο πρόβλημα είναι η χρήση ενός framework μέσω του οποίου θα δημιουργούνται διεπαφές οι οποίες θα βασίζονται στις μητρικές (native) βιβλιοθήκες χειρισμού της οθόνης. Λαμβάνοντας υπόψη τις ειδικές ανάγκες του Atlas επιλέγεται, το Qt της Nokia. Το framework Qt είναι γραμμένο σε C++ και χρησιμοποιείται κυρίως μέσω C και C++ εφαρμογών. Το Qt, αν όχι ιδανικό, είναι περισσότερο από κατάλληλο για το Atlas, διότι:

- Είναι ένα υπερπλήρες framework ανάπτυξης γραφικών διεπαφών (και όχι μόνο[15]). Αξίζει να σημειωθεί ότι περιβάλλον K.D.E. του λειτουργικού συστήματος Linux (ξανά: και όχι μόνο[16]) είναι γραμμένο σε Qt.
- Διανέμεται και ως ελεύθερο λογισμικό.
- Υποστηρίζει όλα τα λειτουργικά συστήματα για τα οποία αναπτύχθηκε και στα οποία υποστηρίζεται το Atlas.
- Μπορεί να χρησιμοποιηθεί μέσα από Java κώδικα με τη χρήση της βιβλιοθήκης “Qt Jambi” χωρίς να γραφεί ειδικός κώδικας σε C ή C++.
- Με τη χρήση του, ο προγραμματιστής μπορεί να διατηρήσει πλήρη έλεγχο, ακόμα και σε χαμηλό επίπεδο, στα αντικείμενα που παρουσιάζονται στην οθόνη.

Ακόμα και μέσα από Java, οι εφαρμογές που χρησιμοποιούν το Qt τρέχουν πολύ γρηγορότερα σε σχέση με αυτές που είναι γραμμένες σε Swing. Επιπλέον διατηρούν τις γραφικές ιδιαιτερότητες του λειτουργικού συστήματος στο οποίο τρέχουν.

LIBREOFFICE

Πέρα από την ανάγκη του εξυπηρετητή να διαβάζει αρχεία Excel τα οποία περιέχουν τα στοιχεία μίας ομάδας εξέτασης σε ειδικά διαμορφωμένη φόρμα, το σύστημα χρειάζεται τις υπηρεσίες μίας σουίτας εφαρμογών γραφείου όπως το LibreOffice, το OpenOffice ή το Microsoft Office. Συγκεκριμένα απαιτείται η χρήση

15 Το Qt διαθέτει δικούς του ενοποιημένους μηχανισμούς, για όλα το λειτουργικά συστήματα που υποστηρίζει, πρόσβασης σε βάσεις δεδομένων, προγραμματισμού πολυνηματικών (multithreaded) εφαρμογών, κ.α. Λόγω της χρήσης της Java, ως βασικής γλώσσας ανάπτυξης του Atlas, αυτοί οι μηχανισμοί δεν χρειάζεται να χρησιμοποιηθούν.

16 Το περιβάλλον K.D.E. τρέχει και κάτω από τα λειτουργικά συστήματα, B.S.D., Mac, ακόμα και Windows.

επεξεργαστή κειμένου ή / και λογιστικού φύλλου (spreadsheet). Η χρήση των προγραμμάτων αυτών εξυπηρετεί την δημιουργία εκτυπώσεων.

Η εκτύπωση μορφοποιημένου κειμένου, ιδιαίτερα όταν αυτό έχει πολύπλοκη διαμόρφωση όπως στήλες, εικόνες, πλαίσια, κ.λ.π., δεν είναι εύκολη υπόθεση διότι ο κώδικας που την υλοποιεί πρέπει να λάβει υπόψη πολλές παραμέτρους. Αρκετές από αυτές έχουν διαφορετικές τιμές από σύστημα σε σύστημα. Παράδειγμα τέτοιων παραμέτρων είναι:

- Ο χώρος που καταλαμβάνει στο χαρτί ένα τμήμα κειμένου, τόσο στο ύψος όσο και στο πλάτος.
- Η διάσταση του χαρτιού στο οποίο θα γίνει η εκτύπωση.
- Η ανάλυση του εκτυπωτή όπως αυτή έχει οριστεί στις ρυθμίσεις του driver.
- Το πραγματικό ωφέλιμο ύψος και πλάτους της σελίδας του χαρτιού (το μέγεθος αυτό είναι διαφορετικό σε κάθε μοντέλο εκτυπωτή).

Ένα πρόγραμμα το οποίο δημιουργεί εκτυπώσεις πρέπει να λάβει υπόψη όλες τις σχετικές παραμέτρους ώστε να υπολογίσει τι πρέπει να εκτυπωθεί, σε ποιο σημείο της σελίδας και με ποια μορφή για να δημιουργηθεί το επιθυμητό αποτέλεσμα.

Η απαιτούμενη προγραμματιστική εργασία είναι δυσανάλογη του αποτελέσματος που θα παράγει. Για το λόγο αυτό, οι εκτυπώσεις πρέπει γίνονται μέσω εξειδικευμένων προγραμμάτων. Οι επεξεργαστές κειμένου και τα λογιστικά φύλλα είναι προγράμματα τα οποία μπορούν εύκολα να χρησιμοποιηθούν για το σκοπό αυτό, με την προϋπόθεση ότι διαθέτουν μηχανισμό λήψης εντολών από άλλα προγράμματα. Ακολουθεί, ως παράδειγμα, τα βήματα που χρειάζεται να πραγματοποιηθούν για να χρησιμοποιηθεί ένα πρόγραμμα επεξεργασίας λογιστικών φύλλων για την εκτύπωση του πρακτικού εξέτασης ενός υποψηφίου του Atlas.

1. Αρχικά δημιουργείται ένα λογιστικό φύλλο το οποίο θα χρησιμοποιηθεί ως φόρμα εκτύπωσης. Σε μία τυπική μορφή του, στο πάνω μέρος της πρώτης σελίδας βρίσκεται το λογότυπο του φορέα που υποστηρίζει την εξέταση και μερικά βασικά στοιχεία για αυτόν (π.χ. τηλέφωνο επικοινωνίας). Έπειτα, βρίσκονται οι γραμμές με τα λεκτικά για τα στοιχεία του εξεταζόμενου και της κατηγορίας εξέτασης και, φυσικά, τα απαιτούμενα κελιά για να συμπληρωθούν τα στοιχεία αυτά. Στην συνέχεια, σε στήλες, θα υπάρχει χώρος για την καταγραφή των στοιχείων της κάθε ερώτησης στην οποία κλήθηκε να απαντήσει ο εξεταζόμενος. Στο τέλος της φόρμας θα υπάρχει χώρος για το τελικό αποτέλεσμα, τις υπογραφές των εξεταστών, κ.λ.π.

2. Τα περιεχόμενα διαμορφώνονται[17] ως προς την αισθητική (πλάτος στηλών, επιλογή γραμματοσειράς και ιδιότητες αυτής, κ.λ.π.) και ορίζονται οι παράμετροι της εκτύπωσης (μέγεθος χαρτιού, περιθώρια, αριθμός γραμμών οι οποίες θα επαναλαμβάνονται σε κάθε αλλαγή σελίδας του πρακτικού, κ.α.). Το διαμορφωμένο αρχείο – φόρμα αποθηκεύεται στον εξυπηρετητή σε συγκεκριμένη θέση και με συγκεκριμένο όνομα.
3. Κάθε φορά που πρέπει να τυπωθεί ένα πρακτικό εξέτασης, ο εξυπηρετητής δίνει εντολή στο υπολογιστικό φύλλο να τρέξει˙ κατά προτίμηση, χωρίς να φαίνεται στην οθόνη.
4. Στην συνέχεια συμπληρώνει τη φόρμα με τα στοιχεία που αφορούν τον εξεταζόμενο και το ερωτηματολόγιό του. Η συμπλήρωση γίνεται στις συγκεκριμένες θέσεις που έχουν προβλεφθεί στη φόρμα.
5. Ο εξυπηρετητής δίνει εντολή στο φύλλο να εκτυπωθεί και στην συνέχεια κλείνει το πρόγραμμα.

Τα δύο πρώτα βήματα πραγματοποιούνται μία φορά εκτός και αν υπάρξουν αλλαγές στα στοιχεία του φορέα ή απαιτηθούν διορθώσεις στην αισθητική του παραγόμενου εγγράφου. Τα επόμενα βήματα επαναλαμβάνονται για κάθε πρακτικό που θα εκτυπωθεί.

Από τις γνωστές σουίτες γραφείου, οι οποίες πληρούν τους όρους ώστε να χρησιμοποιηθούν ως μηχανισμοί δημιουργίας εγγράφων και εκτύπωσης, προτιμάται η χρήση του LibreOffice, για τους εξής δύο λόγους:

- Υποστηρίζει όλα τα λειτουργικά συστήματα στα οποία μπορεί να τρέξει το Atlas.
- Είναι ελεύθερο λογισμικό και δεν απαιτείται οικονομικό τίμημα για την εγκατάστασή του.

Λαμβάνοντας υπόψη ότι είναι πιθανό να χρειαστούν διαφορετικές βιβλιοθήκες για την επικοινωνία εξυπηρετητή – σουίτας, ανάλογα με το λειτουργικό σύστημα, η μη ύπαρξη έκδοσης του Microsoft Office για τα λειτουργικά συστήματα GNU Linux, Free BSD, κ.λ.π. δεν αποτελεί τεχνικό λόγο αποκλεισμού του από τη χρήση αυτή.

17 Η διαμόρφωση αφορά και το σταθερό κείμενο και τα κελιά στα οποία αργότερα θα τοποθετηθεί περιεχόμενο.

Κεφάλαιο 5

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ “ATLAS”

Η πλήρης ανάπτυξη ενός πληροφοριακού συστήματος διεξαγωγής εξετάσεων το οποίο ικανοποιεί τις απαιτήσεις οι οποίες προδιαγράφηκαν στα προηγούμενα κεφάλαια, κάτω από το όνομα “Atlas”, ξεφεύγει από τα όρια μίας διπλωματικής εργασίας, κυρίως λόγω του μεγέθους της απαιτούμενης δουλειάς. Το σύστημα που αναπτύχθηκε, ουσιαστικά, καλύπτει το μεγαλύτερο μέρος της λειτουργικότητας του Μ.Σ.Θ.Ε.Υ.Ο. αλλά λειτουργεί με τη χρήση Android συσκευών ως σταθμούς εξέτασης.

Το σύστημα που υλοποιήθηκε έχει ως στόχο:

- Να επιδείξει την καταλληλότητα των Android συσκευών ως σταθμών εξέτασης ενός συστήματος διεξαγωγής αξιολόγησης γνώσεων.
- Να επιδείξει την απλότητα χρήσης του συστήματος Atlas για όλες τις κατηγορίες χρηστών (εξεταζόμενων, εξεταστών, διαχειριστών συστήματος).
- Να επιδείξει την δυνατότητα ενός μηχανογραφικού συστήματος εξετάσεων να υπηρετήσει ειδικές κατηγορίες πολιτών, όπως τους αναλφάβητους και τους μετανάστες, με οικονομία και διαφάνεια[18].
- Να επιδείξει την καταλληλότητα των (περισσότερων) τεχνολογιών, οι οποίες αναλύθηκαν στο προηγούμενο κεφάλαιο, για την υλοποίηση του συστήματος.
- Να αποδείξει ότι ένα μηχανογραφικό σύστημα διεξαγωγής εξετάσεων μπορεί εύκολα να αντικαταστήσει τις υπάρχουσες χειρόγραφες διαδικασίες, για ορισμένες τουλάχιστον διαδικασίες εξετάσεων.

Στα υποκεφάλαια που ακολουθούν φαίνεται η γενική αρχιτεκτονική του συστήματος. Αναλύονται οι εφαρμογές που το αποτελούν και οι λειτουργίες που υλοποιούν οι εφαρμογές αυτές, η μεταφορά πληροφορίας μέσα στο σύστημα, η

18 Μέχρι την εφαρμογή του Μ.Σ.Θ.Ε.Υ.Ο, οι αλλοδαποί οι οποίοι εξετάζονταν για δίπλωμα οδήγησης, χρησιμοποιούσαν διερμηνέα, ο οποίος τους μετέφραζε την ερώτηση και τις πιθανές απαντήσεις στη γλώσσα τους, και κατέγραφε την απάντηση του αλλοδαπού υποψηφίου στο ελληνικό φύλλο εξέτασης. Ο διερμηνέας αυτός δεν ήταν υπάλληλος της υπηρεσίας μεταφορών και επικοινωνιών. Η αμοιβή του καταβαλλόταν από την οικεία νομαρχία.

δομής της βάσης δεδομένων του συστήματος, κ.α. Προκειμένου να μπορεί να διαχωριστεί από το Atlas, η περιορισμένη έκδοση του νέου συστήματος η οποία υλοποιήθηκε για τις ανάγκες της παρούσας εργασίας, ονομάζεται miniAtlas.

ΟΙ ΕΦΑΡΜΟΓΕΣ ΤΟΥ miniAtlas

Το σύστημα miniAtlas αποτελείται από τρία προγράμματα:

- *Το πρόγραμμα Server*. Το πρόγραμμα αυτό τρέχει στον εξυπηρετητή του συστήματος. Είναι υπεύθυνο για την εκτέλεση των διαχειριστικών λειτουργιών του miniAtlas και συντονίζει τους σταθμούς εξέτασης κατά τη διάρκεια της εξέτασης.

- *Το πρόγραμμα Client*. Το πρόγραμμα αυτό τρέχει στους σταθμούς εξέτασης. Παρουσιάζει στον εξεταζόμενο το ερωτηματολόγιο και καταγράφει τις απαντήσεις του.

- *Το πρόγραμμα Daemon.* Το πρόγραμμα Daemon τρέχει, επίσης, στους σταθμούς εξέτασης και αναλαμβάνει να εκτελεί τις εντολές του εξυπηρετητή. Μέσω του Daemon διεκπεραιώνεται η επικοινωνία μεταξύ εξυπηρετητή και σταθμού εξέτασης.

Το πρόγραμμα Server.

Το πρόγραμμα Server, ο πλήρης πηγαίος κώδικας του οποίου βρίσκεται στο παράρτημα Β, είναι το πρόγραμμα μέσω του οποίου οι εξεταστές διαχειρίζονται και ελέγχουν τις εξετάσεις. Μέσω των επιλογών που παρέχει στους χρήστες του το πρόγραμμα Server, ο εξεταστής έχει τη δυνατότητα:

- Να εισάγει στο σύστημα τα στοιχεία μίας ομάδας εξέτασης μέσω ενός ".xlsx" αρχείου. Αναλυτική περιγραφή της γραμμογράφησης του φύλλου γίνεται στο επόμενο κεφάλαιο.

- Να διαχειριστεί τις ομάδες εξέτασης. Με τον όρο διαχείριση, ουσιαστικά, εννοείται η παρουσίαση των προγραμματισμένων ομάδων και της κατάστασή τους (προγραμματισμένη, ολοκληρωμένη, ακυρωμένη) και η ακύρωση της εξέτασης για κάποιες από αυτές. Διαγραφή στοιχείων από το σύστημα δεν προβλέπεται.

- Να εκκινήσει τη διαδικασία εξέτασης μίας ομάδας. Η εξέταση ολοκληρώνεται με την εκτύπωση των αποτελεσμάτων. Κατά τη διάρκεια της εξέτασης ο εξυπηρετητής δεν χρειάζεται να αλληλεπιδρά με τον χειριστή του.

- Να δει και να επανεκτυπώσει τα αποτελέσματα κάποιας εξέτασης από το αρχείο του συστήματος.

- Να διαχειριστεί τα δεδομένα των σταθμών εξέτασης. Τα δεδομένα των σταθμών εξέτασης είναι: το όνομα του σταθμού, η διεύθυνσή του στο δίκτυο, η κατάστασή του (αν είναι ενεργός, απενεργοποιημένος, κ.λ.π.). Παρουσιάζεται, επίσης, αν υπάρχει επικοινωνία μεταξύ εξυπηρετητή και σταθμού εξέτασης σύμφωνα με τα πρωτόκολλα του Atlas.

Ως πρόγραμμα του miniAtlas γραμμένο σε γλώσσα Java, το Server αποτελείται από ένα σύνολο Java κλάσεων η συνεργασία των οποίων παράγει το τελικό αποτέλεσμα. Οι κλάσεις έχουν χωριστεί σε πακέτα (packages) σύμφωνα με τους κανόνες σωστής δόμησης και επαναχρησιμοποίησης λογισμικού. Κάποια από αυτά τα πακέτα έχουν γραφτεί στο παρελθόν και είναι γενικής χρήσης[19]. Ακολουθεί μία απλή αναφορά των σημαντικότερων από αυτές. Ο αναγνώστης μπορεί να δει την πλήρη περιγραφή τους στα παραρτήματα της εργασίας όπου βρίσκεται ο πηγαίος κώδικας, πλήρως τεκμηριωμένος μέσω “Javadocs”.

- *ClientStations (org.gmele.dissertation.server)*. Περιέχει τις πληροφορίες που αφορούν όλους τους σταθμούς εξέτασης[20]. Παρέχει μεθόδους για την ανάγνωση των πληροφοριών αυτών. Ο κατασκευαστής αρχικοποιεί το αντικείμενο με τις πληροφορίες που βρίσκονται στη βάση δεδομένων.

- *ExamServer (org.gmele.dissertation.server)*. Είναι η κύρια κλάση του Server. Αρχικοποιεί το σύστημα και παρέχει μεθόδους οι οποίες υλοποιούν βασικές λειτουργίες όπως την διεξαγωγή της διαδικασίας εξέτασης, την κλήρωση σταθμών εξέτασης, κ.λ.π.

- *Uhura*[21] (*org.gmele.dissertation.server.communications*). Είναι υπεύθυνη για την επικοινωνία μεταξύ του προγράμματος Server και των σταθμών εξέτασης. Η επικοινωνία πραγματοποιείται μέσω tcp sockets και υλοποιεί συγκεκριμένο πρωτόκολλο το οποίο αναλύεται σε επόμενο υποκεφάλαιο.

- *ESException (org.gmele.dissertation.server.exception)*. Η αναφορά και ο χειρισμός των λαθών, που συμβαίνουν στο Server, γίνεται με την χρήση εξαιρέσεων (exceptions). Όλες οι εξαιρέσεις που δημιουργούνται στο κύριο τμήμα του προγράμματος αναφέρονται ως αντικείμενα της κλάσης *ESException* η οποία κληρονομεί (extends) την γενική κλάση *GmException* η οποία περιέχει τρία attributes: Τον κωδικό του λάθους ως ακέραιο αριθμό, την περιγραφή του λάθους ως συμβολοσειρά και (αν

19 Έχουν χρησιμοποιηθεί ή θα χρησιμοποιηθούν από το γράφοντα και σε άλλες εφαρμογές.

20 Σε ένα στιγμιότυπο της κλάσης.

21 Για τους φίλους της τηλεοπτικής επιστημονικής φαντασίας, η κλάση πήρε το όνομά της από την αξιωματικό επικοινωνιών του αστρόπλοιου “U.S.S. Enterprise” της σειράς Star Trek (The Original Series).

υπάρχει) την εξαίρεση που δημιουργήθηκε από άλλη κλάση και είχε ως αποτέλεσμα τη δημιουργία της τρέχουσας εξαίρεσης[22]. Η κλάση διαθέτει μόνο κατασκευαστές και μεθόδους τύπου getter για τα attributes και υλοποιεί (implements) το interface *ESExceptionConsts* το οποίο βρίσκεται στο ίδιο πακέτο και περιέχει όλους τους κωδικούς λαθών του προγράμματος.

- *ClientForm (org.gmele.dissertation.server.gui)*. Είναι το παράθυρο (φόρμα) μέσω του οποίου γίνεται ο χειρισμός των σταθμών εξέτασης, σε ό,τι αφορά την διεπαφή με το χρήστη.

- *MainForm (org.gmele.dissertation.server.gui)*. Η κλάση υλοποιεί το βασικό παράθυρο, με τα μενού επιλογής, του προγράμματος.

- *ResultForm (org.gmele.dissertation.server.gui)*. Παρουσιάζει τα αποτελέσματα μίας ομάδας εξέτασης και δίνει την δυνατότητα εκτύπωσής τους. Σε περίπτωση που τα αποτελέσματα που παρουσιάζονται αφορούν την εξέταση που μόλις ολοκληρώθηκε, η εκτύπωση ενεργοποιείται αυτόματα.

- *TestSheet (org.gmele.dissertation.testhandler)*. Αποτελεί ένα ολοκληρωμένο φύλλο (ερωτηματολόγιο) εξέτασης. Περιέχει τα στοιχεία του εξεταζόμενου και τις ερωτήσεις στις οποίες θα εξεταστεί. Αντικείμενα της κλάσης αυτής αποστέλλονται στους σταθμούς εξέτασης μέσω serialization και επιστρέφουν συμπληρωμένα, βαθμολογούνται και αποθηκεύονται ως αρχεία σε προκαθορισμένο χώρο.

- *TestHeader (org.gmele.dissertation.testhandler*). Περιέχει τα στοιχεία του εξεταζομένου και της εξέτασης στην οποία συμμετέχει. Χρησιμοποιείται ως δομή (C struct) και δεν έχει μεθόδους. Χρησιμοποιείται από την *TestSheet*.

- *Question (org.gmele.dissertation.testhandler)*. Κάθε στιγμιότυπο της κλάσης περιέχει τα στοιχεία μίας ερώτησης του ερωτηματολογίου (λεκτικό ερώτησης, λεκτικά απαντήσεων, εικόνα ερώτησης, κ.α.). Στα στοιχεία αυτά περιλαμβάνεται και το πεδίο στο οποίο καταγράφεται η απάντηση του εξεταζόμενου για τη συγκεκριμένη ερώτηση.

- *ExamDisDB (org.gmele.dissertation.testhandler)*. Μέσω των μεθόδων της κλάσης αυτής γίνεται η πρόσβαση στη βάση δεδομένων και πραγματοποιούνται οι βασικές λειτουργίες του συστήματος που σχετίζονται με αυτή. Κατά παράβαση κανόνων ορθού σχεδιασμού Java προγραμμάτων,

22 Για παράδειγμα σε μία μέθοδο η οποία διαβάζει κάποια στοιχεία από τη βάση, αν δεν υπάρχουν τα στοιχεία αυτά (π.χ. άκυρος κωδικός), η εξαίρεση που θα δημιουργηθεί θα περιέχει μόνο κωδικό και περιγραφή του λάθους. Αν όμως, κατά την ανάκτηση της πληροφορίας από τη βάση δεδομένων δημιουργηθεί εξαίρεση (π.χ. *SqlException*) τότε εκτός από τον κωδικό λάθους και την περιγραφή του, η εξαίρεση που θα δημιουργηθεί θα περιέχει αναφορά και στην αρχική εξαίρεση την οποία πέταξε (throws) η βάση δεδομένων.

δεν υλοποιεί μόνο λειτουργίες που σχετίζονται με τη δημιουργία ή την αξιολόγηση ερωτηματολογίων αλλά και χειρισμού ομάδων εξέτασης[23].

- *ExamStruct (org.gmele.dissertation.testhandler*). Χρησιμοποιείται για την αποθήκευση (ως C struct) των στοιχείων μίας ομάδας εξέτασης, όπως αυτά ανακτώνται από τον σχετικό πίνακα της βάσης δεδομένων.

- *PeopleStruct (org.gmele.dissertation.testhandler)*. Όπως και η προηγούμενη, αλλά για τα στοιχεία ενός εξεταζομένου.

- *TestHandler (org.gmele.dissertation.testhandler)*. Η κλάση *TestHandler* παρέχει στατικές μεθόδους χειρισμού φύλλων εξέτασης (*TestSheet*). Είναι υπεύθυνη, μεταξύ άλλων, για τη δημιουργία ενός φύλλου, την βαθμολόγησή του και τη δημιουργία των φύλλων μίας ολόκληρης εξέτασης.

- *TestException (org.gmele.dissertation.testhandler.exceptions)*. Αντικείμενα της κλάσης αυτής δημιουργούνται όταν δημιουργούνται λάθη στο τμήμα του προγράμματος που σχετίζεται με τον χειρισμό των φύλλων εξέτασης. Η *TestException*, όπως και η *ESException* κληρονομεί την γενική κλάση *GMException*. Υλοποιεί το interface *TestExceptionConsts* το οποίο βρίσκεται στο ίδιο πακέτο και το οποίο περιέχει τους κωδικούς των λαθών που μπορεί να δημιουργηθούν στη βιβλιοθήκη χειρισμού φύλλων εξέτασης.

- *DBSimlpeConn (org.gmele.general.dbconnections)*. Γενική κλάση η οποία μέσω στατικών μεθόδων δημιουργεί συνδέσεις (connections) προς MySql βάσεις δεδομένων.

- *XlsxSheet (org.gmele.general.sheets.excelx)*. Γενική κλάση η οποία χειρίζεται λογιστικά φύλλα "xlsx" μορφής. Βασίζεται στην βιβλιοθήκη POI του ιδρύματος Apache. Δεν προϋποθέτει την εγκατάσταση του Microsoft Excel για να λειτουργήσει (χειρίζεται τα αρχεία και όχι το πρόγραμμα).

Το πρόγραμμα Server, όπως αυτό υλοποιήθηκε για το σύστημα miniAtlas, δεν χρησιμοποιεί το Qt framework για την διεπαφή με το χρήστη αλλά τη βιβλιοθήκη Swing η οποία είναι ενσωματωμένη στη γλώσσα Java.

23 Η παραβίαση του σωστού σχεδιασμού σχετίζεται με την επαναχρησιμοποίηση του πακέτου στο οποίο βρίσκεται η κλάση. Για παράδειγμα αν το πακέτο "*org.gmele.dissertation.testhandler*" χρησιμοποιηθεί και από το πρόγραμμα Client, το οποίο τρέχει στους σταθμούς εξέτασης, για το χειρισμό των φύλλων εξέτασης, θα εισαχθούν σε αυτό και μέθοδοι οι οποίες αφορούν αποκλειστικά το Server και δεν τις χρειάζεται.

Το πρόγραμμα Client

Το πρόγραμμα Client, ο πηγαίος κώδικας του οποίου φαίνεται στο παράρτημα Γ, είναι υπεύθυνο για την παρουσίαση του ερωτηματολογίου στον εξεταζόμενο και την καταγραφή των απαντήσεων του. Συγκεκριμένα το πρόγραμμα:

- Παρουσιάζει αρχική οθόνη με τα στοιχεία του εξεταζόμενου ο οποίος κληρώθηκε να εξεταστεί στον συγκεκριμένο σταθμό εξέτασης και τα στοιχεία εξέτασης (κατηγορία εξέτασης, ενεργοποίηση ήχου)
- Καταγράφει την πληροφορία της παρουσίας ή της απουσίας του εξεταζόμενου.
- Παρουσιάζει την κάθε ερώτηση με τις πιθανές απαντήσεις, από τις οποίες θα επιλέξει ο εξεταζόμενος την κατά τη γνώμη του σωστή, καθώς και την εικόνα που πιθανώς τη συνοδεύει.
- Εκφωνεί την ερώτηση και τις απαντήσεις της σε περίπτωση που ενεργοποιηθεί η σχετική επιλογή.
- Παρουσιάζει τον υπολειπόμενο χρόνο που διαθέτει ο εξεταζόμενος για να ολοκληρώσει το ερωτηματολόγιο.
- Καταγράφει τις απαντήσεις του εξεταζόμενου.

Το πρόγραμμα Client δεν ενεργοποιείται χειροκίνητα από κάποιον χρήστη. Αντίθετα ξεκινά με εντολή του προγράμματος Daemon. Δεν επικοινωνεί με τον εξυπηρετητή του συστήματος αλλά μόνο με το Daemon από το οποίο δέχεται εντολές και μέσω του οποίου συγχρονίζεται στην όλη διαδικασία. Η επικοινωνία αυτή αναλύεται σε επόμενο υποκεφάλαιο.

Τα δεδομένα που χρειάζεται για να λειτουργήσει τα διαβάζει από ένα αρχείο στο οποίο περιέχεται ένα πλήρως αρχικοποιημένο, από το πρόγραμμα Server, αντικείμενο της κλάσης `TestSheet` με την μέθοδο του serialization. Το αρχείο αυτό, στο τέλος της διαδικασίας, ενημερώνεται με τις απαντήσεις του εξεταζομένου. Οι εικόνες και οι ήχοι που απαιτούνται στην διαδικασία εξέτασης βρίσκονται προεγκατεστημένοι σε φάκελο του σταθμού εξέτασης και δεν συμπεριλαμβάνονται ούτε στους πόρους (resources / assets) του, ούτε στο ερωτηματολόγιο. Με τον τρόπο επιτυγχάνονται τα ακόλουθα:

- Ο όγκος των δεδομένων που μεταδίδονται στο δίκτυο κατά τη διάρκεια της εξέτασης παραμένει μικρός[24], ακόμα και όταν η αίθουσα έχει πολλούς σταθμούς εξέτασης

24 Ένα πλήρες φύλλο εξέτασης το οποίο ανήκει στην κατηγορία εξέτασης του κώδικα (30 ερωτήσεις) έχει μέγεθος περίπου 15 KB.

- Οι πόροι που χρησιμοποιεί το πρόγραμμα (εικόνες / ήχοι) μπορεί να ενημερώνονται (update) ανεξάρτητα, κάθε φορά που υπάρχει ανάγκη, χωρίς τις πολύπλοκες διαδικασίες και τα προβλήματα ασφαλείας που προκύπτουν κατά την αναβάθμιση (upgrade) του λογισμικού.
- Η αρχειοθέτηση των ερωτηματολογίων, στον εξυπηρετητή, δεν καταλαμβάνει πολύ χώρο.

Το πρόγραμμα Client αποτελείται από τις ακόλουθες δύο activities:

- *IntroScreen (org.gmele.dissertation.client)*. Είναι η activity με την η οποία ξεκινά το πρόγραμμα (αρχική οθόνη). Παρουσιάζει τα στοιχεία της εξέτασης και του εξεταζομένου και περιμένει να πατηθεί το κουμπί που δηλώνει ότι ο εξεταζόμενος έχει προσέλθει στην αίθουσα ή αυτό που δηλώνει την απουσία του. Η *IntroScreen* δέχεται πληροφορίες συγχρονισμού και αναφέρει την παρουσία ή την απουσία του εξεταζομένου μέσω του προγράμματος Daemon.
- *ExamActiv (org.gmele.dissertation.client)*.Είναι η activity με την οποία πραγματοποιείται η εξέταση. Παρουσιάζει τις εικόνες και καταγράφει τις απαντήσεις.

Προκειμένου να διαβάσουν και να γράψουν στα φύλλα εξέτασης (κλάση *TestSheet*) και οι δύο προαναφερθείσες activities χρησιμοποιούν το package "*org.gmele.dissertation.testhandler*", όπως αυτό χρησιμοποιείται και στον Server.

Το πρόγραμμα Daemon

Το πρόγραμμα Daemon, ο κώδικας του οποίου βρίσκεται στο παράρτημα Δ, δημιουργεί ένα Service στον σταθμό εξέτασης το οποίο αναλαμβάνει να υλοποιεί την επικοινωνία μεταξύ εξυπηρετητή και σταθμού εξέτασης. Συγκεκριμένα, το Daemon:

- Δέχεται εντολές από τον εξυπηρετητή και τις εκτελεί.
- Ενεργοποιεί το πρόγραμμα Client όταν ξεκινήσει η διαδικασία εξέτασης στην αίθουσα
- Μεταφέρει τις εντολές του Server, οι οποίες έρχονται μέσω του δικτύου, στο Client και καταγράφει την τρέχουσα κατάσταση της διαδικασίας εξέτασης όπως την αναφέρει το πρόγραμμα Client.

Το πρόγραμμα Daemon αποτελείται από τις ακόλουθες δύο κλάσεις:

- Το Service *Daemon (org.gmele.dissertation.daemon)* στο οποίο υλοποιείται όλη η λειτουργικότητα του προγράμματος και το οποίο υλοποιεί

το interface *DaemonConsts* στο οποίο ορίζονται οι κωδικοί των εντολών που υποστηρίζονται από το σύστημα.

- Τη Activity *DaemonActivity (org.gmele.dissertation.daemon)* η οποία ενεργοποιεί το Service και τερματίζει τον εαυτό της χωρίς να έχει καμία αλληλεπίδραση με τον χρήστη (δεν εκτελείται ούτε η μέθοδος "*setContentView*").

Σε πραγματική εφαρμογή του συστήματος, το Service θα ξεκινούσε αυτόματα, με το άνοιγμα της συσκευής, από το λειτουργικό σύστημα (καταγράφοντας το ως *IntentReceiver* για το *BOOT_COMPLETED_ACTION*). Ο τρόπος αυτός δεν επιλέχτηκε στην παρούσα εργασία, διότι αρκετές από τις Android συσκευές στις οποίες έγιναν τα σχετικά πειράματα ήταν δανικές. Στο εξής, όπου γίνεται αναφορά στο Daemon θα εννοείται το Service και όχι η συνολική εφαρμογή.

Το Daemon λειτουργεί ως εξυπηρετητής εντολών για μηνύματα τα οποία έρχονται από το Server. Ανοίγει ένα TCP Socket και το "ακούει" συνεχώς[25]. Τα μηνύματα - εντολές είναι ακέραιοι αριθμοί οι οποίοι έρχονται σε μορφή κειμένου (strings). Οι εντολές μπορούν να ακολουθούνται από παραμέτρους. Το κανάλι διαβάζεται (και γράφεται) ως απλό stream (byte προς byte και όχι γραμμή προς γραμμή), προκειμένου να μπορούν μέσα από αυτό να μεταφερθούν και ολόκληρα binary αρχεία ως σειρές από bytes. Σε επόμενο υποκεφάλαιο αναλύονται οι εντολές που δέχεται το Daemon από τον εξυπηρετητή.

Η ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ miniAtlas.

Η βάση δεδομένων του miniAtlas σχεδιάστηκε ώστε να καλύψει πλήρως τις απαιτήσεις ενός συστήματος διεξαγωγής εξετάσεων, το οποίο θα επιδεικνύει τις δυνατότητες του συστήματος Atlas, σύμφωνα με όσα γράφηκαν στην αρχή του κεφαλαίου. Καλύπτει τις τέσσερις βασικές κατηγορίες εξέτασης για διπλώματα εξέτασης και παρουσιάζει πολλές ομοιότητες με τη βάση δεδομένων που χρησιμοποιεί το Μ.Σ.Θ.Ε.Υ.Ο. όπως αυτή παρουσιάζεται στο επίσημο εγχειρίδιο του συστήματος.

Ακολουθεί σύντομη περιγραφή των πινάκων οι οποίοι περιέχονται στη βάση και αναφορά στα πεδία από τα οποία αποτελούνται. Στο παράρτημα Α ο αναγνώστης μπορεί να βρει την πλήρη εξαγωγή της βάσης σε μορφή sql όπως αυτή γίνεται από το phpMyAdmin. Στην εξαγωγή δεν περιλαμβάνονται, φυσικά, τα δεδομένα της βάσης.

25 Στην επικοινωνία μεταξύ εξυπηρετητή και σταθμών εξέτασης,χρησιμοποιείται Client – Server αρχιτεκτονική. Μόνο που οι σταθμοί εξέτασης είναι, τεχνικά, οι Servers οι οποίοι περιμένουν να δεχτούν εντολές και να τις εξυπηρετήσουν, και ο εξυπηρετητής του Atlas είναι ο πελάτης της επικοινωνίας ο οποίος δίνει τις εντολές και παραλαμβάνει αποτελέσματα.

Ο πίνακας Questions.

Ο πίνακας Questions περιλαμβάνει όλες τις ερωτήσεις, όλων των κατηγοριών και σε όλες τις γλώσσες του συστήματος miniAtlas. Ο πίνακας είναι ενιαίος, διότι όλες τις κατηγορίες εξέτασης τις χειρίζεται το ίδιο module. Ο πίνακας περιέχει τα εξής πεδία:

Όνομα	Τύπος	Περιγραφή
QCod	*Int (10)*	Κωδικός ερώτησης.
QKateg	*Meduimint (8)*	Κωδικός κατηγορίας εξέτασης στην οποία ανήκει (1 κώδικας, 2 μοτοσυκλέτες, κ.λ.π).
QPag	*Meduimint (8)*	Κωδικός εσωτερικής κατηγορίας στην οποία ανήκει.
QLang	*Meduimint (8)*	Κωδικός γλώσσας της ερώτησης.
QLect	*Varchar (500)*	Λεκτικό (κείμενο) της ερώτησης
QPhoto	*Varchar (20)*	Όνομα αρχείου φωτογραφίας η οποία σχετίζεται με την ερώτηση (“0” αν η ερώτηση δεν έχει φωτογραφία)
QSound	*Varchar (20)*	Όνομα αρχείου ήχου για εκφώνηση της ερώτησης (“0” αν δεν υπάρχει)
QBook	*Varchar (50)*	Λεκτικό προσδιορισμού της ερώτησης. Εκτυπώνεται στα πρακτικά εξέτασης[26].

Προκειμένου να απενεργοποιηθεί μία ερώτηση, δηλαδή να μην κληρώνεται πλέον σε φύλλα εξέτασης, παίρνει κωδικό εσωτερικής κατηγορίας μεγαλύτερο από το πλήθος των εσωτερικών κατηγοριών μίας κατηγορίας εξέτασης. Σε καμία περίπτωση μία ερώτηση δεν πρέπει να διαγράφεται από τη βάση διότι δεν θα μπορούν να επανεκτυπωθούν πρακτικά παλαιότερων ερωτηματολογίων στα οποία περιλαμβάνεται.

Ο πίνακας Answers

Ο πίνακας Answers περιέχει τις απαντήσεις όλων των ερωτήσεων, όλων των κατηγοριών και σε όλες τις γλώσσες του συστήματος miniAtlas. Ισχύουν οι παρατηρήσεις του προηγούμενου πίνακα. Ο πίνακας περιέχεται τα εξής πεδία:

Όνομα	Τύπος	Περιγραφή
AQCod	*Int (10)*	Κωδικός ερώτησης στην οποία ανήκει η απάντηση.

26 Το πεδίο ονομάστηκε έτσι διότι στο Μ.Σ.Θ.Ε.Υ.Ο. το πεδίο συσχετίζει την ερώτηση με τα περιεχόμενα του επίσημου βιβλίου του Υ.Μ.Ε.

AAA	*smallint (5)*	Αύξοντας αριθμός απάντησης (για τη συγκεκριμένη ερώτηση).
ALect	*Varchar (500)*	Λεκτικό (κείμενο) της απάντησης.
ACorr	*Tinyint (1)*[27]	Είναι η σωστή απάντηση; (N / O)
QSound	*Varchar (20)*	Όνομα αρχείου ήχου για εκφώνηση της απάντησης (“0” αν δεν υπάρχει)

Ο πίνακας Answers δεν παρουσιάζει τους ίδιους περιορισμούς με τον Questions σε ό,τι αφορά την προσθήκη ή την αφαίρεση απαντήσεων. Αρκεί, πάντα, να σχετίζονται περισσότερες από μία απαντήσεις με μία ερώτηση και να υπάρχει μία και μόνο μία απάντηση η οποία να έχει τιμή true στο πεδίο ACorr.

Ο πίνακας Numbers.

Ο πίνακας Numbers περιέχει τους κωδικούς των εσωτερικών κατηγοριών μίας κατηγορίας εξέτασης και το πλήθος των ερωτήσεων, ανά εσωτερική κατηγορία, από τις οποίες αποτελείται ένα ερωτηματολόγιο. Ο πίνακας παρουσιάζει την εξής γραμμογράφηση.

Όνομα	Τύπος	Περιγραφή
KCod	*Mediumint (8)*	Κωδικός κατηγορίας εξέτασης.
PCod	*Int (10)*	Αύξοντας αριθμός εσωτερικής κατηγορίας.
Numb	*Int (10)*	Πλήθος ερωτήσεων της εσωτερικής κατηγορίας που περιέχονται σε ένα ερωτηματολόγιο της συγκεκριμένης κατηγορίας εξέτασης.

Το συνολικό πλήθος των ερωτήσεων από τις οποίες αποτελείται ένα ερωτηματολόγιο προκύπτει από το άθροισμα του πλήθους των ερωτήσεων όλων των εσωτερικών κατηγοριών[28].

27 Τα πεδία τύπου tinyint (1) είναι, ουσιαστικά, boolean τιμές.

28 Στο Μ.Σ.Θ.Ε.Υ.Ο. κάθε ερώτηση που μπαίνει στο ερωτηματολόγιο θεωρείται ξεχωριστή εσωτερική κατηγορία. Για παράδειγμα, το ερωτηματολόγιο του κώδικα (αυτοκίνητα) το οποίο περιέχει 30 ερωτήσεις αποτελείται από 30 εσωτερικές κατηγορίες. Η απόφαση αυτή πάρθηκε από το Υ.Μ.Ε., για λόγους συμβατότητας της χειρόγραφης διαδικασίας εξέτασης με την μηχανογραφική, και δεν σχετίζεται με τις δυνατότητες του συστήματος.

Ο πίνακας Kateg

Ο πίνακας Kateg περιέχει πληροφορίες για τις κατηγορίες εξέτασης. Ο πίνακας αποτελείται από τα εξής πεδία:

Όνομα	Τύπος	Περιγραφή
KCod	*Mediumint (8)*	Κωδικός κατηγορίας εξέτασης.
KLect	*Varchar (50)*	Λεκτικό κατηγορίας εξέτασης (π.χ. μοτοσυκλέτες).
KTime	*Mediumint (8)*	Χρόνος εξέτασης σε πρώτα λεπτά της ώρας.
AllowedErrors	*Mediumint (8)*	Μέγιστος αριθμός επιτρεπόμενων λάθος απαντήσεων.
KPic	*Varchar (30)*	Όνομα αρχείου εικόνας κατηγορίας για την αρχική οθόνη του Client.

Η πληροφορία του πεδίου *KPic* δεν χρησιμοποιήθηκε στο πρόγραμμα λόγω έλλειψης... καλλιτεχνικών δυνατοτήτων από τον γράφοντα.

Ο πίνακας Langs

Ο πίνακας Langs περιέχει τις γλώσσες τις οποίες υποστηρίζει το miniAtlas και παρουσιάζει την εξής μορφή:

Όνομα	Τύπος	Περιγραφή
LCod	*Mediumint (8)*	Κωδικός γλώσσας εξέτασης.
LLect	*Varchar (30)*	Λεκτικό κατηγορίας εξέτασης (π.χ. μοτοσυκλέτες).

Ο κωδικός της γλώσσας χρησιμοποιείται ως ξένο κλειδί (foreign key) στον πίνακα των ερωτήσεων. Το λεκτικό της κατηγορίας δεν χρησιμοποιείται, επί του παρόντος, προγραμματιστικά και υπάρχει ως επεξήγηση για αυτούς που διαβάζουν τη βάση δεδομένων.

Ο πίνακας Clients

Ο πίνακας Clients περιέχει πληροφορίες για τους σταθμούς εξέτασης οι οποίοι είναι συνδεδεμένοι στο σύστημα. Μέσω των πληροφοριών που βρίσκονται σε αυτόν τον πίνακα, το Server γνωρίζει με ποιους σταθμούς να επικοινωνήσει και σε ποια IP διεύθυνση. Η γραμμογράφηση του πίνακα έχει ως ακολούθως:

Όνομα	Τύπος	Περιγραφή
ClID	*Mediumint (8)*	Κωδικός σταθμού εξέτασης.
ClName	*Varchar (20)*	Λεκτικό (όνομα) σταθμού εξέτασης.
ClIP	*Varchar (15)*	IP διεύθυνση του σταθμού εξέτασης.
ClStatus	*enum('Active', 'InActive','Damaged')*	Τρέχουσα κατάσταση του σταθμού εξέτασης.

Η IP διεύθυνση του σταθμού πρέπει να είναι στατική. Αυτό μπορεί να επιτευχθεί με διάφορους τρόπους. Υπενθυμίζεται ότι, σύμφωνα με τις προδιαγραφές του Atlas, ένα δίκτυο από σταθμούς εξέτασης μπορεί να αποτελείται ταυτόχρονα και από PCs και από Android tablets. Στον πίνακα δεν χρειάζεται να γίνει διαχωρισμός διότι το Server τα μεταχειρίζεται με τον ίδιο ακριβώς τρόπο.

Ο πίνακας Examinations.

Ο πίνακας Examinations περιέχει τις πληροφορίες οι οποίες αφορούν τις ομάδες εξέτασης που προγραμματίστηκαν στο βάθος του χρόνου της λειτουργίας του συστήματος. Περιέχει τα παρακάτω πεδία:

Όνομα	Τύπος	Περιγραφή
ExamID	*Int (10)*	Κωδικός ομάδας εξέτασης
ExamGroup	*Mediumint (8)*	Αριθμός ομάδας εξέτασης σε συγκεκριμένη ημερομηνία
ExamDate	*Datetime*	Προγραμματισμένη ημερομηνία και ώρα διεξαγωγής της εξέτασης.
ExamStatus	*enum('Pending', 'Active','Done','Aborted','Cancelled')*	Κατάσταση εξέτασης.

Ο αριθμός της ομάδας εξέτασης δεν είναι υποχρεωτικό να παίρνει αριθμούς στη σειρά. Δεν μπορεί, όμως, δύο ομάδες εξέτασης που είναι προγραμματισμένες για την ίδια μέρα να έχουν τον ίδιο αριθμό ομάδας.

Η κατάσταση 'Active' σημαίνει ότι η ομάδα εξετάζεται την συγκεκριμένη χρονική στιγμή. Η κατάσταση 'Aborted' αντιπροσωπεύει ομάδες των οποίων η εξέταση ξεκίνησε αλλά διακόπηκε (πιθανόν λόγω τεχνικού προβλήματος), ενώ η κατάσταση 'Cancelled' αφορά ομάδες των οποίων η εξέταση ματαιώθηκε για κάποιο λόγο, πριν ξεκινήσει η εξέτασή τους.

Ο πίνακας People.

Ο πίνακας People περιέχει τις πληροφορίες των εξεταζομένων, για όλες τις ομάδες εξέτασης. Οι πληροφορίες αφορούν τόσο τα στοιχεία ταυτότητας του εξεταζόμενου όσο και την κατηγορία στην οποία εξετάζεται την συγκεκριμένη μέρα και ώρα. Η γραμμογράφηση του πίνακα έχει ως εξής:

Όνομα	Τύπος	Περιγραφή
PID	*Int (10)*	Κωδικός εξεταζομένου.
Surname	*Varchar (50)*	Επώνυμο εξεταζομένου.
Firstame	*Varchar (50)*	Όνομα Εξεταζομένου
Fathername	*Varchar (50)*	Πατρώνυμο εξεταζομένου
IDNumber	*Varchar(15)*	Αριθμός δελτίου ταυτότητας ή άλλου έγγραφου ταυτοποίησης (π.χ. διαβατήριο)
ExamID	*Int (10)*	Κωδικός εξέτασης στην οποία συμμετέχει
ExamKateg	*Mediumint (8)*	Κωδικός Κατηγορίας Εξέτασης
ExamLang	*Mediumint (8)*	Κωδικός γλώσσας εξέτασης
Speech	*Tinyint (1)*	Αναλφάβητος – Θα γίνει εκφώνηση ερωτήσεων (Ν /Ο);
ClientID	*Int (10)*	Κωδικός σταθμού εξέτασης στον οποίο κληρώθηκε ο υποψήφιος να εξεταστεί
Result	*enum('Pass', 'Fail','Absent','-')*	Αποτέλεσμα εξέτασης.

Τα στοιχεία ταυτότητας του κάθε εξεταζόμενου εισάγονται κάθε φορά που συμμετέχει σε κάποια ομάδα εξέτασης και δεν συσχετίζονται με προηγούμενες συμμετοχές σε εξετάσεις. Ο εξεταζόμενος, δηλαδή, δεν αντιμετωπίζεται ως ξεχωριστή οντότητα στο σχεδιασμό της βάσης.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΠΙΚΟΙΝΩΝΙΑΣ

Όπως προαναφέρθηκε, το σύστημα miniAtlas αποτελείται από τρεις συνιστώσες: Το πρόγραμμα Server το οποίο τρέχει στον εξυπηρετητή και δέχεται εντολές από τους εξεταστές, το πρόγραμμα Client το οποίο παρουσιάζει τις ερωτήσεις και καταγράφει τις απαντήσεις των εξεταζομένων και το πρόγραμμα Daemon το οποίο τρέχει ως δαίμονας (υπηρεσία) στον σταθμό εξέτασης και μέσω του οποίου πραγματοποιείται η επικοινωνία εξυπηρετητή και σταθμού εξέτασης.

Κατά τη διάρκεια της εξέτασης ο σταθμός εξέτασης λειτουργεί αυτόνομα. Η ανταλλαγή μηνυμάτων μεταξύ του εξυπηρετητή και του σταθμού, αφορά μόνο την κατάσταση της εξέτασης στον συγκεκριμένο σταθμό, δηλαδή αν ο εξεταζόμενος τελείωσε ή όχι. Αυτό σημαίνει πως η διαδικασία δεν θα διακοπεί και οι εξετάσεις δεν θα καταστραφούν ακόμα και αν “πέσει” το δίκτυο για κάποιο λόγο ή αν κολλήσει ο εξυπηρετητής.

Στις παραγράφους που ακολουθούν γίνεται αναφορά στα μηνύματα που ανταλλάσσει το πρόγραμμα Server με το πρόγραμμα Daemon και το πρόγραμμα Client με το πρόγραμμα Daemon. Τέλος αναλύεται, βήμα προς βήμα, η ανταλλαγή μηνυμάτων, μεταξύ όλων των προγραμμάτων, που λαμβάνει χώρα κατά την πραγματοποίηση της διαδικασίας εξέτασης.

Εντολές προς τον σταθμό εξέτασης.

Προκειμένου να κρατηθεί το μοντέλο επικοινωνίας απλό και να μπορεί το σύστημα Atlas να ανανήπτει από αστοχίες υλικού ή λογισμικού που συμβαίνουν στο δίκτυο και στον εξυπηρετητή, μόνο το πρόγραμμα Server στέλνει μηνύματα / εντολές στους σταθμούς εξέτασης, δηλαδή επικοινωνία πραγματοποιείται μόνο με πρωτοβουλία του εξυπηρετητή. Το Server, όταν στέλνει εντολές στους σταθμούς εξέτασης, “μιλάει” με κάθε ένα σταθμό ξεχωριστά, στον ένα μετά από τον άλλο. Με τον τρόπο αυτό δεν επιβαρύνεται το δίκτυο και ένας ασύρματος δρομολογητής χαμηλού κόστους, μπορεί να χρησιμοποιηθεί για να καλύψει τις ανάγκες αίθουσας με 50 ή και περισσότερους σταθμούς εξέτασης.

Το πρόγραμμα Daemon, το οποίο είναι Android service, εκτελείται στο background της Android συσκευής ως εξυπηρετητής της επικοινωνίας (server socket), δέχεται εντολές και τις εκτελεί χωρίς να επιβαρύνει τη συσκευή όταν το Client εκτελεί απαιτητικές λειτουργίες (π.χ. ανανέωση οθόνης). Οι εντολές οι οποίες αποστέλλονται είναι οι ακόλουθες:

- `CommHello (0)`. Αποστέλλεται κάθε φορά που απαιτείται (επαν)έλεγχος της επικοινωνίας. Το πρόγραμμα Server στέλνει τη συμβολοσειρά “Hello” στο Daemon. Το Daemon στέλνει την ίδια συμβολοσειρά πίσω. Αν η συμβολοσειρά δεν είναι η σωστή δημιουργείται εξαίρεση.

- *CommGetFile (1)*. Το Daemon πρέπει να λάβει από το Server ένα αρχείο και να το αποθηκεύσει σε συγκεκριμένη θέση και με συγκεκριμένο όνομα στον σταθμό εξέτασης. Η εντολή ακολουθείται από μία συμβολοσειρά που αντιστοιχεί στο όνομα του καταλόγου στον οποίο θα αποθηκευτεί το αρχείο, μία συμβολοσειρά που αντιστοιχεί στο όνομα του αρχείου και στα περιεχόμενα του αρχείου σε binary μορφή. Αν το αρχείο υπάρχει ήδη, αντικαθίσταται. Όταν ολοκληρωθεί η μεταφορά και η αποθήκευση του αρχείου, το Daemon στέλνει στο Server τη συμβολοσειρά "Ο.Κ.". Με την εντολή *CommGetFile* δεν αποστέλλονται στον σταθμό εξέτασης μόνο τα φύλλα εξέτασης, αλλά και νέες ή διορθωμένες εικόνες και ήχοι.

- *CommSendFile (2)*. Το Daemon πρέπει να στείλει στο Server ένα αρχείο. Ο κατάλογος στο οποίο βρίσκεται το αρχείο αυτό και το όνομά του προσδιορίζονται όπως στην προηγούμενη εντολή.

- *CommDeleteFile (3)*. Το Daemon διαγράφει ένα αρχείο από τους καταλόγους στους οποίους αποθηκεύει τα αρχεία του. Ο προσδιορισμός του αρχείου γίνεται όπως στις προηγούμενες εντολές.

- *CommStartClient (10)*. Το Daemon θα ενεργοποιήσει το πρόγραμμα Client. Μόλις εκτελέσει την εντολή ενεργοποίησης, στέλνει στον εξυπηρετητή τη συμβολοσειρά "Ο.Κ.".

- *CommSendClientComm (11)*. Το Daemon στέλνει στον εξυπηρετητή τον κωδικό κατάστασης του προγράμματος Client. Η κωδικός αυτός σχετίζεται με την επικοινωνία των προγραμμάτων Daemon και Client και θα αναλυθεί σε επόμενη παράγραφο.

- *CommStartExam (12)*. Ο εξυπηρετητής στέλνει εντολή στο πρόγραμμα Client, μέσω του Daemon, να ξεκινήσει τη διεξαγωγή της εξέτασης. Εφόσον ο εξεταζόμενος έχει παρουσιαστεί, το Client θα προχωρήσει στην ExamAct Activity. Σε αντίθετη περίπτωση το Client μπαίνει σε κατάσταση αναμονής έως ότου ολοκληρωθεί η διαδικασία εξέτασης σε όλους του σταθμούς. Η εντολή στέλνεται σε όλους τους σταθμούς εξέτασης (σχεδόν) ταυτόχρονα ώστε να μην δημιουργούνται προβλήματα στην αίθουσα εξετάσεων.

- *CommExamCompleted (13)*. Ο εξυπηρετητής ενημερώνει το πρόγραμμα Client, πάντα μέσω του Daemon, ότι η συνολική διαδικασία ολοκληρώθηκε. Το Client τερματίζεται και ελευθερώνει την οθόνη του σταθμού εξέτασης.

Όταν ο εξεταζόμενος απαντήσεις σε όλες τις ερωτήσεις του ερωτηματολογίου, το Client μπαίνει σε κατάσταση αναμονής μέχρι να ειδοποιηθεί από το Daemon ότι ο εξυπηρετητής έστειλε την εντολή *CommExamCompleted*. Αυτό συμβαίνει για να δοθεί χρόνος στους εξεταστές να μαζέψουν τους σταθμούς εξέτασης χωρίς οι εξεταζόμενοι να "πειράζουν" το λογισμικό που βρίσκεται σε αυτούς. Η εντολή

CommExamCompleted δεν αφορά την διακοπή της εξέτασης σε ένα σταθμό λόγω υπέρβαση του χρόνου εξέτασης, διότι η υπέρβαση αυτή διαπιστώνεται από τον κάθε σταθμό ξεχωριστά και όχι από το πρόγραμμα Server.

Επικοινωνία Daemon και Client.

Το πρόγραμμα Client ενεργοποιείται από το service Daemon, μετά από εντολή του εξυπηρετητή. Το Client, προκειμένου να δέχεται εντολές και να αναφέρει την κατάστασή του, συνδέεται (bind) με το service Daemon. Η επικοινωνία μεταξύ των δύο προγραμμάτων γίνεται μέσω ακεραίων αριθμών. Η σχετική μεταβλητή ονομάζεται *ClientComm* και ορίζεται ως attribute της κλάσης Daemon. Η πρόσβαση σε αυτήν τόσο από το Client όσο και από το ίδιο το Daemon γίνεται μέσω των μεθόδων *public synchronized void SetClientCom (int NewStatus)* και *public synchronized int GetClientCom ()*. Οι μέθοδοι είναι public ώστε να μπορούν να εκτελούνται από κάθε activity το οποίο είναι συνδεδεμένο (bind) με το Daemon και synchronized ώστε μόνο ένα τμήμα κώδικα να μπορεί να έχει πρόσβαση στη μεταβλητή κάθε φορά.

Οι τιμές οι οποίες δέχεται η μεταβλητή *ClientComm* και η σημασία τους είναι οι εξής:

- *ClCommStarted (0)*. Το πρόγραμμα Client ενεργοποιήθηκε. Γράφεται από το Daemon ως αρχικοποίηση της μεταβλητής κάθε φορά που ξεκινά μία νέα εξέταση.
- *ClCommPressedStart (1)*. Γράφεται από την activity *IntroScreen* αν πατηθεί το κουμπί που σημαίνει ότι ο εξεταζόμενος είναι παρόν.
- *ClCommPressedAbsent (2)*. Γράφεται από την activity *IntroScreen* αν πατηθεί το κουμπί που σημαίνει ότι ο εξεταζόμενος απουσιάζει.
- *ClCommRunningTest (3)*. Γράφεται από το Daemon όταν ο εξυπηρετητής στείλει την εντολή *CommStartExam*. Όλοι οι σταθμοί εξέτασης ξεκινούν τα ερωτηματολόγια σχεδόν ταυτόχρονα.
- *ClCommFinished (4)*. Γράφεται από το activity *ExamActiv* όταν ο εξεταζόμενος απαντήσει σε όλες τις ερωτήσεις του ερωτηματολογίου του.
- *ClCommTimeup (5)*. Γράφεται από το activity *ExamActiv* αν εξαντληθεί ο διαθέσιμος χρόνος ολοκλήρωσης της εξέτασης χωρίς ο εξεταζόμενος να προλάβει να απαντήσει σε όλες τις ερωτήσεις.
- *ClCommProcessFinished (6)*. Γράφεται από το Daemon όταν το Server στείλει την εντολή *CommExamCompleted*. Μόλις διαβάσει το Client αυτή την τιμή τερματίζεται.

Από το ζευγάρι `ClCommPressedStart` και `ClCommPressedAbsent` μόνο μία τιμή γράφεται από το πρόγραμμα Client σε κάθε εξέταση υποψηφίου. Το ίδιο συμβαίνει και για το ζευγάρι `ClCommFinished` και `ClCommTimeup`.

Βήματα διαδικασίας εξέτασης

Στις παραγράφους που ακολουθούν περιγράφεται η ανταλλαγή πληροφορίας μεταξύ των τριών συνιστωσών του συστήματος miniAtlas με την οποία διεκπεραιώνεται η διαδικασία εξέτασης. Η ανάλυση της διαδικασίας γίνεται σε βήματα. Η περιγραφή δεν περιλαμβάνει τις ενέργειες των προγραμμάτων οι οποίες δεν σχετίζονται με την επικοινωνία (π.χ. ο εξυπηρετητής δημιουργεί τα ερωτηματολόγια).

- *Βήμα 1*: Το πρόγραμμα Server στέλνει την εντολή `CommHello` στο σταθμό εξέτασης (και παίρνει την απάντηση). Αν η διαδικασία δεν ολοκληρωθεί σωστά, έστω και για ένα σταθμό, η διαδικασία εξέτασης τερματίζεται. Η προς εξέταση ομάδα παραμένει σε κατάσταση “pending” και μπορεί να εξεταστεί μόλις λυθεί το πρόβλημα.

- *Βήμα 2*: Το Server στέλνει την εντολή `CommGetFile` στο Daemon. Το αρχείο το οποίο αφορά η εντολή είναι το φύλλο εξέτασης.

- *Βήμα 3*: Το Server στέλνει την εντολή `CommStartClient`. Το Daemon ενεργοποιεί το πρόγραμμα Client και αρχικοποιεί τη μεταβλητή `ClientComm` με την τιμή `ClCommStarted`.

- *Βήμα 4*: Το πρόγραμμα Client περιμένει να πατηθεί το κουμπί που δηλώνει ότι ο εξεταζόμενος έχει προσέλθει για την εξέταση ή αυτό που δηλώνει ότι ο εξεταζόμενος απουσιάζει. Μόλις πατηθεί ένα από τα δύο, το Client ενημερώνει το Daemon στέλνοντας την τιμή `ClCommPressedStart` ή την τιμή `ClCommPressedAbsent` αντίστοιχα. Παράλληλα, το Server στέλνει σε τακτά χρονικά διαστήματα[29] την εντολή `CommSendClientComm` ώστε να λάβει την τιμή αυτή από τον σταθμό εξέτασης.

- *Βήμα 5*: Το Server, αφού χειριστεί τις απουσίες, στέλνει την εντολή `CommStartExam` στο πρόγραμμα Daemon το οποίο μεταφέρει την εντολή στο Client γράφοντας την τιμή `ClCommRunningTest` στην μεταβλητή `ClientComm`. Στο μεταξύ, το πρόγραμμα Client εξετάζει σε τακτά χρονικά διαστήματα[30] τη τιμή της μεταβλητής. Μόλις λάβει την τιμή `ClCommRunningTest` και αν ο εξεταζόμενος είναι παρών, ενεργοποιείται το activity `ExamActiv` και πραγματοποιείται η εξέταση.

29 Με τις τρέχουσες ρυθμίσεις του συστήματος, η εντολή αποστέλλεται κάθε 20 δευτερόλεπτα.

30 Με τις τρέχουσες ρυθμίσεις κάθε 5 δευτερόλεπτα. Η ταυτόχρονη έναρξη της προβολής ερωτηματολογίων στους σταθμούς εξέτασης σημαίνει, πρακτικά, ότι όλοι οι σταθμοί θα ξεκινήσουν την προβολή σε χρονικό διάστημα 5 δευτερολέπτων. Το χρονικό αυτό διάστημα είναι αρκετά μικρό ώστε να μην δημιουργηθεί κάποιου είδους αναστάτωση στην αίθουσα.

- *Βήμα 6*: Το πρόγραμμα Server περιμένει να τελειώσει η εξέταση στον σταθμό. Αυτό επιτυγχάνεται με την αποστολή, σε τακτά χρονικά διαστήματα, της εντολής `CommSendClientComm` και μέχρι να ληφθεί τιμή η οποία να σημαίνει ότι η διαδικασία τελείωσε, δηλαδή μέχρι να ληφθεί η `ClCommFinished` ή η `ClCommTimeup`. Η `ClCommFinished` στέλνεται κατευθείαν από το πρόγραμμα Client στο Daemon αν ο εξεταζόμενος απουσιάζει. Αν ο εξεταζόμενος είναι παρών τότε στέλνεται μόλις ο εξεταζόμενος απαντήσει σε όλες τις ερωτήσεις. Σε περίπτωση που εξαντληθεί ο χρόνος στέλνεται η τιμή `ClCommTimeup`. Και στις δύο περιπτώσεις ο Client περιμένει την εντολή τερματισμού της συνολικής διαδικασίας μπλοκάροντας τη χρήση της συσκευής από τον εξεταζόμενο.
- *Βήμα 7*: Το Server στέλνει την εντολή `CommSendFile` και λαμβάνει το συμπληρωμένο φύλλο εξέτασης.
- *Βήμα 8*: Το Server στέλνει την εντολή `CommExamCompleted`. Με την λήψη της το Daemon γράφει την τιμή `ClCommProcessFinished` στην μεταβλητή επικοινωνίας. Μόλις το Client λάβει την τιμή τερματίζεται και ελευθερώνει τον σταθμό εξέτασης.

Τα παραπάνω βήματα εκτελούνται για όλους τους σταθμούς εξέτασης. Το Server στέλνει τις εντολές του κάθε βήματος στους σταθμούς εξέτασης σειριακά.

Κεφάλαιο 6

ΧΡΗΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ miniAtlas

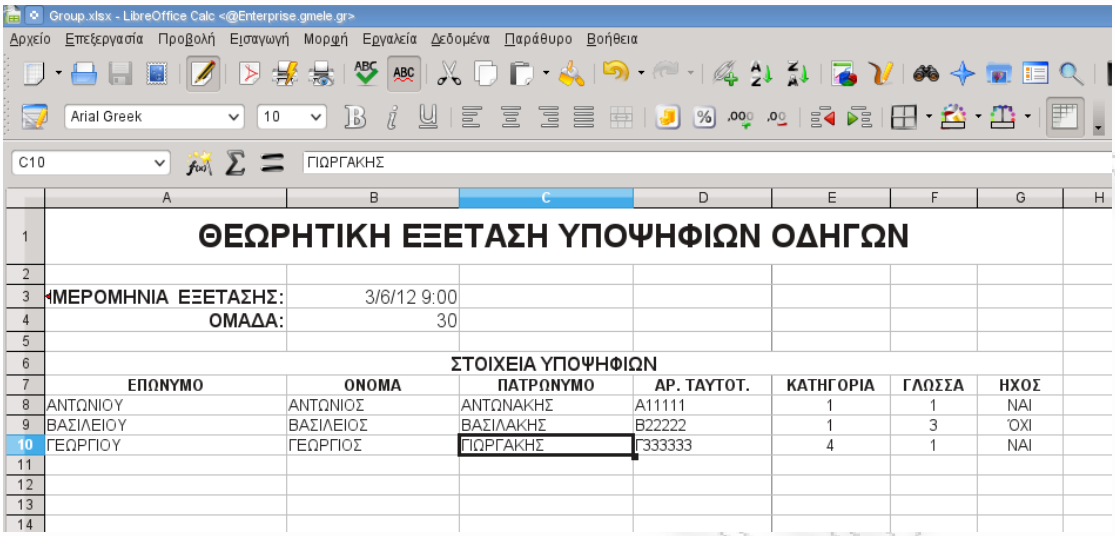
Όπως είναι απαραίτητο για ένα σύστημα διεξαγωγής εξετάσεων του οποίου οι χρήστες, εξεταστές και εξεταζόμενοι, δεν είναι ειδικευμένοι στη χρήση ηλεκτρονικών υπολογιστών, το σύστημα Atlas σχεδιάστηκε να είναι εύκολο στη χρήση του. Το ίδιο ισχύει και για το miniAtlas. Στα υποκεφάλαια που ακολουθούν παρουσιάζεται η χρήση του λογισμικού σε μορφή εγχειριδίου χρήσης.

ΤΟ ΦΥΛΛΟ ΕΙΣΑΓΩΓΗΣ ΟΜΑΔΩΝ ΕΞΕΤΑΣΗΣ

Οι πληροφορίες που αφορούν τις ομάδες εξέτασης εισάγονται στο σύστημα, μέσω του προγράμματος Server, με την χρήση λογιστικών φύλλων τύπου “Microsoft Excel 2007 / 2010 XML”. Ο τύπος αυτός επιλέχτηκε για δύο λόγους:

- Το Microsoft Excel είναι ένα ευρύτατα διαδεδομένο πρόγραμμα και στους περισσότερους υπολογιστές βρίσκεται εγκατεστημένη μία έκδοση του προγράμματος που υποστηρίζει το προαναφερθέντα τύπο αρχείου. Οι περισσότεροι, επίσης, χρήστες προσωπικών υπολογιστών έχουν εμπειρία βασικής, έστω, χρήσης του προγράμματος.
- Ο “Microsoft Excel 2007 / 2010 XML” τύπος αρχείου αποτελεί πλέον παγκόσμιο πρότυπο. Ως τέτοιο υποστηρίζεται χωρίς προβλήματα και από αντίστοιχα προγράμματα ελεύθερου λογισμικού όπως οι σουίτες LibreOffice και OpenOffice.

Η εικόνα (screenshot) που ακολουθεί είναι από το πρόγραμμα LibreOffice Calc το οποίο τρέχει κάτω από το λειτουργικό σύστημα GNU / Linux.



	A	B	C	D	E	F	G	H
1	ΘΕΩΡΗΤΙΚΗ ΕΞΕΤΑΣΗ ΥΠΟΨΗΦΙΩΝ ΟΔΗΓΩΝ							
2								
3	ΗΜΕΡΟΜΗΝΙΑ ΕΞΕΤΑΣΗΣ:	3/6/12 9:00						
4	ΟΜΑΔΑ:	30						
5								
6	ΣΤΟΙΧΕΙΑ ΥΠΟΨΗΦΙΩΝ							
7	ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΠΑΤΡΩΝΥΜΟ	ΑΡ. ΤΑΥΤΟΤ.	ΚΑΤΗΓΟΡΙΑ	ΓΛΩΣΣΑ	ΗΧΟΣ	
8	ΑΝΤΩΝΙΟΥ	ΑΝΤΩΝΙΟΣ	ΑΝΤΩΝΑΚΗΣ	Α11111	1	1	ΝΑΙ	
9	ΒΑΣΙΛΕΙΟΥ	ΒΑΣΙΛΕΙΟΣ	ΒΑΣΙΛΑΚΗΣ	Β22222	1	3	ΟΧΙ	
10	ΓΕΩΡΓΙΟΥ	ΓΕΩΡΓΙΟΣ	ΓΙΩΡΓΑΚΗΣ	Γ333333	4	1	ΝΑΙ	
11								
12								
13								
14								

Εικόνα 6.1

Προκειμένου να δημιουργήσει μία νέα ομάδα εξέτασης, ο εξεταστής συμπληρώνει το προηγούμενο φύλλο, το οποίο χρησιμοποιείται ως φόρμα, χωρίς να τροποποιήσει την γραμμογράφησή του, σύμφωνα με τις οδηγίες που ακολουθούν.

- Στο κελί B3 εισάγεται η προγραμματισμένη ημερομηνία ώρα διεξαγωγής της εξέτασης. Το στοιχείο θα πρέπει να γραφεί με έναν από τους τρόπους οι οποίοι θα του επιτρέπουν να αναγνωριστεί ως πεδίο ημερομηνίας. Η ημερομηνία πρέπει να είναι ακριβής διότι η εξέταση μίας ομάδας ξεκινά από το Server μόνο αν είναι προγραμματισμένη για την τρέχουσα ημερομηνία, σύμφωνα με το ρολόι του εξυπηρετητή. Αντίθετα, η ώρα έναρξης χρησιμοποιείται μόνο στην εκτύπωση των αποτελεσμάτων και δεν λαμβάνεται υπόψη για την έναρξη της διαδικασίας.

- Στο κελί B4 καταγράφεται ένας αριθμός ο οποίος προσδιορίζει την ομάδα εξέτασης για την συγκεκριμένη ημερομηνία. Για οργανωτικούς λόγους συστήνεται να είναι ο Αύξοντας Αριθμός της ομάδας. Τεχνικά, μπορεί να είναι οποιοσδήποτε θετικός ακέραιος αριθμός. Δύο ομάδες προγραμματισμένες να εξεταστούν την ίδια ημέρα δεν μπορούν να έχουν τον ίδιο αριθμό ομάδας.

- Η καταχώρηση των δεδομένων που αφορούν τους εξεταζόμενους εισάγονται από τη γραμμή 8 και κάτω. Η πρώτη κενή γραμμή ορίζει το τέλος της λίστας. Όλα τα πεδία πρέπει να συμπληρώνονται.

- Στην στήλη A γράφονται τα επώνυμα των υποψηφίων.

- Στην στήλη B γράφονται τα ονόματα των υποψηφίων.

- Στην στήλη C γράφονται τα πατρώνυμα των υποψηφίων.

- Στην στήλη D γράφονται οι αριθμοί που ταυτοποιούν μοναδικά το έγγραφο το οποίο, με τη σειρά του, ταυτοποιεί τον υποψήφιο (π.χ. αριθμός δελτίου αστυνομικής ταυτότητας ή αριθμός διαβατηρίου)
- Στην στήλη E εισάγεται ο αριθμός της κατηγορίας εξέτασης. Για το miniAtlas, ο αριθμός 1 αντιστοιχεί στην κατηγορία του κώδικα, ο αριθμός 2 στην κατηγορία των μοτοσυκλετών, ο αριθμός 3 στην κατηγορία των φορτηγών οχημάτων και ο αριθμός 4 στην κατηγορία των λεωφορείων.
- Στην στήλη F εισάγεται ο κωδικός της γλώσσας εξέτασης. Οι κωδικοί που ισχύουν στο miniAtlas είναι: 1 για Ελληνικά, 2 για Αγγλικά, 3 για Ρωσικά και 4 για Αλβανικά.
- Στην στήλη G εισάγεται η λέξη "ΝΑΙ" (κεφαλαία ελληνικά) αν ο υποψήφιος είναι αναλφάβητος και το σύστημα θα πρέπει να εκφωνεί τις ερωτήσεις και τις απαντήσεις. Σε αντίθετη περίπτωση πρέπει να εισαχθεί η λέξη "ΌΧΙ". Η εκφώνηση των κειμένων, στο σύστημα miniAtlas, είναι διαθέσιμη μόνο στην ελληνική γλώσσα.

Το φύλλο εισαγωγής ομάδων εξέτασης δεν είναι απαραίτητο να είναι αποθηκευμένο στον σκληρό δίσκο του εξυπηρετητή.

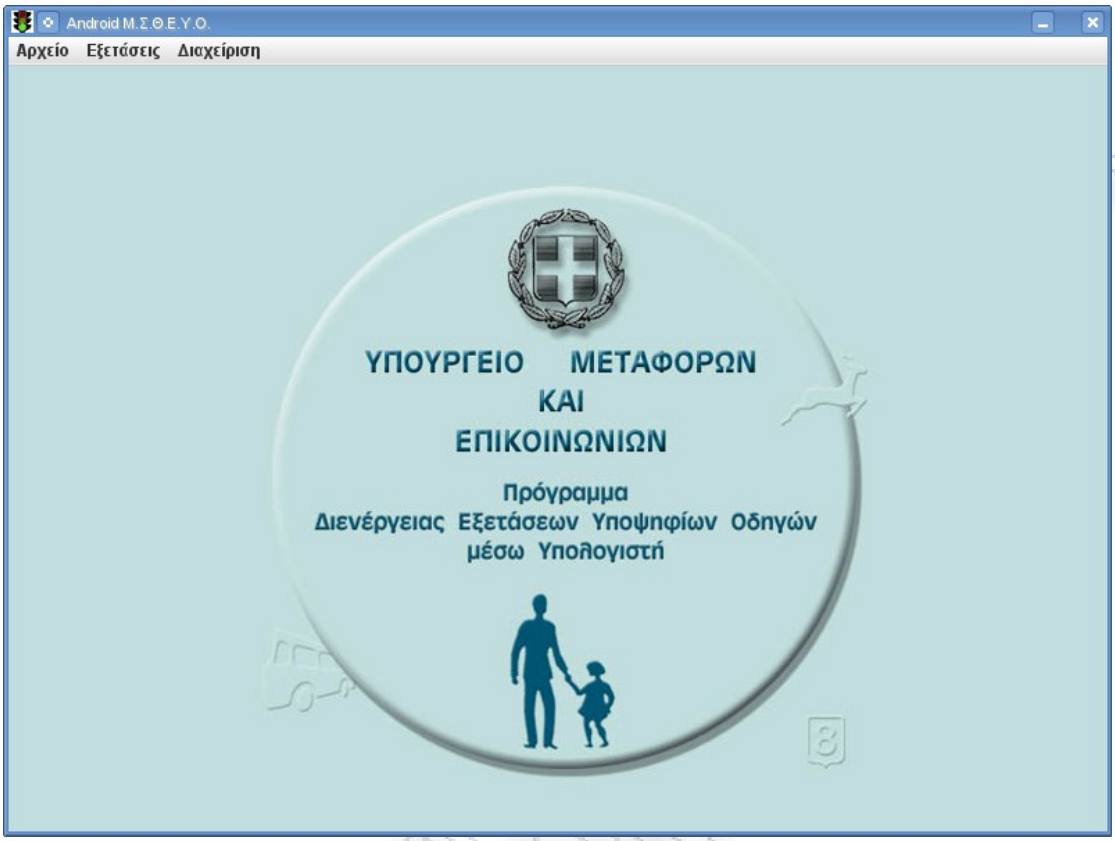
ΤΟ ΠΡΟΓΡΑΜΜΑ SERVER

Το σύστημα miniAtlas έχει γραφεί για να επιδείξει τις βασικές λειτουργίες ενός συστήματος εξετάσεων με τη χρήση Android σταθμών εξέτασης. Για το λόγο αυτό το πρόγραμμα Server, το οποίο υλοποιεί τις λειτουργίες του εξυπηρετητή, εκτελεί μόνο τις βασικές λειτουργίες του συστήματος. Όπως φαίνεται και στην εικόνα[31] 6.2, οι λειτουργίες του είναι κατανεμημένες σε τρία Μενού:

- Το μενού "Αρχείο" το οποίο περιέχει την επιλογή τερματισμού του προγράμματος
- Το μενού "Εξετάσεις" το οποίο περιέχει τις επιλογές που σχετίζονται με την διαχείριση και την εκτέλεση των εξετάσεων
- Το μενού Διαχείριση το οποίο περιέχει την επιλογή διαχείρισης των σταθμών εξέτασης.

Στις παραγράφους που ακολουθούν περιγράφεται συνοπτικά η λειτουργία των επιλογών του προγράμματος Server.

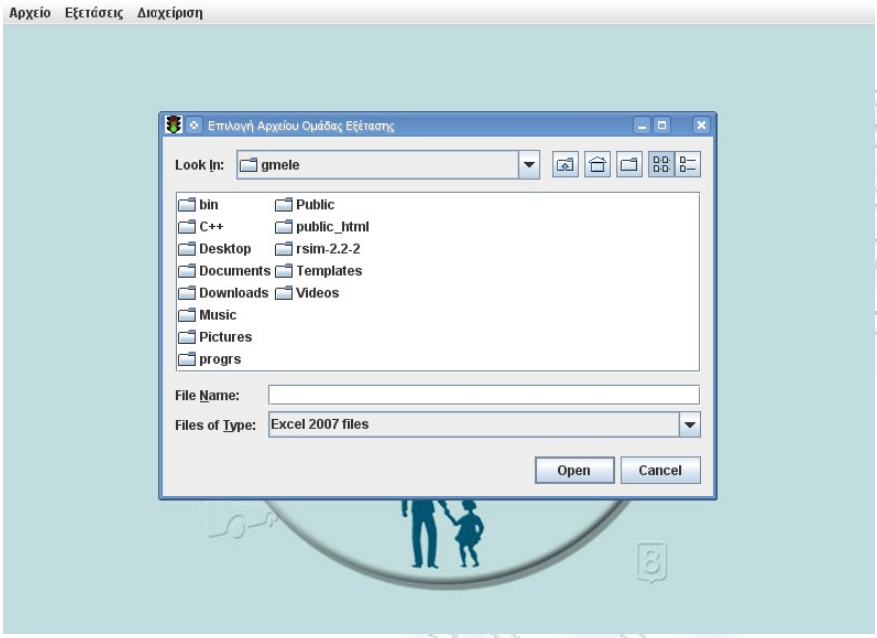
31 Οι εικόνες αποτελούν screenshots τα οποία έχουν τραβηχτεί από ένα Linux Server.



Εικόνα 6.2

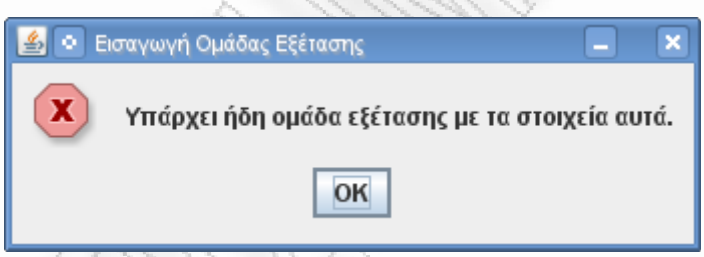
Εισαγωγή Ομάδας

Η επιλογή “Εισαγωγή Ομάδας” του μενού “Εξετάσεις” οδηγεί τον χρήστη σε διάλογο (dialog) επιλογής αρχείου με επέκταση “.xlsx” (εικόνα 6.3). Ο χρήστης επιλέγει το αρχείο και η ομάδα η οποία περιγράφεται σε αυτό εισάγεται στο σύστημα.



Εικόνα 6.3

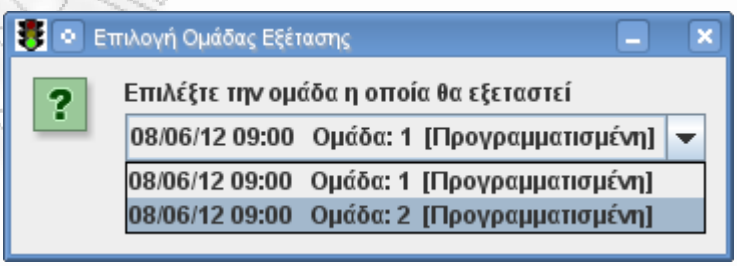
Σε περίπτωση που η ομάδα δεν μπορεί να εισαχθεί, για παράδειγμα αν η ομάδα έχει ήδη εισαχθεί, ο χρήστης ενημερώνεται σχετικά.



Εικόνα 6.4

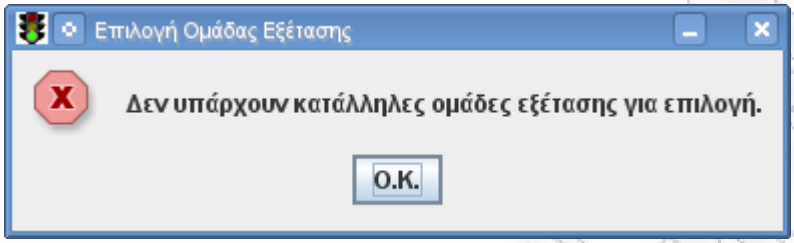
Διενέργεια Εξέτασης

Η επιλογή "Διενέργεια Εξέτασης" ενεργοποιεί την διαδικασία εξέτασης μίας ομάδας. Ο εξεταστής καλείται να επιλέξει την ομάδα η οποία θα εξεταστεί. Μόνο ομάδες οι οποίες έχουν προγραμματιστεί για την συγκεκριμένη ημερομηνία θα εμφανιστούν



Εικόνα 6.5

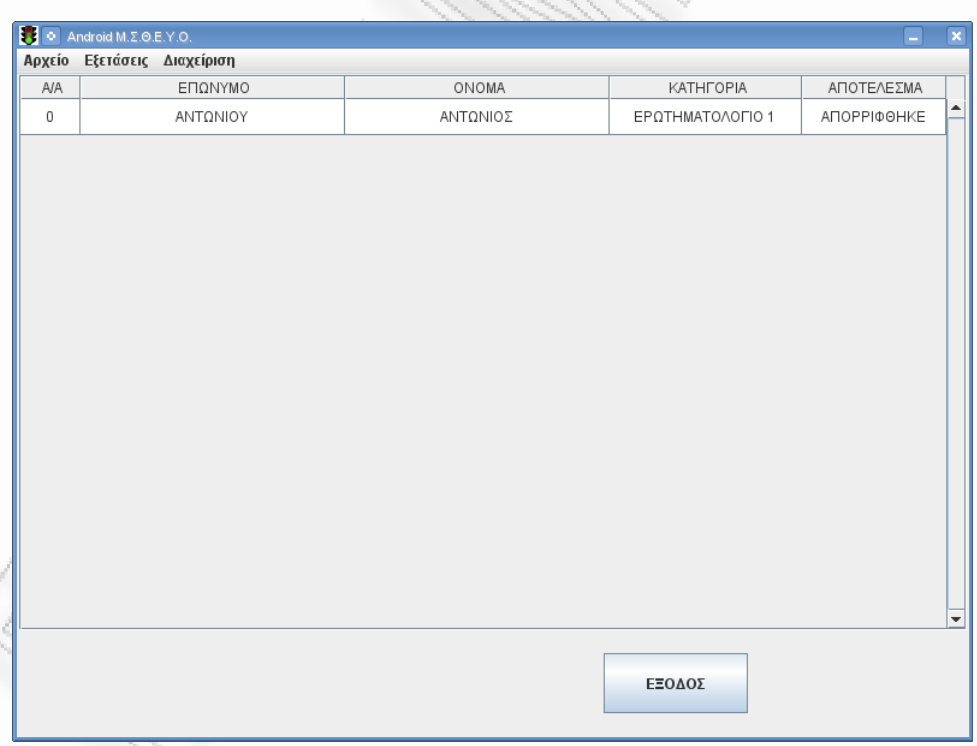
Σε περίπτωση που δεν υπάρχουν διαθέσιμες ομάδες για εξέταση εμφανίζεται σχετικό μήνυμα.



Εικόνα 6.6

Στην συνέχεια το πρόγραμμα Server κλειδώνει μέχρι να ολοκληρωθεί η εξέταση. Από το σημείο αυτό και μέχρι να βγουν τα αποτελέσματα δεν απαιτείται καμία παρέμβαση από τον χρήστη του.

Όταν ολοκληρωθεί η διαδικασία το πρόγραμμα εμφανίζει το παράθυρο παρουσίασης αποτελεσμάτων. Το κουμπί το οποίο δίνει στο χρήστη του προγράμματος να εκτυπώσει τα αποτελέσματα απουσιάζει διότι σε αυτή την περίπτωση τα αποτελέσματα τυπώνονται υποχρεωτικά και άρα αυτόματα.

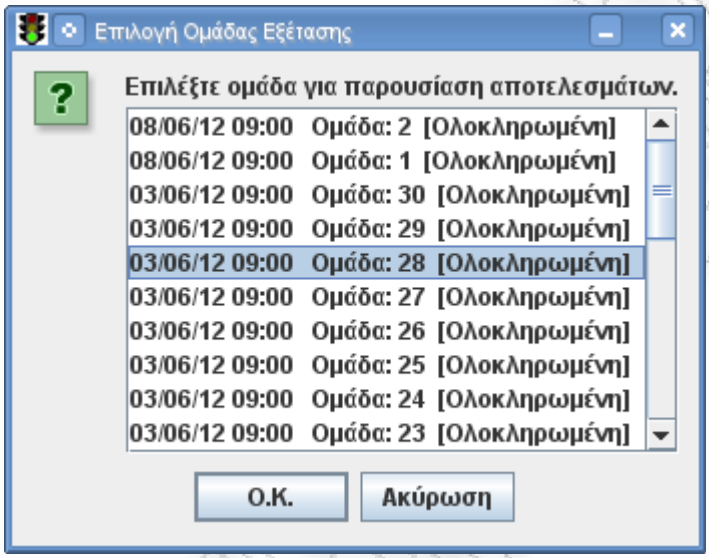


Εικόνα 6.7

Όταν ο χρήστης πατήσει το κουμπί “ΕΞΟΔΟΣ” το πρόγραμμα επιστρέφει στην αρχική του οθόνη και στην αρχική κατάσταση.

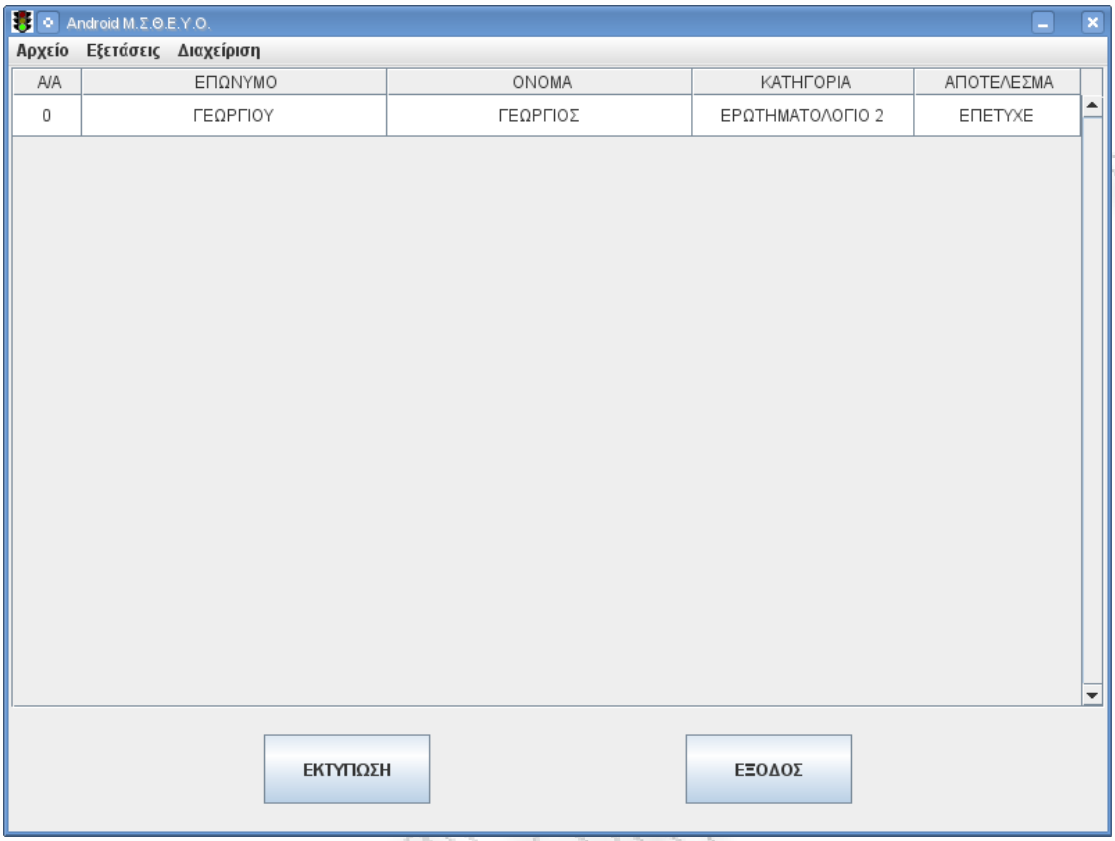
Αποτελέσματα Εξέτασης

Με την επιλογή “Αποτελέσματα Εξέτασης” ο χρήστης του εξυπηρετητή έχει τη δυνατότητα να ανακτήσει τα αποτελέσματα μίας εξέτασης η οποία ολοκληρώθηκε στο παρελθόν. Αρχικά επιλέγει μία ομάδα από την λίστα ολοκληρωμένων ομάδων εξέτασης. Οι ομάδες εμφανίζονται κατά φθίνουσα τάξη ημερομηνίας διεξαγωγής, δηλαδή οι πλέον πρόσφατες εμφανίζονται πρώτες.



Εικόνα 6.8

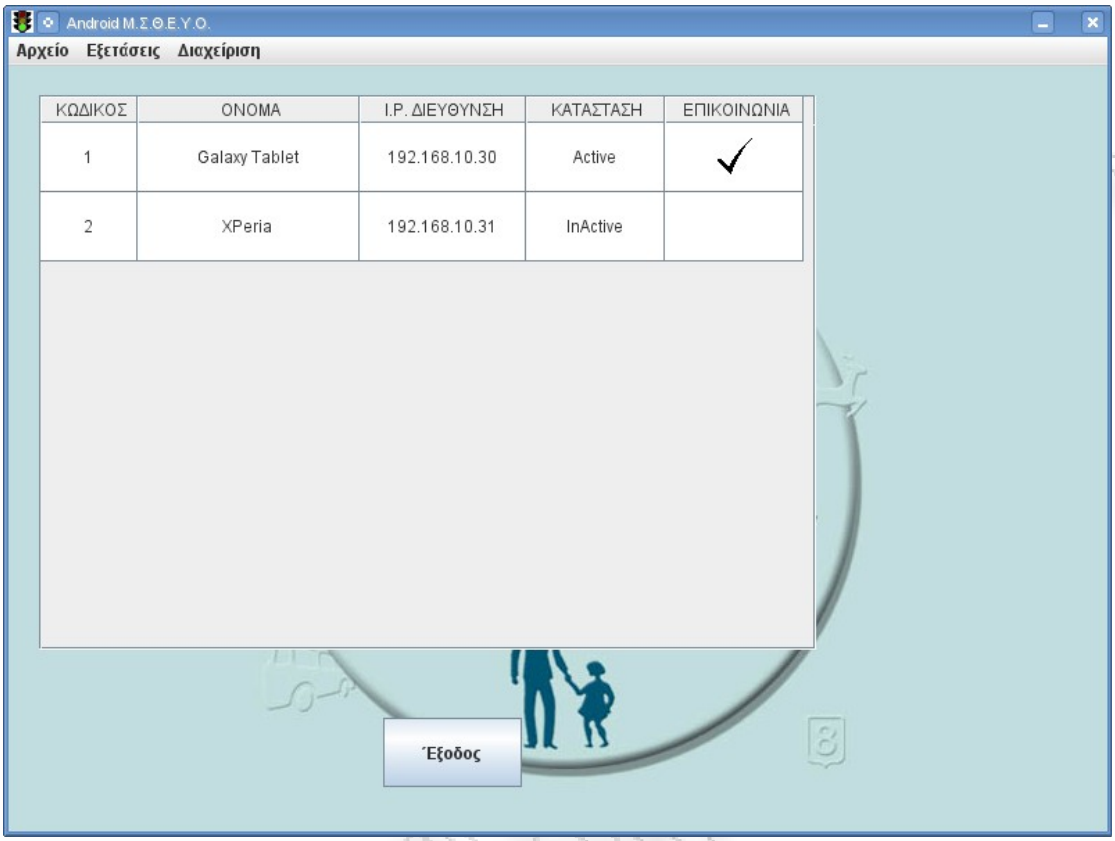
Στην συνέχεια εμφανίζεται το παράθυρο παρουσίασης αποτελεσμάτων. Αυτή τη φορά το παράθυρο περιλαμβάνει το κουμπί “ΕΚΤΥΠΩΣΗ” με τη χρήση του οποίου ο εξεταστής μπορεί να τυπώσει, ξανά, τα πρακτικά της επιλεγμένης εξέτασης.



Εικόνα 6.9

Σταθμοί Εξέτασης

Η επιλογή “Σταθμοί Εξέτασης” που βρίσκεται στο μενού “Διαχείριση” εμφανίζει ένα παράθυρο μέσω του οποίου ο χειριστής του συστήματος μπορεί να διαχειριστεί τις πληροφορίες που σχετίζονται με τους σταθμούς εξέτασης.



Εικόνα 6.10

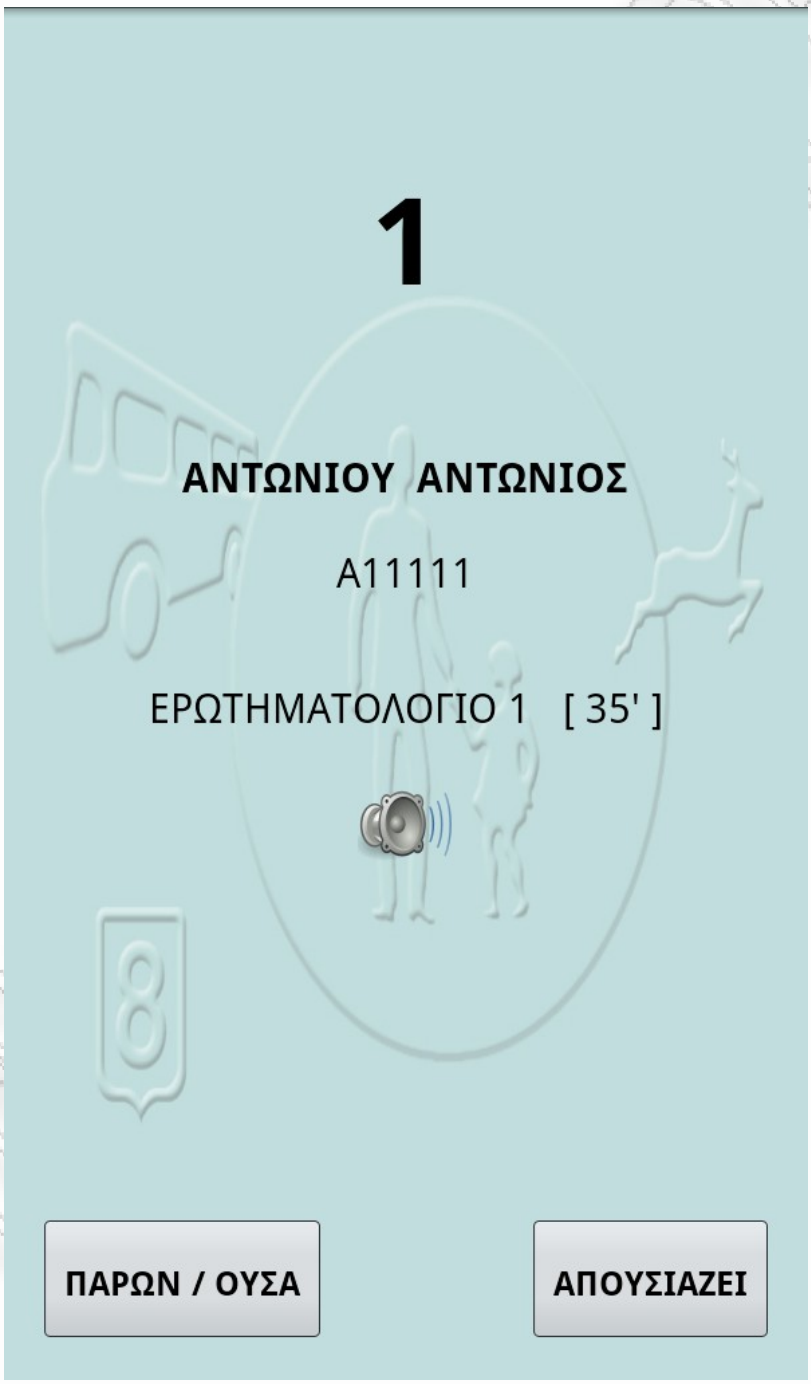
Στο παράθυρο παρουσιάζονται οι εξής πληροφορίες:

- Ο Αύξοντας αριθμός του σταθμού εκπαίδευσης.
- Το όνομα του σταθμού.
- Η IP διεύθυνση του σταθμού εξέτασης
- Η κατάσταση του σταθμού εξέτασης (ενεργός, απενεργοποιημένος, χαλασμένος)
- Το αποτέλεσμα του ελέγχου επικοινωνίας

Πριν ενεργοποιηθεί το παράθυρο, το πρόγραμμα ελέγχει την επικοινωνία με κάθε σταθμό εξέτασης (εντολή `CommHello`) ο οποίος βρίσκεται σε κατάσταση Active. Το αποτέλεσμα του ελέγχου παρουσιάζεται στο παράθυρο όπως φαίνεται και στην προηγούμενη εικόνα.

ΤΟ ΠΡΟΓΡΑΜΜΑ CLIENT

Το πρόγραμμα Client ενεργοποιείται αυτόματα στον σταθμό εξέτασης μόλις ξεκινήσει η διαδικασία εξέτασης στον εξυπηρετητή. Το πρόγραμμα με την εισαγωγική οθόνη η οποία παρουσιάζει τα στοιχεία της εξέτασης. Στην εικόνα που ακολουθεί φαίνεται ένα τυπικό δείγμα εισαγωγικής οθόνης.



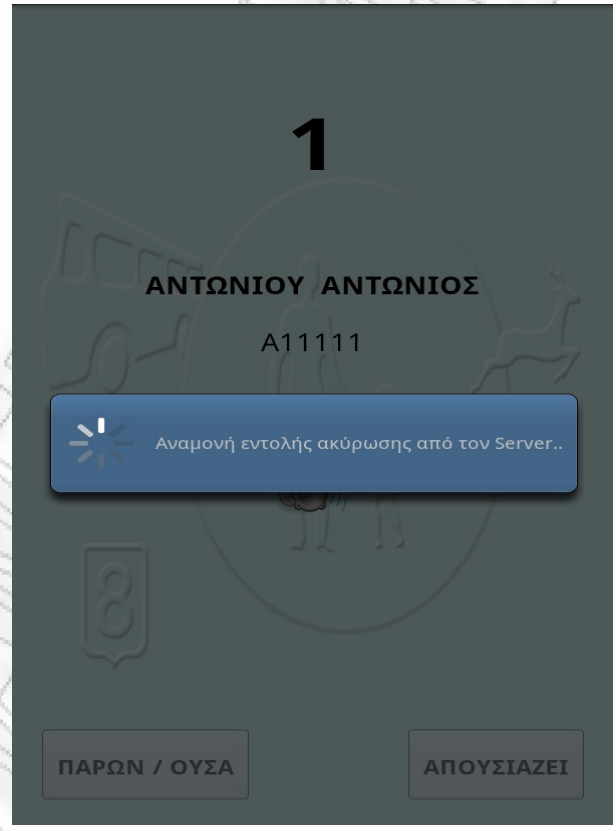
Εικόνα 6.11

Η εισαγωγική οθόνη καταλαμβάνει το σύνολο της οθόνης του σταθμού εξέτασης. Η γραμμή ειδοποιήσεων (notification bar) είναι απενεργοποιημένη. Απενεργοποιημένο είναι, επίσης, το κουμπί επιστροφής στην προηγούμενη activity (back button).

Στο πάνω μέρος της εισαγωγικής οθόνης φαίνεται ο αριθμός του σταθμού εξέτασης ώστε οι εξεταστές να ξέρουν ποια συσκευή δίνουν σε ποιον εξεταζόμενο. Στην συνέχεια αναγράφεται το ονοματεπώνυμο του εξεταζομένου και κατόπιν ο αριθμός του εγγράφου με το οποίο ταυτοποιείται. Ακολουθεί η κατηγορία εξέτασης και ο χρόνος, σε πρώτα λεπτά της ώρας, που έχει στη διάθεσή του ο εξεταζόμενος για να απαντήσει σε όλες τις ερωτήσεις. Ακριβώς από κάτω υπάρχει ένα εικονίδιο το οποίο δείχνει αν η εκφώνηση είναι ενεργοποιημένη. Σε αυτή την περίπτωση ο εξεταστής πρέπει να δώσει και φορητά ακουστικά (hands free) στον εξεταζόμενο.

Στο κάτω μέρος της εισαγωγικής οθόνης υπάρχουν τα κουμπιά με τα οποία δηλώνεται αν ο εξεταζόμενος είναι παρών ή απών.

Αν ο εξεταζόμενος απουσιάζει η οθόνη κλειδώνεται μέχρι ο εξυπηρετητής να σημάνει το συνολικό τέλος της διαδικασίας.



Εικόνα 6.12

Αν ο εξεταζόμενος είναι παρών, μόλις δοθεί το σήμα από τον εξυπηρετητή, παρουσιάζεται η οθόνη στην οποία παρουσιάζονται οι ερωτήσεις και

καταγράφονται οι απαντήσεις του εξεταζόμενου. Η οθόνη διεξαγωγής της εξέτασης έχει ως ακολούθως:

Εικόνα 6.13

Στο πάνω μέρος της οθόνης αναγράφεται ο αριθμός της ερώτησης. Ακριβώς από κάτω φαίνεται η εικόνα που σχετίζεται με την ερώτηση, εφόσον υπάρχει τέτοια. Ακολουθεί το κείμενο της ερώτησης. Το κείμενο της ερώτησης, για να ξεχωρίζει, είναι γραμμένο με πλάγια (italics) γραμματοσειρά. Στην συνέχεια φαίνονται οι διαθέσιμες απαντήσεις[32]. Αν ο διαθέσιμος χώρος δεν φτάνει για να παρουσιαστεί το

32 Οι διαθέσιμες απαντήσεις στο σύστημα miniAtlas είναι από 2 ως 5 ανάλογα με την ερώτηση και την

σύνολο των απαντήσεων “σκρολάρει” με μία απλή κίνηση του χρήστη. Στο κάτω μέρος της οθόνης φαίνεται με γραφικό τρόπο ο διαθέσιμος χρόνος για την ολοκλήρωση της εξέτασης το κουμπί “ΕΠΟΜΕΝΗ” με το οποίο ο εξεταζόμενος μπορεί να αφήσει μία ερώτηση για αργότερα και το κουμπί “ΕΠΙΒΕΒΑΙΩΣΗ” με το οποίο ο εξεταζόμενος επιβεβαιώνει την απάντησή του.

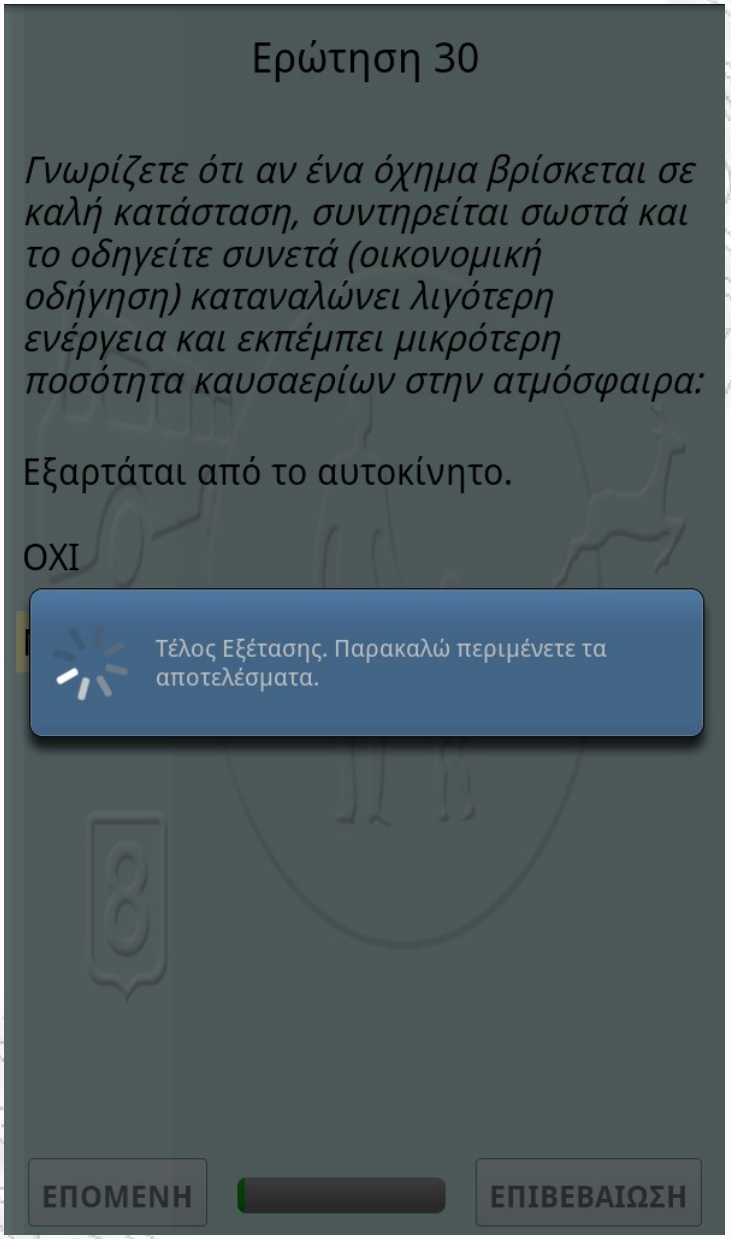
Το κουμπί “ΕΠΙΒΕΒΑΙΩΣΗ” δεν ενεργοποιείται παρά μόνο όταν ο εξεταζόμενος επιλέξει απάντηση. Για να επιλέξει απάντηση, ο εξεταζόμενος ακουμπάει το δάκτυλό του στο κείμενο της απάντησης την οποία θεωρεί σωστή. Το φόντο της απάντησης αλλάζει ώστε η επιλογή του εξεταζόμενου να ξεχωρίζει. Ο χρήστης μπορεί να αλλάξει την επιλογή του όσες φορές θέλει μέχρι να την επιβεβαιώσει.

Εικόνα 6.14

Μόλις ο εξεταζόμενος πατήσει κάποιο από τα δύο κουμπιά, το σύστημα προχωράει στην επόμενη ερώτηση η οποία δεν έχει ήδη απαντηθεί. Όταν επιβεβαιωθεί η απάντηση μίας ερώτησης, αυτή δεν ξαναπαρουσιάζεται. Όταν απαντηθούν όλες οι

κατηγορία στην οποία ανήκει.

ερωτήσεις ή εξαντληθεί ο διαθέσιμος χρόνος, το πρόγραμμα Client κλειδώνει την οθόνη. Το κλείδωμα της οθόνης διατηρείται μέχρι ο εξυπηρετητής να σημάνει το οριστικό τέλος της διαδικασίας εξέτασης.



Εικόνα 6.15

Αν η εκφώνηση είναι ενεργοποιημένη τότε με το που παρουσιάζεται μία ερώτηση το Client αρχίζει την εκφώνησή της. Κάθε φορά που ο εξεταζόμενος ακουμπάει μία απάντηση, το σύστημα αρχίζει την εκφώνησή της. Ο εξεταζόμενος μπορεί να ξαναακούσει την ερώτηση αν ακουμπήσει το κείμενό της. Η έναρξη της εκφώνησης της ερώτησης ή κάποιας απάντησης, διακόπτει την εκφώνηση προηγούμενου κειμένου η οποία πιθανόν δεν έχει ολοκληρωθεί.

Κεφάλαιο 7

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στα προηγούμενα κεφάλαια αναλύθηκε ένα πληροφοριακό σύστημα διεξαγωγής επίσημων εξετάσεων για τον ιδιωτικό και τον δημόσιο τομέα, το Μ.Σ.Θ.Ε.Υ.Ο., μελετήθηκαν οι αδυναμίες του και προτάθηκαν διορθώσεις και επεκτάσεις που καλύπτουν τις αδυναμίες αυτές.

Η ενοποίηση των λύσεων αυτών οδηγεί σε ένα νέο πληροφοριακό σύστημα: το σύστημα Atlas. Όπως περιγράφηκε, τόσο από πλευράς υλικού όσο και από πλευράς λογισμικού, το σύστημα Atlas παρουσιάζει τα εξής πλεονεκτήματα:

- Ευκολία χρήσης για όλα τα εμπλεκόμενα μέρη (εξεταστές και εξεταζόμενους).
- Ασφάλεια και αξιοπιστία.
- Χαμηλό κόστος ανάπτυξης και λειτουργίας.
- Εύκολη ενοποίηση με άλλα συστήματα
- Προσαρμοστικότητα στις ειδικές ανάγκες του κάθε οργανισμού που το υιοθετεί και ευκολία επέκτασης του.

Ειδικά σε ό,τι αφορά τον παράγοντα κόστος, ο οποίος σήμερα αποτελεί το πλέον βασικό κριτήριο επιλογής ή όχι ενός συστήματος, θα πρέπει να τονιστούν τα ακόλουθα πλεονεκτήματα του νέου συστήματος.

- Μπορεί να υλοποιηθεί με απόλυτα φορητό εξοπλισμό ώστε να μην δεσμεύει πολύτιμο οικονομικά χώρο (αίθουσα).
- Δεν έχει υψηλές ενεργειακές ανάγκες λειτουργίας.
- Κατάλληλοι σταθμοί εξέτασης (Android tablets), τουλάχιστον αποδεκτής ποιότητας, μπορούν να αποκτηθούν με λιανικό κόστος το οποίο σήμερα[33] δεν ξεπερνά τα 150 Ευρώ ανά σταθμό. Η τάση στην παγκόσμια αγορά είναι οι τιμές, σε αυτή την κατηγορία εξοπλισμού, να μειώνονται.

33 Ιούνιος 2012

- Ο εξοπλισμός, με πιθανή εξαίρεση τον εξυπηρετητή, δεν έχει ανάγκη συμβολαίων συντήρησης.
- Ο ιδιοκτήτης του μπορεί να το ενοικιάσει, είτε ως εξοπλισμό είτε ως συνολική υπηρεσία, σε άλλους οργανισμούς και να έχει εισοδήματα από αυτό.

Εκτός από οικονομικά εφικτό, για να είναι "εφαρμόσιμο", το σύστημα θα πρέπει να γίνει αποδεκτό από τους χρήστες του. Η βασικότερη τροχοπέδη στην προσπάθεια αυτή θα τοποθετούνταν από το ίδιο αν αυτό ήταν δύσχρηστο ή απαιτούσε γνώσεις χρήσης ηλεκτρονικών υπολογιστών ενώ απευθύνεται και σε κατηγορίες εξεταζομένων οι οποίοι δεν τις διαθέτουν (π.χ. ηλικιωμένοι υποψήφιοι οδηγοί). Το σύστημα miniAtlas αναπτύχθηκε, μεταξύ άλλων, για να αποδείξει ότι το προτεινόμενο σύστημα είναι απλό στη χρήση του διότι απαιτεί:

- Ελάχιστη εμπειρία χρήσης ηλεκτρονικών υπολογιστών από τον εξεταστή (π.χ. επιλογή λειτουργιών με τη χρήση ποντικιού, στοιχειώδης δεξιότητες πληκτρολόγησης, χρήση εκτυπωτή). Οι δεξιότητες αυτές δεν είναι σπάνιες σε προσωπικό το οποίο διενεργεί, σήμερα, όλως των ειδών τις εξετάσεις και όπου δεν υπάρχουν μπορεί εύκολα να αποκτηθούν με την οργάνωση μικρών παρουσιάσεων ή σεμιναρίων.
- Μηδενική γνώση χρήσης ηλεκτρονικών υπολογιστών από τους εξεταζόμενους αν ο σταθμός εξέτασης είναι Android συσκευή.
- Βασικές γνώσεις χρήσης ηλεκτρονικών υπολογιστών, όπως η σύνδεση καλωδίων και η δημιουργία εφεδρικών αντιγράφων, για τους υπεύθυνους λειτουργίας του συστήματος.

Τέλος, πρέπει να τονιστεί ότι, ένα σύστημα διεξαγωγής εξετάσεων το οποίο εφαρμόζεται για να πιστοποιεί κρίσιμες κοινωνικά γνώσεις (π.χ. επαγγελματικά προσόντα) ή σε κρίσιμους τομείς της καθημερινότητας του πολίτη όπου είναι έντονη η "αίσθηση της διαφθοράς", πρέπει όχι μόνο να είναι ασφαλές και αξιόπιστο, αλλά να το επιδεικνύει σε κάθε ευκαιρία. Αυτό σημαίνει πως πρέπει να επιδεικνύει την δύναμή του απέναντι στον πιο αδύναμο, θεωρητικά, κρίκο της αξιοπιστίας του: τον χειριστή του. Το σύστημα Atlas είναι ένα τέτοιο σύστημα.

Κλείνοντας την παρούσα εργασία θα ήθελα να εκφράσω την ελπίδα μου, το σύστημα Atlas, να αναπτυχθεί στο σύνολό του και να αντικαταστήσει, όπου είναι δυνατό, υπάρχουσες χειρόγραφες διαδικασίες ή άλλα παλαιότερα μηχανογραφικά συστήματα. Είμαι στη διάθεση οποιουδήποτε δημόσιου φορέα (π.χ. ενός ΑΕΙ), ο οποίος θα ήθελε να το υλοποιήσει για να καλύψει τις ανάγκες του, ή για να κερδίσει πόρους από αυτό, ή απλά να το διαθέσει στην κοινωνία ως ελεύθερο λογισμικό.

ΠΑΡΑΡΤΗΜΑ Α

ΕΞΑΓΩΓΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΤΟΥ miniAtlas

```
-- phpMyAdmin SQL Dump
-- version 3.1.2
-- http://www.phpmyadmin.net
--
-- Σύστημα: localhost
-- Χρόνος δημιουργίας: 11 Ιούν 2012, στις 06:51 PM
-- Έκδοση Διακομιστή: 5.5.23
-- Έκδοση PHP: 5.3.13

SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";



/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Βάση: `ExamDis`
--

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Answers`
--

CREATE TABLE IF NOT EXISTS `Answers` (
  `AQCod` int(10) unsigned NOT NULL COMMENT 'Κωδικός Ερώτησης',
  `AAA` smallint(5) unsigned NOT NULL COMMENT 'A/A Απάντησης',
  `ALect` varchar(500) NOT NULL COMMENT 'Λεκτικό Απάντησης',
  `ACorr` tinyint(1) NOT NULL COMMENT 'Είναι σωστή απάντηση',
  `ASound` varchar(20) NOT NULL COMMENT 'Αρχείο Ήχου',
  KEY `AQCod` (`AQCod`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Πίνακας Απαντήσεων';

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Clients`
--

CREATE TABLE IF NOT EXISTS `Clients` (
```

```
  `ClID` mediumint(8) unsigned NOT NULL AUTO_INCREMENT COMMENT 'Κωδικός
Τερματικού Εξέτασης',
  `ClName` varchar(20) NOT NULL COMMENT 'Λεκτικό Τερματικού Εξέτασης',
  `ClIP` varchar(15) NOT NULL COMMENT 'IP Σταθμού Εξέτασης',
  `ClStatus` enum('Active','InActive','Damaged') NOT NULL COMMENT 'Κατάσταση
Σταθμού Εξέτασης ',
  PRIMARY KEY (`ClID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT='Πληροφορίες Σταθμών Εξέτασης
(Android Tablets)' AUTO_INCREMENT=4 ;

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Examinations`
--

CREATE TABLE IF NOT EXISTS `Examinations` (
  `ExamID` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT 'Κωδικός
Εξέτασης',
  `ExamGroup` mediumint(8) unsigned NOT NULL COMMENT 'Ομάδα Εξέτασης',
  `ExamDate` datetime NOT NULL COMMENT 'Προγραμματισμένη Ημερομηνία
Εξέτασης',
  `ExamStatus` enum('Pending','Active','Done','Aborted','Cancelled') NOT
NULL COMMENT 'Κατάσταση Εξέτασης (Abort = Ακύρωση κατά τη διάρκεια, Cancel =
Ακύρωση από πριν)',
  PRIMARY KEY (`ExamID`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT='Πίνακας Προγραμματισμένων
Εξετάσεων' AUTO_INCREMENT=45 ;

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Kateg`
--

CREATE TABLE IF NOT EXISTS `Kateg` (
  `KCod` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Κατηγορίας
Εξέτασης',
  `KLect` varchar(50) NOT NULL COMMENT 'Λεκτικό κατηγορίας Εξέτασης',
  `KTime` mediumint(8) unsigned NOT NULL COMMENT 'Χρόνος Εξέτασης σε πρώτα
λεπτά της ώρας',
  `AllowedErrors` mediumint(8) unsigned NOT NULL COMMENT 'Επιτρεπτός Αριθμός
Λαθών',
  `KPic` varchar(30) NOT NULL COMMENT 'Όνομα Φωτογραφίας για αρχική οθόνη',
  PRIMARY KEY (`KCod`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Κατηγορίες Εξέτασης
(Αυτοκίνητα, φορτηγά, κλπ)';

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Langs`
--

CREATE TABLE IF NOT EXISTS `Langs` (
  `LCod` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Γλώσσας',
  `LLect` varchar(30) NOT NULL COMMENT 'Λεκτικό Γλώσσας',
  PRIMARY KEY (`LCod`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Πίνακας Γλωσσών Εξετάσεων';

-- --------------------------------------------------------
```

```
--
-- Δομή Πίνακα για τον Πίνακα `Numbers`
--

CREATE TABLE IF NOT EXISTS `Numbers` (
  `KCod` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Κατηγορίας',
  `PCod` int(10) unsigned NOT NULL COMMENT 'Κωδικός Εσωτερικής Κατηγορίας',
  `Numb` int(10) unsigned NOT NULL COMMENT 'Αριθμός Ερωτήσεων Εσωτερικής
Κατηγορίας'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Πλήθος Ερωτήσεων ανά εσωτερική
κατηγορία';

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `People`
--

CREATE TABLE IF NOT EXISTS `People` (
  `PID` int(10) unsigned NOT NULL AUTO_INCREMENT COMMENT 'Κωδικός
Εξεταζόμενου',
  `Surname` varchar(50) NOT NULL COMMENT 'Επώνυμο Εξεταζόμενου',
  `Firstname` varchar(50) NOT NULL COMMENT 'Όνομα Εξεταζόμενου',
  `Fathername` varchar(50) NOT NULL COMMENT 'Πατρώνυμο Εξεταζόμενου',
  `IDNumber` varchar(15) NOT NULL COMMENT 'Αριθμός Ταυτότητας Εξεταζόμενου',
  `ExamID` int(10) unsigned NOT NULL COMMENT 'Εξέταση στην οποία
συμμετέχει',
  `ExamKateg` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Κατηγορίας
Εξέτασης',
  `ExamLang` mediumint(8) unsigned NOT NULL COMMENT 'Γλώσσα Εξέτασης',
  `Speech` tinyint(1) NOT NULL COMMENT 'Θα υπάρχει ανάγνωση των ερωτήσεων',
  `ClientID` int(10) unsigned NOT NULL DEFAULT '0' COMMENT 'Κωδικός Σταθμού
Εξέτασης (0 = Δεν έχει οριστεί)',
  `Result` enum('Pass','Fail','Absent','-') NOT NULL COMMENT 'Αποτέλεσμα',
  PRIMARY KEY (`PID`),
  KEY `ExamID` (`ExamID`),
  KEY `ExamKateg` (`ExamKateg`),
  KEY `ExamLang` (`ExamLang`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COMMENT='Πίνακας Εξεταζομένων'
AUTO_INCREMENT=45 ;

-- --------------------------------------------------------

--
-- Δομή Πίνακα για τον Πίνακα `Questions`
--

CREATE TABLE IF NOT EXISTS `Questions` (
  `QCod` int(10) unsigned NOT NULL COMMENT 'Κωδικός Ερώτησης',
  `QKateg` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Κατηγορίας',
  `QPag` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Εσωτερικής
Κατηγορίας',
  `QLang` mediumint(8) unsigned NOT NULL COMMENT 'Κωδικός Γλώσσας',
  `QLect` varchar(500) NOT NULL COMMENT 'Λεκτικό Ερώτησης',
  `QPhoto` varchar(20) NOT NULL COMMENT 'Όνομα Φωτογραφίας',
  `QSound` varchar(20) NOT NULL COMMENT 'Όνομα Αρχείου Ήχου',
  `QBook` varchar(50) NOT NULL COMMENT 'Αναφορά Βιβλίου',
  PRIMARY KEY (`QCod`),
  KEY `QKateg` (`QKateg`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='Πίνακας Ερωτήσεων όλων των
Κατηγοριών';
```

```
--
-- Constraints for dumped tables
--

--
-- Constraints for table `Answers`
--
ALTER TABLE `Answers`
  ADD CONSTRAINT `Answers_ibfk_1` FOREIGN KEY (`AQCod`) REFERENCES
`Questions` (`QCod`);

--
-- Constraints for table `People`
--
ALTER TABLE `People`
  ADD CONSTRAINT `People_ibfk_1` FOREIGN KEY (`ExamID`) REFERENCES
`Examinations` (`ExamID`),
  ADD CONSTRAINT `People_ibfk_2` FOREIGN KEY (`ExamKateg`) REFERENCES
`Kateg` (`KCod`),
  ADD CONSTRAINT `People_ibfk_3` FOREIGN KEY (`ExamLang`) REFERENCES `Langs`
(`LCod`);

--
-- Constraints for table `Questions`
--
ALTER TABLE `Questions`
  ADD CONSTRAINT `Questions_ibfk_1` FOREIGN KEY (`QKateg`) REFERENCES
`Kateg` (`KCod`);

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

ΠΑΡΑΡΤΗΜΑ Β

Ο ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ SERVER

PACKAGE: org.gmele.dissertation.server

CLASS: ClientStations

```
package org.gmele.dissertation.server;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.gmele.dissertation.server.exception.ESException;
import org.gmele.general.dbconnections.DBSimlpeConn;
import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση ClientStations χρησιμοποιείται για την αποθήκευση των πληροφοριών
που αφορούν τους τερματικούς σταθμούς εξάτασης. Καταγράφονται
 * οι πληροφορίες που βρίσκονται στη βάση δεδομένων (πίνακας clients).
Παρέχονται μέθοδοι για το διάβασμα, την τροποποίηση και την
 * αποθήκευση των πληροφοριών αυτών.
 * @author Giorgos Meletiou
 * @version 1.0
 */
final public class ClientStations
{
    /**
     * Ο αριθμός των σταθμών εξέτασης.
     */
    public int Numb;
    /**
     * Ο Α.Α. (κωδικός) του σταθμού εξέτασης.
     */
    public int[] ID;
    /**
     * Το λεκτικό του σταθμού εξέτασης.
     */
    public String[] Name;
    /**
```

```
     * Η IP διεύθυνση του σταθμού εξέτασης. Η διεύθυνση αυτή είναι σταθερή
για κάθε σταθμό και χρησιμοποιείται σε ελέγχους ασφαλείας και
     * για την αναγνώριση του σταθμού.
     */
    public String[] IP;

    /**
     * Η κατάσταση του σταθμού εξέτασης. Ο κάθε σταθμός εξέτασης μπορεί να
βρίσκεται σε μία από τις παρακάτω καταστάσεις:
     * <ul>
     * <li> Active
     * <li> InActive
     * <li> Damaged
     * </ul>
     */
    public String[] Status;

    /**
     * Αρχικοποιεί την κλάση Καλώντας την μέθοδο {@link #LoadTable() }
     * @throws GmException Αν δημιουργηθεί πρόβλημα στο διάβασμα των
δεδομένων. Συμπεριλαμβάνεται και η περίπτωση της δημιουργείας εξαίρεσης
     * από την κλάση DBSimpleConn.
     */
    ClientStations () throws ESException
    {
        LoadTable ();
    }

    /**
     * Διαβάζει τον πίνακα Clients και τοποθετεί τα δεδομένα του στις
αντίστοιχες μεταβλητές.
     * @throws GmException Αν δημιουργηθεί πρόβλημα στο διάβασμα των
δεδομένων. Συμπεριλαμβάνεται και η περίπτωση της δημιουργείας εξαίρεσης
     * από την κλάση DBSimpleConn.
     */
    void LoadTable () throws ESException
    {
        Connection conn;
        PreparedStatement stat;
        ResultSet rs;
        String Que;
        int i;
        conn = null;
        try
        {
            Que = "SELECT * From Clients";
            conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis",
"ExamDis");
                    stat = conn.prepareStatement (Que);
                    rs = stat.executeQuery ();
            rs.last ();
            Numb = rs.getRow ();
            ID = new int[Numb];
            Name = new String[Numb];
            IP = new String[Numb];
            Status = new String[Numb];
            rs.beforeFirst ();
                    i = 0;
            while (rs.next())
                    {
                            ID[i] = rs.getInt ("ClID");
                Name[i] = rs.getString ("ClName");
```

```
                IP[i] = rs.getString ("ClIP");
                Status[i] = rs.getString ("ClStatus");
                i++;
            }
                rs.close();
                stat.close();
                conn.close ();
        }
        catch (GmException e)
        {
            throw new ESException ("Error in reading Client Stations data",
ESException.CSCannotReadClientTable, "Connection error", e);
        }
        catch (SQLException e)
        {
            throw new ESException ("Error in reading Client Stations data",
ESException.CSCannotReadClientTable, "Sql error", e);
        }
    }

    /**
     * Δημιουργεί και επιστρέφει ένα πίνακα ο οποίος περιέχει τους κωδικούς
των σταθμών εξέτασης οι οποίοι είναι ενεργοί σύμφωνα με τα
     * στοιχεία της βάσης δεδομένων
     * @return Ο πίνακας με τα IDs.
     */
    public int[] GetActiveNumbs ()
    {
        int i;
        int c;
        int[] Act;
        c = 0;
        for (i = 0; i < Numb; i++)
            if (Status[i].equals ("Active"))
                c++;
        Act = new int[c];
        c = 0;
        for (i = 0; i < Numb; i++)
            if (Status[i].equals ("Active"))
                Act[c++] = ID[i];
        return Act;
    }

    /**
     * Επιστρέφει την IP διεύθυνση ενός σταθμού εξέτασης. Δεν γίνεται
έλεγχος αν ο κωδικός υπάρχει και σε περίπτωση λάθους θα δημιουργηθεί
     * εξαίρεση από το σύστημα.
     * @param s Το ID του σταθμού.
     * @return Η IP διεύθυνση του σταθμού.
     */
    public String GetIP (int s)
    {
        int i = 0;
        while (ID[i] != s)
            i++;
        return IP[i];
    }
}
```

CLASS: ExamServer

```
package org.gmele.dissertation.server;

import java.util.Date;
import javax.swing.JOptionPane;
import org.gmele.dissertation.server.communications.Uhura;
import org.gmele.dissertation.server.exception.ESException;
import org.gmele.dissertation.server.gui.MainForm;
import org.gmele.dissertation.server.gui.ResultForm;
import org.gmele.dissertation.server.resources.ResourceHandler;
import org.gmele.dissertation.testhandler.ExamDisDB;
import org.gmele.dissertation.testhandler.ExamStruct;
import org.gmele.dissertation.testhandler.PeopleStruct;
import org.gmele.dissertation.testhandler.TestHandler;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;
import org.gmele.general.sheets.excelx.XlsxSheet;


/**
 * Η βασική κλάση του project ExamServer. Το project ExamServer είναι ο
Server του συστήματος εξετάσεων υποψηφίων οδηγών (και όχι μόνο) για
 * android συσκευές. Αρχικοποιεί κλάσεις και παρέχει μεθόδους οι οποίες
υλοποιούν λειτουργίες της εφαρμογής οι οποίες δεν σχετίζονται με το
 * User Interface.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public final class ExamServer
{
    /**
     * Το βασικό παράθυρο της εφαρμογής.
     */
    MainForm MainF;

    /**
     * Πληροφορίες για τους σταθμούς εξέτασης.
     */
    ClientStations Clients;

    /**
     * Επικοινωνία με Clients
     */
    Uhura Comm;

    /**
     * Αρχικοποιεί το πρόγραμμα ExamServer. Συγκεκριμένα:
     * <ul>
     *   <li>Φορτώνει τις πληροφορίες των τερματικών σταθμών από τη βάση
δεδομένων στο αντικείμενο Clients.
     *   <li>Δημιουργεί το αντικείμενο το οποίο θα χειρίζεται την
επικοινωνία με τους σταθμούς εξέτασης.
     *   <li>Δημιουργεί το αντικείμενο το οποίο θα χειρίζεταο το user
interface της εφαρμογής.
     * </ul>
     */
    public ExamServer ()
    {
        try
        {
            Clients = new ClientStations ();
        }
        catch (ESException e)
```

```
        {
            String Que;
            Que = "Δημιουργήθηκε Σφάλμα κατά την ανάγνωση των Στοιχείων των
Σταθμών Εξέτασης.";
            UnexpectedSystemError (Que, e.getErrorCode ());
        }
        Comm = new Uhura (this);
        MainF = new MainForm (this);
    }

    /**
     * Εκτελεί τις εργασίες τερματισμού τις εφαρμογής, τακτοποιεί τυχών
εκρεμμότητες και τερματίζει το πρόγραμμα. Καλείται από το μενού
     * "Εξοδος" της βασικής φόρμας.
     */
    public void EndProcs ()
    {
        System.exit (0);
    }

    /**
     * Η μέθοδος getClients είναι η getter μέθοδος για το attribute {@link
#Clients}.
     * @return Αναφορά στο attribute {@link #Clinets}
     */
    public ClientStations getClients ()
    {
        return Clients;
    }

    /**
     * Η μέθοδος getMainF είναι η getter μέθοδος για το attribute {@link
#MainF}.
     * @return Αναφορά στο attribute {@link #MainF}
     */

    public MainForm getMainF ()
    {
        return MainF;
    }

    /**
     * Η μέθοδος getComm είναι η getter μέθοδος για το attribute {@link
#Comm}.
     * @return Αναφορά στο attribute {@link #Comm}
     */

    public Uhura getComm ()
    {
        return Comm;
    }
    /**
     * Η μέθοδος UnexpectedSystemError καλείται όταν συμβεί εξαίρεση η οποία
δεν θα έπρεπε ποτέ να συμβεί (π.χ. λείπει εικονίδιο από τα
     * resources. Το πρόγραμμα ενημερώνει το χρήστη και στη συνέχεια
τερματίζει την εφαρμογή.
     * @param Mess Το μήνυμα που θα τυπωθεί (μετά από επεξεργασία).
     * @param ErrCod Ο κωδικός λάθους της εξαίρεσης που δημιουργήθηκε.
     */
    public void UnexpectedSystemError (String Mess, int ErrCod)
    {
        String Que;
```

```
        Que = "<html>" + Mess + "<br> Το πρόγραμμα θα τερματιστεί. Παρακαλώ,
επικοινωνείστε με τον διαχειριστή του συστήματος. <br>";
        Que = Que + "Error Code : " + ErrCod + "<br> </html>";
        JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
        System.exit (1);
    }

    /**
     * Πραγματοποιεί όλα τα βήματα της διαδικασίας εξέτασης μία ομάδας
     */
    public void ExamGroup ()
    {
        ExamStruct[] Groups;
        PeopleStruct[] Peo;
        String SDir;
        String RDir;
        String[] Fns;
        int[] Terms;
        int ExamID;
        int ExamPos;
        int i;
        int NoPres;
        try
        {
            MainF.SetCursor (MainForm.MouseWait);
            Groups = ExamDisDB.GetTodaysPendingExams ();
            ExamPos = MainF.SelectExam (Groups, "Επιλέξτε την ομάδα η οποία
θα εξεταστεί");
            if (ExamPos == -1)
            {
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            ExamID = Groups[ExamPos].ID;
            Peo = ExamDisDB.GetPeopleInExam (ExamID);
            if (!Comm.SendHelloAll ())
            {
                String Que = "Δεν υπάρχει επικοινωνία με όλους τους σταθμούς
Εξέτασης.\nΠαρακαλώ χειριστείτε το μέσα από το μενού "
                    + "\"Διαχείριση --> Σταθμοί Εξέτασης\"";
                JOptionPane.showMessageDialog (MainF, Que, "Πρόβλημα
Επικοινωνίας", JOptionPane.WARNING_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            if (!DrawStations (Peo))
            {
                String Que = "Δεν υπάρχουν αρκετοί ενεργοί σταθμοί εξέτασης
για να καλύψουν την προγραμματισμένη ομάδα.";
                JOptionPane.showMessageDialog (MainF, Que, "Επιλογή Θέσεων
Εξέτασης", JOptionPane.WARNING_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            SDir = ResourceHandler.GetProperty ("Options", "SendTestDir");
            RDir = ResourceHandler.GetProperty ("Options",
"RecievedTestDir");
            TestHandler.MakeTestFiles (Peo, SDir);
```

```
            Fns = new String[Peo.length];
            Terms = new int[Peo.length];
            for (i = 0; i < Peo.length; i++)
            {
                Fns[i] = SDir + "/" + Peo[i].ExamID + "/" + Peo[i].ClientID
+ "-" + Peo[i].Surname + ".test";
                Terms[i] = Peo[i].ClientID;
            }
            if (!Comm.SendAllTestFiles (Fns, Terms))
            {
                String Que = "Δεν ήταν δυνατή η αποστολλή των
ερωτηματολογίων σε κάποιους σταθμούς εξέτασης.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία και επανεκκινείστε τη
διαδικασία εξέτασης.";
                JOptionPane.showMessageDialog (MainF, Que, "Αποστολή
ερωτηματολογίων", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            if (!Comm.SendAllSimpleCom (Terms, Comm.CommStartClient))
            {
                String Que = "Δεν ήταν δυνατή η αποστολή εντολής έναρξης
εξέτασης σε κάποιους σταθμούς εξέτασης.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία και επανεκκινείστε τη
διαδικασία εξέτασης.";
                JOptionPane.showMessageDialog (MainF, Que, "Έναρξη Client",
JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            NoPres = Comm.HandleAbsences (Peo);
            if (NoPres == -1)
            {
                String Que = "Δεν ήταν δυνατή η λήψη του καταλόγου παρουσιών
/ απουσιών.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία και επανεκκινείστε τη
διαδικασία εξέτασης.";
                JOptionPane.showMessageDialog (MainF, Que, "Κατάλογος
παρουσιών", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            if (!Comm.SendAllSimpleCom (Terms, Comm.CommStartExam))
            {
                String Que = "Δεν ήταν δυνατή η αποστολή εντολής έναρξης της
κύριας διαδικασίας εξέτασης.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία και επανεκκινείστε τη
διαδικασία εξέτασης.";
                JOptionPane.showMessageDialog (MainF, Que, "Έναρξη
Εξέτασης", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            if (!Comm.WaitExamCompletition (Peo))
            {
                String Que = "Δεν ήταν δυνατή η παραλαβή γνωστοποίησης
ολοκλήρωσης της διαδικασίας εξέτασης από όλους τους σταθμούς.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία.";
```

```
                JOptionPane.showMessageDialog (MainF, Que, "Αναμονή
ολοκλήρωσης εξέτασης", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            for (i = 0; i < Peo.length; i++)
            {
                Fns[i] = RDir + "/" + Peo[i].ExamID + "/" + Peo[i].ClientID
+ "-" + Peo[i].Surname + ".test";
                Terms[i] = Peo[i].ClientID;
            }
            if (!Comm.GetAllTestFiles (Fns, Terms))
            {
                String Que = "Δεν ήταν δυνατή η παραλλαβή των
ερωτηματολογίων από κάποιους σταθμούς εξέτασης.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία και επανεκκινείστε τη
διαδικασία εξέτασης.";
                JOptionPane.showMessageDialog (MainF, Que, "Ανάκτηση
ερωτηματολογίων", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            MainF.SetCursor (MainForm.MouseWait);
            MarkActiveGroup (Peo, Fns);
            ResultForm RF = new ResultForm (MainF, Peo[0].ExamID, true);
            MainF.add (RF);
            RF.repaint ();
            if (!Comm.SendAllSimpleCom (Terms, Comm.CommExamCompleted))
            {
                String Que = "Δεν ήταν δυνατή η αποστολή εντολής ολοκλήρωσης
της διαδικασίας σε κάποιους σταθμούς εξέτασης.\n" +
                    "Παρακαλώ ελέγξτε την επικοινωνία.";
                JOptionPane.showMessageDialog (MainF, Que, "Ολοκλήρωση
Διαδικασίας", JOptionPane.ERROR_MESSAGE);
                MainF.SetCursor (MainForm.MouseNormal);
                return;
            }
            System.out.println ("Examination completed");
            MainF.SetCursor (MainForm.MouseNormal);
        }
        catch (TestException e)
        {
            System.out.println (e.getMessage ());
            System.out.println (e.getData ());
            if (e.getSysException () != null)
                System.out.println (e.getSysException ().getMessage ());
            if (e.getErrorCode () == TestException.EDSqlException)
                UnexpectedSystemError ("Δημιουργήθηκε Λάθος κατά την
πρόσβαση στη Βάση Δεδομένων", e.getErrorCode ());
            if (e.getErrorCode () == TestException.EDNoPeopleInExam)
                UnexpectedSystemError ("Δεν υπάρχουν εξεταζόμενοι στην
συγκεκριμένη ομάδα εξέτασης:" + e.getData (), e.getErrorCode ());
            System.out.println ("@@@@@" + e.getMessage () + " @ " +
e.getErrorCode ());
        }
        catch (ESException e)
        {
            if (e.getErrorCode () == ESException.RHIOErrorReading ||
e.getErrorCode () == ESException.RHNoSuchResource ||
                e.getErrorCode ()  == ESException.RHNoSuchPropertyKey)
            {
```

```
                UnexpectedSystemError ("Δημιουργήθηκε Λάθος κατά την
πρόσβαση σε δεοδομένα της εφαρμογής", e.getErrorCode ());
            }
            System.out.println ("@@@" + e.getMessage () + " @ " +
e.getErrorCode ());
        }
        catch (GmException e)
        {
            System.out.println (e.getMessage ());
            System.out.println (e.getData ());
            System.out.println (e.getSysException ().getMessage ());
            UnexpectedSystemError ("Δημιουργήθηκε Γενικό Λάθος στην
εφαρμογή.", e.getErrorCode ());
            System.out.println ("@" + e.getMessage () + " @ " +
e.getErrorCode ());
        }
    }

    public static void main (String[] args)
    {
        ExamServer ES = new ExamServer();
    }

    /**
     * Κληρώνει τους σταθμούς εξέτασης στους οποίους θα εξεταστούν οι
εξεταζόμενοι ο οποίοι εισάγονται ως είσοδος. Η κλήρωση
     * πραγματοποιείται με την χρήση της "TestHandler.Suffle". Ο σταθμός
στον οποίο θα εξεταστεί ο κάθε εξεταζόμενος καταγράφεται στο πεδίο
     * "ClientID" της δομής PeopleStruct στον πίνακα εισόδου. <p>
     * @param Peo Ο πίνακας των εξεταζόμενων.
     * @return true αν η διαδικασία ήταν επιτυχής, false αν οι ενεργοί
σταθμοί είναι λιγότεροι από τους εξεταζόμενους.
     */
    public boolean DrawStations (PeopleStruct[] Peo)
    {
        int i;
        int[] Cl;
        Cl = Clients.GetActiveNumbs ();
        if (Cl.length < Peo.length)
            return false;
        TestHandler.Suffle (Cl);
        for (i = 0; i < Peo.length; i++)
        {
            Peo[i].ClientID = Cl[i];
        }
        return true;
    }

    /**
     * Βαθμολογεί όλα τα φύλλα μίας ομάδας εξέτασης (αυτής που εξετάζεται
κατά τη στιγμή της κλήσης) και ενημερώνει τους σχετικούς πίνακες.
     * @param Peo Ο πίνακας των εξεταζομένων όπως έχει διαμορφωθεί από τα
προηγούμενα στάδια της διαδικασίας εξέτασης (περιέχει τερματικά,
     * κατάσταση παρουσίας / απουσίας, κλπ).
     * @param Fns Τα ονόματα των αρχείων των φύλλων εξέτασης που
αντιστοιχούν στους εξεταζόμενους.
     * @throws TestException Όπως δημιουργούνται από τις {@link
TestHandler#MarkTest(java.lang.String)}, {@link ExamDisDB#SetPeopleResult}
     * και {@link ExamDisDB#SetExamStatus}.
     * @throws GmException Όπως δημιουργούνται από τις {@link
TestHandler#MarkTest(java.lang.String)}, {@link ExamDisDB#SetPeopleResult}
και
```

```
     * {@link ExamDisDB#SetExamStatus}.
     */
    public void MarkActiveGroup (PeopleStruct[] Peo, String[] Fns) throws
TestException, GmException
    {
        int i;
        boolean res;
        for (i = 0; i < Peo.length; i++)
        {
            //System.out.println ("@");
            res = TestHandler.MarkTest (Fns[i]);
            //System.out.println ("@@");
            if (Peo[i].Result.equals ("Absent"))
                ExamDisDB.SetPeopleResult (Peo[i].PID, "Absent");
            else
                if (res)
                    ExamDisDB.SetPeopleResult (Peo[i].PID, "Pass");
                else
                    ExamDisDB.SetPeopleResult (Peo[i].PID, "Fail");
            //System.out.println ("@@@");
        }
        ExamDisDB.SetExamStatus (Peo[0].ExamID, "Done");
        //System.out.println ("@@@@");
    }

    /**
     * Διαβάζει τα στοιχεία μίας ομάδας εξέτασης από ένα φύλλο του Excel
(τύπος .xlsx) και τα καταχωρεί στη βάση δεδομένων. Η καταχώρηση
     * γίνεται με τη χρήση των μεθόδων της κλάσης ExamDisDB. Αν υπάρχει
ομάδα με τα ίδια στοιχεία (ημερομηνία και αριθμός group)
     * παρουσιάζεται μήνυμα λάθους.
     * @param Fn Το όνομα του .xlsx αρχείου το οποίο περιέχει τα στοιχεία
της ομάδας.
     */
    public void LoadGroupData (String Fn)
    {
        XlsxSheet xl;
        ExamStruct Exam;
        PeopleStruct Pe;
        Date D;
        int EID;
        int G;
        int CurL;
        try
        {
            xl = new XlsxSheet (Fn);
            D = xl.GetCellDate (2, 1);
            G = (int) xl.GetCellNumeric (3, 1);
            EID = ExamDisDB.GetGroupID (D, G);
            if (EID != -1)
            {
                JOptionPane.showMessageDialog (null, "Υπάρχει ήδη ομάδα
εξέτασης με τα στοιχεία αυτά.", "Εισαγωγή Ομάδας Εξέτασης"
                    , JOptionPane.ERROR_MESSAGE);
                return;
            }
            CurL = 7;
            Exam = new ExamStruct ();
            Exam.DateTime = D;
            Exam.Group = G;
            Exam.Status = "Pending";
            EID = ExamDisDB.AddExam (Exam);
```

```
            while (CurL <= xl.GetLastRow ())
            {
                Pe = new PeopleStruct ();
                Pe.Surname = xl.GetCellString (CurL, 0);
                Pe.Firstname = xl.GetCellString (CurL, 1);
                Pe.Fathername = xl.GetCellString (CurL, 2);
                Pe.IDNumber = xl.GetCellString (CurL, 3);
                Pe.ExamID = EID;
                Pe.ExamKateg = (int) xl.GetCellNumeric (CurL, 4);
                Pe.ExamLang = (int) xl.GetCellNumeric (CurL, 5);
                if (xl.GetCellString (CurL, 6).equals ("NAI"))
                    Pe.Speech = true;
                else
                    Pe.Speech = false;
                Pe.ClientID = 0;
                Pe.Result = "-";
                ExamDisDB.AddPeople (Pe);
                CurL++;
            }
        }
        catch (Exception e)
        {
            System.out.println ("Exception...." + e.getMessage ());
        }

    }
}
```

PACKAGE: org.gmele.dissertation.server.communications

CLASS: Uhura

```
package org.gmele.dissertation.server.communications;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.Socket;
import java.net.UnknownHostException;
import javax.swing.JOptionPane;
import org.gmele.dissertation.server.ClientStations;
import org.gmele.dissertation.server.ExamServer;
import org.gmele.dissertation.testhandler.PeopleStruct;

/**
 * Η κλάση Uhura είναι υπεύθυνη για τον συντονισμό της επικοινωνίας του
Server με τους Clients. Η επικοινωνία γίνεται μέσω tcp socket. Ως
 * εξυπηρετητής των συνδέσεων αυτών δρα το service "Daemon" που τρέχει σε
κάθε σταθμό εξέτασης. <p>
 * Η κλάση στέλνει εντολές στους clients. Οι εντολές μπορούν να στέλνονται
σε ξεχωριστά threads μπορεί και όχι. Οι εντολές είναι ακέραιοι
```

```
 * αριθμοί σε string μορφή. Πάνω στα sockets ανοίγονται απλοί streamers ώστε
να είναι δυνατό στο ίδιο socket να στέλνονται τόσο binary όσο
 * και formatted data.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class Uhura
{
    /**
     * Αποστέλλεται κάθε φορά που (επαν)εγκαθίσταται επικοινωνία,
προκειμένου να ελεγχθεί η επικοινωνία.
     */
    public final int CommHello = 0;

    /**
     * Δίνει εντολή στον σταθμό εξέτασης να παραλλάβει ένα αρχείο
     */
    public final int CommGetFile = 1;

    /**
     * Δίνει εντολή στον σταθμό εξέτασης να αποστείλει ένα αρχείο.
     */
    public final int CommSendFile = 2;

    /**
     * Δίνει εντολή στον σταθμό εξέτασης να διαγράψει ένα αρχείο.
     */
    public final int CommDeleteFile = 3;

    /**
     * Δίνει εντολή ενεργοποίησης του client προγράμματος και προβολής της
αρχικής οθόνης με τα στοιχεία του εξεταζόμενου.
     */
    public final int CommStartClient = 10;

    /**
     * Δίνει εντολή στο Service να επιστρέψει την τιμή της μεταβλητής
επικοινωνίας με τον Client (ClientComm).
     */
    public final int CommSendClientComm = 11;
    /**
     * Δίνει εντολή για έναρξη της εξέτασης.
     */
    public final int CommStartExam = 12;

    /**
     * Ενημερώνει τον σταθμό εξέτασης ότι ο server κήρυξε τη διαδικασία
εξέτασης ολοκληρωμένη και κατά συνέπεια το πρόγραμμα client το
     * οποίο περιμένει πρέπει να τερματιστεί.
     */
    public final int CommExamCompleted = 13;
    /**
     * Η πόρτα στην οποία ακούνε οι Δαίμονες στους σταθμούς εξέτασης.
     */
    final int DaemonPort = 25123;

    /**
     * Το μήνυμα που ανταλλάσεται στην εντολή "Hello"
     */
    final String HelloString = "HELLO";

    /**
```

```
     * Αναφορά στην βάσική κλάση της εφαρμογής
     */
    ExamServer Father;

    /**
     * Αναφορά στον πίνακα πληροφοριών των σταθμών εξέτασης.
     */
    ClientStations Clients;

    /**
     * Αριθμός Σταθμών Εξάτασης που έχουν καταγραφεί ως ενεργοί.
     */
    int NoActive;

    /**
     * IDs των Σταθμών εξέτασης που θεωρούντια ενεργοί. Με αυτό το ID μπορεί
να αναζητηθούν τα λοιπά στοιχεία του σταθμού, π.χ. η IP του.
     */
    int[] ActID;

    /**
     * Αρχικοποιεί τα attributes {@link #Father} και {@link Clients}.
     * @param father
     */
    public Uhura (ExamServer father)
    {
        Father = father;
        Clients = Father.getClients ();
        ActID = Clients.GetActiveNumbs ();
        NoActive = ActID.length;
    }

    /**
     * Στέλνει την εντολή "Hello" σε ένα σταθμό εξέτασης.
     * @param StatID Ο κωδικός του σταθμού εξέτασης.
     * @return true Αν η διαδικασία ολοκληρώθηκε κανονικά, false αν
δημιουργήθηκε εξαίρεση στο socket η ο σταθμός εξέτασης έστειλε άσχετο
     * string.
     */
    public boolean SendHello (int StatID)
    {
        String IP;
        boolean res;
        Socket soc;
        InputStream is;
        OutputStream os;
        String S;
        IP = Clients.GetIP (StatID);
        res = true;
        try
        {
            res = true;
            soc = new Socket (IP, DaemonPort);
            is = soc.getInputStream ();
            os = soc.getOutputStream ();
            SendLine (os, Integer.toString (CommHello));
            SendLine (os, HelloString);
            S = GetLine (is);
            if (!S.equals (HelloString))
                res = false;
        }
        catch (UnknownHostException e)
```

```
        {
            String Que = "Άκυρη IP Διεύθυνση. Δεν θα έπρεπε να συμβεί ποτέ.
" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
        }
        catch (IOException ex)
        {
            res = false;
        }
        return res;
    }

    /**
     * Στέλνει σε όλους τους ενεργούς σταθμούς εξέτασης, δηλαδή σε όλους
όσους βρίσκονται στον πίνακα ActID, την εντολή Hello. Αν αποτύχει
     * έστω και σε ένα σταθμό επιστρέφει false, αλλιώς επιστρέφει true.
     * @return True αν όλοι οι σταθμοί απάντησαν κανονικά, false
διαφορετικά.
     */
    public boolean SendHelloAll ()
    {
       boolean flag;
       flag = true;
       for (int i : ActID)
       {
           if (!SendHello (i))
               flag = false;
       }
       return flag;
    }

    /**
     * Στέλνει την εντολή "GetFile" σε ένα σταθμό εξέτασης. Ο σταθμός πρέπει
να παραλάβει το αρχείο και να το αποθηκεύσει. Στην αρχή
     * αποστέλλεται η εντολή, στην συνέχεια ο σχετικός κατάλογος αποθήκευσης
του αρχείου, έπειτα το τοπικό (στον σταθμό εξέτασης) όνομα
     * του αρχείου και τέλος το αρχείο ως πίνακας από bytes.
     * Ο σταθμός εξέτασης απαντάει με "O.K". <p>
     * Το σχετικό όνομα του φακέλου θα προστεθεί από το σταθμό εξέτασης στο
πλήρες όνομα του φακέλου δεδομένων της εφαρμογής (YME) ο οποίος
     * βρίσκεται στην εξωτερική του κάρτα. Το όνομα δεν περιέχει τον
χαρακτήρα "/" και μπορεί να είναι "." προκειμένου το αρχείο να σωθεί
     * στον κατάλογο YME.
     * @param StatID Ο κωδικός του σταθμού εξέτασης.
     * @param Fn Το πλήρες όνομα του αρχείου που θα αποσταλλεί.
     * @param Dir Το όνομα του καταλόγου του σταθμού εξέτασης στον οποίο θα
αποθηκευτεί το αρχείο.
     * @param RFn Το όνομα που θα έχει το αρχείο στον σταθμό εξέτασης.
     * @return true αν η διαδικασία ολοκληρωθεί με επιτυχία, false
διαφορετικά.
     */
    public boolean SendFile (int StatID, String Fn, String Dir, String RFn)
    {
        String IP;
        boolean res;
        Socket soc;
        InputStream is;
        OutputStream os;
        FileInputStream fil;
        byte[] buf;
```

```
        int br;
        String S;
        buf = new byte[4096];
        IP = Clients.GetIP (StatID);
        res = true;
        try
        {
            res = true;
            soc = new Socket (IP, DaemonPort);
            is = soc.getInputStream ();
            os = soc.getOutputStream ();
            SendLine (os, Integer.toString (CommGetFile));
            SendLine (os, Dir);
            SendLine (os, RFn);
            fil = new FileInputStream (Fn);
            while ((br = fil.read (buf)) != -1)
                os.write (buf, 0, br);
            fil.close ();
            os.flush ();
            soc.shutdownOutput ();
            S = GetLine (is);
            if (!S.equals ("O.K."))
            {
                res = false;
            }
            os.close ();
        }
        catch (UnknownHostException e)
        {
            String Que = "Άκυρη IP Διεύθυνση. Δεν θα έπρεπε να συμβεί ποτέ.
" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
        }
        catch (FileNotFoundException e)
        {
            String Que = "Το αρχείο προς αποστολή σε σταθμό εξέτασης δεν
βρέθηκε. Δεν θα έπρεπε να συμβεί ποτέ.\n" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
        }
        catch (IOException e)
        {
           res = false;
           System.out.println ("Here:" + e.getLocalizedMessage ());
        }
        return res;
    }

    /**
     * Αποστέλει τα TestSheets σε όλους τους σταθμούς εξέτασης οι οποίοι
έχουν κληρωθεί να χρησιμοποιηθούν στην εξέταση. Τα αρχεία, στους
     * σταθμούς εξέτασης, τοποθετούνται στην σχετική ρίζα με το κοινό όνομα
"TestFile.test". Η αποστολή των αρχείων πραγματοποιείται μέσω
     * της {@link #SendFile}
     * @param Files Ο πίνακας με τα πλήρη ονόματα αρχείων που θα αποσταλούν.
     * @param Stations Ο πίνακας με τους κωδικούς των σταθμών εξέτασης στους
οποίους θα γίνουν οι αποστολές των αρχείων. Το μέγεθος και
     * οι θέσεις του πίνακα Stations είναι αντίστοιχα με αυτά του πίνακα
Files.
```

```
     * @return true αν η αποστολή ολοκληρωθεί με επιτυχία σε όλους τους
σταθμούς, false διαφορετικά.
     */
    public boolean SendAllTestFiles (String[] Files, int[] Stations)
    {
        int i;
        boolean res;
        res = true;
        for (i = 0; i < Files.length; i++)
        {
            if (!SendFile (Stations[i], Files[i], ".", "TestFile.test"))
                res = false;
        }
        return res;
    }

    /**
     * Στέλνει μία απλή εντολή σε ένα σταθμό εξέτασης. Ως απλή εντολή
ορίζεται μία εντολή η οποία ενεργοποιεί μία λειτουργία χωρίς να
     * απαιτείται περαιτέρω ανταλλαγή δεδομένων ή απάντηση πέρα από το
string "O.K." το οποίο επιβεβαιώνει την λήψη της εντολής.
     * @param StatID Ο κωδικός του σταθμού εξέτασης.
     * @param Command Η εντολή που θα αποσταλεί. Η εντολή πρέπει να είναι
μία από αυτές που έχουν οριστεί με σταθερές στην κλάση.
     * @return true αν η διαδικασία ολοκληρώθηκε με επιτυχία, false
διαφορετικά.
     */
    public boolean SendSimpleCommand (int StatID, int Command)
    {
        String IP;
        boolean res;
        Socket soc;
        InputStream is;
        OutputStream os;
        String S;
        IP = Clients.GetIP (StatID);
        res = true;
        try
        {
            res = true;
            soc = new Socket (IP, DaemonPort);
            is = soc.getInputStream ();
            os = soc.getOutputStream ();
            SendLine (os, Integer.toString (Command));
            S = GetLine (is);
            if (!S.equals ("O.K."))
                res = false;
        }
        catch (UnknownHostException e)
        {
            String Que = "Άκυρη IP Διεύθυνση. Δεν θα έπρεπε να συμβεί ποτέ.
" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
        }
        catch (IOException ex)
        {
            res = false;
        }
        return res;
    }
```

```
    /**
     * Στέλνει μία απλή εντολή σε λίστα σταθμών εξέτασης. Η αποστολή γίνεται
σειριακά χωρίς δημιουργία thread. Για την αποστολή
     * χρησιμοποιείται η {@link #SendSimpleCommand}.
     * @param Stations Ο πίνακας με τους κωδικούς των σταθμών εξέτασης στους
οποίους θα γίνει η αποστολή.
     * @param Command Η εντολή η οποία θα αποσταλεί.
     * @return true αν η αποστολή ολοκληρώθηκε με επιτυχία σεόλους τους
σταθμούς, false αν η αποστολή απέτυχε έστω και σε ένα σταθμό.
     */
    public boolean SendAllSimpleCom (int[] Stations, int Command)
    {
        int i;
        boolean res;
        res = true;
        for (i = 0; i < Stations.length; i++)
        {
            if (!SendSimpleCommand (Stations[i], Command))
                res = false;
        }
        return res;
    }

    /**
     * Στέλνει την εντολή "Send Client Comm Value" ({@link
Uhura#CommSendClientComm}) στον σταθμό εξέτασης και περιμένει έως ότου ο
σταθμός
     * να στείλει την τιμή της μεταβλητής. Η τιμή επιστρέφεται σε μορφή
string και μετατρέπεται σε ακέραιο. Σε περίπτωση προβλήματος
     * επικοινωνίας ή παραλαβής άκυρων δεδομένων, η μέθοδος επιστρέφει την
τιμή -1.
     * @param StatID Ο κωδικός του σταθμού εξέτασης.
     * @return Η τιμή που στάλθηκε από τον σταθμό εξέτασης ή -1 σε περίπτωση
λάθούς.
     */
    public int GetClientComm (int StatID)
    {
        String IP;
        int res;
        Socket soc;
        InputStream is;
        OutputStream os;
        String S;
        IP = Clients.GetIP (StatID);
        res = -1;
        try
        {
            soc = new Socket (IP, DaemonPort);
            is = soc.getInputStream ();
            os = soc.getOutputStream ();
            SendLine (os, Integer.toString (CommSendClientComm));
            S = GetLine (is);
            res = Integer.parseInt (S);
        }
        catch (UnknownHostException e)
        {
            String Que = "Άκυρη IP Διεύθυνση. Δεν θα έπρεπε να συμβεί ποτέ.
" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
```

```
        }
        catch (NumberFormatException e)
        {
            res = -1;
        }
        catch (IOException e)
        {
            res = -1;
        }
        return res;
    }

    /**
     * Διαβάζει την τιμή του ClientComm των σταθμών εξέτασης στους οποίους
εξετάζονται οι υποψήφιοι και καταγράφει τους παρόντες και τους
     * απόντες. Το διάβασμα γίνεται κάθε 20 δευτερόλεπτα και μέχρι να πάρει
την τιμή 1 (παρών) ή την τιμή 2 (απών) από όλους τους σταθμούς
     * εξέτασης. Η τιμή καταγράφεται στο πεδίο {@link PeopleStruct#Result}
ως εξής: Αν ο εξεταζόμενος απουσιάζει στο πεδίο εκχωρείται το
     * string "Absent" το οποίο αποτελεί την επίσημη τιμή του πεδίου για την
απουσία. Αν ο εξεταζόμενος είναι παρών στη διαδικασία, στο
     * πεδίο τοποθετείται προσωρινά το string "Testing" το οποίο, για το
υπόλοιπο της διαδικασίας εξέτασης, σημαίνει ότι ο εξεταζόμενος
     * είναι παρών και εξετάζεται. Η τιμή αυτή θα μείνει στο πεδίο μέχρι να
αντικατασταθεί από το τελικό αποτέλεσμα της εξέτασης.
     * @param Peo Ο πίνακας των εξεταζομένων
     * @return Ο αριθμός των παρόντων εξεταζομένων, -1 αν υπάρξει πρόβλημα
επικοινωνίας ή επιστραφεί άσχετη τιμή.
     */
    public int HandleAbsences (PeopleStruct[] Peo)
    {
        int res;
        int r;
        int i;
        int NoAns;
        NoAns = 0;
        res = 0;
        while (NoAns < Peo.length)
        {
            for (i = 0; i < Peo.length; i++)
                if (Peo[i].Result.equals ("-"))
                {
                    r = GetClientComm (Peo[i].ClientID);
                    switch (r)
                    {
                        case -1: return -1;
                        case 0: break;
                        case 1: Peo[i].Result = "Testing"; NoAns++; res++;
break;
                        case 2: Peo[i].Result = "Absent"; NoAns++; break;
                        default: return -1;
                    }
                }
            try
            {
                Thread.sleep (5000);
            }
            catch (InterruptedException ex)
            {

            }
        }
```

```
        return res;
    }

    /**
     * Περιμένει εως ότου όλοι οι σταθμοί εξέτασεις στους οποίους έχουν
εκχωρηθεί εξεταζόμενοι να αναφέρουν ολοκλήρωση διαδικασίας εξέτασης.
     * Η αναφορά έρχεται μέσω της εντολής "Send Client Comm Value" η οποία
στέλνεται σε κάθε σταθμό ξεχωριστά με την εκτέλεση της
     * {@link #GetClientComm(int)}. Η διαδιακασία εξέτασης θεωρείται
ολοκληρωμένη όταν η μεταβλητή ClientComm στον δαίμονα περιέχει τη τιμή
     * "4" (Examination Finished) ή την τιμή "5" (Examination Timed Out). Η
μέθοδος ελέγχει την ολοκλήρωση της διαδικασίας και στους
     * σταθμούς στους οποίους εξετάζονται υποψήφιοι που απουσιάζουν.
     * @param Peo Ο πίνακας των εξεταζομένων.
     * @return true αν η διαδικασία ολοκληρωθεί κανονικά, false αν
επιστραφεί μη αναμενόμενος κωδικός κατάστασης ή υπάρξει πρόβλημα
     * επικοινωνίας.
     */
    public boolean WaitExamCompletition (PeopleStruct[] Peo)
    {
        boolean[] Fin;
        int i;
        int r;
        int NoAns;
        Fin = new boolean[Peo.length];
        for (i = 0; i < Peo.length; i++)
            Fin[i] = false;
        NoAns = 0;
        while (NoAns < Peo.length)
        {
            for (i = 0; i < Peo.length; i++)
                if (!Fin[i])
                {
                    r = GetClientComm (Peo[i].ClientID);
                    switch (r)
                    {
                        case -1: return false;
                        case 3: break;
                        case 4:
                        case 5: NoAns++; Fin[i] = true; break;
                        default: return false;
                    }
                }
            try
            {
                Thread.sleep (5000);
            }
            catch (InterruptedException ex)
            {

            }
        }
        return true;
    }


    /**
     * Στέλνει την εντολή "SendFile" σε ένα σταθμό εξέτασης. Ο σταθμός
πρέπει να στείλει το αρχείο το οποίο θα αποθηκευτεί σε τοπικό φάκελο
     * Στην αρχή αποστέλλεται η εντολή, στην συνέχεια ο σχετικός κατάλογος
αποθήκευσης του αρχείου που θα  αποσταλεί, έπειτα το τοπικό
```

```
     * (στον σταθμό εξέτασης) όνομα του αρχείου. Ο σταθμός εξέτασης στέλνει
το αρχείο ως πίνακα από bytes.
     * @param StatID Ο κωδικός του σταθμού εξέτασης.
     * @param Fn Το πλήρες όνομα με το οποίο θα αποθηκευτεί το αρχείο.
     * @param Dir Το όνομα του καταλόγου του σταθμού εξέτασης στον οποίο
βρίσκεται το αρχείο.
     * @param RFn Το όνομα του αρχείου στον σταθμό εξέτασης.
     * @return true αν η διαδικασία ολοκληρωθεί με επιτυχία, false
διαφορετικά.
     */
    public boolean GetFile (int StatID, String Fn, String Dir, String RFn)
    {
        String IP;
        boolean res;
        Socket soc;
        InputStream is;
        OutputStream os;
        FileOutputStream fil;
        byte[] buf;
        int br;
        String S;
        String SDir;
        buf = new byte[4096];
        IP = Clients.GetIP (StatID);
        res = true;
        try
        {
            res = true;
            soc = new Socket (IP, DaemonPort);
            is = soc.getInputStream ();
            os = soc.getOutputStream ();
            SendLine (os, Integer.toString (CommSendFile));
            SendLine (os, Dir);
            SendLine (os, RFn);
            os.flush ();
            SDir = Fn.substring (0, Fn.lastIndexOf ("/"));
            if (!new File (SDir).exists ())
                new File (SDir).mkdirs ();
            fil = new FileOutputStream (Fn);
            while ((br = is.read (buf)) != -1)
                fil.write (buf, 0, br);
            fil.close ();
            os.close ();
            is.close ();
            soc.close ();
        }
        catch (UnknownHostException e)
        {
            String Que = "Άκυρη IP Διεύθυνση. Δεν θα έπρεπε να συμβεί ποτέ.
" + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
        }
        catch (FileNotFoundException e)
        {
            String Que = "Το αρχείο που λαμβάνεται από το σταθμό εξέτασης
δεν δημιουργήθηκε. Δεν θα έπρεπε να συμβεί ποτέ.\n"
                + e.getMessage ();
            JOptionPane.showMessageDialog (null, Que, "Κρίσιμο Λάθος",
JOptionPane.ERROR_MESSAGE);
            System.exit (1);
```

```
        }
        catch (IOException e)
        {
           res = false;
           System.out.println ("Here:" + e.getLocalizedMessage ());
        }
        return res;
    }


     /**
     * Ανακτά τα TestSheets από όλους τους σταθμούς εξέτασης οι οποίοι έχουν
κληρωθεί να χρησιμοποιηθούν στην εξέταση. Τα αρχεία, στους
     * σταθμούς εξέτασης, βρίσκονται στην σχετική ρίζα με το κοινό όνομα
"TestFile.test". Η αποστολή των αρχείων πραγματοποιείται μέσω της
     * {@link #GetFile}
     * @param Files Ο πίνακας με τα πλήρη ονόματα αρχείων με τα οποία θα
σωθούν τα αρχεία που θα σταλθούν από τους σταθμούς εξέτασης.
     * @param Stations Ο πίνακας με τους κωδικούς των σταθμών εξέτασης από
τους οποίους θα ανακτηθούν τα αρχεία. Το μέγεθος και
     * οι θέσεις του πίνακα Stations είναι αντίστοιχα με αυτά του πίνακα
Files.
     * @return true αν η ανάκτηση ολοκληρωθεί με επιτυχία από όλους τους
σταθμούς, false διαφορετικά.
     */
    public boolean GetAllTestFiles (String[] Files, int[] Stations)
    {
        int i;
        boolean res;
        res = true;
        for (i = 0; i < Files.length; i++)
        {
            if (!GetFile (Stations[i], Files[i], ".", "TestFile.test"))
                res = false;
        }
        return res;
    }

     /**
        * Διαβάζει μία γραμμή ως string από ένα InputStream. Χρησιμοποιείται
προκειμένου τα sockets να διαβάζονται με InputStream για να
     * γίνεται και μεταφορά block από bytes.
        * @param is Το inputstream από το οποίο διαβάζονται bytes.
        * @return Το string που διαβάστηκε
        * @throws IOException Σε περίπτωση που τελειώσουν τα δεδομένα εισόδου
χωρίς να βρεθεί "\n" ή δημιουργηθεί πρόβλημα στην κλήση της
     * μεθόδου read.
        */
       private String GetLine (InputStream is) throws IOException
       {
            ByteArrayOutputStream buffer = new ByteArrayOutputStream ();
            int b;
            String str;
          while ((b = is.read()) != 0x0A)
          {
            if (b < 0)
            {
                  throw new IOException("Not enough socket input");
            }
            buffer.write(b);
          }
          str = new String (buffer.toByteArray(), "UTF-8");
```

```
            return  str;
        }

        /**
         * Η μέθοδος γράφει ένα string σε ένα OutputStream ως σύνολο από
bytes. Το string κωδικοποιείται κατά UTF-8 και στο τέλος του
     * προστίθεται η αλλαγή γραμμής.
         * @param os Το OutputStream στο οποίο θα γραφεί η γραμμή. Κανονικά,
πρέπει να ανήκει σε socket.
         * @param Line Το string στο οποίο θα προστεθεί η αλλαγη γραμμής και
θα γραφεί στο stream.
         * @throws IOException σε περίπτωση που υπάρχει πρόβλημα με το socket.
         */
        private void SendLine (OutputStream os, String Line) throws
IOException
        {
               byte[] buffer;
               try
               {
                      Line = Line + "\n";
                      buffer = Line.getBytes ("UTF-8");
                      os.write (buffer);
              os.flush ();
               }
               catch (UnsupportedEncodingException e)
               {
                      //Δεν θα συμβεί ποτέ..... (Ελπίζω!)
               }
    }
}
```

PACKAGE: org.gmele.dissertation.server.exception

CLASS: ESException

```
package org.gmele.dissertation.server.exception;

import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση χειρισμού εξαιρέσεων του Examination Server. Κληρονομεί την
{@link GmException} και υλοποιεί το interface
 * {@link ESExceptionConsts}. Η κλάση δεν προσθέτει λειτουργικότητα στην
κλάση που κληρονομεί.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class ESException extends GmException implements ESExceptionConsts
{
    /**
     * Κατασκευάζει το αντικείμενο για εξαίρεση η οποία δεν περιλαμβάνει
πληροφορία System Exception. Περνάει τις παραμέτρους στον
     * αντίστοιχο κατασκευαστή της κλάσης {@link GmException} χωρίς καμμία
επεξεργασία.
     * @param message Το μήνυμα λάθους.
     * @param errCode Ο κωδικός λάθους. Πρέπει να έχει οριστεί ως σταθερά
στο interface {@link ESExceptionConsts} ή στο
     * {@link GmExceptionConsts}.
```

```
     * @param data Έξτρα πληροφορίες για το λάθος σε μορφή string.
     */
    public ESException (String message, int errCode, String data)
    {
        super (message, errCode, data);
    }

    /**
     * Κατασκευάζει το αντικείμενο για εξαίρεση η οποία περιλαμβάνει
πληροφορία System Exception. Περνάει τις παραμέτρους στον αντίστοιχο
     * κατασκευαστή της κλάσης {@link GmException} χωρίς καμμία επεξεργασία.
     * @param message Το μήνυμα λάθους
     * @param errCode Ο κωδικός λάθους. Πρέπει να έχει οριστεί ως σταθερά
στο interface {@link ESExceptionConsts} ή στο
     * {@link GmExceptionConsts}.
     * @param data Έξτρα πληροφορίες για το λάθος σε μορφή string.
     * @param sysExc Η εξαίρεση, ο χειρισμός της οποίας, δημιουργεί την
τρέχουσα εξαίρεση.
     */
    public ESException (String message, int errCode, String data, Exception
sysExc)
    {
        super (message, errCode, data, sysExc);
    }
}
```

INTERFACE: ESExceptionConsts

```
package org.gmele.dissertation.server.exception;

import org.gmele.general.exceptions.GmExceptionConsts;

/**
 * Οι κωδικοί λαθών για το project ExamServer. Υλοποιείται από την κλάση
{@link ESException}. Δεν χρειάζεται να κληρονομεί το
 * {@link GmExceptionConsts}. Οι σταθερές του περνούν μέσω της
κληρονομικότητας των κλάσεων.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public interface ESExceptionConsts
{
    /**
     * Δεν μπορεί να διαβαστεί η εικόνα από τα resources διότι δεν υπάρχει ή
δημιουργήθηκε πρόβλημα κατά το διάβασμα.
     */
    final int RHIOErrorReading = 10000;

    /**
     * Το resource file που διαβάστηκε δεν είναι εικόνα ή είναι εικόνα με
ιδιότητες που δεν υποστηρίζονται.
     */
    final int RHNotImageFile = 10001;

    /**
     * Το αρχείο το οποίο έγινε προσπάθεια να διαβαστεί δεν υπάρχει στο
package των resources.
     */
    final int RHNoSuchResource = 10002;

    /**
```

```
     * Δεν υπάρχει τέτοιο κλειδί (key) στο συγκεκριμένο property αρχείο.
     */
    final int RHNoSuchPropertyKey = 10003;

    /**
     * Δημιουργήθηκε πρόβλημα κατά την ανάγνωση των δεδομένων για τους
σταθμούς εξέτασης. Περιλαμβάνει και την περίπτωση να δημιουργήθηκε
     * εξαίρεση κατά την προσπάθεια δημιουργίας νέου connection στη βάση
     */
    final int CSCannotReadClientTable = 10010;

}
```

PACKAGE: org.gmele.dissertation.server.gui

CLASS: ClientForm

```
package org.gmele.dissertation.server.gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.SwingConstants;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import org.gmele.dissertation.server.ClientStations;
import org.gmele.dissertation.server.ExamServer;
import org.gmele.dissertation.server.communications.Uhura;
import org.gmele.dissertation.server.exception.ESException;
import org.gmele.dissertation.server.resources.ResourceHandler;

/**
 * Η φόρμα χειρισμού των σταθμών εξέτασης.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public final class ClientForm extends JPanel implements ActionListener
{
    MainForm Father;
    ExamServer ES;
    ClientStations CL;
    Uhura Comm;
    JScrollPane Scroll;
    JTable Table;
    Object[][] Data;
    String[] Header;
    ImageIcon IconOK;
    ImageIcon IconErr;
    ImageIcon IconEmpty;
    JButton BtClose;

    /**
     * Κατασκευάζει τη φόρμα χειρισμού των σταθμών εξέτασης
```

```
     * @param fath Αναφορά προς τη βασική φόρμα του προγράμματος {@link
MainForm}
     */
    ClientForm (MainForm fath)
    {
        Father = fath;
        ES = Father.Father;
        CL = ES.getClients ();
        Comm = ES.getComm ();
        IconEmpty = null;
        try
        {
            IconOK = ResourceHandler.GetIcon ("OKPic.png");
            IconErr = ResourceHandler.GetIcon ("ErrPic.png");
        }
        catch (ESException e)
        {
            ES.UnexpectedSystemError ("Πρόβλημα κατά τη δημιουργία της
φόρμας των σταθμών εξέτασης (δεν βρέθηκαν εικονίδια).",
                e.getErrorCode ());
        }
        setLayout (null);
        setBounds (20, 20, 600, 550);
        MakeScrollPane ();
        MakeRest ();
        setOpaque (false);
        setVisible (true);
    }

    /**
     * Δημιουργεί το ScrollPane με τις πληροφορίες για τους σταθμούς
εξέτασης. Προκειμένου να δημιουργηθεί ο σχετικός πίνακας ελέγχεται
     * η επικοινωνία με τους ενεργούς σταθμούς με την αποστολή της εντολής
"HELLO" {@link Uhura#SendHello(int)}
     */
    void MakeScrollPane ()
    {
        int NoC;
        int i;
        Header = new String[5];
        Header[0] = "ΚΩΔΙΚΟΣ";
        Header[1] = "ΟΝΟΜΑ";
        Header[2] = "I.P. ΔΙΕΥΘΥΝΣΗ";
        Header[3] = "ΚΑΤΑΣΤΑΣΗ";
        Header[4] = "ΕΠΙΚΟΙΝΩΝΙΑ";
        NoC = CL.Numb;
        Data = new Object[NoC][5];
        for (i = 0; i < NoC; i++)
        {
            Data[i][0] = Integer.toString (CL.ID[i]);
            Data[i][1] = CL.Name[i];
            Data[i][2] = CL.IP[i];
            Data[i][3] = CL.Status[i];
            if (CL.Status[i].equals ("Active"))
                Data[i][4] = Comm.SendHello (CL.ID[i]) ? IconOK : IconErr;
            else
                Data[i][4] = IconEmpty;
        }
        DefaultTableModel model = new DefaultTableModel (Data, Header);
        Table = new JTable (model)
        {
            @Override
```

```
            public Class getColumnClass(int column)
            {
                return getValueAt (0, column).getClass();
            }
        };
        //Table.setPreferredScrollableViewportSize
(Table.getPreferredSize());
        Table.setLayout (null);
        Table.setAutoResizeMode (JTable.AUTO_RESIZE_OFF);
        Table.setRowHeight (50);
        Table.setColumnSelectionAllowed (false);
        Table.setRowSelectionAllowed (true);
        Table.getTableHeader ().setResizingAllowed (false);
        Table.getTableHeader ().setReorderingAllowed (false);
        Table.getColumnModel ().getColumn (0).setPreferredWidth (70);
        Table.getColumnModel ().getColumn (1).setPreferredWidth (160);
        Table.getColumnModel ().getColumn (2).setPreferredWidth (120);
        Table.getColumnModel ().getColumn (3).setPreferredWidth (100);
        Table.getColumnModel ().getColumn (4).setPreferredWidth (100);
        DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();
        dtcr.setHorizontalAlignment(SwingConstants.CENTER);
        Table.getColumnModel ().getColumn (0).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (1).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (2).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (3).setCellRenderer (dtcr);
        Scroll = new JScrollPane (Table);
        Scroll.setBounds (2, 1, 560, 400);
        Scroll.setOpaque (true);
        Scroll.setVisible (true);
        add (Scroll);
    }

    /**
     * Δημιουργεί όλα τα υπόλοιπα αντικείμενα της κλάσης ClientForm.
     */
    void MakeRest ()
    {
        BtClose = new JButton ("Εξοδος");
        BtClose.setBounds (250,450, 100, 50);
        BtClose.addActionListener (this);
        add (BtClose);
    }

    @Override
    public void actionPerformed (ActionEvent e)
    {
        if (e.getSource () == BtClose)
        {
            System.out.println ("**");
            Father.getContentPane ().remove (this);
            Father.repaint ();
        }
    }

    @Override
    protected void finalize ()
    {
        System.out.println ("*************************");
    }
}
```

CLASS: MainForm

```
package org.gmele.dissertation.server.gui;

import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSeparator;
import javax.swing.SwingConstants;
import javax.swing.UIManager;
import javax.swing.filechooser.FileNameExtensionFilter;
import org.gmele.dissertation.server.ExamServer;
import org.gmele.dissertation.server.exception.ESException;
import org.gmele.dissertation.server.resources.ResourceHandler;
import org.gmele.dissertation.testhandler.ExamDisDB;
import org.gmele.dissertation.testhandler.ExamStruct;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;


/**
 * Το user interface της εφαρμογής ExamServer
 * @author Giorgos Meletiou
 * @version 1.0
 */
public final class MainForm extends JFrame implements ActionListener
{
    public static int MouseNormal = 1;
    public static int MouseWait = 2;
    JMenuBar Men;
    JMenu MFile;
    JMenuItem MFExit;
    JMenu MExams;
    JMenuItem MEInsexam;
    JMenuItem MEAdmexams;
    JMenuItem MEDoexam;
    JMenuItem MEResults;
    JMenu MAdmin;
    JMenuItem MAClients;
    ExamServer Father;
    ImagePanel MainPanel;

    /**
     * Αρχικοποιεί το παράθυρο. Η αρχικοποίηση γίνεται με τη βοήθεια και
βοηθητικών μεθόδων.
     * @param Father Αναφορά σε αντικείμενο της βασικής κλάσης {@link
ExamServer}.
     */
    public MainForm (ExamServer father)
```

```
    {
        super ("Android Μ.Σ.Θ.Ε.Υ.Ο.");
        BufferedImage img;
        Father = father;
        Dimension dim = Toolkit.getDefaultToolkit ().getScreenSize ();
        setLayout (null);
        getContentPane ().setLayout (null);
        setSize (800, 600);
        setLocation ((dim.width - getWidth ()) / 2, (dim.height - getHeight
()) / 2);
        setResizable (false);
        try
        {
            img = ResourceHandler.GetImage ("ServIco.png");
            setIconImage (img);
        }
        catch (ESException e)
        {
            System.out.println (e.getMessage () + " " + e.getData () + " " +
e.getSysException ().getMessage ());
            System.exit (1);
        }
        MainPanel = new ImagePanel (this);
        setContentPane (MainPanel);
        CreateMenus ();
        setDefaultCloseOperation (EXIT_ON_CLOSE);
        setVisible (true);

    }

    /**
     * Δημιουργεί τη βασική γραμμή μενού του προγράμματος.
     */
    void CreateMenus ()
    {
        MFile = new JMenu ("Αρχείο");
        MFExit = new JMenuItem ("Έξοδος");
        MFile.add (MFExit);
        MExams = new JMenu ("Εξετάσεις");
        MEInsexam = new JMenuItem ("Είσαγωγή Ομάδας");
        MEAdmexams = new JMenuItem ("Ομάδες εξέτασης...");
        MEDoexam = new JMenuItem ("Διενέργεια εξέτασης");
        MEResults = new JMenuItem ("Αποτελέσματα εξέτασης...");
        MExams.add (MEInsexam);
        MExams.add (MEAdmexams);
        MExams.add (new JSeparator (SwingConstants.HORIZONTAL));
        MExams.add (MEDoexam);
        MExams.add (new JSeparator (SwingConstants.HORIZONTAL));
        MExams.add (MEResults);
        MAdmin = new JMenu ("Διαχείριση");
        MAClients = new JMenuItem ("Σταθμοί Εξέτασης");
        MAdmin.add (MAClients);
        Men = new JMenuBar ();
        Men.add (MFile);
        Men.add (MExams);
        Men.add (MAdmin);
        setJMenuBar (Men);
        MFExit.addActionListener (this);
        MEInsexam.addActionListener (this);
        MEDoexam.addActionListener (this);
        MEResults.addActionListener (this);
        MAClients.addActionListener (this);
```

```
    }

    /**
     * Ο χρειριστής των ActionEvents της κύριας φόρμας.
     * @param e Το αντικείμενο το οποίο δημιούργησε το event.
     */
    @Override
    public void actionPerformed (ActionEvent e)
    {
        if (e.getSource () == MFExit)
        {
            Father.EndProcs ();
        }
        if (e.getSource () == MAClients)
        {
            DoAdminClients ();
        }
        if (e.getSource () == MEDoexam)
        {
            DoExamGroup ();
        }
        if (e.getSource () == MEResults)
        {
            DoExamResults ();
        }
        if (e.getSource () == MEInsexam)
            DoInsertExam ();
    }

    /**
     * Ενεργοποιεί την φόρμα διαχείρισης σταθμών εξέτασης.
     */
    void DoAdminClients ()
    {
        ClientForm Cl = new ClientForm (this);
        getContentPane ().add (Cl);
        Cl.repaint ();
        System.out.println (Cl.getWidth () + " --- " + Cl.getHeight ());
    }

    /**
     * Ενεργοποιεί τη διαδικασία εξέτασης ομάδας
     */
    void DoExamGroup ()
    {
        Father.ExamGroup ();
    }

    /**
     * Υλοποιεί την επιλογή "Αποτελέσματα Εξέτασης" του μενού "Εξετάσεις".
Παρουσιάζει στο χρήστη ένα Combo Box με όλες τις ολοκληρωμένες
     * εξετάσεις και στην συνέχεια δημιουργεί ένα αντικείμενο {@link
ResultForm} για να παρουσιαστούν τα αποτελέσματα.
     *
     */
    void DoExamResults ()
    {
        int ExamPos;
        ExamStruct[] Groups;
        Groups = null;
        try
        {
```

```
            Groups = ExamDisDB.GetDoneExams ();
        }
        catch (TestException e)
        {
            Father.UnexpectedSystemError ("Λάθος κατά την ανάκτηση ομάδων
εξέτασης." , e.getErrorCode ());
        }
        catch (GmException e)
        {
            Father.UnexpectedSystemError ("Λάθος κατά την ανάκτηση ομάδων
εξέτασης." , e.getErrorCode ());
        }
        ExamPos = SelectExam (Groups, "Επιλέξτε ομάδα για παρουσίαση
αποτελεσμάτων.");
        if (ExamPos != -1)
        {
            ResultForm RF = new ResultForm (this, Groups[ExamPos].ID,
false);
            getContentPane ().add (RF);
            RF.repaint ();
        }
    }

    /**
     * Υλοποιεί την επιλογή "Εισαγωγή Ομάδας" του μενού "Εξετάσεις".
Παρουσιάζει στο χρήστη ένα διάλογο επιλογής ".xlsx" αρχείου και στην
     * συνέχεια καλεί την {@link ExamServer#LoadGroupData} για να
πραγματοποιηθεί η διαδικασία.
     *
     */
    void DoInsertExam ()
    {
        JFileChooser fc;
        FileNameExtensionFilter ff;
        String FN;
        int r;
        ff = new FileNameExtensionFilter ("Excel 2007 files", "xlsx");
        fc = new JFileChooser ();
        fc.setAcceptAllFileFilterUsed (false);
        fc.setDialogTitle ("Επιλογή Αρχείου Ομάδας Εξέτασης");
        fc.setFileFilter (ff);
        fc.setMultiSelectionEnabled (false);
        r = fc.showOpenDialog (this);
        if (r != JFileChooser.APPROVE_OPTION )
            return;
        FN = fc.getSelectedFile ().getAbsolutePath ();
        if (!FN.endsWith (".xlsx"))
            return;
        System.out.println (FN);
        Father.LoadGroupData (FN);
    }

    /**
     * Εμφανίζει ένα διάλογο επιλογής ομάδας εξέτασης. Ως στοιχεία ομάδας
φαίνεται το string το οποίο επιστρέφει η μέθοδος "toString"
     * της κλάσης ExamStruct.
     * @param Grps Ο πίνακας με τις ομάδες από τις οποίες θα γίνει η
επιλογή.
     * @param mess Το μήνυμα που θα εμφανιστεί.
     * @return Τη θέση της επιλεγμένης ομάδας εξέτασης στον πίνακα επιλογών
ή -1 αν ο χρήστης πάτησε ακύρωση ή ο πίνακας Grps έχει
     * μηδενικό μέγεθος.
```

```
     */
    public int SelectExam (ExamStruct[] Grps, String mess)
    {
        int i;
        ExamStruct s;
        UIManager.put("OptionPane.cancelButtonText", "Ακύρωση");
        UIManager.put("OptionPane.okButtonText", "O.K.");
        if (Grps.length == 0)
        {
            JOptionPane.showMessageDialog (this, "Δεν υπάρχουν κατάλληλες
ομάδες εξέτασης για επιλογή.", "Επιλογή Ομάδας Εξέτασης",
                JOptionPane.ERROR_MESSAGE);
            return -1;
        }
        s = (ExamStruct) JOptionPane.showInputDialog (this, mess, "Επιλογή
Ομάδας Εξέτασης", JOptionPane.QUESTION_MESSAGE,
            null, Grps, Grps[0].toString ());
        if (s == null)
            return -1;
        else
        {
            for (i = 0; i < Grps.length; i++)
                if (Grps[i] == s)
                    return i;
        }
        return -2;
    }

    /**
     * Αλλάζει το εικονίδιο του ποντικιού σύμφωνα με την παράμετρο που
δίδεται ως είσοδος. Οι αποδεκτές τιμές έχουν οριστεί ως σταθερές.
     * Οι μη αποδεκτές τιμές αγνοούνται.
     * @param st Η κατάσταση στην οποία θα αντιστοιχεί το εικονίδιο το οποίο
θα οριστεί.
     */
    public void SetCursor (int st)
    {
        if (st == MouseNormal)
            setCursor(new Cursor  (Cursor.DEFAULT_CURSOR));
        if (st == MouseWait)
            setCursor(new Cursor  (Cursor.WAIT_CURSOR));
    }
}


/**
 * Το ImagePanel χρησιμοποιείται προκειμένου να ζωγραφίζεται το background
του υπουργείου στη βασική φόρμα της εφαρμογής. Η εικόνα που
 * ζωγραφίζεται έχει το όνομα "server.jpg" και πρέπει να έχει το σωστό
μέγεθος.
 * @author Giorgos Meletiou
 * @version 1.0
 */
class ImagePanel extends JPanel
{

    Image img;

    public ImagePanel (MainForm Fath)
    {
        try
        {
```

```
            img = ResourceHandler.GetImage ("server.jpg");
        }
        catch (ESException ex)
        {
            System.out.println ("Error in MainForms Picture");
            System.exit (1);
        }
        Dimension size = Fath.getContentPane ().getSize ();
        setSize (size);
        setLayout (null);
    }

    @Override
    public void paintComponent (Graphics g)
    {

        g.drawImage (img, 0, 0, null);
    }
}
```

CLASS: ResultForm

```
package org.gmele.dissertation.server.gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.ScrollPaneConstants;
import javax.swing.SwingConstants;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import org.gmele.dissertation.server.ExamServer;
import org.gmele.dissertation.testhandler.ExamDisDB;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;

/**
 * Παρουσιάζει τα αποτελέσματα της εξέτασης στο βασικό παράθυρο της
εφαρμογής.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public final class ResultForm extends JPanel implements ActionListener
{
    MainForm Father;
    ExamServer ES;
    int ExamID;
    boolean AutoPrint;
    JTable Table;
    JScrollPane Scroll;
    JButton BtPrint;
    JButton BtClose;
    String[][] Data;
    String[] Header;
    int Wid;
    int Hei;
```

```
    /**
     * Δημιουργεί το Panel το οποίο θα παρουσιάζει τα αποτελέσματα μίας
εξέτασης.
     * @param fath Το βασικό παράθυρο της εφαρμογής.
     * @param eid Ο κωδικός της εξέτασης της οποίας τα αποτελέσματα θα
παρουσιαστούν.
     * @param autoprint true αν πρέπει να τυπωθούν τα αποτελέσματα
κατευθείαν, false αν θα εμφανιστούν σχετικά κουμπιά στη οθόνη.
     */
    public ResultForm (MainForm fath, int eid, boolean autoprint)
    {
        Father = fath;
        ES = Father.Father;
        ExamID = eid;
        AutoPrint = autoprint;
        setLayout (null);
        Wid = Father.getContentPane ().getWidth ();
        Hei = Father.getContentPane ().getHeight ();
        setBounds (0, 0, Wid, Hei);
        MakeScrollPane ();
        MakeRest ();
        setOpaque (true);
        setVisible (true);
    }

    /**
     * Δημιουργεί το ScrollPane με τα αποτελέσματα της εξέτασης. Τα
αποτελέσματα επιστρέφονται από την {@link ExamDisDB#GetExamResults}.
     */
    void MakeScrollPane ()
    {
        int i;
        Header = new String[5];
        Header[0] = "A/A";
        Header[1] = "ΕΠΩΝΥΜΟ";
        Header[2] = "ΟΝΟΜΑ";
        Header[3] = "ΚΑΤΗΓΟΡΙΑ";
        Header[4] = "ΑΠΟΤΕΛΕΣΜΑ";
        try
        {
            Data = ExamDisDB.GetExamResults (ExamID);
            for (i = 0; i < Data.length; i++)
            {
                if (Data[i][4].equals ("Pass"))
                    Data[i][4] = "ΕΠΕΤΥΧΕ";
                if (Data[i][4].equals ("Fail"))
                    Data[i][4] = "ΑΠΟΡΡΙΦΘΗΚΕ";
                if (Data[i][4].equals ("Absent"))
                    Data[i][4] = "ΑΠΟΥΣΙΑΖΕ";
            }
        }
        catch (TestException e)
        {
            ES.UnexpectedSystemError ("Λάθος κατά την ανάκτηση αποτελεσμάτων
εξέτασης: " + e.getData (), e.getErrorCode ());
        }
        catch (GmException e)
        {
            ES.UnexpectedSystemError ("Λάθος κατά την ανάκτηση αποτελεσμάτων
εξέτασης: " + e.getData (), e.getErrorCode ());
        }
        DefaultTableModel model = new DefaultTableModel (Data, Header);
```

```
        Table = new JTable (model)
        {
            @Override
            public Class getColumnClass(int column)
            {
                return getValueAt (0, column).getClass();
            }
        };
        //Table.setPreferredScrollableViewportSize
(Table.getPreferredSize());
        Table.setLayout (null);
        Table.setAutoResizeMode (JTable.AUTO_RESIZE_OFF);
        Table.setRowHeight (30);
        Table.setColumnSelectionAllowed (false);
        Table.setRowSelectionAllowed (false);
        Table.getTableHeader ().setResizingAllowed (false);
        Table.getTableHeader ().setReorderingAllowed (false);
        Table.getColumnModel ().getColumn (0).setPreferredWidth (50);
        Table.getColumnModel ().getColumn (1).setPreferredWidth (220);
        Table.getColumnModel ().getColumn (2).setPreferredWidth (220);
        Table.getColumnModel ().getColumn (3).setPreferredWidth (160);
        Table.getColumnModel ().getColumn (4).setPreferredWidth (120);
        DefaultTableCellRenderer dtcr = new DefaultTableCellRenderer();
        dtcr.setHorizontalAlignment(SwingConstants.CENTER);
        Table.getColumnModel ().getColumn (0).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (1).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (2).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (3).setCellRenderer (dtcr);
        Table.getColumnModel ().getColumn (4).setCellRenderer (dtcr);
        Scroll = new JScrollPane (Table);
        Scroll.setHorizontalScrollBarPolicy
(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        Scroll.setVerticalScrollBarPolicy
(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        Scroll.setBounds (2, 1, Wid - 4, Hei - 90);
        Scroll.setOpaque (true);
        Scroll.setVisible (true);
        add (Scroll);
    }

    void MakeRest ()
    {
        int Z;
        Z = (Wid - 240) / 3;
        if (!AutoPrint)
        {
            BtPrint = new JButton ("ΕΚΤΥΠΩΣΗ");
            BtPrint.setBounds (Z, Hei - 70, 120, 50);
            BtPrint.addActionListener (this);
            add (BtPrint);
        }
        else
            PrintResults ();
        BtClose = new JButton ("ΕΞΟΔΟΣ");
        BtClose.setBounds (2 * Z + 120, Hei - 70, 120, 50);
        BtClose.addActionListener (this);
        add (BtClose);
    }

    @Override
    public void actionPerformed (ActionEvent e)
    {
```

```
        if (e.getSource () == BtClose)
        {
            Father.getContentPane ().remove (this);
            Father.repaint ();
        }
        if (e.getSource () == BtPrint)
            PrintResults ();
    }

    void PrintResults ()
    {
        JOptionPane.showMessageDialog (this, "Η εκτύπωση αποτελεσμάτων
ολοκληρώθηκε.", "Εκτύπωση Αποτελεσμάτων",
                JOptionPane.INFORMATION_MESSAGE);
    }
}
```

PACKAGE: org.gmele.dissertation.server.resources

CLASS: ResourceHandler

```
package org.gmele.dissertation.server.resources;

import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import org.gmele.dissertation.server.exception.ESException;

/**
 * Η κλάση ResourceHandler παρέχει βασικές μεθόδους για ανάκτηση πόρων
(resources) της εφαρμογής..
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class ResourceHandler
{
      /**
     * Επιστρέφει ένα InputStream από το οποίο μπορεί να διαβαστεί το
περιεχόμενο ενός πόρου.
     * @param name Το όνομα του αρχείου στο package των πόρων.
     * @return Το InputStream του πόρου.
     */
    public static InputStream GetStream (String name)
    {
            return ResourceHandler.class.getResourceAsStream (name);
    }

    /**
     * Επιστρέφει μία εικόνα η οποία βρίσκεται στα resources ως
BufferedImage
     * @param name Το όνομα του αρχείου.
     * @return Το αρχείο ως εικόνα.
```

```
     * @throws ESException Σε περίπτωση που το αρχείο δεν υπάρχει ή δεν
είναι κατάλληλου τύπου το περιεχόμενό του.
     */
    public static BufferedImage GetImage (String name) throws ESException
    {
        BufferedImage image = null;
        try
        {
            image = ImageIO.read (ResourceHandler.class.getResource (name));
            if (image == null)
                throw new ESException ("Resource File not Image",
ESException.RHNotImageFile, name);
            return image;
        }
        catch (IOException e)
        {
            throw new ESException ("Cannot read Image from resources1",
ESException.RHIOErrorReading, name, e);
        }
        catch (IllegalArgumentException e)
        {
            throw new ESException ("Resource File not Image2",
ESException.RHIOErrorReading, name, e);
        }
    }

    /**
     * Επιστρέφει μία εικόνα η οποία βρίσκεται στα resources ως ImageIcon.
     * @param name Το όνομα του αρχείου.
     * @return Το αρχείο ως εικονίδιο.
     * @throws ESException Σε περίπτωση που το αρχείο δεν υπάρχει ή δεν
είναι κατάλληλου τύπου το περιεχόμενό του.
     */
    public static ImageIcon GetIcon (String name) throws ESException
    {
        Image image = null;
        try
        {
            image = ImageIO.read (ResourceHandler.class.getResource (name));
            if (image == null)
                throw new ESException ("Resource File not Image",
ESException.RHNotImageFile, name);
            return new ImageIcon (image);
        }
        catch (IOException e)
        {
            throw new ESException ("Cannot read Image from resources1",
ESException.RHIOErrorReading, name, e);
        }
        catch (IllegalArgumentException e)
        {
            throw new ESException ("Resource File not Image2",
ESException.RHIOErrorReading, name, e);
        }
    }

    /**
     * Διαβάζει την τιμή ενός κλειδιού σε ένα property file το οποίο
βρίσκεται στα resources.
     * @param File Το όνομα του property file. Στο όνομα προστίθεται η
επέκταση ".properties".
     * @param Key Το κλειδί του οποίου η τιμή αναζητείται.
```

```
     * @return Η τιμή του κλειδιού.
     * @throws ESException Σε περίπτωση που δεν υπάρχει το αρχείο, το κλειδί
ή δημιουργηθεί IO εξαίρεση κατά την ανάγνωση του αρχείου.
     */
    public static String GetProperty (String File, String Key) throws
ESException
    {
        Properties prop;
        InputStream is;
        String Value;
        try
        {
            is = GetStream (File + ".properties");
            if (is == null)
                throw new ESException ("Property File does not exist.",
ESException.RHNoSuchResource, File);
            prop = new Properties ();
            prop.load (is);
            Value = prop.getProperty (Key);
            if (Value == null)
                throw new ESException ("Property Key does not exist.",
ESException.RHNoSuchPropertyKey, Key);
            return Value;
        }
        catch (IOException e)
        {
            throw new ESException ("Unexpected IO Error in reading property
file", ESException.RHIOErrorReading, File, e);
        }
    }
}
```

PACKAGE: org.gmele.dissertation.testhandler

CLASS: ExamDisDB

```
package org.gmele.dissertation.testhandler;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.dbconnections.DBSimlpeConn;
import org.gmele.general.exceptions.GmException;

/**
 * Η ExamDisDB χρησιμοποιείται για την πρόσβαση στη Βάση Δεδομένων ExamDisDB
οι οποία περιέχει όλους τους πίνακες για τις εξετάσεις. Για
 * λόγους απλότητας της ανάπτυξης και κατά παράβαση του σωστού σχεδιασμού,
χρησιμοποιείται και για πρόσβαση στους πίνακες που στο αρχικό
 * ΜΣΘΕΥΟ βρίσκονται στην MainBase, με αποτέλεσμα η κλάση να έχει κώδικα ο
οποίος δεν χρειάζεται στους σταθμούς εξέτασης αν ενσωματωθεί και
```

```
 * στην android εφαρμογή.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class ExamDisDB
{
    /**
     * Επιστρέφει τον αριθμών των ερωτήσεων ανά εσωτερική κατηγορία που
πρέπει να έχει ένα ερωτηματολόγιο για μία συγκεκριμένη κατηγορία
     * εξέτασης
     * @param Kateg Η κατηγορία εξέτασης (Αυτοκίνητα, μοτοσυκλέτες, κλπ)
     * @return Ο πίνακας με τους αριθμούς ερωτήσεων ανά κατηγορία
     * @throws GmException Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection με τη Βάση Δεδομένων.
     * @throws TestException Σε περίπτωση που δεν υπάρχουν στοιχεία για την
κατηγορία ή συμβεί SQL εξαίρεση.
     */
    public static int[] GetInterKatNumbs (int Kateg) throws TestException,
GmException
    {

        int[] Nums;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT PCod, Numb FROM Numbers Where KCod = ? ORDER BY PCod";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Kateg);
            rs = st.executeQuery ();
            if (!rs.last ())
                throw new TestException ("No Internal Kategory Numbers",
TestException.EDNoDataForCategory, Integer.toString (Kateg));
            Nums = new int[rs.getRow ()];
            rs.beforeFirst ();
            i = 0;
            while (rs.next ())
                Nums[i++] = rs.getInt ("Numb");
            rs.close ();
            st.close ();
            Conn.close ();
            return Nums;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Internal Kategory
Numbers", TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Βρίσκει όλες τις ερωτήσεις μίας συγκεκριμένης εσωτερικής κατηγορίας
και επιστρέφει τους κωδικούς των.
     * @param Kateg Η κατηγορία εξέτασης.
     * @param Inter Η εσωτερική κατηγορία.
     * @param Lang Η γλώσσα για την οποία θα βρεθούν οι ερωτήσεις.
     * @return Οι κωδικοί των ερωτήσεων που πληρούν τους όρους.
```

```
     * @throws GmException Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection με τη Βάση Δεδομένων.
     * @throws TestException Σε περίπτωση που δεν υπάρχουν ερωτήσεις με τα
χαρακτηριστικά αυτά ή που δημιουργηθεί SQL Εξαίρεση.
     */
    public static int[] GetInterKategQCods (int Kateg, int Inter, int Lang)
throws TestException, GmException
    {
        int[] Nums;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT QCod FROM Questions Where QKateg = ? AND QPag = ? AND
QLang = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Kateg);
            st.setInt (2, Inter);
            st.setInt (3, Lang);
            rs = st.executeQuery ();
            if (!rs.last ())
                throw new TestException ("No Such Questions",
TestException.EDNoSuchQuestions, "[" + Kateg + " " + Inter + " " + Lang +
"]");
            Nums = new int[rs.getRow ()];
            rs.beforeFirst ();
            i = 0;
            while (rs.next ())
                Nums[i++] = rs.getInt ("QCod");
            rs.close ();
            st.close ();
            Conn.close ();
            return Nums;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in finding Questions of a
specific internal kategory", TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τις πληροφορίες μίας ερώτησης και των απαντήσεων της σε
ένα αντικείμενο τύπου {@link Question}. Οι απαντήσεις δεν είναι
     * ανακατεμένες.
     * @param QID Ο κωδικός της ερώτησης
     * @return Τις πληροφορίες της ερώτησης που σχετίζονται με την κλάση
{@link Question}
     * @throws GmException Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection με τη Βάση Δεδομένων.
     * @throws TestException Σε περίπτωση που δεν υπάρχουν ερωτήσεις ή
απαντήσεις με το συγκεκριμένο κωδικό ή δημιουργηθεί SQL εξαίρεση
     */
    public static Question GetFullQuestion (int QID) throws TestException,
GmException
    {
        Question Q;
```

```
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Q = new Question ();
        Que = null;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        try
        {
            Que = "SELECT * FROM Questions Where QCod = ?";
            st = Conn.prepareStatement (Que);
            st.setInt (1, QID);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("No Such Question",
TestException.EDNoSuchQuestions, "[" + QID + "]");
            Q.Code = rs.getInt ("QCod");
            Q.Verbal = rs.getString ("QLect");
            Q.Photo = rs.getString ("QPhoto");
            Q.Sound = rs.getString ("QSound");
            Q.Book = rs.getString ("QBook");
            rs.close ();
            st.close ();
            Que = "SELECT * FROM Answers WHERE AQCod = ? ORDER BY AAA";
            st = Conn.prepareStatement (Que);
            st.setInt (1, QID);
            rs = st.executeQuery ();
            if (!rs.last ())
                throw new TestException ("No Such Answers",
TestException.EDNoSuchAnswers, "[" + QID + "]");
            Q.NoAns = rs.getRow ();
            Q.AnsAA = new int[Q.NoAns];
            Q.AnsVerbal = new String[Q.NoAns];
            Q.AnsSound = new String[Q.NoAns];
            Q.Choice = 0;
            rs.beforeFirst ();
            i = 0;
            while (rs.next ())
            {
                Q.AnsAA[i] = rs.getInt ("AAA");
                Q.AnsVerbal[i] = rs.getString ("ALect");
                Q.AnsSound[i] = rs.getString ("ASound");
                i++;
            }
            rs.close ();
            st.close ();
            Conn.close ();
            return Q;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting a full Question",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Διαβάζει από τη Βάση Δεδομένων τις ομάδες εξέτασης οι οποίες έχουν
προγραμματιστεί για την ημέρα την οποία καλείται και βρίσκονται σε
     * κατάσταση "Pending" και τις επιστρέφει ως πίνακα {@link ExamStruct}
ταξινομημένο κατά ExamGroup. Αν δεν υπάρχουν τέτοιες,
```

```
     * επιστρέφεται κενός πίνακας (length = 0).
     * @return Ο πίνακας με τις ομάδες εξέτασης.
     * @throws GmException Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection με τη Βάση Δεδομένων.
     * @throws TestException Σε περίπτωση που συμβεί SQL εξαίρεση.
     */
    public static ExamStruct[] GetTodaysPendingExams () throws
TestException, GmException
    {
        ExamStruct[] Grps;
        String Que;
        String Today;
        SimpleDateFormat formatter;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT * FROM Examinations WHERE Date (ExamDate) = ? AND
ExamStatus = 'Pending' ORDER BY ExamGroup";
        try
        {
            st = Conn.prepareStatement (Que);
            formatter = new SimpleDateFormat("yyyy-MM-dd");
            Today = formatter.format (new java.util.Date ());
            st.setString (1, Today);
            rs = st.executeQuery ();
            if (!rs.last ())
                Grps = new ExamStruct[0];
            else
            {
                Grps = new ExamStruct[rs.getRow ()];
                rs.beforeFirst ();
                i = 0;
                while (rs.next ())
                {
                    Grps[i] = new ExamStruct ();
                    Grps[i].ID = rs.getInt ("ExamID");
                    Grps[i].Group = rs.getInt ("ExamGroup");
                    Grps[i].DateTime = rs.getTimestamp ("ExamDate");
                    Grps[i].Status = rs.getString ("ExamStatus");
                    i++;
                }
            }
            rs.close ();
            st.close ();
            Conn.close ();
            return Grps;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Todays Exam groups",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Διαβάζει από τη Βάση Δεδομένων τις ομάδες εξέτασης οι οποίες έχουν
ολοκληρωθεί (κατάσταση "Done" και τις επιστρέφει ως πίνακα
     * {@link ExamStruct} ταξινομημένο κατά φθίνουσα τάξη ExamID. Αν δεν
υπάρχουν τέτοιες, επιστρέφεται κενός πίνακας (length = 0).
     * @return Ο πίνακας με τις ομάδες εξέτασης.
```

```
     * @throws GmException Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection με τη Βάση Δεδομένων.
     * @throws TestException Σε περίπτωση που συμβεί SQL εξαίρεση.
     */
    public static ExamStruct[] GetDoneExams () throws TestException,
GmException
    {
        ExamStruct[] Grps;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT * FROM Examinations WHERE ExamStatus = 'Done' ORDER BY
ExamID DESC";
        try
        {
            st = Conn.prepareStatement (Que);
            rs = st.executeQuery ();
            if (!rs.last ())
                Grps = new ExamStruct[0];
            else
            {
                Grps = new ExamStruct[rs.getRow ()];
                rs.beforeFirst ();
                i = 0;
                while (rs.next ())
                {
                    Grps[i] = new ExamStruct ();
                    Grps[i].ID = rs.getInt ("ExamID");
                    Grps[i].Group = rs.getInt ("ExamGroup");
                    Grps[i].DateTime = rs.getTimestamp ("ExamDate");
                    Grps[i].Status = rs.getString ("ExamStatus");
                    i++;
                }
            }
            rs.close ();
            st.close ();
            Conn.close ();
            return Grps;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Done Exam groups",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Αναζητεί και επιστρέφει τα στοιχεία των εξεταζόμενων μίας εξέτασης.
Τα στοιχεία επιστρέφονται ως πίνακας {@link PeopleStruct}.
     * @param ExamID Ο κωδικός της εξέτασης.
     * @return Ο πίνακας με τους εξεταζόμενους
     * @throws TestException Σε περίπτωση που δεν βρεθούν εξεταζόμενοι για
τον συγκεκριμένο κωδικό εξέτασης ή δημιουργηθεί SQL εξαίρεση.
     * @throws GmException  Σε περίπτωση που δεν μπορεί να δημιουργηθεί
connection στη Βάση Δεδομένων.
     */
    public static PeopleStruct[] GetPeopleInExam (int ExamID) throws
TestException, GmException
    {
```

```
        PeopleStruct[] Peo;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT * FROM People WHERE ExamID = ? ORDER BY Surname,
Firstname, Fathername";
        try
        {
            st = Conn.prepareStatement (Que);
            st.setInt (1, ExamID);
            rs = st.executeQuery ();
            if (!rs.last ())
                throw new TestException ("Error in getting Examinee's Data",
TestException.EDNoPeopleInExam, " " + ExamID);
            else
            {
                Peo = new PeopleStruct[rs.getRow ()];
                rs.beforeFirst ();
                i = 0;
                while (rs.next ())
                {
                    Peo[i] = new PeopleStruct ();
                    Peo[i].PID = rs.getInt ("PID");
                    Peo[i].Surname = rs.getString ("Surname");
                    Peo[i].Firstname = rs.getString ("Firstname");
                    Peo[i].Fathername = rs.getString ("Fathername");
                    Peo[i].IDNumber = rs.getString ("IDNumber");
                    Peo[i].ExamID = rs.getInt ("ExamID");
                    Peo[i].ExamKateg = rs.getInt ("ExamKateg");
                    Peo[i].ExamLang = rs.getInt ("ExamLang");
                    Peo[i].Speech = rs.getBoolean ("Speech");
                    Peo[i].Result = rs.getString ("Result");
                    i++;
                }
            }
            rs.close ();
            st.close ();
            Conn.close ();
            return Peo;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Examinee's Data",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τον μέγιστο χρόνο εξέτασης για μία κατηγορία εξέτασης.
     * @param Kateg Η κατηγορία εξέτασης
     * @return Ο χρόνος εξέτασης σε πρώτα λεπτά της ώρας.
     * @throws TestException Σε περίπτωση που δεν βρεθούν στοιχεία για την
κατηγορία ή η δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί connection στη Βάση
Δεδομένων.
     */
    public static int GetExamTime (int Kateg) throws TestException,
GmException
    {
```

```
        int Durat;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT KTime FROM Kateg WHERE KCod = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Kateg);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Examination Time
for Kategory", TestException.EDNoDataForCategory, " " + Kateg);
            Durat = rs.getInt ("KTime");
            rs.close ();
            st.close ();
            Conn.close ();
            return Durat;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Examination Time for
Kategory", TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει το λεκτικό μίας κατηγορίας εξέτασης.
     * @param Kateg Ο κωδικός της κατηγορίας.
     * @return Το λεκτικό της κατηγορίας.
     * @throws TestException Σε περίπτωση που δεν βρεθεί η κατηγορία ή
δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί connection στη Βάση
Δεδομένων.
     */
    public static String GetKategoryVerbal (int Kateg) throws TestException,
GmException
    {
        String Verb;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT KLect FROM Kateg WHERE KCod = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Kateg);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Verbal for
Kategory", TestException.EDNoDataForCategory, " " + Kateg);
            Verb = rs.getString ("KLect");
            rs.close ();
            st.close ();
```

```
            Conn.close ();
            return Verb;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Verbal for Kategory",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τον κωδικό της κατηγορίας εξέτασης που αντιστοιχεί σε
συγκεκριμένο λεκτικό.
     * @param Ver Το λεκτικό της κατηγορίας εξέτασης.
     * @return Ο κωδικός της κατηγορίας εξέτασης.
     * @throws TestException Αν δεν βρέθηκε σχετική εξέταση ή δημιουργηθεί
SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση με τη Βάση
Δεδομένων.
     */
    public static int GetKategoryCode (String Ver) throws TestException,
GmException
    {
        int Cod;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT KCod FROM Kateg WHERE KLect = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setString (1, Ver);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Code for
Kategory", TestException.EDNoDataForCategory, " " + Ver);
            Cod = rs.getInt ("KCod");
            rs.close ();
            st.close ();
            Conn.close ();
            return Cod;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Code for Kategory",
TestException.EDSqlException, Que, e);
        }
    }

        /**
     * Επιστρέφει το λεκτικό μίας γλώσσας.
     * @param Lang Ο κωδικός της γλώσσας.
     * @return Το λεκτικό της γλώσσας.
     * @throws TestException Σε περίπτωση που δεν βρεθεί η γλώσσα ή
δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί connection στη Βάση
Δεδομένων.
     */
```

```
    public static String GetLanguageVerbal (int Lang) throws TestException,
GmException
    {
        String Verb;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT LLect FROM Langs WHERE LCod = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Lang);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Verbal for
Language", TestException.EDNoDataForLanguage, " " + Lang);
            Verb = rs.getString ("LLect");
            rs.close ();
            st.close ();
            Conn.close ();
            return Verb;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Verbal for Language",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τον Αύξοντα Αριθμό της σωστής απάντησης μίας ερώτησης.
     * @param QID Ο κωδικός της ερώτησης.
     * @return Ο Αύξοντας Αριθμός της σωστής απάντησης
     * @throws TestException Σε περίπτωση που δεν υπάρχει η ερώτηση ή δεν
έχει καταγραφεί σωστή απάντηση για αυτή ή συμβεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση στη Βάση.
     */
    public static int GetCorrectAns (int QID) throws TestException,
GmException
    {
        int CorrA;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT AAA FROM Answers WHERE AQCod = ? AND ACorr = 1";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, QID);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Correct Answer.",
TestException.EDNoSuchAnswers, " " + QID);
            CorrA = rs.getInt ("AAA");
            rs.close ();
            st.close ();
```

```
            Conn.close ();
            return CorrA;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Correct Answer",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τον μέγιστο επιτρεπτό αριθμό λαθών ώστε η εξέταση μίας
κατηγορίας να θεωρηθεί επιτυχής.
     * @param Kateg Ο κωδικός της κατηγορίας.
     * @return Ο μέγιστος επιτρεπτός αριθμός λαθών για την κατηγορία
     * @throws TestException Σε περίπτωση ανύπαρκτης κατηγορίας ή SQL
εξαίρεσης.
     * @throws GmException Σε περίπτωση προβλήματος δημιουργίας σύνδεσης με
τη Βάση Δεδομένων.
     */
    public static int GetKategAErrors (int Kateg) throws TestException,
GmException
    {
        int Errs;
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT AllowedErrors FROM Kateg WHERE KCod = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setInt (1, Kateg);
            rs = st.executeQuery ();
            if (!rs.first ())
                throw new TestException ("Error in Getting Allowed Errors.",
TestException.EDNoDataForCategory, " " + Kateg);
            Errs = rs.getInt ("AllowedErrors");
            rs.close ();
            st.close ();
            Conn.close ();
            return Errs;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Getting Allowed Errors",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Καταγράφει το αποτέλεσμα της εξέτασης για έναν εξεταζόμενο στον
πίνακα People της Βάσης Δεδομένων. Το αποτέλεσμα πρέπει να είναι
     * ένα από τα ακόλουθα: "Pass", "Fail", "Absent", "-".
     * @param ID Ο κωδικός του εξεταζόμενου.
     * @param Result Το αποτέλεσμα που θα καταγραφεί.
     * @throws TestException Σε περίπτωση άκυρου κωδικού ή SQL εξαίρεσης.
     * @throws GmException Σε περίπτωση προβλήματος σύνδεσης με τη Βάση
Δεδομένων.
     */
```

```
    public static void SetPeopleResult (int ID, String Result) throws
TestException, GmException
    {
        String Que;
        Connection Conn;
        PreparedStatement st;
        int rs;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "UPDATE People SET Result = ? WHERE PID = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setString (1, Result);
            st.setInt (2, ID);
            rs = st.executeUpdate ();
            if (rs != 1)
                throw new TestException ("Error in Setting People Result",
TestException.EDNoSuchPeople, " " + ID);
            st.close ();
            Conn.close ();
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in Setting People Result",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Αλλάζει την κατάσταση (status) μίας ομάδας εξέτασης στον πίνακα
Examinations της Βάσης Δεδομένων. Αποδεκτά λεκτικά κατάστασης είναι
     * τα: "Pending", "Active", "Done", "Aborted", "Cancelled"
     * @param ID
     * @param Status
     * @throws TestException
     * @throws GmException
     */
    public static void SetExamStatus (int ID, String Status) throws
TestException, GmException
    {
        String Que;
        Connection Conn;
        PreparedStatement st;
        int rs;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "UPDATE Examinations SET ExamStatus = ? WHERE ExamID = ?";
        try
        {

            st = Conn.prepareStatement (Que);
            st.setString (1, Status);
            st.setInt (2, ID);
            rs = st.executeUpdate ();
            if (rs != 1)
                throw new TestException ("Error in Setting Examination
Status", TestException.EDNoSuchExam, " " + ID);
            st.close ();
            Conn.close ();
        }
        catch (SQLException e)
        {
```

```
            throw new TestException ("Error in Setting Examination Status",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Δημιουργεί ένα δισδιάστατο πίνακα με Strings ο οποίος σε κάθε του
γραμμή περιέχει τα στοιχεία ενός εξεταζόμενου που εμφανίζονται σε
     * πίνακα αποτελεσμάτων. Ο πίνακας είναι έτοιμος να μετατραπεί σε
JTable. Περιλαμβάνει τα ακόλουθα στοιχεία:
     * <ul>
     * <li> Α/Α.
     * <li> Επώνυμο.
     * <li> Όνομα.
     * <li> Λεκτικό κατηγορίας εξέτασης.
     * <li> Αποτέλεσμα.
     * </ul>
     * @param ExamID Ο κωδικός της εξέτασης.
     * @return Ο πίνακας των αποτελεσμάτων.
     * @throws TestException Σε περίπτωση που επιστρέφεται κενό result set
(δεν υπάρχει ο κωδικός εξέτασης ή δεν υπάρχουν εξεταζόμενοι σε
     * αυτή την εξέταση) ή δημιουργηθεί SQL εξαίρεση. Κανονικά δεν πρέπει να
συμβεί τίποτα από τα δύο.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση με τη Βάση
Δεδομένων.
     */
    public static String[][] GetExamResults (int ExamID) throws
TestException, GmException
    {
        String[][] Res;
        String Que;
        Connection Conn=null;
        PreparedStatement st;
        ResultSet rs;
        int i;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT Surname, Firstname, Kateg.KLect, Result FROM People,
Kateg WHERE People.ExamKateg = Kateg.KCod AND ExamID = ? " +
            "ORDER BY Surname, Firstname";
        try
        {
            st = Conn.prepareStatement (Que);
            st.setInt (1, ExamID);
            rs = st.executeQuery ();
            if (!rs.last ())
                throw new TestException ("Error in getting Examinee's
Results", TestException.EDNoPeopleInExam, " " + ExamID);
            else
            {
                Res = new String[rs.getRow ()][5];
                rs.beforeFirst ();
                i = 0;
                while (rs.next ())
                {
                    Res[i][0] = Integer.toString (i + 1);
                    Res[i][1] = rs.getString ("Surname");
                    Res[i][2] = rs.getString ("Firstname");
                    Res[i][3] = rs.getString ("KLect");
                    Res[i][4] = rs.getString ("Result");
                    i++;
                }
            }
```

```
            rs.close ();
            st.close ();
            Conn.close ();
            return Res;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Examinee's Results",
TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Επιστρέφει τον κωδικό μίας ομάδας εξέτασης από τον πίνακα
Examinations η οποία προσδιορίζεται από την ημερομηνία διεξαγωγής της και
     * το group μέσα στη μέρα. Η ώρα εξέτασης δεν λαμβάνεται υπόψη. Σε
περίπτωση που η ομάδα δεν βρεθεί επιστρέφεται -1.
     * @param Dat Η ημερομηνία της εξέτασης σε μορφή Date.
     * @param Grp Ο αριθμός της ομάδας μέσα στη μέρα.
     * @return Ο κωδικός της εξέτασης ή -1 αν δεν βρέθηκε αντίστοιχη
εξέταση.
     * @throws TestException Σε περίπτωση που δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση με τη Βάση
Δεδομένων.
     */
    public static int GetGroupID (Date Dat, int Grp) throws TestException,
GmException
    {
        String Que;
        String D;
        SimpleDateFormat formatter;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int ID;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "SELECT * FROM Examinations WHERE Date (ExamDate) = ? AND
ExamGroup = ?";
        try
        {
            st = Conn.prepareStatement (Que);
            formatter = new SimpleDateFormat("yyyy-MM-dd");
            D = formatter.format (Dat);
            st.setString (1, D);
            st.setInt (2, Grp);
            rs = st.executeQuery ();
            if (!rs.first ())
                ID = -1;
            else
                ID = rs.getInt ("ExamID");
            rs.close ();
            st.close ();
            Conn.close ();
            return ID;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in getting Exam Group ID",
TestException.EDSqlException, Que, e);
        }
    }
```

```
    /**
     * Εισάγει μία νέα εξέταση στον πίνακα Examinations. Δεν γίνεται έλεγχος
αν υπάρχει προγραμματισμένη εξέταση με την ίδια ημερομηνία και
     * την ίδια ομάδα.
     * @param E Τα στοιχεία της ομάδας που θα εισαχθεί.
     * @return Το ExamID της εξέτασης που δημιουργήθηκε.
     * @throws TestException Σε περίπτωση που δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση με τη Βάση
Δεδομένων.
     */
    public static int AddExam (ExamStruct E) throws TestException,
GmException
    {
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        SimpleDateFormat formatter;
        int ID;
        ID = -1;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "INSERT Examinations (ExamGroup, ExamDate, ExamStatus) Values
(?, ?, ?)";
        try
        {
            st = Conn.prepareStatement (Que,
Statement.RETURN_GENERATED_KEYS);
            st.setInt (1, E.Group);
            formatter = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
            st.setString (2, formatter.format (E.DateTime));
            st.setString (3, E.Status);
            st.executeUpdate ();
            rs = st.getGeneratedKeys ();
            rs.first ();
            ID = rs.getInt (1);
            rs.close ();
            st.close ();
            Conn.close ();
            return ID;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in adding new Examination
Group", TestException.EDSqlException, Que, e);
        }
    }

    /**
     * Εισάγει τα στοιχεία ενός εξεταζόμενου στον πίνακα People. Η εισαγωγή
γίνεται από αντικείμενο PeopleStruct το οποίο πρέπει να
     * περιέχει δεδομένα σε όλα τα πεδία εκτός του κωδικού. Δεν γίνεται
έλεγχος αν υπάρχει ο εξεταζόμενος αυτός ή αν ο κωδικός της εξέτασης
     * είναι σωστός.
     * @param P Τα στοιχεία της ομάδας που θα εισαχθεί.
     * @return Το PID του εξεταζόμενου που εισήχθη.
     * @throws TestException Σε περίπτωση που δημιουργηθεί SQL εξαίρεση.
     * @throws GmException Αν δεν μπορεί να δημιουργηθεί σύνδεση με τη Βάση
Δεδομένων.
     */
    public static int AddPeople (PeopleStruct P) throws TestException,
GmException
    {
```

```
        String Que;
        Connection Conn;
        PreparedStatement st;
        ResultSet rs;
        int ID;
        ID = -1;
        Conn = DBSimlpeConn.GetConnection ("ExamDis", "ExamDis", "ExamDis");
        Que = "INSERT People (Surname, Firstname, Fathername, IDNumber,
ExamID, ExamKateg, ExamLang, Speech, ClientID, Result) "
            + "values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        try
        {
            st = Conn.prepareStatement (Que,
Statement.RETURN_GENERATED_KEYS);
            st.setString (1, P.Surname);
            st.setString (2, P.Firstname);
            st.setString (3, P.Fathername);
            st.setString (4, P.IDNumber);
            st.setInt (5, P.ExamID);
            st.setInt (6, P.ExamKateg);
            st.setInt (7, P.ExamLang);
            st.setBoolean (8, P.Speech);
            st.setInt (9, P.ClientID);
            st.setString (10, P.Result);
            st.executeUpdate ();
            rs = st.getGeneratedKeys ();
            rs.first ();
            ID = rs.getInt (1);
            rs.close ();
            st.close ();
            Conn.close ();
            return ID;
        }
        catch (SQLException e)
        {
            throw new TestException ("Error in adding new People",
TestException.EDSqlException, Que, e);
        }
    }
}
```

CLASS: ExamStruct

```
package org.gmele.dissertation.testhandler;

import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Η κλάση ExamStruct αποτελεί τη δομή στην οποία θα φυλλάσονται δεδομένα
του πίνακα Examinations.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class ExamStruct
{
    /**
     * Ο κωδικός της ομάδας εξέτασης.
     */
    public int ID;
```

```
    /**
     * Η ομάδα (group) της εξέτασης μέσα στη μέρα.
     */
    public int Group;

    /**
     * Η προγραμματισμένη ημερομηνία και ώρα διενέργειας της εξέτασης.
     */
    public Date DateTime;

    /**
     * Η κατάσταση της εξέτασης. Μπορεί να έχει μία από τις ακόλουθες τιμές:
     * <ul>
     * <li> Pending: Έχει προγραμματιστεί
     * <li> Active: Εξετάζεται αυτή τη στιγμή
     * <li> Done: Ολοκληρώθηκε κανονικά
     * <li> Aborted: Ακυρώθηκε λόγω προβλήματος κατά τη διενέργειά της
     * <li> Cancelled: Ακυρώθηκε ως ομάδα πριν ξεκινήσει η εξέτασή της
     * </ul>
     */
    public String Status;


    /**
     * Μετατρέπει μία ExamStruct σε string προκειμένου να μπορεί να
εισαχθεί, για παράδειγμα, σε combo boxes.
     * @return Το string που περιέχει τα δεδομένα της κλάσης σε αναγνώσιμη
μορφή.
     */
    @Override
    public String toString ()
    {
        String Que;
        SimpleDateFormat frm;
        frm = new SimpleDateFormat ("dd/MM/yy HH:mm");
        Que =  frm.format (DateTime) + "   Ομάδα: " + Group + "  [";
        if (Status.equals ("Pending"))
            Que = Que + "Προγραμματισμένη";
        if (Status.equals ("Active"))
            Que = Que + "Εξετάζεται";
        if (Status.equals ("Done"))
            Que = Que + "Ολοκληρωμένη";
        if (Status.equals ("Aborted"))
            Que = Que + "Διακεκομμένη";
        if (Status.equals ("Cancelled"))
            Que = Que + "Ακυρωμένη";
        Que = Que + "]";
        return Que;
    }
}
```

CLASS: PeopleStruct

```
package org.gmele.dissertation.testhandler;

/**
 * Η κλάση PeolpleStruct χρησιμοποιείται ως η δομή για το διάβασμα και το
γράψιμο εγγραφών από τον πίνακα People ο οποίος περιέχει τα
 * στοιχεία των εξεταζομένων.
 * @author Giorgos Meletiou
 * @version 1.0
```

```
 */
public class PeopleStruct
{
    /**
     * Κωδικός Εξεταζόμενου.
     */
    public int PID;

    /**
     * Επώνυμο εξεταζόμενου.
     */
    public String Surname;

    /**
     * Όνομα Εξεταζόμενου.
     */
    public String Firstname;

    /**
     * Πατρώνυμο Εξεταζόμενου.
     */
    public String Fathername;

    /**
     * Αριθμός ταυτότητας ή σχετικού εγγράφου.
     */
    public String IDNumber;

    /**
     * Κωδικός Ομάδας Εξέτασης στην οποία συμμετέχει.
     */
    public int ExamID;

    /**
     * Κωδικός Κατηγορίας Εξέτασης.
     */
    public int ExamKateg;

    /**
     * Κωδικός Γλώσσας Εξέτασης.
     */
    public int ExamLang;

    /**
     * Ομιλία στην εξέταση;
     */
    public boolean Speech;

    /**
     * Κωδικός Σταθμού Εξέτασης.
     */
    public int ClientID;

    /**
     * Αποτέλεσμα Εξέτασης. Ένα από: "Pass", "Fail", "Absent", "-".
     */
    public String Result;
}
```

CLASS: Question

```
package org.gmele.dissertation.testhandler;

import java.io.Serializable;

/**
 * Τα πλήρη δεδομένα που αφορούν μία ερώτηση στο ερωτηματολόγιο.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class Question implements Serializable
{
    /**
     * Κωδικός Ερώτησης στη Βάση Δεδομένων
     */
    public int Code;

    /**
     * Λεκτικό Ερώτησης
     */
    public String Verbal;

    /**
     * Όνομα αρχείου φωτογραφίας ερώτησης ("0" = χωρίς φωτογραφία)
     */
    public String Photo;

    /**
     * Ονομασία αρχείου ήχου ερώτησης ("0" = δεν υπάρχει)
     */
    public String Sound;

    /**
     * Λεκτικό συσχέτησης ερώτησης με βιβλίο υπουργείου
     */
    public String Book;

    /**
     * Αριθμός πιθανών απαντήσεων
     */
    public int NoAns;

    /**
     * Αύξωντες αριθμοί απαντήσεων σύμφωνα με τον πίνακα των απαντήσεων στη
βάση δεδομένων
     */
    public int[] AnsAA;

    /**
     * Λεκτικά απαντήσεων
     */
    public String[] AnsVerbal;

    /**
     * Ονόματα αρχείων ήχου απαντήσεων ("0" = δεν υπάρχει)
     */
    public String[] AnsSound;

    /**
     * Επιλογή απάντησης σύμφωνα με τους αύξωντες αριθμούς στη βάση και όχι
σύμφωνα με την σειρά παρουσίασης στην οθόνη (0 = αναπάντητη)
     */
    public int Choice;
```

```
    /**
     * Σωστή απάντηση του εξεταζόμενου;
     */
    public boolean Correct;
}
```

CLASS: TestHandler

```
package org.gmele.dissertation.testhandler;

import java.io.File;
import java.util.Random;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση TestHandler παρέχει στατικές μεθόδους για τον χειρισμό
αντικειμένων της κλάσης {@link TestSheet}. Είναι, δηλαδή, υπεύθυνη για
 * την δημιουργία, την βαθμολόγιση κλπ φύλλων εξέτασης. Υλοποιεί εργασίες
που αφορούν μόνο τα test και δεν σχετίζεται με την παλαιά Mainbase
 * του YME. <p>
 * Χρησιμοποιεί την {@link ExamDisDB} για την πρόσβαση στη βάση.
 * @author Giorgos Meletiou
 * @version 1.0
 */

public class TestHandler
{
    /**
     * Δημιουργεί ερωτηματολόγιο συγκεκριμένης κατηγορίας και γλώσσας και το
τοποθετεί σε ένα {@link TestSheet}. Η μέθοδος τροποποιεί στο
     * Header μόνο τα δεδομένα που την αφορούν ({@link TestHeader#ExamTime}
- {@link TestHeader#Result}) χωρίς να πειράζει τα στοιχεία του
     * χρήστη, της εξέτασης, κλπ.
     * @param Sheet Το TestSheet το οποίο θα γεμίσει.
     * @param Kateg Η κατηγορία της εξέτασης του ερωτηματολογίου.
     * @param Lang Η γλώσσα του ερωτηματολογίου.
     * @throws GmException Όπως αυτό μεταφέρεται από τις μεθόδους που
καλούνται.
     */
    public static void CreateQuestionnaire (TestSheet Sheet, int Kateg, int
Lang) throws GmException
    {
        int[] QPag;
        int[] QCods;
        int CurQ;
        int i, j;
        QPag = ExamDisDB.GetInterKatNumbs (Kateg);
        Sheet.NoQ = SumIntArray (QPag);
        Sheet.Quests = new Question[Sheet.NoQ];
        CurQ = 0;
        for (i = 0; i < QPag.length; i++)
        {
            QCods = ExamDisDB.GetInterKategQCods (Kateg, i + 1, Lang);
            Suffle (QCods);
            for (j = 0; j < QPag[i]; j++)
            {
                Sheet.Quests[CurQ] = ExamDisDB.GetFullQuestion (QCods[j]);
                SuffleAnswers (Sheet.Quests[CurQ]);
                Sheet.Quests[CurQ].Choice = 0;
```

```
                Sheet.Quests[CurQ].Correct = false;
                CurQ++;
            }
        }
        Sheet.Header.ExamTime = ExamDisDB.GetExamTime (Kateg);
        Sheet.Header.Result = "-";
    }

    /**
     * Υπολογίζει το άθροισμα των στοιχειών ενός πίνακα ακεραίων. Αν ο
πίνακας έχει μηδενικό μέγεθος ή είναι null επιστρέφεται 0
     * @param mat Ο πίνακας του οποίου τα στοιχεία θα αθροιστούν
     * @return Το άθροισμα των στοιχείων
     */
    public static int SumIntArray (int[] mat)
    {
        int Sum;
        if (mat == null || mat.length == 0)
            return 0;
        Sum = 0;
        for (int i: mat)
            Sum += i;
        return Sum;
    }

    /**
     * Ανακατεύει τα στοιχεία ενός πινακα ακεραίων με τυχαίο τρόπο.
Χρησιμοποιείται για κληρώσεις.
     * @param mat Ο πίνακας προς αναδιάταξη
     */
    public static void Suffle (int[] mat)
    {
        int i;
        int tmp;
        int rp;
        Random rgen;
        if (mat == null)
            return;
        if (mat.length == 0)
            return;
        rgen = new Random ();
        for (i = 0; i < mat.length; i++)
        {
            rp = rgen.nextInt (mat.length);
            tmp = mat[i];
            mat[i] = mat [rp];
            mat[rp] = tmp;
        }
    }

    /**
     * Ανακατεύει την σειρά των απαντήσεων μίας ερώτησης.
     * @param Q Η ερώτηση σε μορφή {@link Question}
     */
    public static void SuffleAnswers (Question Q)
    {
        int i;
        int[] AnsP;
        int[] F1;
        String[] F2;
        String[] F3;
        AnsP = new int[Q.NoAns];
```

```
        for (i = 0; i < Q.NoAns; i++)
            AnsP[i] = i;
        Suffle (AnsP);
        F1 = new int[Q.NoAns];
        F2 = new String[Q.NoAns];
        F3 = new String[Q.NoAns];
        for (i = 0; i < Q.NoAns; i++)
        {
            F1[i] = Q.AnsAA[AnsP[i]];
            F2[i] = Q.AnsVerbal[AnsP[i]];
            F3[i] = Q.AnsSound[AnsP[i]];
        }
        Q.AnsAA = F1;
        Q.AnsVerbal = F2;
        Q.AnsSound = F3;
    }

    /**
     * Δημιουργεί ένα Test Header {@link TestHeader} και τοποθετεί μέσα του
τα δεδομένα ενός εξεταζόμενου όπως αυτά βρίσκονται στην κλάση
     * {@link PeopleStruct}. Δεδομένα τα οποία δεν βρίσκονται στη δομή του
εξεταζόμενου δεν αρχικοποιούνται ή τροποποιούνται.
     * @param peo Τα στοιχεία του εξεταζόμενου.
     * @return Ο TestHeader αρχικοποιημένος.
     * @throws TestException Αν δημιουργηθεί στην {@link
ExamDisDB#GetKategoryVerbal(int)} ή στην {@link
ExamDisDB#GetLanguageVerbal(int)}.
     * @throws GmException Αν δημιουργηθεί στην {@link
ExamDisDB#GetKategoryVerbal(int)} ή στην {@link
ExamDisDB#GetLanguageVerbal(int)}.
     */
    public static TestHeader PeopleToHeader (PeopleStruct peo) throws
TestException, GmException
    {
        TestHeader header;
        header = new TestHeader ();
        header.ExamID = peo.ExamID;
        header.Surname = peo.Surname;
        header.Firstname = peo.Firstname;
        header.Fathername = peo.Fathername;
        header.IDNum = peo.IDNumber;
        header.Kateg = ExamDisDB.GetKategoryVerbal (peo.ExamKateg);
        header.Lang = ExamDisDB.GetLanguageVerbal (peo.ExamLang);
        header.Speech = peo.Speech;
        header.ClientID = peo.ClientID;
        return header;
    }

    /**
     * Δημιουργεί και επιστρέφει ένα πλήρες {@link TestSheet} για κάποιον
συγκεκριμένο εξεταζόμενο.
     * @param Peo Τα στοιχεία του εξεταζόμενου.
     * @return Το TestSheet
     * @throws TestException Αν δημιουργηθεί στην {@link #PeopleToHeader} ή
στην {@link #CreateQuestionnaire}
     * @throws GmException Αν δημιουργηθεί στην {@link #PeopleToHeader} ή
στην {@link #CreateQuestionnaire}
     */
    public static TestSheet CreateTestSheet (PeopleStruct Peo) throws
TestException, GmException
    {
        TestSheet sheet;
```

```
        sheet = new TestSheet ();
        sheet.Header = PeopleToHeader (Peo);
        CreateQuestionnaire (sheet, Peo.ExamKateg, Peo.ExamLang);
        return sheet;
    }

    /**
     * Δημιουργεί ένα σύνολο από TestFiles και τα αποθηκεύει στο δίσκο. Τα
αρχεία αντιστοιχούν σε συγκεκριμένους εξεταζόμενους τα δεδομένα
     * των οποίων εισάγονται μέσω δομών {@link PeopleStruct}. Τα αρχεία που
δημιουργούνται αποθηκεύονται στον φάκελο "Send" του συστήματος
     * σε υποφάκελο με όνομα τον κωδικό της εξέτασης. Το όνομα του αρχείου
έχει τη μορφή "<Αριθμός Τερματικού>-<Επώνυμο Εξεταζόμενου>.test.
     * Αν κάποιος από τους φακέλους δεν υπάρχει, δημιουργείται.
     * @param Peo Ο πίνακας με τα στοιχεία των εξεταζομένων.
     * @param SendDir Το όνομα του φακέλου στον οποίο αποθηκεύονται όλα τα
αρχεία των εξετάσεων προς αποστολή.
     * @throws TestException Σε περίπτωση που δημιουργηθεί από την {@link
#CreateTestSheet} ή την {@link TestSheet#Save}.
     * @throws GmException Σε περίπτωση που δημιουργηθεί από την {@link
#CreateTestSheet}.
     */
    public static void MakeTestFiles (PeopleStruct[] Peo, String SendDir)
throws TestException, GmException
    {
        int i;
        TestSheet Sheet;
        String Dn;
        for (i = 0; i < Peo.length; i++)
        {
            Sheet = CreateTestSheet (Peo[i]);
            Dn = SendDir + "/" + Peo[i].ExamID;
            if (!new File (Dn).exists ())
                new File (Dn).mkdirs ();
            TestSheet.Save (Sheet, Dn + "/" + Sheet.Header.ClientID + "-" +
Sheet.Header.Surname + ".test");
        }
    }

    /**
     * Βαθμολογεί ένα φύλλο εξέτασης, καταγράφει σε αυτό τις σχετικές
πληροφορίες (σωστό / λάθος ανά ερώτηση, τελικό αποτέλεσμα) και
     * επιστρέφει το αποτέλεσμα σε boolean μορφή. Το τελικό αποτέλεσμα
καταγράφεται στο φύλλο εξέτασης με τα λεκτικά "ΕΠΕΤΥΧΕ" και
     * "ΑΠΟΡΡΙΦΘΗΚΕ". Η μέθοδος δεν χειρίζεται το θέμα της απουσίας. Οι
αναπάντητες ερωτήσεις μετράνε ως λανθασμένες απαντήσεις.
     * @param Sheet Το ερωτηματολόγιο που θα βαθμολογηθεί.
     * @return true αν ο εξεταζόμενος πέρασε, false διαφορετικά.
     * @throws TestException Σε περίπτωση λανθασμένων δεδομένων.
Δημιουργείται από τις μεθόδους της {@link ExamDisDB} που χρησιμοποιούνται.
     * @throws GmException Σε περίπτωση προβλήματος σύνδεσης με τη Βάση
Δεδομένων. Δημιουργείται από τις μεθόδους της {@link ExamDisDB} που
     * χρησιμοποιούνται.
     */
    public static boolean MarkTest (TestSheet Sheet) throws TestException,
GmException
    {
        int Errs;
        int i;
        Errs = 0;
        for (i = 0; i < Sheet.NoQ; i++)
        {
```

```
            if (Sheet.Quests[i].Choice == ExamDisDB.GetCorrectAns
(Sheet.Quests[i].Code))
                Sheet.Quests[i].Correct = true;
            else
            {
                Sheet.Quests[i].Correct = false;
                Errs++;
            }
        }
        if (Errs > ExamDisDB.GetKategAErrors (ExamDisDB.GetKategoryCode
(Sheet.Header.Kateg)))
        {
            Sheet.Header.Result = "ΑΠΟΡΡΙΦΘΗΚΕ";
            return false;
        }
        else
        {
            Sheet.Header.Result = "ΕΠΕΤΥΧΕ";
            return true;
        }
    }

    /**
     * Βαθμολογεί ένα φύλλο εξέτασης το οποίο είναι αποθηκευμένο σε αρχείο.
Η βαθμολόγηση γίνεται μέσω της
     * {@link #MarkTest(org.gmele.dissertation.testhandler.TestSheet)} και
το βαθμολογημένο φύλλο αποθηκεύεται με το ίδιο όνομα.
     * @param Fn Το όνομα του αρχείου.
     * @return true αν ο εξεταζόμενος πέρασε, false διαφορετικά
     * @throws TestException Αν υπάρξει πρόβλημα στο διάβασμα ή το γράψιμο
του αρχείου ή αν δημιουργηθεί από την
     * {@link #MarkTest(org.gmele.dissertation.testhandler.TestSheet)}
     * @throws GmException Αν υπάρξει πρόβλημα στο serialization ή από την
{@link #MarkTest(org.gmele.dissertation.testhandler.TestSheet)}
     */
    public static boolean MarkTest (String Fn) throws TestException,
GmException
    {
        boolean res;
        TestSheet Sheet;
        Sheet = TestSheet.Load (Fn);
        res = MarkTest (Sheet);
        TestSheet.Save (Sheet, Fn);
        return res;
    }
}
```

CLASS: TestHeader

```
package org.gmele.dissertation.testhandler;

import java.io.Serializable;
import java.sql.Date;


/**
 * Η κλάση TestHeader περιέχει τις πληροφορίες που αφορούν τον εξεταζόμενο
και την εξέταση στην οποία συμμετέχει. Αποτελεί τμήμα ενός
 * Φύλλου Εξέτασης {@link TestSheet}
 * @author Giorgos Meletiou
 * @version 1.0
```

```
 */
public class TestHeader implements Serializable
{
    /**
     * Κωδικός Εξέτασης.
     */
    public int ExamID;

    /**
     * Επώνυμο Εξεταζόμενου.
     */
    public String Surname;

    /**
     * Όνομα Εξεταζόμενου
     */
    public String Firstname;

    /**
     * Πατρώνυμο Εξεταζόμενου.
     */
    public String Fathername;

    /**
     * Στοιχεία ταυτότητας Εξεταζόμενου.
     */
    public String IDNum;

    /**
     * Λεκτικό κατηγορίας Εξέτασης.
     */
    public String Kateg;

    /**
     * Λεκτικό Γλώσσας Εξέτασης.
     */
    public String Lang;

    /**
     * Ανάγνωση των ερωτήσεων για αγράμματους;
     */
    public boolean Speech;

    /**
     * Χρόνος εξέτασης σε πρώτα λεπτά της ώρας.
     */
    public int ExamTime;

    /**
     * Ημερομηνία και ώρα πραγματικής έναρξης εξέτασης.
     */
    public Date DateStarted;

    /**
     * Ημερομηνία και ώρα πραγματικής λήξης εξέτασης.
     */
    public Date DateEnded;

    /**
     * Κωδικός Σταθμού Εξέτασης.
     */
    public int ClientID;
```

```
    /**
     * Λεκτικό Αποτελέσματος εξέτασης. (ΕΠΕΤΥΧΕ - ΑΠΟΡΡΙΦΘΗΚΕ - "-")
     */
    public String Result;

}
```

CLASS: TestSheet

```
package org.gmele.dissertation.testhandler;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.io.Serializable;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;

/**
 * Ένα ολοκληρωμένο φύλλο εξέτασης.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class TestSheet implements Serializable
{
    public TestHeader Header;
    public int NoQ;
    public Question[] Quests;

    /**
     * Δημιουργεί ένα κενό φύλλο εξέτασης.
     */
    public TestSheet ()
    {
        Header = new TestHeader ();
        Header.ExamID =-1;
        NoQ = 0;
    }

    /**
     * Αποθηκεύει ένα Testsheet ως αρχείο μέσω serialization.
     * @param sheet Το Testsheet που θα αποθηκευτεί.
     * @param FileName Το όνομα του αρχείου με το οποίο θα αποθηκευτεί.
     * @throws TestException Σε περίπτωση προβλήματος στο όνομα του αρχείου
ή κατά την διάρκεια της εγγραφής των δεδομένων.
     */
    public static void Save (TestSheet sheet, String FileName) throws
TestException
    {
        try
        {
            ObjectOutputStream os;
            os = new ObjectOutputStream (new FileOutputStream (FileName));
            os.writeObject (sheet);
        }
        catch (FileNotFoundException e)
```

```
        {
            throw new TestException ("Invalid Filename to Save Testsheet.",
TestException.TEInvalidFilename, FileName, e);
        }
        catch (IOException e)
        {
            throw new TestException ("IO error during saving Testsheet.",
TestException.TEErrorInWrittingData, FileName, e);
        }

    }

    /**
     * Διαβάζει και επιστρέφει ένα TestSheet από αρχείο μέσω serialization.
     * @param FileName Το όνομα του φύλλου στο αποθηκευτικό μέσο.
     * @return Το TestSheet που διαβάστηκε
     * @throws TestException Σε περίπτωση που δεν ήταν δυνατό το διάβασμα
του αρχείου.
     * @throws GmException Σε περίπτωση που αποτύχει το serialization λόγω
άγνωστης κλάσης.
     */
    public static TestSheet Load (String FileName) throws TestException,
GmException
    {
        try
        {
            ObjectInputStream is;
            TestSheet ts;
            is = new ObjectInputStream (new FileInputStream (FileName));
            ts = (TestSheet) is.readObject ();
            return ts;
        }
        catch (IOException e)
        {
            throw new TestException ("IO error during loading Testsheet.",
TestException.TEErrorInLoadingData, FileName, e);
        }
        catch (ClassNotFoundException e)
        {
            throw new GmException ("Error during loading Testsheet.",
GmException.GenImpossibleException, FileName, e);
        }
    }

    /**
     * Τυπώνει τα δεδομένα ενός TestSheet γραμμή - γραμμή σε Κανάλι.
     * @param os Το κανάλι στο οποίο θα γραφτούν τα δεδομένα. Πρέπει να
είναι ανοιχτό.
     */
    public void Print (PrintStream os)
    {
        int i, j;
        if (Header.ExamID == -1)
            os.println ("*** NO HEADER ***");
        else
        {
            os.println (Header.ExamID);
            os.println (Header.Surname + ", " + Header.Firstname + ", " +
Header.Fathername);
            os.println (Header.IDNum);
            os.println (Header.Kateg);
            os.println (Header.Lang);
```

```
            os.println (Header.Speech ? "Με Ήχο" : "Χωρίς Ήχο");
            os.println (Header.ExamTime);
            os.println (Header.DateStarted + " - " + Header.DateEnded);
            os.println (Header.Result);
        }
        os.println
("================================================================================
======");
        if (NoQ == 0)
            os.println ("*** NO QUESTIONS ***");
        else
        {
            for (i = 0; i < NoQ; i++)
            {
                os.println ((i + 1) + ") " + Quests[i].Verbal);
                os.println (Quests[i].Code + ", " + Quests[i].Photo + ", " +
Quests[i].Sound + ", " + Quests[i].Book);
                os.println (Quests[i].NoAns);
                for (j = 0; j < Quests[i].NoAns; j ++)
                {
                    os.println ("     " + Quests[i].AnsAA[j] + ") " +
Quests[i].AnsVerbal[j]);
                    os.println ("     " + Quests[i].AnsSound[j] + "   " +
Quests[i].Choice);
                }
                os.println ("     " + Quests[i].Choice);
                os.println
("--------------------------------------------------------------------------")
;
            }
        }
        os.println
("================================================================================
======\n");
    }
}
```

PACKAGE: org.gmele.dissertation.testhandler.exceptions

CLASS: TestException

```
package org.gmele.dissertation.testhandler.exceptions;

import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση χειρισμού εξαιρέσεων του ExamTest. Κληρονομεί την {@link
GmException} και υλοποιεί το interface  {@link TestExceptionConsts}.
 * Η κλάση δεν προσθέτει λειτουργικότητα στην κλάση που κληρονομεί.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class TestException extends GmException implements
TestExceptionConsts
{

    /**
```

```
     * Κατασκευάζει το αντικείμενο για εξαίρεση η οποία δεν περιλαμβάνει
πληροφορία System Exception. Περνάει τις παραμέτρους στον
     * αντίστοιχο κατασκευαστή της κλάσης {@link GmException} χωρίς καμμία
επεξεργασία.
     * @param message Το μήνυμα λάθους.
     * @param errCode Ο κωδικός λάθους. Πρέπει να έχει οριστεί ως σταθερά
στο interface "ESExceptionConsts" ή στο "GmExceptionConsts".
     * @param data Έξτρα πληροφορίες για το λάθος σε μορφή string.
     */
    public TestException (String message, int errCode, String data)
    {
        super (message, errCode, data);
    }

    /**
     * Κατασκευάζει το αντικείμενο για εξαίρεση η οποία περιλαμβάνει
πληροφορία System Exception. Περνάει τις παραμέτρους στον αντίστοιχο
     * κατασκευαστή της κλάσης {@link GmException} χωρίς καμμία επεξεργασία.
     * @param message Το μήνυμα λάθους
     * @param errCode Ο κωδικός λάθους. Πρέπει να έχει οριστεί ως σταθερά
στο interface "ESExceptionConsts" ή στο "GmExceptionConsts".
     * @param data Έξτρα πληροφορίες για το λάθος σε μορφή string.
     * @param sysExc Η εξαίρεση, ο χειρισμός της οποίας, δημιουργεί την
τρέχουσα εξαίρεση.
     */
    public TestException (String message, int errCode, String data,
Exception sysExc)
    {
        super (message, errCode, data, sysExc);
    }

}
```

INTERFACE: TestExceptionConsts

```
package org.gmele.dissertation.testhandler.exceptions;

/**
 * Το Interface TestExceptionConsts περιέχει τους κωδικούς λάθους για το
Project ExamTests. Υλοποιείται από την κλάση {@link TestException}
 * @author Giorgos Meletiou
 * @version 1.0
 */
public interface TestExceptionConsts
{
    /**
     * Λάθος όνομα αρχείου για γράψιμο ή διάβασμα δεδομένων.
     */
    final int TEInvalidFilename = 11000;

    /**
     * Δημιουργήθηκε IOError κατά το γράψιμο ενός TestSheet ή άλλων σχετικών
δεδομένων.
     */
    final int TEErrorInWrittingData = 11001;

    /**
     * Δημιουργήθηκε IOError κατά το διάβασμα ενός TestSheet ή άλλων
σχετικών δεδομένων.
     */
    final int TEErrorInLoadingData = 11002;
```

```
    /**
     * Δημιουργήθηκε SqlException κατά την πραγματοποίηση λειτουργίας στην
κλάση ExamDisDB.
     */
    final int EDSqlException = 11100;

    /**
     * Ζητήθηκαν δεδομένα για κατηγορία και επιστράφηκε κενό result set.
Πιθανό η κατηγορία να μην υπάρχει.
     */
    final int EDNoDataForCategory = 11101;

    /**
     * Ζητήθηκε να βρεθούν ερωτήσεις που πληρούν συγκεκριμένες ιδιότητες και
δεν βρέθηκαν.
     */
    final int EDNoSuchQuestions = 11102;

    /**
     * Ζητήθηκε να βρεθούν απαντήσεις που πληρούν κάποια κριτήρια και δεν
βρέθηκαν.
     */
    final int EDNoSuchAnswers = 11103;

    /**
     * Δεν βρέθηκαν Εξεταζόμενοι σε μία ομάδα εξέτασης. Μπορεί να μην
υπάρχει και η ομάδα.
     */
    final int EDNoPeopleInExam = 11104;

    /**
     * Ζητήθηκαν δεδομένα για γλώσσα και επιστράφηκε κενό result set. Πιθανό
η κατηγορία να μην υπάρχει.
     */
    final int EDNoDataForLanguage = 11105;

    /**
     * Δεν βρέθηκε εγγραφή στον πίνακα People που να πληροί τους όρους
αναζήτησης.
     */
    final int EDNoSuchPeople = 11106;

    /**
     * Δεν βρεθηκε εγγραφή στον πίνακα Examinations που να πληροί τους όρους
αναζήτησης.
     */
    final int EDNoSuchExam = 11107;

}
```

PACKAGE: org.gmele.general.dbconnections

CLASS: DBSimpleConn

```
package org.gmele.general.dbconnections;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση DBSimpleConn παρέχει στατικές και μη στατικές μεθόδους οι οποίες
επιστρέφουν αντικείμενα τύπου {@link Connection} για πρόσβαση
 * σε Βάση Δεδομένων (αρχικά Mysql). Θα χρησιμοποιηθεί για απλές περιπτώσεις
και ώσπου να υλοποιηθεί η κλάση που υλοποιεί Pools στο ίδιο
 * πακέτο. Οι εξαιρέσεις έχουν κωδικούς από 1000 ... 1099
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class DBSimlpeConn
{
    /**
     * Δημιουργεί και επιστρέφει ένα MySql Connection. Το Connection πρέπει
να κλείσει και να καταστραφεί από το πρόγραμμα που το
     * χρησιμοποιεί. Το Connection γίνεται στο localhost στη default πόρτα
και με κωδικοποίηση UTF-8
     * @param DbName Το όνομα της Βάσης Δεδομένων
     * @param LogName Το όνομα του Χρήστη της Βάσης
     * @param Password Το Password του Χρήστη της Βάσης
     * @return Το Connection που δημιουργείται.
     * @throws GmException Σε περίπτωση οποιουδήποτε προβλήματος αφορά την
δημιουργία σύνδεσης
     */
    public static Connection GetConnection (String DbName, String LogName,
String Password) throws GmException
    {
        try
        {
            Connection Con;
            String Que;
            Class.forName ("com.mysql.jdbc.Driver").newInstance ();
            Que = "jdbc:mysql://localhost/" + DbName + "?
characterEncoding=UTF-8";
            Con = DriverManager.getConnection (Que, LogName, Password);
            return Con;
        }
        catch (IllegalAccessException e)
        {
            throw new GmException ("Unexpected Error Getting MySql
Connection", GmException.GenUnexpectedException, "Mysql:" + DbName, e);
        }
        catch (ClassNotFoundException ex)
        {
            throw new GmException ("MySql Driver Not Found",
GmException.GenCannotLoadClass, "Mysql:" + DbName, ex);
        }
        catch (SQLException e)
        {
            if (e.getErrorCode () == 1040)
                throw new GmException ("Too many connections ",
GmException.SimConDataError, DbName, e);
            else
                throw new GmException ("Invalid Connection Data (db,
logname, password)", GmException.SimConDataError, DbName + ":" +
                    LogName + ":" + Password, e);
        }
        catch (InstantiationException e)
```

```
        {
            throw new GmException ("Unexpected Error Getting MySql
Connection", GmException.GenUnexpectedException, "Mysql:" + DbName, e);
        }
    }


}
```

PACKAGE: org.gmele.general.exceptions

CLASS: GmException

```
package org.gmele.general.exceptions;

/**
 * Η κλάση GmException είναι η γενική κλάση για τις εξαιρέσεις όλων των
έργων "org.gmele". Η κλάση κληρονομεί την κλάση {@link Exception}
 * και ορίζει νέα attributes. Θα μπορεί να κληρονομηθεί από άλλες κλάσεις αν
χρειάζεται (π.χ. για να υλοποιηθούν κάποια interfaces με
 * κωδικούς λαθών), αλλά αυτό δεν θα είναι απαραίτητο. Υλοποιεί το interface
{@link GmExceptionConsts}.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class GmException extends Exception implements GmExceptionConsts
{
       /**
        * Ο κωδικός λάθος του Exception. Πρέπει να πάρει τιμές από αυτές που
ορίζονται στα σχετικά interfacesc
        */
       private int ErrorCode;

       /**
        * Δεδομένα σχετικά με το πρόβλημα που δημιουργήθηκε. Μπορεί να
περιέχει και προγραμματιστικές πληροφορίες, π.χ. τις τιμές κάποιων
     * μεταβλητών κλπ
        */
       private String Data;

    /**
        * Το Exception του συστήματος που προκάλεσε τη δημιουργία του
συγκεκριμένου Exception (αν υπάρχει τέτοιο).
        */
       private Exception SysException;

       /**
        * Ο κατασκευαστής του Exception και ο μοναδικός τρόπος να οριστούν
τιμές στα πεδία του (δεν υπάρχουν setters)
        * @param message Το μήνυμα για τον κατασκευαστή της κλάσης {@link
Exception} που κληρονομείται
        * @param errCode Ο κωδικός σφάλματος. Οι κωδικοί πρέπει να πέρνουν
τιμές από το interface {@link GmExceptionConsts} ή από αντίστοιχο
        * @param data Τα δεδομένα που αφορούν το σφάλμα σε String.
        * @param sysExc Αν το exception δημιουργείται για να ενημερώσει για
κάποιο άλλο exception που δημιουργήθηκε από το σύστημα
```

```
	 * (π.χ. ένα SQLException κατά την εκτέλεση εντολών σχετικών με τη
βάση δεδομένων) εδώ καταχωρείται το αρχικό exception. Αν δεν υπάρχει
	 * τέτοια περίπτωση η παράμετρος έχει τιμή null
	 */
	public GmException (String message, int errCode, String data,
Exception sysExc )
	{
		super (message);
		ErrorCode = errCode;
		Data = data;
		SysException = sysExc;
	}

	/**
	 * Βοηθητικός κατασκευαστής για την δημιουργία αντικειμένου χωρίς την
παράμετρο SysException. Καλεί τον βασικό κατασκευαστή περνώντας
	 * του null στην σχετική παράμετρο
	 * @param message Το μήνυμα για τον κατασκευαστή της κλάσης Exception
που κληρονομείται
	 * @param errCode Ο κωδικός σφάλματος. Οι κωδικοί πρέπει να πέρνουν
τιμές από την κλάση YMEExceptionConstants
	 * @param data Τα δεδομέναπου αφορούν το σφάλμα σε String.
	 */
	public GmException (String message, int errCode, String data)
	{
		this(message, errCode, data, (Exception) null);
	}

	/**
	 * Επιστρέφει τον κωδικό λάθους που έχει καταγραφεί στο exception
	 * @return Ο κωδικός λάθους
	 */
	public int getErrorCode ()
	{
		return ErrorCode;
	}

	/**
	 * Επιστρέφει τα δεδομένα που αφορούν το σφάλμα που δημιουργήθηκε
	 * @return Η αναλυτική περιγραφή
	 */
	public String getData ()
	{
		return Data;
	}

	/**
	 * Επιστρέφει το Exception Συστήματος που δημιούργησε το τρέχων
exception ή null αν δεν έχει οριστεί τέτοιο.
	 * @return Το System Exception
	 */
	public Exception getSysException ()
	{
		return SysException;
	}

}
```

INTERFACE: GmExceptionsConsts

```
package org.gmele.general.exceptions;
```

```
/**
 * Το interface GmExceptionConsts περιέχει ένα δείγμα ορισμού σταθερών για
την κλάση {@link GmException} καθώς και τις σταθερές που θα
 * χρησιμοποιηθούν από την χρήση της κλάσης χωρίς αυτή να κληρονομηθεί (π.χ
σε γενικές κλάσεις που βρίσκονται σε πακέτα κάτω από το
 * "org.gmele.general"). Οι κωδικοί λαθών για κάθε εφαρμογή ή κλάση
ορίζονται σε πακέτα των 100. Οι αριθμοί από το 0 ως το 99999 είναι
 * δεσμευμένοι για την κλάση.<p>
 * Κάθε κωδικός λάθους ορίζεται με σταθερά που να δηλώνει την κλάση ή την
εφαρμογή που τον δημιουργεί και συνοδεύται από αναλυτική περιγραφή
 * σε μορφή javadoc
 * @author Giorgos Meletiou
 * @version 1.0
 */
public interface GmExceptionConsts
{
//Κωδικοί από 0 ... 999. Όσο πιο system και γενικοί γίνεται.
    /**
     * Κωδικός λάθους 0. Δεν θα συμβεί ποτέ.
     */
    final int GenSystemNoError = 0;
    /**
     * Λείπει κάποια βιβλιοθήκη και δημιουργήθηκε ClassNotFoundException.
Π.Χ. λείπει ο mysql driver
     */
    final int GenCannotLoadClass = 1;

    /**
     * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
     * Δημιουργήθηκε κάποια εξαίρεση η οποία δεν είχε προβλεφτεί στο
συγκεκριμένο σημείο του προγράμματος. Για παράδειγμα, σε μία λειτουργία
     * πρόσβασης σε βάση δεδομένων αναμένεται να συμβούν δύο διαφορετικού
τύπου εξαιρέσεις η παγίδευση των οποίων δημιουργεί
     * {@link GmException} με δύο σχετικούς κωδικούς λάθους. Αν συμβεί μία
τρίτη εξαίρεση, μη αναμενόμενη στο συγκεκριμένο τμήμα κώδικα,
     * η GmException θα έχει ως κωδικό λάθους GenUnexpectedException
     * σε μία βάση δεδομένων αν γίνει έλεγχος για ένα ή δύο τύπους εξαίρεσης
που αναμένεται να συμβούν από το σύστημα και που η και
     * δημιουργηθεί κάποιου άλλου τύπου εξαίρεσης τότε η κλάση δημιουργεί
εξαίρεση
     */
    final int GenUnexpectedException = 2;

    /**
     * Δημιουργήθηκε εξαίρεση η οποία δεν θα έπρεπε να συμβεί με κανένα
τρόπο. Για παράδειγμα κατά τη διάρκεια διαβάσματος αντικειμένου από
     * αρχείο (serialiazation) η κλάση του αντικειμένου δεν έχει οριστεί.
<p>
     * Χρησιμοποιείται στις GMele βιβλιοθήκες όταν συμβεί εξαίρεση που η
Java απαιτεί τον χειρισμό της, αλλά κανονικά δεν υπάρχει περίπτωση
     * να συμβεί. Στο προηγούμενο παράδειγμα ο κωδικός θα δημιουργούνταν αν
η μέθοδος ObjectInputStream.readObject δημιουργούσε
     * ClassNotFoundException
      */
    final int GenImpossibleException = 3;

//Κωδικοί 1000 ... 1099. Εξαιρέσεις για SQL Connections
    /**
     * Δεν μπορεί να δημιουργηθεί νέα απλή σύνδεση με τη MySql. Υπάρχουν
πολλές ενεργές συνδέσεις.
     */
```

```
    final int SimConCannotGetCon = 1000;

    /**
     * Λάθος όνομα Βάσης Δεδομένων ή logname ή password
     */
    final int SimConDataError = 1001;

//Κωδικοί 1100 ... 1199. Εξαιρέσεις για Υπολογιστικά Φύλλα (Spreadsheets).
    /**
     * Έγινε προσπάθεια να ανοίξει αρχείο υπολογιστικού φύλλου το οποίο δεν
υπάρχει.
     */
    final int SheetNoSuchFile = 1100;

    /**
     * Δημιουργήθηκε IOerror κατά την ανάγνωση ή την εγγραφή ενός
υπολογιστικού φύλλου.
     */
    final int SheetIOError = 1101;

    /**
     * Δεν υπάρχει το εσωτερικό φύλλο που επιλέχτηκε.
     */
    final int SheetNoSuchSheet = 1102;

    /**
     * Δεν υπάρχει η συγκεκριμένη γραμμή στο φύλλο.
     */
    final int SheetNoSuchRow = 1103;

    /**
     * Δεν υπάρχει το συγκεκριμένο κελί, ως στήλη, στη γραμμή του φύλλου.
     */
    final int SheetNoSuchCell = 1104;

    /**
     * Το κελί περιέχει διαφορετικού τύπου δεδομένα από αυτά που ζητήθηκαν
να διαβαστούν.
     */
    final int SheetTypeMismatch = 1105;

    /**
     * Το όνομα με το οποίο έγινε προσπάθεια να αποθηκευτεί ένα excelόφυλλο
δεν μπορεί να δημιουργηθεί στο σύστημα αρχείων.
     */
    final int SheetInvalidFilename = 1106;
}
```

PACKAGE: org.gmele.general.sheets.excelx

CLASS: XlsxSheet

```
package org.gmele.general.sheets.excelx;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Date;
import org.apache.poi.ss.usermodel.PrintOrientation;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.gmele.general.exceptions.GmException;

/**
 * Η κλάση χειρίζεται ".xlsx" αρχεία παρέχοντας υψηλότερου επιπέδου
λειτουργίες από αυτές της βιβλιοθήκης Apache POI μέσω της οποίας γίνεται
 * ο πραγματικός χειρισμός. Η κλάση χειρίζεται το αρχείο χωρίς να απαιτείται
η παρουσία του Microsoft Excel. Η κλάση δημιουργεί GmException
 * εξαιρέσεις.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class XlsxSheet
{
    XSSFWorkbook wb;
    XSSFSheet sheet;
    XSSFRow row;
    XSSFCell cell;
    String Filename;

    /**
     * Ανοίγει ένα excelόφυλλο (το οποίο πρέπει να υπάρχει στο δίσκο) και
ενεργοποιεί το πρώτο φύλλο. Αν δεν υπάρχουν φύλλα δημιουργείται
     * εξαίρεση.
     * @param Fn Το πλήρες όνομα του αρχείου που θα δημιουργηθεί.
     * @throws GmException Σε περίπτωση που δεν υπάρχει το αρχείο ή που δεν
μπορεί να διαβαστεί ή που το αρχείο δεν περιέχει φύλλα.
     */
    public XlsxSheet (String Fn) throws GmException
    {
        try
        {
            Filename = Fn;
            InputStream Stream = new FileInputStream (Fn);
            wb = new XSSFWorkbook (Stream);
            sheet = wb.getSheetAt (0);
            if (sheet == null)
                throw new GmException ("No internal sheets in file",
GmException.SheetNoSuchSheet, "sheet 0");
        }
        catch (FileNotFoundException e)
        {
            throw new GmException ("Xlsx file does not exist.",
GmException.SheetNoSuchFile, Fn, e);
        }
        catch (IOException e)
        {
            throw new GmException ("IO Error reading Xlsx file.",
GmException.SheetIOError, Fn, e);
        }
    }

    /**
```

```
     * Επιστρέφει την τιμή ένός κελιού το οποίο πρέπει να περιέχει
αριθμητική τιμή. Αν δεν έχει αριθμητική τιμή θα δημιουργηθεί εξαίρεση.
     * @param r Η γραμμή του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @param c Η στήλη (αριθμητικά) του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @return Έναν αριθμό διπλής ακρίβειας που αντιπροσωπεύει την τιμή του
κελιού.
     * @throws GmException Σε περίπτωση ανύπαρκτης γραμμής, ή ανύπαρκτου
κελιού, ή σε περίπτωση που το περιεχόμενο του κελιού δεν είναι
     * αριθμός.
     */
    public double GetCellNumeric (int r, int c) throws GmException
    {
        double val;
        row = sheet.getRow (r);
        if (row == null)
            throw new GmException ("Row does not exist in current sheet.",
GmException.SheetNoSuchRow, Integer.toString (r));
        cell = row.getCell (c);
        if (cell == null)
            throw new GmException ("Cell (column) does not exist in row",
GmException.SheetNoSuchCell, Integer.toString (c));
        try
        {
            val = cell.getNumericCellValue ();
        }
        catch (IllegalStateException e)
        {
            throw new GmException ("Cell type mismatch. Data not numeric.",
GmException.SheetTypeMismatch, "", e);
        }
        return val;
    }

    /**
     * Επιστρέφει την τιμή ένός κελιού το οποίο πρέπει να περιέχει
αριθμητική τιμή. Αν δεν έχει αριθμητική τιμή θα δημιουργηθεί εξαίρεση.
     * @param r Η γραμμή του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @param c Η στήλη (αριθμητικά) του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @return Έναν αριθμό διπλής ακρίβειας που αντιπροσωπεύει την τιμή του
κελιού.
     * @throws GmException Σε περίπτωση ανύπαρκτης γραμμής, ή ανύπαρκτου
κελιού, ή σε περίπτωση που το περιεχόμενο του κελιού δεν είναι
     * αριθμός.
     */
    public String GetCellString (int r, int c) throws GmException
    {
        String val;
        val = null;
        row = sheet.getRow (r);
        if (row == null)
            throw new GmException ("Row does not exist in current sheet.",
GmException.SheetNoSuchRow, Integer.toString (r));
        cell = row.getCell (c);
        if (cell == null)
            throw new GmException ("Cell (column) does not exist in row",
GmException.SheetNoSuchCell, Integer.toString (c));
        try
        {
            val = cell.getStringCellValue ();
        }
        catch (IllegalStateException e)
        {
```

```
            throw new GmException ("Cell type mismatch. Data not String.",
GmException.SheetTypeMismatch, "", e);
        }
        return val;
    }

    /**
     * Επιστρέφει την τιμή ένός κελιού το οποίο πρέπει να περιέχει Date
τιμή. Αν δεν έχει τιμή ημερομηνίας θα δημιουργηθεί εξαίρεση.
     * @param r Η γραμμή του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @param c Η στήλη (αριθμητικά) του κελιού. Η αρίθμηση ξεκινά από το 0.
     * @return Ημερομηνία σε μορφή κλάσης Date.
     * @throws GmException Σε περίπτωση ανύπαρκτης γραμμής, ή ανύπαρκτου
κελιού, ή σε περίπτωση που το περιεχόμενο του κελιού δεν είναι
     * ημερομηνία.
     */
    public Date GetCellDate (int r, int c) throws GmException
    {
        Date val;
        val = null;
        row = sheet.getRow (r);
        if (row == null)
            throw new GmException ("Row does not exist in current sheet.",
GmException.SheetNoSuchRow, Integer.toString (r));
        cell = row.getCell (c);
        if (cell == null)
            throw new GmException ("Cell (column) does not exist in row",
GmException.SheetNoSuchCell, Integer.toString (c));
        try
        {
            val = cell.getDateCellValue ();
        }
        catch (IllegalStateException e)
        {
            throw new GmException ("Cell type mismatch. Data not Date",
GmException.SheetTypeMismatch, "", e);
        }
        return val;
    }

    /**
     * Επιστρέφει τον τύπο του περιεχομένου ενός κελιού. Ο τύπος
επιστρέφεται όπως αναφέρεται από την βιβλιοθήκη και πρέπει να συγκριθεί με
     * τις αντίστοιχες σταθερές της βιβλιοθήκης π.χ. Cell.CELL_TYPE_NUMERIC.
     * @param r Η γραμμή του κελιού (από 0).
     * @param c Η στήλη του κελιού (από 0).
     * @return Ο τύπος του κελιού.
     * @throws GmException Σε περίπτωση που το κελί δεν υπάρχει είτε λόγω
γραμμής, είτε λόγω στήλης.
     */
    public int GetCellType (int r, int c) throws GmException
    {
        row = sheet.getRow (r);
        if (row == null)
            throw new GmException ("Row does not exist in current sheet.",
GmException.SheetNoSuchRow, Integer.toString (r));
        cell = row.getCell (c);
        if (cell == null)
            throw new GmException ("Cell (column) does not exist in row",
GmException.SheetNoSuchCell, Integer.toString (c));
        return cell.getCellType ();
    }
```

```
    /**
     * Ορίζει την τιμή ενός κελιού σε συγκεκριμένο αριθμό. Αν το κελί δεν
υπάρχει θα δημιουργηθεί.
     * @param r Η γραμμή του κελιού.
     * @param c Η στήλη του κελιού.
     * @param Num Ο αριθμός που θα τοποθετηθεί.
     */
    public void SetCell (int r, int c, double Num)
    {
        row = sheet.getRow (r);
        if (row == null)
            row = sheet.createRow (r);
        cell = row.getCell (c);
        if (cell == null)
            cell = row.createCell (c);
        cell.setCellValue (Num);
    }

    /**
     * Ορίζει την τιμή ενός κελιού σε συγκεκριμένο String. Αν το κελί δεν
υπάρχει θα δημιουργηθεί.
     * @param r Η γραμμή του κελιού.
     * @param c Η στήλη του κελιού.
     * @param S Το String που θα τοποθετηθεί.
     */
    public void SetCell (int r, int c, String S)
    {
        row = sheet.getRow (r);
        if (row == null)
            row = sheet.createRow (r);
        cell = row.getCell (c);
        if (cell == null)
            cell = row.createCell (c);
        cell.setCellValue (S);
    }

    /**
     * Ορίζει την τιμή ενός κελιού σε συγκεκριμένη ημερομηνία. Αν το κελί
δεν υπάρχει θα δημιουργηθεί.
     * @param r Η γραμμή του κελιού.
     * @param c Η στήλη του κελιού.
     * @param D Η ημερομηνία που θα τοποθετηθεί.
     */
    public void SetCell (int r, int c, Date D)
    {
        row = sheet.getRow (r);
        if (row == null)
            row = sheet.createRow (r);
        cell = row.getCell (c);
        if (cell == null)
            cell = row.createCell (c);
        cell.setCellValue (D);
    }

    /**
     * Επιστρέφει τον αριθμό της τελευταίας γραμμής του φύλλου.
     * @return Ο αριθμός της τελευταίας γραμμής.
     */
    public int GetLastRow ()
    {
        return sheet.getLastRowNum ();
```

```
    }

    /**
     * Αποθηκεύει το workbook στον δίσκο με το όνομα που δίδεται. Αν το
όνομα είναι null, το αρχείο αποθηκεύεται με το όνομα που άνοιξε
     * όπως αυτό αποθηκεύτηκε στον κατασκευαστή.
     * @param fn Το πλήρες όνομα του αρχείου, null για αποθήκευση με το
αρχικό όνομα.
     * @throws GmException Σε περίπτωση που το αρχείο δεν μπορεί να
δημιουργηθεί με αυτό το όνομα ή δημιουργηθεί IO εξαίρεση κατά την
     * εγγραφή των δεδομένων.
     */
    public void Save (String fn) throws GmException
    {
        OutputStream out = null;
        try
        {
            if (fn == null)
            {
                fn = Filename;
            }
            out = new FileOutputStream (fn);
            wb.write (out);
            out.close ();
        }
        catch (FileNotFoundException e)
        {
            throw new GmException ("Could not create file to save sheet. ",
GmException.SheetInvalidFilename, fn, e);
        }
        catch (IOException e)
        {
            throw new GmException ("IO Error during saving excel file. ",
GmException.SheetIOError, fn, e);
        }
    }

    /**
     * Ορίζει αν το orientation της τρέχουσας σελίδας θα είναι landscape
(στο πλάι).
     * @param Landscape true για landscape, false για portrait
     */
    public void SetLandscape (boolean Landscape)
    {
        if (Landscape)
            sheet.getPrintSetup ().setOrientation
(PrintOrientation.LANDSCAPE);
        else
            sheet.getPrintSetup ().setOrientation
(PrintOrientation.PORTRAIT);
    }
}
```

ΠΑΡΑΡΤΗΜΑ Γ

Ο ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ CLIENT

PACKAGE: org.gmele.dissertation.client

CLASS: ExamActiv

```
package org.gmele.dissertation.client;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;

import org.gmele.dissertation.daemon.IDaemonServ;
import org.gmele.dissertation.testhandler.Question;
import org.gmele.dissertation.testhandler.TestSheet;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.graphics.BitmapFactory;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.RemoteException;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;
```

```
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

public class ExamActiv extends Activity implements OnClickListener
{
	final String YMEPath	= "/sdcard/external_sd/YME/";
	TestSheet sheet;
	Question CurQ;
	int SelectedAns;
	int CurQuest;
	int NoAnsQuests;
	TextView TvQNumber;
	ImageView IvQuestPic;
	TextView TvQuest;
	TextView[] TvAns;
	Button BtNext;
	Button BtOk;
	ProgressDialog Pdialog;
	ProgressBar  PbTimeleft;
	MediaPlayer  mp;
	long StartTime;
	long EndTime;
	IDaemonServ MyDaemonServ = null;
	Timer		timer;

	private ServiceConnection MyConnection = new ServiceConnection()
	{
	 public void onServiceConnected (ComponentName className, IBinder
service)
	 {
	     MyDaemonServ = IDaemonServ.Stub.asInterface (service);
	     //try... register callback here
	 }

	 public void onServiceDisconnected (ComponentName className)
	 {
	     //MyDaemonServ = null;
	     MakeToast ("Service Died. Why???");
	 }
    };

	TimerTask task = new TimerTask ()
	{
	      @Override
	      public void run ()
	      {
	            long CurTime = System.currentTimeMillis () / 1000;
	            if (CurTime > EndTime)
	            {
	                  try
	                  {
	                  PbTimeleft.setProgress (PbTimeleft.getMax ());
	                  timer.cancel ();
	                  task.cancel ();
	                  timer.purge ();
	                  Bundle B = new Bundle ();
	                  B.putString ("Command", "EndExamin");
	                  Message M = new Message ();
	                  M.setData (B);
	                  handler.sendMessage (M);
	                  return;
```

```
                        }
                        catch (Exception e)
                        {
                                MakeToast ("EGINE" + e.getMessage ());
                        }
                }
                int P = (int) (CurTime - StartTime);
                PbTimeleft.setProgress (P);
        }
    };

    Handler handler = new Handler ()
    {
        @Override
        public void handleMessage (Message message)
        {
                String Comm = message.getData ().getString ("Command");
                if (Comm.equals ("Toast"))
                {
                        Toast toast = Toast.makeText (ExamActiv.this,
message.getData ().getString ("Message"),
                                        Toast.LENGTH_LONG);
                        toast.setGravity (Gravity.CENTER, 0, 0);
                        toast.show ();
                        return;
                }
                if (Comm.equals ("EndExamin"))
                {
                        EndExamin ();
                        return;
                }
                //MakeToast ("$$$$$$$$$$$$$");
        }
    };

    @Override
    public void onCreate (Bundle savedInstanceState)
    {
        super.onCreate (savedInstanceState);
        int i;
        requestWindowFeature (Window.FEATURE_NO_TITLE);
        getWindow ().setFlags
(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView (R.layout.exam);
        TvQNumber = (TextView) findViewById (R.id.TvQNumber);
        IvQuestPic = (ImageView) findViewById (R.id.IvQuestPic);
        TvQuest = (TextView) findViewById (R.id.TvQuest);
        TvAns = new TextView[5];
        TvAns[0] = (TextView) findViewById (R.id.TvAns1);
        TvAns[1] = (TextView) findViewById (R.id.TvAns2);
        TvAns[2] = (TextView) findViewById (R.id.TvAns3);
        TvAns[3] = (TextView) findViewById (R.id.TvAns4);
        TvAns[4] = (TextView) findViewById (R.id.TvAns5);
        BtNext = (Button) findViewById (R.id.BtNext);
        BtOk = (Button) findViewById (R.id.BtOK);
        PbTimeleft = (ProgressBar) findViewById (R.id.PbTimeleft);
        TvQuest.setOnClickListener (this);
        for (i = 0; i < 5; i++)
                TvAns[i].setOnClickListener (this);
        BtNext.setOnClickListener (this);
        BtOk.setOnClickListener (this);
```

```
            Intent int1 = new Intent ();
            int1.setClassName ("org.gmele.dissertation.daemon",
"org.gmele.dissertation.daemon.Daemon");
            bindService (int1, MyConnection, Context.BIND_AUTO_CREATE);
            InitTest ();
            InitTimer ();
            NextQuest ();
      }

      @Override
      public void onDestroy ()
      {
            super.onDestroy ();
      }

      public void MakeToast (String Mess)
      {
            Bundle B = new Bundle ();
            B.putString ("Command", "Toast");
            B.putString ("Message", Mess);
            Message M = new Message ();
            M.setData (B);
            handler.sendMessage (M);
      }

      public void InitTest ()
      {
            try
            {
                  sheet = TestSheet.Load (YMEPath + "TestFile.test");
                  sheet.Header.DateStarted = new java.sql.Date
(System.currentTimeMillis ());
                  if (sheet.Header.Speech)
                  {
                        mp = new MediaPlayer ();
                  }
            }
            catch (TestException e)
            {
                  MakeToast ("TestException in reading TestFile.test");
                  finish ();

            }
            catch (GmException e)
            {
                  MakeToast ("GmException in reading TestFile.test");
                  finish ();
            }
            CurQuest = -1;
            NoAnsQuests = 0;
      }

      public void InitTimer ()
      {
            StartTime = System.currentTimeMillis () / 1000;
            EndTime = StartTime + sheet.Header.ExamTime * 60;
            PbTimeleft.setMax ((int) (EndTime - StartTime));
            PbTimeleft.setProgress (0);
            timer = new Timer ();
            timer.schedule (task, 3000, 3000);
      }
```

```
	public void NextQuest ()
	{
		int i;
		do
		{
			CurQuest++;
			if (CurQuest == sheet.NoQ)
				CurQuest = 0;
		}
		while (sheet.Quests[CurQuest].Choice != 0);
		CurQ = sheet.Quests[CurQuest];
		TvQNumber.setText ("Ερώτηση " + (CurQuest + 1));
		LoadQPhoto ();
		TvQuest.setText (CurQ.Verbal);
		for (i = 0; i < 5; i++)
		{
			TvAns[i].setBackgroundColor (0x00000000);
			if (i < CurQ.NoAns)
			{
				TvAns[i].setText (CurQ.AnsVerbal[i]);
				TvAns[i].setVisibility (View.VISIBLE);
			}
			else
			{
				TvAns[i].setText (null);
				TvAns[i].setVisibility (View.GONE);
			}
		}
		BtOk.setEnabled (false);
		SelectedAns = -1;
		PlaySound (10);

	}

	public void PlaySound (int W)
	{
		String Sfn = null;
		int p = 0;
		if (sheet.Header.Speech)
		{
			try
			{
				p = 1;
				mp.stop ();
				p=11;
				mp.reset ();
				p = 2;
				if (W == 11)
					return;
				if (W == 10)
					Sfn = YMEPath + "SOUNDS/" + CurQ.Sound +
".ogg";

				else
					Sfn = YMEPath + "SOUNDS/" + CurQ.AnsSound[W]
+ ".ogg";
				p = 3;
				mp.setDataSource (Sfn);
				p = 4;
				mp.prepare ();
				p = 5;
				mp.start ();
```

```
                        p = 6;
                    }
                    catch (IOException e)
                    {
                        MakeToast ("Sound Error: " + p + Sfn);
                    }
                    catch (IllegalStateException e)
                    {
                        MakeToast ("State " + p + " " +Sfn);
                    }
                    catch (IllegalArgumentException e)
                    {
                        MakeToast ("Argument " + p + e.getMessage ());
                    }
                }
        }

        public void LoadQPhoto ()
        {
                FileInputStream is;
                if (CurQ.Photo.equals ("0"))
                        IvQuestPic.setImageDrawable (null);
                else
                {
                        try
                        {
                                is = new FileInputStream (YMEPath + "PICTURES/" +
CurQ.Photo + ".jpg");
                                IvQuestPic.setImageBitmap
(BitmapFactory.decodeStream (is));
                        }
                        catch (FileNotFoundException e)
                        {
                                MakeToast ("Picture Error:" + CurQ.Photo);
                        }
                }
        }

        public void EndExamin ()
        {
                int p = 0;
                try
                {
                        p = 1;
                        Pdialog = ProgressDialog.show (ExamActiv.this, "",
                                        "Τέλος Εξέτασης. Παρακαλώ περιμένετε τα
αποτελέσματα.", true, false);
                        p = 2;
                        try
                        {
                                sheet.Header.DateEnded = new java.sql.Date
(System.currentTimeMillis ());
                                TestSheet.Save (sheet, YMEPath + "TestFile.test");
                        }
                        catch (TestException e)
                        {
                                MakeToast ("Αδυναμία αποθήκευσης απαντήσεων");
                        }
                        p = 3;
                        if (sheet.Header.Speech)
                                mp.release ();
                        p = 4;
```

```
                    try
                    {
                          if (NoAnsQuests == sheet.NoQ)
                                MyDaemonServ.SetCommVal (4);
                          else
                                MyDaemonServ.SetCommVal (5);
                    }
                    catch (RemoteException e)
                    {
                          MakeToast ("Service Error...");
                    }
                    p = 5;
                    WaitAndTermin ();
                    //p = 51;
                    //unbindService (MyConnection);
                    //p = 6;
                    //Pdialog.dismiss ();
                    //p = 7;
                    //finish ();
                    //p = 8;
              }
              catch (Exception e)
              {
                    MakeToast (e.getMessage () + " " + e.getClass ().getName
() + " " + p);
              }
       }

       /**
        * Περιμένει μέχρι η τιμή της μεταβλητής ClientComm στο Δαίμονα να
πάρει την τιμή 6 που σημαίνει ότι ο server
        * έδωσε εντολή ολοκλήρωσης της όλης διαδικασίας εξέτασης.
        */
       public void WaitAndTermin ()
       {
              new Thread ()
              {
                    @Override
                    public void run ()
                    {
                          int V;
                          do
                          {
                                try
                                {
                                      Thread.sleep (10000);
                                      V = MyDaemonServ.GetCommVal ();
                                }
                                catch (RemoteException e)
                                {
                                      MakeToast ("Service error!");
                                      V = 6;
                                }
                                catch (Exception e)
                                {
                                      //MakeToast ("EEEEEEE????");
                                      V = 6;
                                }
                          }
                          while (V != 6);
                          unbindService (MyConnection);
                          Pdialog.dismiss ();
```

```
                        finish ();
                    }
                }.start ();
        }

        @Override
        public void onClick (View v)
        {
                int i;
                for (i = 0; i < CurQ.NoAns; i++)
                {
                        if (v == TvQuest)
                        {
                                PlaySound (10);
                                return;
                        }
                        if (v == TvAns[i])
                        {
                                if (SelectedAns != -1)
                                        TvAns[SelectedAns].setBackgroundColor
(0x00000000);
                                TvAns[i].setBackgroundColor (0xFFFFFFC0);
                                TvAns[i].refreshDrawableState ();
                                SelectedAns = i;
                                BtOk.setEnabled (true);
                                PlaySound (i);
                                return;
                        }
                }
                if (v == BtOk)
                {
                        PlaySound (11);
                        CurQ.Choice = CurQ.AnsAA[SelectedAns];
                        NoAnsQuests++;
                        if (NoAnsQuests < sheet.NoQ)
                                NextQuest ();
                        else
                        {
                                timer.cancel ();
                                task.cancel ();
                                timer.purge ();
                                BtOk.setEnabled (false);
                                BtNext.setEnabled (false);
                                //Pdialog = ProgressDialog.show (ExamActiv.this,
"",
                                //     "Τέλος Εξέτασης. Παρακαλώ περιμένετε τα
αποτελέσματα.", true, false);
                                Bundle B = new Bundle ();
                                B.putString ("Command", "EndExamin");
                                Message M = new Message ();
                                M.setData (B);
                                handler.sendMessage (M);
                        }
                }
                if (v == BtNext)
                {
                        PlaySound (11);
                        NextQuest ();
                }
        }
}
```

CLASS: IntroScreen

```
package org.gmele.dissertation.client;

import java.util.Timer;
import java.util.TimerTask;

import org.gmele.dissertation.daemon.IDaemonServ;
import org.gmele.dissertation.testhandler.TestSheet;
import org.gmele.dissertation.testhandler.exceptions.TestException;
import org.gmele.general.exceptions.GmException;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.RemoteException;
import android.os.SystemClock;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

public class IntroScreen extends Activity implements OnClickListener
{
	TestSheet sheet;
	final String YMEPath = "/sdcard/external_sd/YME/";
	Button BtStart;
	Button BtAbsent;
	ProgressDialog Pdialog;
	IDaemonServ MyDaemonServ = null;
	boolean Present;
	Timer timer = null;

	Handler handler = new Handler ()
	{
		@Override
		public void handleMessage (Message message)
		{
			String Comm = message.getData ().getString ("Command");
			if (Comm.equals ("Toast"))
			{
				Toast toast = Toast.makeText (IntroScreen.this,
message.getData ().getString ("Message"),
					Toast.LENGTH_LONG);
				toast.setGravity (Gravity.CENTER, 0, 0);
				toast.show ();
			}
		}
	};
```

```
	TimerTask task = new TimerTask ()
	{
		@Override
		public void run ()
		{
			int V;
			try
			{
				V = MyDaemonServ.GetCommVal ();
				if (V == 3)
				{
					timer.cancel ();
					timer.purge ();
					task.cancel ();
					NextStep ();
				}
			}
			catch (RemoteException e)
			{
				MakeToast ("Service Error...");
				finish ();
			}
		}
	};

	private ServiceConnection MyConnection = new ServiceConnection()
	{
	  public void onServiceConnected (ComponentName className, IBinder
service)
	  {
	      MyDaemonServ = IDaemonServ.Stub.asInterface (service);
	      //try... register callback here
	  }

	  public void onServiceDisconnected (ComponentName className)
	  {
	      //MyDaemonServ = null;
	      MakeToast ("Service Died. Why???");
	  }
    };

	@Override
	public void onCreate (Bundle savedInstanceState)
	{
		TextView TvTmp;
		ImageView IvTmp;

		super.onCreate (savedInstanceState);
		requestWindowFeature(Window.FEATURE_NO_TITLE);
	  getWindow().setFlags (WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
		setContentView (R.layout.main);
		MakeToast ("/// Starting");
		Intent int1 = new Intent ();
		int1.setClassName ("org.gmele.dissertation.daemon",
"org.gmele.dissertation.daemon.Daemon");
		bindService (int1, MyConnection, Context.BIND_AUTO_CREATE);
		try
		{
			sheet = TestSheet.Load (YMEPath + "TestFile.test");
		}
		catch (TestException e)
```

```
			{
				MakeToast ("TestException in reading TestFile.test");
				//finish ();

			}
			catch (GmException e)
			{
				MakeToast ("GmException in reading TestFile.test");
				//finish ();
			}
			TvTmp = (TextView) findViewById (R.id.TvClientID);
			TvTmp.setText (Integer.toString (sheet.Header.ClientID));
			TvTmp = (TextView) findViewById (R.id.TvName);
			TvTmp.setText (sheet.Header.Surname + "  " +
sheet.Header.Firstname);
			TvTmp = (TextView) findViewById (R.id.TvID);
			TvTmp.setText (sheet.Header.IDNum);
			TvTmp = (TextView) findViewById (R.id.TvQuestTime);
			TvTmp.setText (sheet.Header.Kateg + "   [ " +
sheet.Header.ExamTime + "' ]");
			IvTmp = (ImageView) findViewById (R.id.IvSound);
			if (sheet.Header.Speech)
				IvTmp.setImageResource (R.drawable.speechon);
			else
				IvTmp.setImageResource (R.drawable.speechoff);
			BtStart = (Button) findViewById (R.id.BtStart);
			BtStart.setOnClickListener (this);
			BtAbsent = (Button) findViewById (R.id.BtAbsent);
			BtAbsent.setOnClickListener (this);
		}

		@Override
		public void onBackPressed()
		{
			return;
		}

		public void MakeToast (String Mess)
		{

			Bundle B = new Bundle ();
			B.putString ("Command", "Toast");
			B.putString ("Message", Mess);
			Message M = new Message ();
			M.setData (B);
			handler.sendMessage (M);
		}

		@Override
		public void onClick (View v)
		{
			if (v == BtStart)
			{
				BtStart.setEnabled (false);
				BtAbsent.setEnabled (false);
				Present = true;
				Pdialog = ProgressDialog.show (this, "Αναμονή Έναρξης",
"Παρακαλώ περιμένετε την έναρξη της διαδικασίας",
					true, false);
				try
				{
					if (MyDaemonServ == null)
```

```
                                MakeToast ("No Service Connection");
                        else
                        {
                                MyDaemonServ.SetCommVal (1);
                                timer = new Timer ();
                                timer.schedule (task, 3000, 5000);
                        }
                }
                catch (RemoteException e)
                {
                        MakeToast ("Service Error...");
                        finish ();
                }
        }
        if (v == BtAbsent)
        {
                BtStart.setEnabled (false);
                BtAbsent.setEnabled (false);
                Present = false;
                Pdialog = ProgressDialog.show (this, "", "Αναμονή εντολής
ακύρωσης από τον Server..",
                                true, false);
                try
                {
                        MyDaemonServ.SetCommVal (2);
                        timer = new Timer ();
                        timer.schedule (task, 3000, 5000);
                }
                catch (RemoteException e)
                {
                        MakeToast ("Service Error...");
                        finish ();
                }
        }
}

@Override
public void onDestroy ()
{
        super.onDestroy ();
}

public void NextStep ()
{
        if (Present)
        {
                unbindService (MyConnection);
                Pdialog.cancel ();
                Intent intent = new Intent (this, ExamActiv.class);
                startActivity (intent);
                finish ();
        }
        else
        {
                try
                {
                        MyDaemonServ.SetCommVal (4);
                        WaitTermin ();
                }
                catch (RemoteException e)
                {
                        MakeToast ("Service Error.....");
```

```
                    }
                    Pdialog.cancel ();
                    unbindService (MyConnection);
                    //MakeToast ("/// Finishing...");
                    finish ();
                    //MakeToast ("/// Finished!!!!");
             }
      }

      /**
       * Περιμένει μέχρι η τιμή της μεταβλητής ClientComm στο Δαίμονα να
πάρει την τιμή 6 που σημαίνει ότι ο server
       * έδωσε εντολή ολοκλήρωσης της όλης διαδικασίας εξέτασης.
       */
      public void WaitTermin ()
      {
             int V;
             do
             {
                    SystemClock.sleep (5000);
                    try
                    {
                           V = MyDaemonServ.GetCommVal ();
                    }
                    catch (RemoteException e)
                    {
                           MakeToast ("Service error!");
                           V = 6;
                    }
             }
             while (V != 6);
      }

      /*
      @Override
      protected void onStart ()
      {
             super.onStart ();

      }

      @Override
    protected void onRestart ()
      {
             super.onRestart ();
      }

      @Override
    protected void onResume ()
      {
             super.onResume ();
      }

      @Override
    protected void onPause()
      {
             super.onPause ();
      }

      @Override
    protected void onStop()
      {
```

```
            super.onStop ();
        }
        */
}
```

RESOURCES

exam.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LoMain"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bgr"
    android:orientation="vertical"
    android:weightSum="1" >

    <TextView
        android:id="@+id/TvQNumber"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="15px"
        android:layout_marginTop="20px"
        android:gravity="center_vertical|center_horizontal"
        android:textColor="#FF000000"
        android:textSize="35px"
        android:textStyle="normal" >
    </TextView>

    <ImageView
        android:id="@+id/IvQuestPic"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="15px"
        android:layout_marginTop="15px"
        android:clickable="false" >
    </ImageView>

    <TextView
        android:id="@+id/TvQuest"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="25px"
        android:layout_marginLeft="10px"
        android:layout_marginRight="10px"
        android:clickable="true"
        android:paddingBottom="5px"
        android:paddingLeft="5px"
        android:paddingRight="5px"
        android:paddingTop="5px"
        android:text="Ερώτηση: Αρνάκι άσπρο και παχύ της μάνας το καμάρι"
        android:textColor="#FF000000"
        android:textSize="30px"
        android:textStyle="italic" >
    </TextView>
```

```
<ScrollView
    android:id="@+id/SvAnswers"
    android:layout_width="match_parent"
    android:layout_height="0px"
    android:layout_weight="1"
    android:scrollbarStyle="insideInset" >

    <LinearLayout
        android:id="@+id/LoAnswers"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/TvAns1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20px"
            android:layout_marginLeft="10px"
            android:layout_marginRight="10px"
            android:paddingBottom="5px"
            android:paddingLeft="5px"
            android:paddingRight="5px"
            android:paddingTop="5px"
            android:text="TextView"
            android:textColor="#000000"
            android:textSize="30px" >
        </TextView>

        <TextView
            android:id="@+id/TvAns2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20px"
            android:layout_marginLeft="10px"
            android:layout_marginRight="10px"
            android:paddingBottom="5px"
            android:paddingLeft="5px"
            android:paddingRight="5px"
            android:paddingTop="5px"
            android:text="TextView"
            android:textColor="#000000"
            android:textSize="30px" >
        </TextView>

        <TextView
            android:id="@+id/TvAns3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20px"
            android:layout_marginLeft="10px"
            android:layout_marginRight="10px"
            android:paddingBottom="5px"
            android:paddingLeft="5px"
            android:paddingRight="5px"
            android:paddingTop="5px"
            android:text="TextView"
            android:textColor="#000000"
            android:textSize="30px" >
        </TextView>
```

```
            <TextView
                android:id="@+id/TvAns4"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="20px"
                android:layout_marginLeft="10px"
                android:layout_marginRight="10px"
                android:paddingBottom="5px"
                android:paddingLeft="5px"
                android:paddingRight="5px"
                android:paddingTop="5px"
                android:text="TextView"
                android:textColor="#000000"
                android:textSize="30px" >
            </TextView>

            <TextView
                android:id="@+id/TvAns5"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="20px"
                android:layout_marginLeft="10px"
                android:layout_marginRight="10px"
                android:paddingBottom="5px"
                android:paddingLeft="5px"
                android:paddingRight="5px"
                android:paddingTop="5px"
                android:text="TextView"
                android:textColor="#000000"
                android:textSize="30px" >
            </TextView>
        </LinearLayout>
    </ScrollView>

    <RelativeLayout
        android:id="@+id/BtLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10px" >

        <Button
            android:id="@+id/BtNext"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginBottom="15px"
            android:layout_marginLeft="15px"
            android:paddingBottom="15px"
            android:paddingTop="15px"
            android:text="ΕΠΟΜΕΝΗ"
            android:textSize="25px"
            android:textStyle="bold" >
        </Button>

        <ProgressBar
            android:id="@+id/PbTimeleft"
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:layout_marginLeft="20px"
```

```
            android:layout_marginRight="20px"
            android:layout_toLeftOf="@+id/BtOK"
            android:layout_toRightOf="@+id/BtNext"
            android:clickable="false" >
        </ProgressBar>

        <Button
            android:id="@+id/BtOK"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="false"
            android:layout_marginBottom="15px"
            android:layout_marginRight="15px"
            android:paddingBottom="15px"
            android:paddingTop="15px"
            android:text="ΕΠΙΒΕΒΑΙΩΣΗ"
            android:textSize="25px"
            android:textStyle="bold" >
        </Button>
    </RelativeLayout>
</LinearLayout>
```

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/LoMain"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bgr"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/TvClientID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="80px"
        android:layout_marginTop="15mm"
        android:gravity="center"
        android:paddingBottom="15px"
        android:paddingLeft="25px"
        android:paddingRight="25px"
        android:paddingTop="10px"
        android:textColor="#000000"
        android:textSize="90px"
        android:textStyle="bold" >

        <requestFocus>
        </requestFocus>
    </TextView>

    <TextView
        android:id="@+id/TvName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="30px"
        android:gravity="center_horizontal"
```

```
    android:textColor="#000000"
    android:textSize="30px"
    android:textStyle="bold" >
</TextView>

<TextView
    android:id="@+id/TvID"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginBottom="60px"
    android:gravity="center_horizontal"
    android:textColor="#000000"
    android:textSize="30px"
    android:textStyle="normal" >
</TextView>

<TextView
    android:id="@+id/TvQuestTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginBottom="30px"
    android:editable="false"
    android:textColor="#000000"
    android:textSize="30px" >
</TextView>

<ImageView
    android:id="@+id/IvSound"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:clickable="false"
    android:focusable="false" >
</ImageView>

<RelativeLayout
    android:id="@+id/LoButtons"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/BtStart"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_marginBottom="25px"
        android:layout_marginLeft="25px"
        android:paddingBottom="30px"
        android:paddingLeft="20px"
        android:paddingRight="20px"
        android:paddingTop="30px"
        android:text="ΠΑΡΩΝ / ΟΥΣΑ"
        android:textSize="25px"
        android:textStyle="bold" >
    </Button>

    <Button
        android:id="@+id/BtAbsent"
        android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_marginBottom="25px"
            android:layout_marginRight="25px"
            android:paddingBottom="30px"
            android:paddingLeft="20px"
            android:paddingRight="20px"
            android:paddingTop="30px"
            android:text="ΑΠΟΥΣΙΑΖΕΙ"
            android:textSize="25px"
            android:textStyle="bold" >
        </Button>
    </RelativeLayout>

</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.gmele.dissertation.client"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="9" />

    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" >
    </uses-permission>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".IntroScreen"
            android:enabled="true"
            android:exported="true"
            android:label="@string/app_name"
            android:launchMode="standard" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ExamActiv"
            android:label="@string/act_name" >
        </activity>
    </application>

</manifest>
```

ΠΑΡΑΡΤΗΜΑ Δ

Ο ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ DAEMON

PACKAGE: org.gmele.dissertation.daemon

CLASS: Daemon

```
package org.gmele.dissertation.daemon;

import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.ServerSocket;
import java.net.Socket;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.os.Message;
import android.os.RemoteException;
import android.view.Gravity;
import android.widget.Toast;

/**
 * Η κλάση Daemon υλοποιεί το service το οποίο είναι υπεύθυνο για τον έλεγχο
ενός client και την επικοινωνία με τον
 * server. Το service, κανονικά, εκκινείται με την έναρξη του λειτουργικού
της συσκευής.<p>
 * Το Daemon δέχεται εντολές από το server μέσα από TCP socket το οποίο
ακούει την πόρτα 25123. Οι εντολές είναι
 * ακέραιοι αριθμοί οι οποίοι αποστέλλονται ως strings και μπορεί να έχουν
παραμέτρους. Οι εντολές περιγράφονται στο
 * παρόν javadoc στα σημεία που ορίζονται και που γίνεται ο χειρισμός τους.
 * @author Giorgos Meletiou
 * @version 1.0
 */
```

```
public class Daemon extends Service implements DaemonConsts
{

	Toast  toast;
	MyHandler handler;
	int          Counter;
	Thread NetThread = null;
	int Status;
	int ClientComm;

	private final IDaemonServ.Stub MyBinder = new IDaemonServ.Stub()
	{


		@Override
		public void SetCommVal (int Val) throws RemoteException
		{
			SetClientCom (Val);
		}

		@Override
		public int GetCommVal () throws RemoteException
		{
			return GetClientCom ();
		}
	};

	@Override
	public void onCreate ()
	{
		handler = new MyHandler (getApplicationContext ());
		//MakeToast ("Daemon Started..", 2);
		ClientComm = 6;
		//Status = StatusHelloWaiting;
		NetThread = new Thread ()
		{
			public void run ()
			{
				NetHandler ();
			}
		};
		NetThread.start ();

	}

	@Override
	public void onDestroy ()
	{
		MakeToast ("Destroyed????", 1);
	}

	@Override
	public IBinder onBind (Intent intent)
	{
		//MakeToast ("Binding...", 1);
		return MyBinder;
	}

	@Override
	public boolean onUnbind (Intent intent)
	{
		//MakeToast ("UnBinding...", 1);
```

```
			return false;
		}
		/**
		 * Η μέθοδος NetHandler τρέχει σε ξεχωριστό thread και είναι υπεύθυνη για την επικοινωνία με τον Examination
		 *  server. Λαμβάνει εντολές από το Server και τις εκτελεί. Οι εντολές είναι ακέραιοι αριθμοί οι οποίοι διαβάζονται
		 *  από tcp socket ως string. Έχουν οριστεί σε σταθερές και οποιαδήποτε μη αναγνωρίσημη εντολή δημιουργεί σχετική
		 *  αντίδραση. <p>
		 * Η μέθοδος δημιουργεί server socket μέσω του οποίου φτάνουν οι εντολές. Κάθε εντολή εξυπηρετείται από ένα accept
		 * και μετά tosocket κλείνει και πρέπει να ξαναγίνει accept.  Άγνωστες εξαιρέσεις τυπώνονται στην οθόνη (toast) για
		 *  λόγους εκσφαλμάτωσης. <p>
		 * Η μέθοδος διατηρεί μόνο ένα ανοιχτό κανάλι τη φορά. Έτσι επικοινωνεί μόνο με ένα Examination Server. Η
		 * επεξεργασία και η εκτέλεση κάθε εντολής γίνεται μέσω ξεχωριστής μεθόδου, αλλά στο ίδιο thread.
		 */
		public void NetHandler ()
		{
			ServerSocket ServSoc;
			Socket Soc;
			InputStream is;
			String S;
			boolean cont;
			S = null;
			Soc = null;
			is = null;
			ServSoc = null;
			cont = true;

			try
			{
				ServSoc = new ServerSocket (25123);
				//MakeToast ("Waiting Connections...", 2);
			}
			catch (IOException e)
			{
				MakeToast ("Could not open server socket. Service ends now...", 1);
				stopSelf ();
			}
			do
			{
				try
				{
					Soc = ServSoc.accept ();
					is = Soc.getInputStream ();
					S = GetLine (is);
					//MakeToast ("[" + S + "]", 3);
					if (Integer.parseInt (S) == CommHello)
						DoHello (Soc, is);
					if (Integer.parseInt (S) == CommGetFile)
						DoGetFile (Soc, is);
					if (Integer.parseInt (S) == CommSendFile)
						DoSendFile (Soc, is);
					if (Integer.parseInt (S) == CommStartClient)
						DoStartClient (Soc, is);
					if (Integer.parseInt (S) == CommSendClientComm)
						DoSendClientComm (Soc, is);
```

```
                        if (Integer.parseInt (S) == CommStartExam)
                              DoStartExam (Soc, is);
                        if (Integer.parseInt (S) == CommExamCompleted)
                              DoExamCompleted (Soc, is);
                        is.close ();
                        Soc.close ();
                  }
                  catch (IOException e)
                  {
                        MakeToast ("IO...: " + e.getMessage () + " " +
e.toString (), 1);
                  }
                  catch (NullPointerException e)
                  {
                        MakeToast ("Type : " + e.getClass
().getCanonicalName (), 1);
                        if (Soc == null)
                              MakeToast ("Socket is Null", 1);
                        if (is == null)
                              MakeToast ("Input buffer is Null", 1);
                        if (S == null)
                              MakeToast ("String is Null", 1);
                  }
                  catch (Exception e)
                  {
                        MakeToast ("Unexpected Exception : " + e.getClass
().getCanonicalName (), 1);
                  }
                  finally
                  {
                        try
                        {
                              if (!Soc.isClosed ())
                                    Soc.close ();
                        }
                        catch (IOException e)
                        {
                              MakeToast ("In Here??????? " + e.getClass
().getCanonicalName (), 1);
                        }
                  }

            }
            while (cont);
            //MakeToast ("Service shuting down....", 2);
      }

      @Override
      public int onStartCommand (Intent intent, int flags, int id)
      {
            return START_STICKY;
      }

      /**
       * Δημιουργεί ένα toast μεγάλης διάρκειας με οριζόντια κεντρική
στοίχιση. Το κείμενο του toast και η θέση του (πάνω,
       * μέση, κάτω) δίδονται ως ορίσματα.
       * @param mess Το μήνυμα που θα τυπωθεί.
       * @param Posis Η θέση του μηνύματος. 1 = πάνω, 2 = μέση, 3 = κάτω.
Οποιαδήποτε άλλη τιμή αγνοείται και το μήνυμα
       * βγαίνει στη μέση.
       */
```

```
	private void MakeToast (String mess, int Posis)
	{
		int P = 2;
		if (Posis == 1)
			P = Gravity.CENTER_HORIZONTAL | Gravity.TOP;
		if (Posis == 2)
			P = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;
		if (Posis == 3)
			P = Gravity.CENTER_HORIZONTAL | Gravity.BOTTOM;
		Bundle B = new Bundle ();
		B.putString ("Message", mess);
		B.putInt ("Position", P);
		Message M = new Message ();
		M.setData (B);
		handler.sendMessage (M);
	}

	/**
	 * Διαβάζει μία γραμμή ως string από ένα InputStream. Χρησιμοποιείται
προκειμένου τα sockets να διαβάζονται με
	 * InputStream για να γίνεται και μεταφορά block από bytes.
	 * @param is Το inputstream από το οποίο διαβάζονται bytes.
	 * @return Το string που διαβάστηκε
	 * @throws IOException Σε περίπτωση που τελειώσουν τα δεδομένα εισόδου
χωρίς να βρεθεί "\n" ή δημιουργηθεί πρόβλημα
	 * στην κλήση της μεθόδου read.
	 */
	private String GetLine (InputStream is) throws IOException
	{
		ByteArrayOutputStream buffer = new ByteArrayOutputStream ();
		int b;
		String str;
	    while ((b = is.read()) != 0x0A)
	    {
		if (b < 0)
		{
			throw new IOException("Not enough socket input");
		}
		buffer.write(b);
	    }
	    str = new String (buffer.toByteArray(), "UTF-8");
	    return  str;
	}

	/**
	 * Η μέθοδος γράφει ένα string σε ένα OutputStream ως σύνολο από
bytes. Το string κωδικοποιείται κατά UTF-8 και στο
	 * τέλος του προστίθεται η αλλαγή γραμμής.
	 * @param os Το OutputStream στο οποίο θα γραφεί η γραμμή. Κανονικά,
πρέπει να ανήκει σε socket.
	 * @param Line Το string στο οποίο θα προστεθεί η αλλαγη γραμμής και
θα γραφεί στο stream.
	 * @throws IOException σε περίπτωση που υπάρχει πρόβλημα με το socket.
	 */
	private void SendLine (OutputStream os, String Line) throws
IOException
	{
		byte[] buffer;
		try
		{
			Line = Line + "\n";
			buffer = Line.getBytes ("UTF-8");
```

```
                os.write (buffer);
                os.flush ();
            }
            catch (UnsupportedEncodingException e)
            {
                MakeToast (e.getMessage (), 3);
            }
        }

        /**
         * Υλοποιεί την εντολή "Hello". Η εντολή διαβάζει ένα string το οποίο
πρέπει να είναι το "HELLO" και το στέλνει
         * πίσω στο Server. Αν η κατάσταση του Daemon είναι {@link
#StatusHelloWaiting} μετατρέπεται σε
         * {@link #StatusCommandWaiting}.
         * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
         * @param is Το InputStream του socket.
         * @throws IOException Σε περίπτωση που δημιουργηθεί πρόβλημα με το
Socket.
         */
        void DoHello (Socket Soc, InputStream is) throws IOException
        {
            OutputStream os;
            String Hello;
            Hello = GetLine (is);
            if (Hello.equals ("HELLO"))
            {
                os = Soc.getOutputStream ();
                SendLine (os, "HELLO");
                //if (Status == StatusHelloWaiting)
                //    Status = StatusCommandWaiting;
                os.close ();
            }
            else
                MakeToast ("Invalid Hello : " + Hello, 3);
        }

        /**
         * Υλοποιεί την εντολή "Get File". Δέχεται να αρχείο ως σύνολο από
bytes στον server και το αποθηκεύει τοπικά.
         * Ο κατάλογος αποθήκευσης και το όνομα του αρχείου διαβάζονται ως
strings από το socket.
         * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
         * @param is Το InputStream του socket.
         * @throws IOException Σε περίπτωση που δημιουργηθεί πρόβλημα με το
Socket.
         */
        void DoGetFile (Socket Soc, InputStream is) throws IOException
        {
            int c;
            OutputStream os;
            os = Soc.getOutputStream ();
            String Dirname = GetLine (is);
            String Filename = GetLine (is);
            FileOutputStream OutF = new FileOutputStream
("/sdcard/external_sd/YME/" + Dirname + "/" + Filename);
            byte[] buffer = new byte[4096];
            while ((c = is.read (buffer)) != -1)
            {
                OutF.write (buffer, 0, c);
            }
            OutF.close ();
```

```
			SendLine (os, "O.K.");
			os.close ();
		}

		/**
		 * Υλοποιεί την εντολή "Send File". Διαβάζει ενά τοπικό αρχείο και το αποστέλει στον Server ως σύνολο από bytes.
		 * Ο τοπικός κατάλογος και το όνομα του αρχείου διαβάζονται ως strings από το socket.
		 * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
		 * @param is Το InputStream του socket.
		 * @throws IOException Σε περίπτωση που δημιουργηθεί πρόβλημα με το Socket.
		 */
		void DoSendFile (Socket Soc, InputStream is) throws IOException
		{
			int c;
			OutputStream os;
			os = Soc.getOutputStream ();
			String Dirname = GetLine (is);
			String Filename = GetLine (is);
			FileInputStream InF = new FileInputStream ("/sdcard/external_sd/YME/" + Dirname + "/" + Filename);
			byte[] buffer = new byte[4096];
			while ((c = InF.read (buffer)) != -1)
			{
				os.write (buffer, 0, c);
			}
			InF.close ();
			os.close ();
		}

		/**
		 * Υλοποιεί την εντολή "Start Client". Ενεργοποιεί τον client ως εφαρμογή και αλλάζει το Status του δαίμονα.
		 * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
		 * @param is Το InputStream του socket.
		 * @throws IOException Σε περίπτωση που υπάρχει πρόβλημα με το socket.
		 */
		void DoStartClient (Socket Soc, InputStream is) throws IOException
		{
			//MakeToast ("///I am Starting Client!!!", 1);
			OutputStream os;
			Intent intent = new Intent ();
			intent.setClassName ("org.gmele.dissertation.client", "org.gmele.dissertation.client.IntroScreen");
			intent.addFlags (Intent.FLAG_ACTIVITY_NEW_TASK);
			intent.addFlags (Intent.FLAG_FROM_BACKGROUND);
			//intent.addFlags (Intent.FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET);
			//intent.addFlags (Intent.FLAG_ACTIVITY_NO_HISTORY);
			//intent.addFlags (Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
			startActivity (intent);
			SetClientCom (ClCommStarted);
			os = Soc.getOutputStream ();
			SendLine (os, "O.K.");
			os.close ();
			//Status = StatusFirstScreen;
		}

		/**
		 * Υλοποιεί την εντολή "Send Client Comm", δηλαδή στέλνει στον Server την τιμή της μεταβλητής {@link #ClientComm}.
```

```
	 *  Η τιμή διαβάζεται μέσω της {@link #GetClientCom()} για λόγους συγχρονισμού μεταξύ threads.
	 * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
	 * @param is Το InputStream του socket.
	 * @throws IOException Σε περίπτωση που υπάρχει πρόβλημα με το socket.
	 */
	void DoSendClientComm (Socket Soc, InputStream is) throws IOException
	{
		OutputStream os;
		String M;
		os = Soc.getOutputStream ();
		M = Integer.toString (GetClientCom ());
		SendLine (os, M);
		os.close ();
	}

	/**
	 * Υλοποιεί την εντολή "Start Client". Αλλάζει το Status του δαίμονα και γράφει την τιμή
	 * {@link DaemonConsts#ClCommRunningTest} στη μεταβλητή ClientComm. Η τιμή εκχωρείται μέσω της {@link #SetClientCom}
	 * για λόγους συγχρονισμού μεταξύ threads.
	 * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
	 * @param is Το InputStream του socket.
	 * @throws IOException Σε περίπτωση που υπάρχει πρόβλημα με το socket.
	 */
	void DoStartExam (Socket Soc, InputStream is) throws IOException
	{
		OutputStream os;
		SetClientCom (ClCommRunningTest);
		os = Soc.getOutputStream ();
		SendLine (os, "O.K.");
		os.close ();
		//Status = StatusRunningTest;
	}

	/**
	 * Υλοποιεί την εντολή "Examination Completed". Εκχωρεί την τιμή {@link DaemonConsts#ClCommProcessFinished} στην
	 * μεταβλητή ClientComm. Η τιμή εκχωρείται μέσω της {@link #SetClientCom} για λόγους συγχρονισμού μεταξύ threads.
	 * @param Soc Το socket με το οποίο πραγματοποιείται η λειτουργία.
	 * @param is Το InputStream του socket.
	 * @throws IOException Σε περίπτωση που υπάρχει πρόβλημα με το socket.
	 */
	void DoExamCompleted (Socket Soc, InputStream is) throws IOException
	{
		OutputStream os;
		SetClientCom (ClCommProcessFinished);
		os = Soc.getOutputStream ();
		SendLine (os, "O.K.");
		os.close ();
	}

	/**
	 * Εκχωρεί τιμή στην μεταβλητή ClientComm η οποία σχετίζεται με την επικοινωνία με τον client. Καλείται τόσο από
	 * το thread επικοινωνίας με τον server, όσο και από τον client μέσω service binding. Η τιμή που θα εκχωρηθεί πρέπει
	 * να είναι μία από αυτές που έχουν οριστεί ως σταθερές στο interface {@link DaemonConsts}.
	 * @param NewStatus Η νέα τιμή κατάστασης.
```

```
	 */
	public synchronized void SetClientCom (int NewStatus)
	{
		ClientComm = NewStatus;
	}

	/**
	 * Επιστρέφει την τιμή της μεταβλητής ClientComm η οποία σχετίζεται με την επικοινωνία με τον client. Καλείται τόσο
	 * από το thread επικοινωνίας με τον server, όσο και από τον client μέσω service binding. Η τιμή που επιστέφεται
	 * είναι μία από αυτές που έχουν οριστεί ως σταθερές στο interface {@link DaemonConsts}.
	 * @return Η τρέχουσα κατάσταση.
	 */
	public synchronized int GetClientCom ()
	{
		return ClientComm;
	}
}

/**
 * Η κλάση MyHandler αποτελεί χειριστή που χρησιμοποιείται ώστε threads του Service {@link Daemon} να μπορούν να
 * δημιουργούν toasts μέσω της μεθόδου {@link #MakeToast(String, int)}.
 * @author Giorgos Meletiou
 * @version 1.0
 */
class MyHandler extends Handler
{
	Context AppContext;

	/**
	 * Ο κατασκευαστής του χειριστή.
	 * @param cont Το Context του Service στο οποίο βρίσκεται. Το χρειάζεται η {@link Toast#makeText
	 * (Context, CharSequence, int)}
	 */
	MyHandler (Context cont)
	{
		super ();
		AppContext = cont;
	}

	@Override
	public void handleMessage (Message message)
	{
		Toast toast;
		String Que = message.getData ().getString ("Message");
		int pos = message.getData ().getInt ("Position");
		toast = Toast.makeText (AppContext, Que, Toast.LENGTH_LONG);
		toast.setGravity (pos, 0, 0);
		toast.show ();
	}
};
```

CLASS: DaemonActivity

```
package org.gmele.dissertation.daemon;

import android.app.Activity;
```

```
import android.content.Intent;
import android.os.Bundle;

/**
 * Το activity DaemonActivity είναι υπεύθυνο για την έναρξη του service
Daemon το οποίο ελέγχει την λειτουργία του
 * Client στο σύστημα εξετάσεων. Στην τελική του μορφή θα ξεκινάει με την
έναρξη της συσκευής. Θα καταργηθεί μόλις
 * βρεθεί τρόπος να ξεκινάει service αυτόματα από το σύστημα χωρίς να είναι
μέρος κάποιας εφαρμογής.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public class DaemonActivity extends Activity
{
	/** Called when the activity is first created. */
	@Override
	public void onCreate (Bundle savedInstanceState)
	{
		super.onCreate (savedInstanceState);
		//setContentView (R.layout.main);
		//Toast msg = Toast.makeText (this, "Starting Service",
Toast.LENGTH_LONG);
		//msg.setGravity (Gravity.CENTER, 0, 0);
		//msg.show();

		Intent myintent = new Intent (this, Daemon.class);
		startService (myintent);


		//msg = Toast.makeText (this, "Ending program",
Toast.LENGTH_LONG);
		//msg.setGravity (Gravity.CENTER, 0, 0);
		//msg.show();

		this.finish ();
	}
}
```

INTERFACE: DaemonConsts

```
package org.gmele.dissertation.daemon;

/**
 * Στο iterface DaemonConsts ορίζονται οι σταθερές για το Service {@link
Daemon}.
 * @author Giorgos Meletiou
 * @version 1.0
 */
public interface DaemonConsts
{
	//Κωδικοί εντολών από Server.
	/**
	 * Αποστέλλεται κάθε φορά που (επαν)εγκαθίσταται επικοινωνία,
προκειμένου να ελεγχθεί η επικοινωνία.
	 */
	final int CommHello = 0;
	/**
	 * Ο Client πρέπει να λάβει ένα αρχείο.
	 */
	final int CommGetFile = 1;
	/**
```

```
        * Ο Client πρέπει να στείλει ένα αρχείο.
        */
       final int CommSendFile = 2;
       /**
        * Ο Client πρέπει να διαγράψει αρχείο.
        */
       final int CommDeleteFile = 3;
       /**
        * Ενεργοποίηση του προγράμματος Client.
        */
       final int CommStartClient = 10;
       /**
        * Αποστολή τιμής της μεταβλητής ClientComm.
        */
       final int CommSendClientComm = 11;
       /**
        * Εντολή έναρξης εξέτασης, δηλαδή πέρασμα από την αρχική οθόνη στο
activity της εξέτασης ή στο κλείσιμο λόγω
        * απουσίας.
        */
       final int CommStartExam = 12;
       /**
        * Ενημερώνει τον σταθμό εξέτασης ότι ο server κήρυξε τη διαδικασία
εξέτασης ολοκληρωμένη και κατά συνέπεια
        * το πρόγραμμα client το οποίο περιμένει πρέπει να τερματιστεί.
        */
       final int CommExamCompleted = 13;

       //Ανταλλαγή πληροφοριών κατάστασης και μετάδοσης εντολών με / στον
Client.
       /**
        * Ενεργοποιήθηκε η αρχική οθόνη του client.
        */
       final int ClCommStarted = 0;
       /**
        * Πατήθηκε το κουμπί "Παρών" για τον εξεταζόμενο.
        */
       final int ClCommPressedStart = 1;
       /**
        * Πατήθηκε το κουμπί "Απουσιάζει" για τον εξεταζόμενο.
        */
       final int ClCommPressedAbsent = 2;
       /**
        * Διενέργεια εξέτασης για τον παρόντα. Για απόντα απλά τερματισμός
του client.
        */
       final int ClCommRunningTest = 3;
       /**
        * Ο εξεταζόμενος τελείωσε τις ερωτήσεις του.
        */
       final int ClCommFinished = 4;
       /**
        * Τελείωσε ο χρόνος του εξεταζόμενου ή τελείωσε ο client λόγω
απουσίας
        */
       final int ClCommTimeup = 5;
       /**
        * Η διαδικασία εξέτασης ολοκληρώθηκε. Δεν χρειάζεται να τρέχει
activity του client.
        */
       final int ClCommProcessFinished = 6;
}
```

PACKAGE: org.gmele.dissertation.daemon

CLASS: Daemon

```
package org.gmele.dissertation.daemon;

/**
 * Το Interface IDaemonServ είναι το AIDl inrterface για πρόσβαση στο
service {@link Daemon} από τον έξω κόσμο, δηλαδή
 * activities που δεν ανήκουν στην ίδια εφαρμογή στην οποία ανήκει το
Daemon. Δίνει πρόσβαση στην μεταβλητή ClientComm
 * του Service μέσω των μεθόδων {@link Daemon#GetClientCom()} και {@link
Daemon#SetClientCom(int)}.
 * @author Giorgos Meletiou
 * @version 1.0
 */
interface IDaemonServ
{
	int GetCommVal ();
	void SetCommVal (int Val);
}
```

RESOURCES

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
</LinearLayout>
```

AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.gmele.dissertation.daemon"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="7"
        android:targetSdkVersion="9" />
```

```
    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" >
    </uses-permission>

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <activity
            android:name=".DaemonActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".Daemon"
            android:enabled="true"
            android:exported="true" >
        </service>
    </application>

</manifest>
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Κεκλίκογλου Ι., Μελετίου Γ., Ζυγούρης Π. (2005), Πληροφοριακό Σύστημα. Θεωρητικής Εξέτασης Υποψηφίων Οδηγών, ΤΕΙ Αθηνών, Αθήνα.

Pressman R, Darrel Ince (2000). Software Engineering. A Practitioner's Approach. European Adaptation, Fifth Edition, McGraw-Hill, Berkshire.

Thelin J. (2007). Foundations of Qt Development, Apress,USA.

Schildt H. (2005). Java the complete reference, Fifth Edition, Osborne, Emeryville.

Βασιλακόπουλος Γ. (2009). Σχεδιασμός Βάσεων Δεδομένων, Πειραιάς