oard 

Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη διαδικτυακής εφαρμογής αντικειμενοστρεφούς αρχιτεκτονικής για επικοινωνία με γεωγραφικό πληροφοριακό σύστημα με τη χρήση τεχνικών χωρικής επεξεργασίας**
Masters Thesis Title	**Development of a web application based on object-oriented architecture and supported by a GIS using spatial processing techniques**
Ονοματεπώνυμο Φοιτητή	**Απόστολος Χριστοδούλου**
Πατρώνυμο	**Νικόλαος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 10002**
Επιβλέπων	**Χρήστος Δουληγέρης, Καθηγητής**
Υπεύθυνος Ερευνητής	**Δρ. Βασίλειος Μενεκλής**

Ημερομηνία Παράδοσης **Μάιος 2013**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Χρήστος Δουληγέρης Καθηγητής	Δέσποινα Πολέμη Επικ. Καθηγήτρια	Παναγιώτης Κοτζανικολάου Λέκτορας

Περίληψη

Καθημερινά γινόμαστε μάρτυρες της ολοένα αυξανόμενης διαθεσιμότητας και χρήσης υπηρεσιών που σχετίζονται με τη διαχείριση και την ανάλυση γεωγραφικών/χωρικών δεδομένων. Λόγω του μεγάλου όγκου των γεωγραφικών δεδομένων και της ανάγκης για υψηλή αποδοτικότητα κατά την επεξεργασία τους, έχουν προκύψει τα γεωγραφικά πληροφοριακά συστήματα (ΓΠΣ) τα οποία με τη χρήση καινοτόμων τεχνολογιών καθιστούν ικανή τη συλλογή, διαχείριση, επεξεργασία, ανάλυση, μοντελοποίηση και απεικόνιση γεωγραφικών δεδομένων.

Στο πλαίσιο αυτής της εργασίας αναπτύχθηκε μία διαδικτυακή εφαρμογή η οποία είναι βασισμένη στην αντικειμενοστρεφή αρχιτεκτονική και εξυπηρετείται από ένα γεωγραφικό πληροφοριακό σύστημα. Οι υπηρεσίες που προσφέρει προς τους χρήστες στηρίζονται στις τεχνικές χωρικής επεξεργασίας της γεωκωδικοποίησης και της αντίστροφης γεωκωδικοποίησης. Για τη κάλυψη των γεωγραφικών/χωρικών απαιτήσεων χρησιμοποιήθηκε το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL/PostGIS και η προγραμματιστική διεπαφή των Google Maps. Η υλοποίηση της εφαρμογής πραγματοποιήθηκε στο πλαίσιο εργασίας CodeIgniter με αντικειμενοστρεφή PHP και τη λογική του σχεδιαστικού προτύπου MVC και του σχεδιαστικού προτύπου του εμπρόσθιου ελεγκτή. Από τη μεριά του πελάτη χρησιμοποιήθηκαν τεχνολογίες όπως είναι η JavaScript, η βιβλιοθήκη jQuery, το πρότυπο JSON, η τεχνική AJAX, η HTML και το Twitter Bootstrap.

Abstract

Nowadays we are witnessing the increasing availability and use of services closely related to the management and analysis of geographic/spatial data. Due to the large volume of geographic data and the need for high efficiency when processing them, there have emerged geographic information systems (GIS) that are using innovative technologies capable of realizing the collection, management, processing, analysis, modeling and visualization of geographic data.

In this masters thesis, a web application is developed that is based on object-oriented architecture and is supported by a geographic information system. It offers several services to the users which are based on the spatial processing techniques of geocoding and reverse geocoding. In order to address the geographic/spatial requirements, the database management system PostgreSQL/PostGIS and the application programming interface (API) of Google Maps were used. The application was implemented in the CodeIgniter framework with the use of object-oriented PHP and the logic of both MVC and Front Controller design patterns. As for the client-side, technologies such as, JavaScript, jQuery library, JSON format standard, AJAX technique, HTML and Twitter Bootstrap were used.

Ευχαριστίες

Θα ήθελα σε αυτό το σημείο να ευχαριστήσω όλους όσους βοήθησαν για την εκπόνηση αυτής της μεταπτυχιακής διατριβής και κυρίως τον κ. Δουληγέρη Χρήστο, Καθηγητή του Πανεπιστημίου Πειραιώς και τον κ. Μενεκλή Βασίλειο, Διδάκτορα του Πανεπιστημίου Πειραιώς για τη καθοδήγηση και υποστήριξη που μου παρείχαν σε όλα τα στάδια της εργασίας και για το γεγονός ότι μου έδωσαν τη δυνατότητα να ασχοληθώ με ένα αξιόλογο και επίκαιρο θέμα.

Πίνακας περιεχομένων

Πίνακας εικόνων

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή στα γεωγραφικά πληροφοριακά συστήματα

Στην εποχή μας η διαχείριση και η ανάλυση γεωγραφικών δεδομένων γίνεται ολοένα και περισσότερο μέρος της δραστηριότητας μιας μεγάλης γκάμας επιχειρήσεων, επαγγελματιών και ιδρυμάτων. Συνεπώς, είναι ένα ερευνητικό πεδίο μείζονος σημασίας αν αναλογιστούμε επίσης ότι το 80% των πολιτικών και οικονομικών αποφάσεων εμπλέκουν άμεσα ή έμμεσα γεωγραφικές πληροφορίες. Ο ολοένα αυξανόμενος όγκος αυτών των γεωγραφικών πληροφοριών οδήγησε στην ανάπτυξη συστημάτων που να είναι σε θέση να τις αποθηκεύσουν, διαχειριστούν και ανακτήσουν με αποτελεσματικό τρόπο. Έτσι έχουν προκύψει τα Γεωγραφικά Πληροφοριακά Συστήματα – ΓΠΣ (Geographic Information System - GIS). Ένα γεωγραφικό πληροφοριακό σύστημα είναι ένα υπολογιστικό σύστημα σχεδιασμένο για να υποστηρίζει τη συλλογή, διαχείριση, επεξεργασία, ανάλυση, μοντελοποίηση και απεικόνιση δεδομένων που αναφέρονται στο χώρο.

Τα γεωγραφικά πληροφοριακά συστήματα βρίσκουν εφαρμογή σε ποικίλα πεδία όπως τα παρακάτω:

- Στο **κτηματολόγιο** ώστε να είναι εφικτή η γεωμετρική περιγραφή και διαχείριση της κτηματικής περιουσίας και η οριοθέτησή της επάνω στο χάρτη, στο **περιουσιολόγιο** ώστε να είναι γνωστές οι τοποθεσίες των διαφόρων περιουσιακών στοιχείων και στο **δασολόγιο** ώστε να πραγματοποιείται ακριβέστερη ανάλυση, καταγραφή και επεξεργασία χωρικών πληροφοριών σε δασικούς χάρτες.

- Σε **δίκτυα κοινής ωφέλειας** ώστε να μπορεί να συντελεστεί μια ολοκληρωμένη ανάλυση και μοντελοποίηση σε δίκτυα, όπως για παράδειγμα του ΟΤΕ, της ΔΕΗ κτλ.

- Στη **διαχείριση φυσικών πόρων** ώστε να υπάρχει μια εποπτεία ως προς το εύρος της διαθεσιμότητας διάφορων φυσικών πόρων.

- Στη **θεματική χαρτογραφία** - **υδρογραφία** ώστε να πραγματοποιούνται διάφορες γεωγραφικές μετρήσεις. Επιπλέον να προβάλλονται χάρτες με δυνατότητες όπως η εύρεση σημείων ενδιαφέροντος σε έναν δήμο ή η εύρεση της συντομότερης διαδρομής μεταξύ δύο διευθύνσεων.

- Στη **φωτογραμμετρία – τηλεπισκόπηση** συμβάλλοντας σημαντικά στην παροχή εναλλακτικών τρόπων απεικόνισης των γεωγραφικών χαρτών (για παράδειγμα από δορυφόρο).

- Στην **ψηφιακή αναπαράσταση του τοπογραφικού ανάγλυφου** ώστε να εκφράζονται σε μορφή ψηφιακών μοντέλων ορισμένα γεωμετρικά χαρακτηριστικά.

- Στο **σχεδιασμό και ανάπτυξη των μεταφορών** ώστε να μοντελοποιούνται κατάλληλα οι δρόμοι και να παρέχονται ακριβείς χωρικές πληροφορίες σχετικά με τις τροχιές των πλοίων, τρένων, αεροπλάνων κτλ.

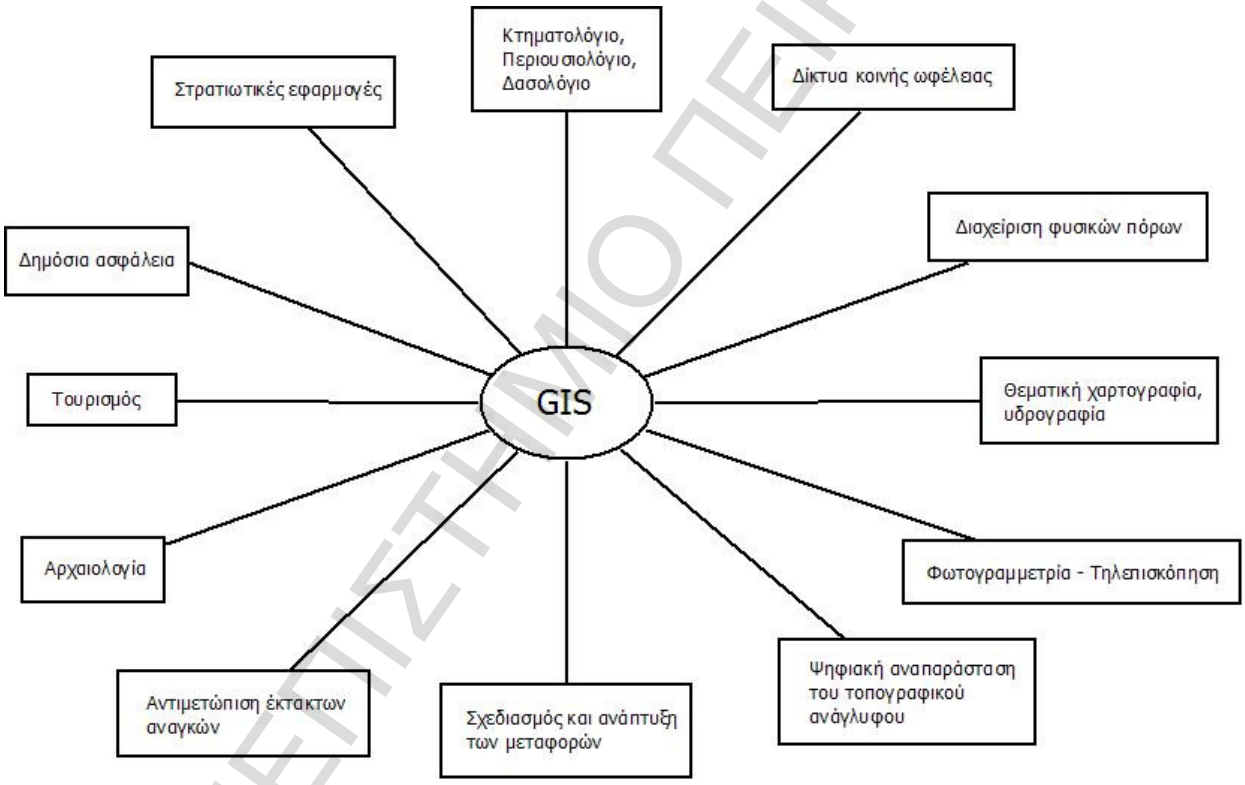
- Στην **αντιμετώπιση έκτακτων αναγκών** όπως στον περιορισμό της πυρκαγιάς σε μια συγκεκριμένη περιοχή με τον ταχύτατο προσδιορισμό της γεωγραφικής τοποθεσίας και της βέλτιστης διαδρομής που πρέπει να ακολουθηθεί για την πρόσβαση.

- Στην **αρχαιολογία** ώστε να είναι διαθέσιμες γεωγραφικές πληροφορίες για την εύρεση αρχαιολογικών χώρων, μνημείων και μουσείων μιας χώρας.

- Στο **τουρισμό** ώστε να είναι εφικτή η εύρεση ξενοδοχείων και γενικά σημείων ενδιαφέροντος σύμφωνα με γεωγραφικά κριτήρια.

- Στη **δημόσια ασφάλεια** για τον άμεσο εντοπισμό και αντιμετώπιση περιστατικών που σχετίζονται με τη δημόσια ασφάλεια αλλά και την καταγραφή επικίνδυνων συμβάντων.

- Σε **στρατιωτικές εφαρμογές** λόγω του ότι τα γεωγραφικά συστήματα πληροφοριών συνεργάζονται άριστα με το σύστημα GPS και έτσι μπορεί να πραγματοποιηθεί με επιτυχία μια επιχειρησιακή πλοήγηση σύμφωνα με διάφορα γεωγραφικά κριτήρια.



Εικόνα 1 - Εφαρμογές των γεωγραφικών πληροφοριακών συστημάτων

1.2 Συνοπτική περιγραφή του θέματος της εργασίας

Αντικείμενο της εργασίας είναι η ανάπτυξη μιας εφαρμογής ιστού η οποία είναι βασισμένη στην αντικειμενοστρεφή αρχιτεκτονική (Object Oriented Architecture - OOA) και εξυπηρετείται από ένα γεωγραφικό πληροφοριακό σύστημα (GIS).

Εικόνα 2 - Το ΓΠΣ σε περιβάλλον Ιστού (Web) και οι αλληλεπιδράσεις που συντελούνται

Στην εικόνα 2 φαίνεται ότι ένα γεωγραφικό πληροφοριακό σύστημα που προβάλλεται στο Web χρησιμοποιείται από πολλές ομάδες χρηστών. Εσωτερικά ένα τέτοιο σύστημα επικοινωνεί με εξυπηρετητές ΓΠΣ (για παράδειγμα PostgreSQL/PostGIS) ώστε να μπορεί να ανακτήσει, να διαγράψει, να εισαγάγει ή και να τροποποιήσει χωρικά δεδομένα και σχετικές πληροφορίες.

Η εφαρμογή της παρούσας εργασίας βασίζεται σε δύο κύριους άξονες λειτουργικότητας:

- **Γεωκωδικοποίηση (Geocoding)**: Είναι η διαδικασία εύρεσης σχετιζόμενων γεωγραφικών συντεταγμένων (γεωγραφικό μήκος και πλάτος) από άλλα γεωγραφικά δεδομένα, όπως διευθύνσεις, ταχυδρομικοί κώδικες, κ.α.

- **Αντίστροφη γεωκωδικοποίηση (Reverse geocoding)**: Είναι η αντίστροφη διαδικασία, δηλαδή, η εύρεση πληροφοριών γεωγραφικής τοποθεσίας, όπως για παράδειγμα η διεύθυνση κάποιας επιχείρησης, από γεωγραφικές συντεταγμένες.

Το θέμα σχετίζεται με τον φανταστικό κόσμο της Μέσης Γης (Middle Earth) και επικεντρώνεται στα βασίλειά της, τα κάστρα που κάθε βασίλειο περιλαμβάνει και τα μονοπάτια που διασχίζουν τα βασίλεια. Τα βασίλεια ουσιαστικά μοντελοποιούνται ως πολύγωνα δηλαδή οριοθετημένες χωρικές εκτάσεις, τα κάστρα ως σημεία ενδιαφέροντος και τα μονοπάτια ως τεθλασμένες γραμμές.

1.3 Συνοπτική περιγραφή της εφαρμογής

Η εφαρμογή προσφέρει τρείς βασικές υπηρεσίες προς τους χρήστες.

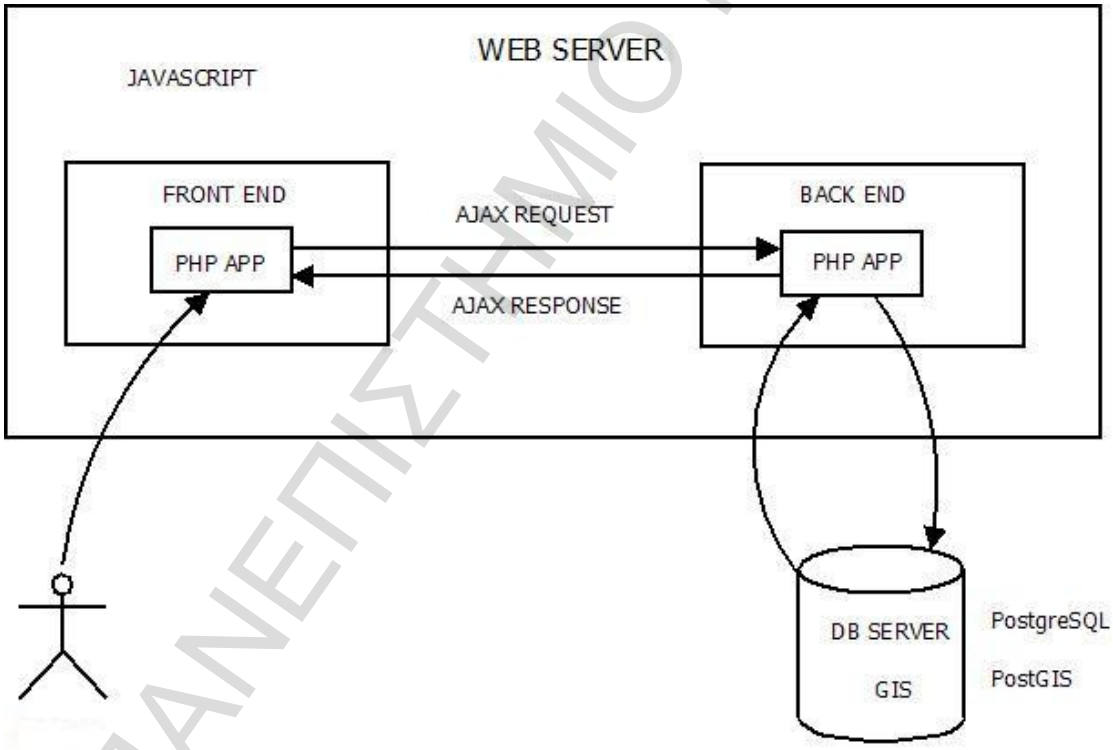
- **Εύρεση γεωγραφικής τοποθεσίας (Geocoding)**: Ο χρήστης επιλέγει μέσα από κατάλληλο μενού επιλογών κάποιο από τα διαθέσιμα βασίλεια και στη συνέχεια αφενός κεντράρεται ο χάρτης της Μέσης Γης στο κέντρο του επιλεγμένου βασιλείου και αφετέρου γεμίζει ένα δεύτερο μενού επιλογών με όλα τα κάστρα που το επιλεγμένο βασίλειο περιλαμβάνει. Με την επιλογή κάποιου κάστρου εμφανίζεται στο χάρτη το εικονίδιο του στην τοποθεσία στο χάρτη που βρίσκεται το κάστρο.

- **Εύρεση πληροφοριών τοποθεσίας (Reverse geocoding)**: Ο χρήστης επιλέγει κάποιο σημείο πάνω στον χάρτη της μέσης γης και παίρνει πληροφορίες σχετικά με το βασίλειο όπου το σημείο αυτό ανήκει και το γεωγραφικό μήκος και πλάτος του κέντρου του βασιλείου. Επιπρόσθετα, ο χάρτης κεντράρεται στο κέντρο του βασιλείου.

- **Εύρεση σημείων ενδιαφέροντος σε συγκεκριμένη ακτίνα (Points of interest (POI) detection in a specific range)**: Αυτή η λειτουργικότητα αποτελεί συνδυασμό των δύο προηγούμενων. Ο χρήστης επιλέγει κάποιο σημείο πάνω στο χάρτη της μέσης γης αφού αρχικά έχει επιλέξει την επιθυμητή ακτίνα μέσα από κατάλληλο μενού επιλογών και παίρνει αφενός πληροφορίες για το γεωγραφικό μήκος και πλάτος του επιλεγμένου σημείου καθώς και για το μήκος της επιλεγμένης ακτίνας και αφετέρου εμφανίζονται στον χάρτη τα κάστρα (σημεία ενδιαφέροντος) που βρίσκονται γύρω από το σημείο του χάρτη που επιλέχθηκε και εμπίπτουν σε αυτή την ακτίνα.

Για την ανάπτυξη της εφαρμογής αρχικά υλοποιήθηκε η κατάλληλη χωρική βάση δεδομένων στο σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) της PostgreSQL (με την επέκταση του PostGIS για υποστήριξη χωρικών λειτουργιών). Τα βασίλεια μοντελοποιήθηκαν ως πολύγωνα (polygon), τα κάστρα μοντελοποιήθηκαν ως σημεία (point) και τα μονοπάτια μοντελοποιήθηκαν ως γραμμές (linestring). Δόθηκε μεγάλη βαρύτητα στην ελαχιστοποίηση του χρόνου αναζήτησης στους χωρικούς πίνακες της βάσης με τη δημιουργία κατάλληλων χωρικών ευρετηρίων.

Στη συνέχεια, υλοποιήθηκε η διαδικτυακή εφαρμογή με χρήση του πλαισίου εργασίας CodeIgniter και του εξυπηρετητή ιστού Apache. Έγινε χρήση της προγραμματιστικής διεπαφής των Google Maps (Google Maps Javascript API v3) για την απεικόνιση/οπτικοποίηση των δεδομένων της βάσης σε χάρτη. Για την ανάπτυξη του στατικού (markup) μέρους της εφαρμογής χρησιμοποιήθηκε το Twitter Bootstrap.



Εικόνα 3 - Τεχνική προσέγγιση αρχιτεκτονικής (απλοποιημένης)

Ο χρήστης (πελάτης) επικοινωνεί με το εμπρόσθιο τμήμα (front-end) της εφαρμογής το οποίο στέλνει αιτήσεις AJAX (ανάλογα με την απαίτηση του χρήστη) στο τελικό τμήμα (back-end) της εφαρμογής, το οποίο με τη σειρά του δέχεται και εξυπηρετεί αυτές τις αιτήσεις

ανακτώντας δεδομένα από τη βάση και στέλνοντάς τα κατάλληλα μορφοποιημένα πίσω στο εμπρόσθιο τμήμα από όπου ο χρήστης μπορεί να τα δει.

Χρησιμοποιήθηκαν τεχνολογίες web τόσο από τη μεριά του εξυπηρετητή (server-side) όσο και από τη μεριά του πελάτη (client-side) όπως PHP, AJAX, JSON, JavaScript, JQuery, HTML και CSS.

1.4 Δομή των κεφαλαίων

Στο δεύτερο κεφάλαιο γίνεται μια παρουσίαση των επιστημονικών εργασιών που σχετίζονται με την παρούσα εργασία και που έχουν δημοσιευτεί κατά καιρούς σε επιστημονικά περιοδικά, βιβλία και συνέδρια. Η παρούσα εργασία συγκρίνεται με αυτές ως προς τις διαφορές και τις ομοιότητες που υπάρχουν μεταξύ τους καθώς επίσης τις βελτιώσεις που έχουν συντελεστεί και τις τυχούσες παραλείψεις. Στο τέλος του κεφαλαίου πραγματοποιείται μια ανασκόπηση των συγκριτικών αποτελεσμάτων και συνοψίζονται τα «θεωρητικά» χαρακτηριστικά τα οποία έχει η εφαρμογή.

Στο τρίτο κεφάλαιο παρουσιάζονται πλήρως όλες οι τεχνολογίες, οι αρχιτεκτονικές και τα σχεδιαστικά πρότυπα (design patterns) που χρησιμοποιήθηκαν τόσο για το σχεδιασμό όσο και για την υλοποίηση της εφαρμογής με απλό και κατανοητό τρόπο.

Στο τέταρτο κεφάλαιο παρουσιάζεται η εφαρμογή που αναπτύχθηκε. Πιο συγκεκριμένα, περιγράφεται και αναλύεται η αρχιτεκτονική και η λειτουργία της πλατφόρμας – εφαρμογής με διαγράμματα UML, σχήματα και κείμενο. Επιδεικνύεται ο τρόπος με τον οποίο συνδέονται όλα όσα έχουν ειπωθεί στα προηγούμενα κεφάλαια με σκοπό τη δημιουργία της εφαρμογής.

Στο πέμπτο κεφάλαιο γίνεται παρουσίαση των συμπερασμάτων αναφορικά με τις τεχνολογίες και τη διαδικασία που ακολουθήθηκε για τη διεκπεραίωση της εργασίας (συγγραφή, σχεδιασμός της εφαρμογής, υλοποίηση, κτλ.), των περιορισμών που τέθηκαν και εν συνεχεία των μελλοντικών επεκτάσεων και προκλήσεων.

Στο παράρτημα παρουσιάζονται συμπληρωματικές πληροφορίες, όπως ο κώδικας της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2: ΣΧΕΤΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

2.1 Περιγραφή / Σύγκριση σχετικών επιστημονικών εργασιών

Σε αυτό το κεφάλαιο πραγματοποιείται μια παρουσίαση των επιστημονικών εργασιών που σχετίζονται με την παρούσα εργασία και που έχουν δημοσιευτεί κατά καιρούς σε επιστημονικά περιοδικά, βιβλία και συνέδρια. Η παρούσα εργασία συγκρίνεται με αυτές ως προς τις διαφορές και τις ομοιότητες που υπάρχουν μεταξύ τους καθώς επίσης τις βελτιώσεις που έχουν συντελεστεί και τις τυχούσες παραλείψεις. Στο τέλος του κεφαλαίου πραγματοποιείται μια ανασκόπηση των συγκριτικών αποτελεσμάτων και συνοψίζονται τα «θεωρητικά» χαρακτηριστικά τα οποία έχει η εφαρμογή.

Στην πρώτη επιστημονική εργασία, οι Boondao, Esichaikul και Tripathi (2003) έχουν αναπτύξει μια web εφαρμογή που προσφέρει υπηρεσίες γύρω από μία συγκεκριμένη τοποθεσία (location based services) στοχεύοντας στην προστασία των πολιτών από επικίνδυνες καταστάσεις όπως για παράδειγμα εγκληματικές ενέργειες. Λαμβάνοντας υπόψη τις απαιτήσεις τόσο των πολιτών όσο και των αστυνομικών επιχειρείται να δημιουργηθεί ένα μοντέλο προς αυτή την κατεύθυνση. Η ανάπτυξη του συστήματος ενοποιεί το Minnesota MapServer, ένα εξυπηρετητή web και μια βάση δεδομένων σε μια αρχιτεκτονική web που ακολουθεί το πρότυπο πελάτη-εξυπηρετητή. Η βάση δεδομένων περιλαμβάνει το σύνολο των δεδομένων που έχουν παρασχεθεί στα πλαίσια της εργασίας όπως τα αστυνομικά τμήματα, τους δρόμους και τις κεντρικές αρτηρίες.

Η εφαρμογή των Boondao, Esichaikul και Tripathi προσφέρει στους χρήστες λειτουργίες σχετικά με την αναφορά της τοποθεσίας όπου έχει συμβεί κάποιο έγκλημα και τη διαδρομή τόσο για την τοποθεσία του εγκλήματος όσο για το κοντινότερο αστυνομικό τμήμα. Οι χρήστες έχουν τη δυνατότητα να συνδέονται στο σύστημα από PDA, κινητά τηλέφωνα, υπολογιστές, κτλ. μέσω μιας web διεπαφής. Το σύστημα αρχικά ελέγχει και ταυτοποιεί το χρήστη επιτρέποντάς του ή όχι την πρόσβαση στις υπηρεσίες του. Τα δεδομένα της τοποθεσίας του χρήστη λαμβάνονται και κατόπιν επεξεργασίας μετασχηματίζονται κατάλληλα σε συντεταγμένες (γεωγραφικό μήκος και πλάτος) και στέλνονται στη βάση δεδομένων η οποία εκτός από χωρικά δεδομένα περιλαμβάνει δεδομένα εγκλημάτων. Τα δεδομένα αυτά ενοποιούνται και στέλνονται στη συνέχεια στον εξυπηρετητή (HTTP server/Map server/WAP server) ο οποίος επιτελεί τους κατάλληλους υπολογισμούς και στέλνει τα αποτελέσματα στο φυλλομετρητή (Web/WAP) όπου εμφανίζονται.

Ως σύστημα διαχείρισης της βάσης δεδομένων έχει επιλεχθεί η PostgreSQL/PostGIS. Το εμπρόσθιο τμήμα (front-end) και το τελικό τμήμα (back-end) της εφαρμογής έχουν βασιστεί σε PHP. Ως εξυπηρετητής web έχει χρησιμοποιηθεί ο Apache.

Στην παρούσα εργασία έχει χρησιμοποιηθεί η προγραμματιστική διεπαφή των Google Maps σε αντίθεση με τον Minnesota MapServer. Επιπλέον, η πρόσβαση στις υπηρεσίες της εφαρμογής της παρούσας εργασίας είναι ανοιχτή για όλους τους χρήστες σε αντίθεση με την εφαρμογή της εργασίας που συγκρίνεται, στην οποία έχουν εισαχθεί πολιτικές ασφαλείας με κατάλληλη αυθεντικοποίηση του κάθε χρήστη.

Ομοιότητα παρατηρείται στο ότι ως web εξυπηρετητής έχει επιλεχθεί ο Apache. Επίσης, έχει χρησιμοποιηθεί το σύστημα της PostgreSQL/PostGIS για τη βάση δεδομένων. Επιπλέον, τόσο το εμπρόσθιο τμήμα όσο και το τελικό τμήμα της εφαρμογής βασίζονται σε PHP.

Έχει βελτιωθεί η ταχύτητα αναζήτησης χωρικών δεδομένων στη βάση με την προσθήκη ευρετηρίων στις γεωμετρικές στήλες των πινάκων και την κατάλληλη σύνταξη των ερωτημάτων. Επίσης ολόκληρη η αρχιτεκτονική του συστήματος έχει βασιστεί στο πρότυπο MVC και όχι στο τετριμμένο μοντέλο πελάτη-εξυπηρετητή.

Λόγω του ότι η βαρύτητα έχει δοθεί στις λειτουργίες που προσφέρει προς τους χρήστες η εφαρμογή της παρούσας εργασίας, έχουν παραλειφθεί θέματα που σχετίζονται με την ασφάλεια της εφαρμογής.

Σε μία άλλη επιστημονική εργασία (Piarsa, Wiranatha & Putra, 2012) αναπτύχθηκε ένα Web/Mobile γεωγραφικό πληροφοριακό σύστημα για την αγορά και πώληση γης. Στο σύστημα αυτό γίνεται διαχωρισμός των χρηστών σε τρείς κατηγορίες: επισκέπτες, εγγεγραμμένους και διαχειριστές. Οι επισκέπτες και συνεπώς όλοι οι χρήστες έχουν πρόσβαση στην αναζήτηση γης μέσω του Google Map με διαθέσιμα πολλαπλά κριτήρια αλλά και στη προβολή λεπτομερειών σχετικά με αυτή. Οι επισκέπτες μπορούν να εγγραφούν στο σύστημα αλλάζοντας με αυτό τον τρόπο τη κατηγορία τους σε εγγεγραμμένους. Οι εγγεγραμμένοι χρήστες, αφού συνδεθούν στο σύστημα, μπορούν να ανοίξουν θέμα στο φόρουμ σχετικά με την ανάθεση της γης που επιθυμούν να πουλήσουν, να επεξεργαστούν σχετικές πληροφορίες και να σχεδιάσουν πολύγωνα με σκοπό την οριοθέτηση της γης που πουλάνε. Ο χρήστης επιλέγει σημεία επάνω στο χάρτη και με τη βοήθεια των συντεταγμένων που παρέχονται από τη προγραμματιστική διεπαφή των Google Maps δημιουργούνται τα αντίστοιχα πολύγωνα στο χάρτη. Οι εγγεγραμμένοι χρήστες μπορούν να σχολιάζουν τόσο τα θέματα που έχουν ανοίξει οι ίδιοι όσο και θέματα άλλων χρηστών. Τέλος έχουν τη δυνατότητα επικοινωνίας μέσω προσωπικών μηνυμάτων.

Το σύστημα έχει υλοποιηθεί ακολουθώντας την αντικειμενοστρεφή προσέγγιση ανάπτυξης. Χρησιμοποιείται Ruby On Rails και βασίζεται στο πρότυπο MVC. Επίσης χρησιμοποιούνται Google Maps. Υιοθετείται η τεχνική της δυναμικής διάταξης (responsive layout) ώστε η διεπαφή του συστήματος να προσαρμόζεται αυτόματα και δυναμικά ανάλογα με τη συσκευή στην οποία επιχειρείται να προβληθεί.

Οι χρήστες, χρησιμοποιώντας είτε σταθερούς υπολογιστές είτε κινητές συσκευές, έχουν πρόσβαση στην όψη (view) της εφαρμογής η οποία στέλνει αιτήσεις δεδομένων στον ελεγκτή (controller) ο οποίος με τη σειρά του επικοινωνεί με το μοντέλο (model) λαμβάνοντας έτσι από τη βάση τα ζητούμενα δεδομένα. Τα δεδομένα επιστρέφονται στην όψη σε μορφή JSON και επεξεργάζονται με JavaScript. Τέλος, οι συντεταγμένες στέλνονται με τεχνική AJAX αφενός στη συνάρτηση γεωκωδικοποίησης η οποία εντοπίζει τη περιοχή που ανήκει η γη επιστρέφοντας πληροφορίες σε μορφή JSON για τη περιοχή αυτή και αφετέρου στον εξυπηρετητή Google Map με αποτέλεσμα την εμφάνιση του χάρτη και των σημείων (markers) που οριοθετούν τη τοποθεσία.

Το σύστημα παρέχει ειδοποιήσεις σχετικά με νέα σχόλια, νέα προσωπικά μηνύματα, νέες προσφορές και νέες πωλήσεις γης. Σημαντικό μειονέκτημα του συστήματος αποτελεί το γεγονός ότι δεν ελέγχεται το τι εισάγει ο χρήστης οπότε δημιουργείται σε κάποιο βαθμό θέμα αξιοπιστίας.

Στην παρούσα εργασία έχει χρησιμοποιηθεί το CodeIgniter σε αντίθεση με το Ruby On Rails. Επιπλέον το σύστημα που παρουσιάζεται στην παρούσα εργασία υποστηρίζεται εξολοκλήρου από ένα αυτόνομο γεωγραφικό πληροφοριακό σύστημα που ασχολείται με την ανάκτηση και την επεξεργασία χωρικών και μη δεδομένων που βρίσκονται αποθηκευμένα στη βάση δεδομένων PostgreSQL/PostGIS σε μορφή γεωμετριών. Η προγραμματιστική διεπαφή των Google Maps χρησιμοποιείται μόνο για την οπτικοποίηση τους σε χάρτη. Ωστόσο, η προκειμένη εργασία στηρίζεται αποκλειστικά και μόνο στην προγραμματιστική διεπαφή των Google Maps και στον εξυπηρετητή Google Maps τόσο για την αρχικοποίηση του χάρτη όσο και για την αναζήτηση περιοχών μέσω των έτοιμων υπηρεσιών που παρέχει. Στη βάση είναι αποθηκευμένα απλά δεδομένα και δεδομένα συντεταγμένων. Επιπλέον στη παρούσα εργασία δεν κατηγοριοποιούνται οι χρήστες λόγω του ότι όλοι οι χρήστες θεωρούνται ομότιμοι.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει ακολουθηθεί το σχεδιαστικό πρότυπο MVC και έχει γίνει χρήση της προγραμματιστικής διεπαφής των Google Maps. Επίσης, οι τεχνολογίες AJAX και JSON έχουν χρησιμοποιηθεί εκτεταμένα και από τις δύο.

Στην παρούσα εργασία έχει βελτιωθεί η αρχιτεκτονική του συστήματος με σκοπό η εφαρμογή να είναι επεκτάσιμη. Γίνεται χρήση ενός εμπρόσθιου ελεγκτή (front controller) και δύο άλλων ελεγκτών που τον επεκτείνουν και εξειδικεύονται ο ένας στην αλληλεπίδραση με τις όψεις και ο άλλος στην αλληλεπίδραση με το μοντέλο. Ενώ στην άλλη εργασία, ένας ελεγκτής είναι επιφορτισμένος με όλα.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση τεχνικών για δυναμική διάταξη (responsive layout) και συνεπώς δεν παρέχεται υποστήριξη για την ορθή απεικόνιση της διάταξης της εφαρμογής από άλλες συσκευές εκτός των υπολογιστών.

Στην επιστημονική εργασία (Oussalah et al., 2012) αναπτύχθηκε ένα ολοκληρωμένο σύστημα λογισμικού που επικεντρώνεται στη σημασιολογική και χωρική (γεωγραφική) ανάλυση δεδομένων του Twitter. Πιο συγκεκριμένα, περιγράφεται μια αρχιτεκτονική για την αυτοματοποιημένη συλλογή των «tweets» από μια προκαθορισμένη τοποθεσία. Η προγραμματιστική διεπαφή του Twitter Streaming χρησιμοποιήθηκε ως βάση για τη συλλογή των «tweets» που αποθηκεύονται σε μία βάση MySQL. Από τη μία συνδέθηκε το σύστημα Apache Lucene με το WordNet και από την άλλη χρησιμοποιήθηκε επεξεργασία φυσικής γλώσσας και η PostGIS πλατφόρμα για να εξασφαλιστεί η σημασιολογική και η χωρική ανάλυση των δεδομένων αντίστοιχα. Επιπρόσθετα, υλοποιήθηκε και αξιολογήθηκε μια λειτουργική προσέγγιση σχετικά με τη μικρή ανοχή σφαλμάτων που διέπει το σύστημα κατά την συλλογή των δεδομένων.

Το σύστημα επιτρέπει στους χρήστες τη συλλογή μεγάλου όγκου από «tweets» που έχουν δημοσιευτεί από διάφορες γεωγραφικές περιοχές, την αναζήτηση σχετικών «tweets» από το σύνολο των δεδομένων χρησιμοποιώντας τόσο χωρικά όσο και θεματικά ερωτήματα, την εξαγωγή τους σε κατάλληλα αρχεία για περαιτέρω έρευνα και την ανάλυση και την προβολή της τοποθεσίας τους μέσω ενός χάρτη.

Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε το web πλαίσιο εργασίας Django το οποίο βασίζεται σε Python και στο πρότυπο MTV (Model – Template - View). Επίσης χρησιμοποιείται η επέκταση GeoDjango για την υποστήριξη του γεωγραφικού μέρους της εφαρμογής και τη διασύνδεσή της με την προγραμματιστική διεπαφή των Google Maps και τη μηχανή αναζήτησης Apache SOLR. Ο συνδυασμός της PosgreSQL/PostGIS επιτρέπει τη διαχείριση χωρικών ερωτημάτων με αποδοτικό τρόπο και χρησιμοποιήθηκε για την αντιμετώπιση των απαιτήσεων του GeoDjango. Το σύστημα Apache Lucene χρησιμοποιήθηκε για την ενσωμάτωση και διαχείριση των «tweets» σε μια επεκτάσιμη και ευρετηριασμένη βάση δεδομένων MySQL. Το WordNet χρησιμοποιήθηκε για τη σημασιολογική ανάλυση λέξεων, προτάσεων και εκφράσεων σε συνδυασμό με μια βάση δεδομένων που χρησιμοποιείται ως μονάδα προ-επεξεργασίας πριν την ενεργοποίηση του μηχανισμού αναζήτησης. Η διαχείριση πληροφοριών για γεωγραφικές τοποθεσίες ενισχύεται με την ενσωμάτωση δύο βιβλιοθηκών.

Μέσω της web διεπαφής επιτρέπεται στους χρήστες να θέτουν συγκεκριμένες τοποθεσίες παρακολούθησης και να επεξεργάζονται συλλογές δεδομένων. Αρχικά συνδέεται η προγραμματιστική διεπαφή Streaming η οποίο συλλέγει «tweets» και τα αποθηκεύει σε μια βάση δεδομένων MySQL μέσω μιας εφαρμογής Java. Χρησιμοποιείται η βιβλιοθήκη Twitter4J για να την ενεργοποίηση της εφαρμογής Java και την επικοινωνία της με διάφορες προγραμματιστικές διεπαφές του Twitter, όπως η Streaming. Τα αποθηκευμένα δεδομένα περιέχουν μεταξύ άλλων συντεταγμένες γεωγραφικού μήκους και πλάτους οι οποίες αρχικά συγκρίνονται και αναλύονται ως προς την απόστασή τους από περιοχές, οι συντεταγμένες των οποίων παρέχονται από δύο βιβλιοθήκες, και στη συνέχεια αποθηκεύονται (αφού πρώτα τροποποιηθούν κατάλληλα από ένα σενάριο Python) στη PostgreSQL/PostGIS. Η χρήση της PostgreSQL/PostGIS αποτελεί προϋπόθεση ώστε να μπορεί να χρησιμοποιηθεί σωστά η επέκταση GeoDjango για ερωτήματα γεωγραφικού περιεχομένου τα οποία εκτελούνται μέσω του GeoQuerySetAPI. Ο χρήστης ορίζει την περιοχή και την ακτίνα δράσης και έτσι του επιστρέφονται «tweets» εντός της κυκλικής περιοχής που δημιουργείται. Επίσης χρησιμοποιείται η μηχανή αναζήτησης Apache SOLR ενοποιημένη με το Haystack το οποίο παρέχει μια υψηλού επιπέδου διεπαφή για αναζήτηση χωρικών δεδομένων μέσω του Apache SOLR. Επίσης παρέχει το SearchQuerySetAPI που επιτρέπει την ανάκτηση ευρετηριασμένων δεδομένων. Το τελευταίο στάδιο της εφαρμογής περιλαμβάνει μία υπηρεσία εμφάνισης χαρτών στο web για την εμφάνιση των αποτελεσμάτων («tweets») επάνω σε Google Map. Για το λόγο αυτό γίνεται χρήση των Google Maps. Επίσης, δίνεται η δυνατότητα στο χρήστη να εξαγάγει τα αποτελέσματα σε αρχείο τύπου CSV.

Στην παρούσα εργασία έχει χρησιμοποιηθεί το CodeIgniter σε αντίθεση με το Django και συνεπώς υπάρχει διαφορά ως προς την υλοποίηση που έχει γίνει με PHP αντί για Python και Java. Επιπλέον το σύστημα που παρουσιάζεται στη παρούσα εργασία χρησιμοποιεί το σχεδιαστικό πρότυπο MVC σε αντίθεση με το MTV του Django.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει ακολουθηθεί αντικειμενοστρεφής προσέγγιση ανάπτυξης. Επίσης έχει χρησιμοποιηθεί η βάση δεδομένων PostgreSQL/PostGIS για την υλοποίηση και την υποστήριξη της χωρικής βάσης δεδομένων και των δύο εφαρμογών. Τα Google Maps και η JavaScript χρησιμοποιούνται εξίσου και στις δύο εφαρμογές για την αρχικοποίηση, τη δημιουργία και τη προβολή χωρικών δεδομένων στο χάρτη. Επίσης, γίνεται χρήση αντικειμένων τύπου GeoJSON.

Στην παρούσα εργασία έχει γίνει βελτίωση στο ότι δίνεται η δυνατότητα στο χρήστη να αλληλεπιδρά κατευθείαν με τη διεπαφή του χάρτη επιτελώντας με αυτό τον τρόπο αντίστροφη γεωκωδικοποίηση (reverse geocoding).

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση τεχνικών επεξεργασίας φυσικής γλώσσας και σημασιολογικής ανάλυσης λόγω του ότι ο χρήστης κάνει αναζήτηση στα δεδομένα που επιθυμεί μέσω κατάλληλων μενού επιλογών που περιέχουν συγκεκριμένες τιμές από τη βάση δεδομένων. Επίσης, για τη μορφοποίηση των δεδομένων χρησιμοποιείται μόνο JSON σε αντίθεση με την υπό σύγκριση εργασία που χρησιμοποιείται και XML. Τέλος, δε δίνεται η δυνατότητα στο χρήστη να εξάγει τα αποτελέσματα σε αρχεία τύπου CSV όπως στην παραπάνω εργασία.

Στην επιστημονική εργασία των Singh, Chutia & Sudhakar (2012) περιγράφονται τα βήματα και οι μεθοδολογίες στις οποίες στηρίχτηκε η ανάπτυξη ενός γεωγραφικού πληροφοριακού συστήματος (GIS) ιστού με αντικειμενικό στόχο την οπτικοποίηση σε χάρτη των διάφορων τοποθεσιών που χρησιμοποιούνται για την παραγωγή μεταξιού στη συνοικία Champhai του Mizoram της Ινδίας. Πιο συγκεκριμένα, αναπτύχθηκε μία διαδικτυακή εφαρμογή GIS που επιτρέπει στο χρήστη την προβολή, ενημέρωση, προσαρμοσμένη ανάκτηση, ερώτηση και ανάλυση πληροφοριών για διαθέσιμους φυσικούς πόρους για τη σηροτροφία (μεταξουργία).

Η υλοποίηση πραγματοποιήθηκε με την ενοποίηση των παρακάτω τεχνολογιών: Minnesota MapServer, PHP, Apache Web Server, Chameleon, PostgreSQL/PostGIS και χρησιμοποιώντας το διαδικτυακό μοντέλο πελάτη – εξυπηρετητή.

Η ανάπτυξη του συστήματος βασίστηκε σε μία αρχιτεκτονική όπου ο εξυπηρετητής επιφορτίζεται το σύνολο της επεξεργασίας των δεδομένων και ο πελάτης ασχολείται μόνο με τη παρουσίαση και τη διαχείριση της αλληλεπίδρασης με το χρήστη. Επίσης, λήφθηκαν υπόψη θέματα που σχετίζονται με τη φορητότητα (μεταφερσιμότητα) και την επεκτασιμότητα του λογισμικού.

Η αρχιτεκτονική της εφαρμογής στηρίχθηκε στον Minnesota MapServer ο οποίος δέχεται και επεξεργάζεται αιτήσεις του χρήστη και του επιστρέφει αποτελέσματα με κατάλληλη ενημέρωση του χάρτη. Αποτελείται από τρία μέρη: το αρχείο του χάρτη (map file), τα αρχεία template και το πρόγραμμα CGI. Στο αρχείο του χάρτη τίθενται οι χαρτογραφικές παράμετροι, τα χαρτογραφικά αντικείμενα και η φόρτωση των δεδομένων και ορίζονται οι τρόποι εμφάνισης, αναζήτησης και χρήσης. Τα αρχεία template είναι σελίδες HTML εφοδιασμένες με μεταβλητές και παραμέτρους του αρχείου χάρτη, οι οποίες χρησιμοποιούνται για την εμφάνιση χαρτών, χαρτογραφικών αντικειμένων κτλ. Το πρόγραμμα CGI είναι ο πυρήνας του Web ΓΠΣ και ασχολείται με την είσοδο και επεξεργασία των παραμέτρων, ρυθμίσεων και μεταβλητών τόσο του αρχείου χάρτη όσο και του αρχείου template επιστρέφοντας επεξεργασμένους χάρτες, αντικείμενα, τιμές μεταβλητών και αποτελέσματα ερωτημάτων. Τα δεδομένα από τα αρχεία shape αποθηκεύτηκαν σε βάση δεδομένων της PostgreSQL/PostGIS η οποία αποτελεί σημαντικό τμήμα του συστήματος από το οποίο το Web ΓΠΣ φορτώνει τα δεδομένα που πρέπει να εμφανιστούν στο χάρτη. Οι πίνακες της βάσης καλούνται από τον MapServer ο οποίος χρησιμοποιεί τύπο σύνδεσης PostGIS και κάποιες παραμέτρους. Για την δημιουργία της διεπαφής (GUI) του χάρτη στο εμπρόσθιο τμήμα της εφαρμογής έχει χρησιμοποιηθεί το πλαίσιο εργασίας Chameleon το οποίο αποτελείται από κώδικα PHP που παρέχει πρόσβαση σε widgets χαρτών και συναρτήσεις για την επεξεργασία τους. Βασίζεται σε CWC2 το οποίο παρέχει

παραμετροποιήσιμα και προσαρμόσιμα widgets και εξειδικευμένες ετικέτες για τις σελίδες HTML με σκοπό τη προσθήκη περιεχομένου στο χάρτη με απλό τρόπο.

Το σύστημα χωρίζεται σε δύο μέρη, τα Chameleon widgets από όπου ο χρήστης αλληλεπιδρά και το κομμάτι της εφαρμογής στον εξυπηρετητή το οποίο καλείται από τα Chameleon widgets και λαμβάνει όλες τις απαραίτητες πληροφορίες επιστρέφοντας κατόπιν επεξεργασίας το τελικό αποτέλεσμα.

Στην παρούσα εργασία τόσο το εμπρόσθιο τμήμα όσο και το τελικό τμήμα της εφαρμογής έχουν υλοποιηθεί σε CodeIgniter ενώ στην υπό σύγκριση εργασία το εμπρόσθιο τμήμα υλοποιείται σε Chameleon και το τελικό τμήμα σε Minnesota MapServer. Στη παρούσα εργασία για το εμπρόσθιο τμήμα έχουν χρησιμοποιηθεί JavaScript, jQuery και το Bootstrap ενώ στην άλλη HTML και εξειδικευμένος κώδικας δημιουργίας και διαχείρισης των widgets. Επιπλέον στη παρούσα εργασία χρησιμοποιείται η προγραμματιστική διεπαφή των Google Maps και τα Google Maps ως υπηρεσία εμφάνισης χαρτών στο Web σε αντίθεση με το Chameleon και το CWC2 της άλλης.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει χρησιμοποιηθεί το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL/PostGIS για την υλοποίηση και την υποστήριξη των χωρικών βάσεων δεδομένων. Επίσης και στις δύο έχει γίνει χρήση της PostGIS συνάρτησης AddGeometryColumn για την εισαγωγή στήλης γεωμετρίας σε κάθε πίνακα. Τέλος, έχει χρησιμοποιηθεί γλώσσα προγραμματισμού PHP με τη πλευρά του εξυπηρετητή να συνίσταται από τον Apache.

Στην παρούσα εργασία έχει γίνει βελτίωση στη μεθοδολογία της ανάπτυξης η οποία έχει στηριχτεί στο πρότυπο MVC και το πρότυπο εμπρόσθιου ελεγκτή σε αντίθεση με το απλό μοντέλο πελάτη – εξυπηρετητή της άλλης εργασίας. Επίσης, χρησιμοποιείται η μορφοποίηση JSON για την εύκολη διαχείριση των δεδομένων που επιστρέφονται.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση των δεδομένων για τη βάση από αρχεία shape και αντ' αυτού τα δεδομένα έχουν εισαχθεί με πολλαπλές εντολές SQL INSERT και με τη βοήθεια του Quantum GIS.

Στην εργασία των Yuan & Cheng (2007) αφενός εξετάζεται η τρέχουσα έρευνα σχετικά με την ενοποίηση ενός γεωγραφικού πληροφοριακού συστήματος με διάφορα υδρολογικά μοντέλα και αφετέρου παρουσιάζεται ένα Web ΓΠΣ που συνδέει τα Google Maps με το μοντέλο IHACRES για την πρόβλεψη της απορροής των υδάτων στο Oak Ridge Moraine του Southern Ontario του Καναδά.

Το IHACRES είναι ένα μοντέλο βροχής-απορροής που προσομοιώνει τα υδρολογικά γεγονότα που λαμβάνουν χώρα κατά την απορροή και υπολογίζει τους χρόνους που απαιτούνται από το κάθε σημείο της λεκάνης για να γίνει πλήρης απορροή αυτής. Το μοντέλο αυτό περιλαμβάνει δύο τμήματα για να προσομοιώσει τη διαδικασία βροχής-απορροής. Το τμήμα της μη γραμμικής απώλειας μετατρέπει τη βροχή σε ωφέλιμη βροχή με την εξάλειψη του ποσού της βροχής που εξατμίστηκε λόγω της θερμοκρασίας. Στη συνέχεια περνάει μέσα από το δεύτερο τμήμα, της γραμμικής δρομολόγησης, στο οποίο παρέχονται δύο επιλογές, γρήγορης ροής και αργής ροής και τελικά εξάγεται στην τελική μορφή της απορροής.

Το σύστημα χωρίζεται σε τρία μέρη. Το πρώτο μέρος είναι ένα πρόγραμμα βασισμένο σε JavaScript και AJAX το οποίο αναλαμβάνει να φορτώσει το Google Map από τον εξυπηρετητή της Google Maps και τα χωρικά δεδομένα από τον Web εξυπηρετητή (που βρίσκονται σε βάσεις δεδομένων). Τα δεδομένα επιστρέφονται στον πελάτη από τον εξυπηρετητή σε μορφή XML, αναλύονται και εμφανίζονται επάνω στο χάρτη, για παράδειγμα οι σταθμοί μέτρησης της ροής των υδάτων. Το δεύτερο μέρος περιλαμβάνει τρία προγράμματα σε JSP τα οποία αλληλεπιδρούν με τις βάσεις δεδομένων ανακτώντας τα ζητούμενα δεδομένα τα οποία στη συνέχεια επιστρέφουν σε μορφή XML ή εικόνας στον εξυπηρετητή Web ο οποίος μέσω της τεχνικής AJAX τα επιστρέφει στο πελάτη. Το πρώτο πρόγραμμα JSP ασχολείται με την ανάκτηση εικόνων, ανάλογα με τη τοποθεσία, από τη βάση δεδομένων εικόνων. Το δεύτερο πρόγραμμα JSP ανακτά τα χωρικά δεδομένα (για παράδειγμα τοποθεσίες σταθμών μέτρησης της ροής των υδάτων, τα όρια της λεκάνης, κτλ.) από τη χωρική βάση δεδομένων. Το τρίτο πρόγραμμα JSP ανακτά συμπληρωματικά δεδομένα μετρήσεων όπως για παράδειγμα

θερμοκρασία, αναμενόμενη ροή υδάτων, μετρήσιμη ροή υδάτων κτλ. Το τρίτο και τελευταίο μέρος, περιλαμβάνει το μοντέλο βροχής-απορροής IHACRES το οποίο χρησιμοποιεί ως είσοδο δεδομένα βροχόπτωσης και θερμοκρασίας και μετρήσιμες ροές υδάτων για τη βαθμονόμηση των παραμέτρων του μοντέλου. Στη συνέχεια το μοντέλο χρησιμοποιείται για την πρόβλεψη της απορροής λαμβάνοντας υπόψη τα δεδομένα της βροχόπτωσης και της θερμοκρασίας.

Το σύστημα παρέχει τις ακόλουθες λειτουργικότητες προς τους χρήστες. Πρώτον, οι χρήστες μπορούν να δούν τις τοποθεσίες των σταθμών μέτρησης της ροής των υδάτων σε κάποιο χρονικό σημείο. Δεύτερον, οι χρήστες μπορούν να επιλέξουν οποιοδήποτε σταθμό λαμβάνοντας πληροφορίες για αυτόν, εικόνες του, τα όρια της λεκάνης που καλύπτει και ένα γράφημα για τη πρόβλεψη της απορροής.

Στην παρούσα εργασία έχει χρησιμοποιηθεί η βιβλιοθήκη jQuery για την υλοποίηση της τεχνικής AJAX ενώ στην υπό σύγκριση εργασία χρησιμοποιείται απλή JavaScript. Στη παρούσα εργασία έχει χρησιμοποιηθεί JSON για τη μορφοποίηση των δεδομένων που επιστρέφει ο εξυπηρετητής μετά από μια αίτηση AJAX ενώ στην άλλη εργασία χρησιμοποιείται XML. Επιπρόσθετα, στη παρούσα εργασία χρησιμοποιείται PHP για την ανάκτηση των δεδομένων από τη βάση (μέσω του μοντέλου) σε αντίθεση με τη JSP που χρησιμοποιεί η άλλη εργασία.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχουν χρησιμοποιηθεί τα Google Maps και η τεχνική AJAX.

Στην παρούσα εργασία έχει γίνει βελτίωση ως προς τις παρεχόμενες υπηρεσίες προς τους χρήστες με την προσθήκη λειτουργικότητας αντίστροφης γεωκωδικοποίησης. Επίσης, έχει γίνει χρήση του σχεδιαστικού προτύπου εμπρόσθιου ελεγκτή, με αποτέλεσμα την ενθυλάκωση γενικών (κοινών) λειτουργιών επεξεργασίας σε ένα κύριο πρόγραμμα (ελεγκτής - υπερκλάση) και την επέκτασή του, μέσω της κληρονομικότητας, σε δύο άλλα προγράμματα (ελεγκτές - υποκλάσεις) σε αντίθεση με τα τρία αρχεία JSP τα οποία είναι ανεξάρτητα μεταξύ τους.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση υδρολογικών μοντέλων λόγω του ότι η θεματολογία δεν το απαιτεί.

Στην επιστημονική εργασία (Wood et al., 2007) περιγράφεται μία προσέγγιση για τη διερευνητική οπτική ανάλυση η οποία είναι χρήσιμη για την προκαταρκτική έρευνα μεγάλων δομημένων και πολύπλευρων συνόλων δεδομένων με χωρική και χρονική υπόσταση.

Για το σκοπό αυτό, προτείνεται μια μελέτη περίπτωσης γεω-οπτικοποίησης η οποία χρησιμοποιείται για την οπτική ανάλυση, σύνθεση και αλληλεπίδραση των κωδικοποιημένων δεδομένων που παρέχονται μέσω KML στο Google Earth και τα οποία προκύπτουν χρησιμοποιώντας ένα συνδυασμό από: MySQL για την αποθήκευση των δεδομένων, PHP για τη μεσολάβηση μεταξύ του εξυπηρετητή web και της βάσης για την ανάκτηση των δεδομένων και του γεωγραφικού πληροφοριακού συστήματος LandSerf για την ανάλυση και την επεξεργασία (εδώ πραγματοποιείται στη πραγματικότητα και εξερευνούνται στη συνέχεια τα αποτελέσματα KML από το Google Earth) της επιφάνειας και τον υπολογισμό του οπτικού αποτελέσματος λαμβάνοντας υπόψη διάφορους χωρικούς παράγοντες. Η KML χρησιμοποιείται για τη περιγραφή των κωδικοποιημένων δεδομένων και τον ορισμό των μεταξύ τους αλληλεπιδράσεων. Αυτή η ενοποίηση των τεχνολογιών επιτρέπει την αποθήκευση, την αναζήτηση (query), την επεξεργασία και τη δημιουργία αφηρημένων γραφικών. Το Google Earth χρησιμοποιείται ως μέσο για την γρήγορη οπτικοποίηση των δεδομένων και για τη διαχείρισή τους μέσω των ειδικών εργαλείων πλοήγησης που περιλαμβάνει λόγω της ικανότητας του να παρέχει μεγάλες ροές δεδομένων από τους εξυπηρετητές ως απόκριση στην αλληλεπίδραση του χρήστη.

Χρησιμοποιείται σύννεφο ετικετών (tag cloud) για τη σύνοψη της σχετικής συχνότητας των ζητούμενων επιχειρήσεων σε μια συγκεκριμένη γεωγραφική ζώνη. Η επιλογή κάποιου ονόματος επιχείρησης από το σύννεφο έχει ως αποτέλεσμα τη δημιουργία KML που μπορεί να εστιάσει στη περιοχή όπου η συγκεκριμένη επιχείρηση ανήκει οριοθετώντας τη περιοχή αυτή στο χάρτη. Για το σκοπό αυτό χρησιμοποιείται χάρτης ετικετών για να προσδώσει χωρική υπόσταση στο σύννεφο ετικετών. Επίσης, χρησιμοποιείται μία τεχνική ομαδοποίησης με γραφικό τρόπο με την οποία ομαδοποιούνται οι ζητούμενες εγγραφές ανά γεωγραφική τοποθεσία, χρονικό σημείο και διάφορα άλλα χαρακτηριστικά, κάθε ένα από τα οποία αντιστοιχεί σε μια γεωγραφική περιοχή.

Εξαιτίας της μεγάλης διακύμανσης ως προς την κατανομή του πληθυσμού στις διάφορες γεωγραφικές περιοχές δημιουργείται πρόβλημα κατά την οπτικοποίηση της πυκνότητας του πληθυσμού στο χάρτη. Το παραπάνω πρόβλημα επιλύθηκε με τη χρήση του LandSerf όπου δημιουργήθηκαν επιφάνειες αποτύπωσης της πυκνότητας του πληθυσμού μέσω της τεχνικής της τετραγωνικής εξομάλυνσης.

Τέλος, αξίζει να σημειωθεί ότι αξιολογήθηκαν τέσσερις πτυχές της προσέγγισης: οι οπτικές κωδικοποιήσεις που χρησιμοποιήθηκαν, η συμβολή τους στην οπτική εξερεύνηση του συνόλου δεδομένων, τα εργαλεία που χρησιμοποιήθηκαν και η προσέγγιση του «mashup».

Στην παρούσα εργασία έχει χρησιμοποιηθεί το ΣΔΒΔ PostgreSQL/PostGIS για την υλοποίηση της βάσης δεδομένων ενώ στην υπό σύγκριση εργασία χρησιμοποιείται βάση δεδομένων MySQL. Στη παρούσα εργασία έχει χρησιμοποιηθεί JSON για τη μορφοποίηση των δεδομένων που επιστρέφονται από τον εξυπηρετητή και HTML/CSS/JavaScript/jQuery για την περιγραφή και παρουσίασή τους ενώ στην άλλη εργασία χρησιμοποιείται KML. Τέλος, στη παρούσα εργασία χρησιμοποιούνται τα Google Maps σε αντίθεση με το Google Earth που χρησιμοποιεί η άλλη εργασία.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει χρησιμοποιηθεί PHP για την αλληλεπίδραση μεταξύ βάσης δεδομένων και εξυπηρετητή.

Στην παρούσα εργασία έχει γίνει βελτίωση ως προς την απόδοση των χωρικών ερωτημάτων λόγω του ότι χρησιμοποιούνται αφενός συναρτήσεις PostGIS και αφετέρου εξειδικευμένα ευρετήρια (GiST).

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση κάποιου εξειδικευμένου γεωγραφικού πληροφοριακού συστήματος (GIS) όπως το LandSerf που χρησιμοποιείται στην άλλη εργασία για την ανάλυση και επεξεργασία των χωρικών δεδομένων λόγω του ότι οι απαιτήσεις της παρούσας εργασίας καλύπτονται άριστα με το συνδυασμό της προγραμματιστικής διεπαφής των Google Maps και των παρεχόμενων συναρτήσεων της PostgreSQL/PostGIS.

Στην επιστημονική εργασία (Rao, Govardhan & Rao, 2012) υλοποιήθηκε ένα σύστημα λογισμικού που επικεντρώνεται στην ανάπτυξη μεθοδολογιών για την εξόρυξη προτύπων τοπολογικών σχέσεων από χωροχρονικές βάσεις δεδομένων. Το σύστημα βασίζεται στο πρότυπο αρχιτεκτονικής MVC (Model – View – Controller) και υποστηρίζεται από χωροχρονική βάση δεδομένων η οποία σχεδιάστηκε και υλοποιήθηκε για το σκοπό αυτό.

Η μεθοδολογία εξόρυξης προτύπων τοπολογικών σχέσεων περιλαμβάνει τρία βήματα. Πρώτο βήμα είναι η ανάκτηση των ζητούμενων χωροχρονικών αντικειμένων από τη βάση, η εξεύρεση των γεωμετρικών σχέσεων μεταξύ κάθε ζεύγους χωροχρονικών αντικειμένων και η αποθήκευσή τους στο πολυδιάστατο μοντέλο (ενδιάμεσο πίνακα). Δεύτερο βήμα είναι η ανίχνευση των αλλαγών, με το πέρασμα του χρόνου, σε αυτές τις σχέσεις (ανά δύο) επιτελώντας επεξεργασία χωρικών και χρονικών δεδομένων. Τρίτο βήμα είναι η εξόρυξη των τοπολογικών προτύπων από αυτές τις σχέσεις με τελικό αποτέλεσμα την εξόρυξη γνώσης.

Το μοντέλο εμπεριέχει τη λογική της εφαρμογής. Λαμβάνει τα κατάλληλα δεδομένα από τη βάση ανάλογα με την αίτηση που πρέπει να εξυπηρετήσει, τα επεξεργάζεται ανακαλύπτοντας έτσι τοπολογικές σχέσεις τις οποίες αποθηκεύει, προσωρινά, σε ένα πολυδιάστατο μοντέλο. Στη συνέχεια αφού εφαρμόσει κατάλληλους αλγορίθμους εξάγει τα πρότυπα των τοπολογικών σχέσεων και ενημερώνει τις αντίστοιχες όψεις. Η όψη ασχολείται με τη παρουσίαση των προτύπων τοπολογικών σχέσεων στο χρήστη, λαμβάνοντας όλες τις απαραίτητες πληροφορίες από το μοντέλο μέσω του ελεγκτή. Ο ελεγκτής λαμβάνει την είσοδο του χρήστη και προωθεί κατάλληλα αιτήματα υπηρεσιών στο μοντέλο όπου διεκπεραιώνονται και λαμβάνει τα αποτελέσματα τα οποία στέλνει στην αντίστοιχη όψη.

Η υλοποίηση της χωροχρονικής βάσης δεδομένων πραγματοποιήθηκε στο σύστημα διαχείρισης βάσεων δεδομένων (ΣΔΒΔ) PostgreSQL/PostGIS. Χρησιμοποιήθηκε η αντικειμενοστρεφής γλώσσα προγραμματισμού Java (με σχεδιαστικό πρότυπο MVC) τόσο για την υλοποίηση των αλγορίθμων όσο και για τη πρόσβαση στο επίπεδο της βάσης δεδομένων σε συνδυασμό με το JDBC (συγκεκριμένα χρησιμοποιήθηκε ο οδηγός PostgreSQL JDBC).

Τέλος, έγινε χρήση του OpenJump GIS για τη δημιουργία ενός συνόλου από δεδομένα κτηματολογίου, τα οποία φορτώθηκαν στη βάση, με σκοπό την εφαρμογή αλγορίθμων πάνω σε αυτά.

Στην παρούσα εργασία έχει χρησιμοποιηθεί PHP για την ανάπτυξη της εφαρμογής σε αντίθεση με την άλλη εργασία που χρησιμοποιεί Java. Στη παρούσα εργασία χρησιμοποιείται το Quantum GIS σε αντίθεση με το OpenJump που χρησιμοποιεί η άλλη εργασία. Επίσης, η παρούσα εργασία επικεντρώνεται μόνο στη χωρική διάσταση των δεδομένων σε αντίθεση με την άλλη που επικεντρώνεται τόσο στη χωρική όσο και στη χρονική διάσταση.

Ομοιότητα παρατηρείται στο ότι και στις δύο εφαρμογές έχει χρησιμοποιηθεί αντικειμενοστρεφής προσέγγιση ανάπτυξης με PHP και Java αντίστοιχα και κοινό σχεδιαστικό πρότυπο, το MVC. Επίσης, και στις δύο εφαρμογές χρησιμοποιείται το ΣΔΒΔ PostgreSQL/PostGIS για την υλοποίηση των βάσεων δεδομένων τους και των αντίστοιχων πινάκων τους. Θα πρέπει επιπλέον να σημειωθεί πως οι γεωμετρικοί τύποι που χρησιμοποιούνται και στις δύο εργασίες είναι: τα πολύγωνα, οι γραμμές και τα σημεία. Τέλος, και στις δύο εργασίες χρησιμοποιούνται τοπολογικές σχέσεις.

Στην παρούσα εργασία έχει γίνει βελτίωση ως προς την υλοποίηση του ελεγκτή κάνοντας χρήση του σχεδιαστικού προτύπου εμπρόσθιου ελεγκτή, με αποτέλεσμα την ενθυλάκωση των γενικών λειτουργιών επεξεργασίας σε ένα κύριο ελεγκτή και την επέκτασή του, μέσω της κληρονομικότητας, σε δύο άλλους ελεγκτές.

Στην παρούσα εργασία έχουν παραληφθεί κάποιες από τις τοπολογικές σχέσεις που περιλαμβάνονται στην άλλη εργασία. Επιπρόσθετα, στη παρούσα εργασία δε χρησιμοποιούνται αλγόριθμοι για εξόρυξη προτύπων και γενικά γνώσης.

Σε μία άλλη επιστημονική εργασία (Serrano et al., 2008) αναπτύχθηκε ένα γεωγραφικό πληροφοριακό σύστημα Web για τη διαχείριση της μελισσοκομίας στη Sierra Morena (Ανδαλουσία, Νότια Ισπανία). Ο σκοπός της εργασίας είναι να περιγράψει το σχεδιασμό και τις δυνατότητες ενός γεωγραφικού πληροφοριακού συστήματος ως εργαλείο απεικόνισης για τον χαρακτηρισμό του μελιού και επίσης ως εργαλείο διαχείρισης για τη ρύθμιση της προστατευόμενης ονομασίας προέλευσής του. Επίσης, δίνει τη δυνατότητα εμφάνισης των τοποθεσιών των μελισσιών και πληροφορίες συσχέτισής τους με περιβαλλοντολογικούς παράγοντες και παραμέτρους ποιότητας του μελιού. Η εφαρμογή καλείται SMHGIS (Sierra Morena Honey GIS). Έχουν ληφθεί υπόψη: η επέκταση και η οριοθέτηση των περιοχών όπως επίσης ορογραφικά, ορογενετικά, κλιματικά και γεωπονικά χαρακτηριστικά (για παράδειγμα πηγές νέκταρ). Για το σκοπό της εφαρμογής χρησιμοποιήθηκαν: ένας χάρτης, στοιχεία απογραφής των μελισσοκομείων και της παραγωγής του μελιού και στοιχεία χαρακτηρισμού του μελιού όσον αφορά την προστατευόμενη ονομασία προέλευσής του.

Για την αποθήκευση των διαφόρων προελεύσεων, των ιδιοτήτων και των γεωγραφικών χαρακτηριστικών του παραγόμενου μελιού σχεδιάστηκε και υλοποιήθηκε μια βάση δεδομένων. Η βάση σχεδιάστηκε σύμφωνα με ένα εννοιολογικό σχήμα οντοτήτων. Η κάθε οντότητα ακολουθεί μια συγκεκριμένη δομή που περιλαμβάνει τη περιγραφή της οντότητας, κάποια χαρακτηριστικά (όπως το πλήθος των πεδίων της και το κλειδί) και μια περιγραφή του κάθε πεδίου. Η βάση υλοποιήθηκε με το σύστημα PostgreSQL/PostGIS με τέτοιο τρόπο ώστε να είναι εφικτή η χρήση γεωμετρικών τύπων δεδομένων, να διευκολύνεται η εκτέλεση σύνθετων ερωτημάτων χωρίς επιπτώσεις στην απόδοση και να είναι ασφαλές και εύρωστο. Για την εισαγωγή των δεδομένων από τα αρχεία shape χρησιμοποιήθηκε το «shp2pg» το οποίο είναι ένα ενσωματωμένο εργαλείο του PostGIS. Δίνεται η δυνατότητα απομακρυσμένης αναζήτησης και τροποποίησης στη βάση δεδομένων μέσω κατάλληλων φορμών web διευκολύνοντας έτσι τη διαδικασία εισόδου δεδομένων. Στα πλαίσια της υλοποίησης των φορμών έχει χρησιμοποιηθεί PHP σε συνδυασμό με το Apache Struts το οποίο εισάγει μια αρχιτεκτονική ανάπτυξης MVC.

Για την υλοποίηση του συστήματος χρησιμοποιήθηκε ένας συνδυασμός από Apache και PHP. Για τη γραφική απεικόνιση χρησιμοποιήθηκε το περιβάλλον MapServer το οποίο επιτρέπει την απεικόνιση και την ανάλυση γεωγραφικών πληροφοριών μέσω της τεχνολογίας IMS. Η διεπαφή της εφαρμογής υλοποιήθηκε με χρήση του συστήματος Chameleon το οποίο βρίσκεται μέσα στο MapServer και περιλαμβάνει ένα σύνολο από widgets.

Το κύριο μενού επιτρέπει στον χρήστη την πρόσβαση σε διάφορες λειτουργίες της εφαρμογής (χαρτογραφία, ερωτήματα, εγγραφή χρήστη) με επιλογή του αντίστοιχου πλήκτρου. Το σύστημα διαχειρίζεται και εξυπηρετεί τρείς τύπους χρηστών, τον απλό επισκέπτη ο οποίος έχει πρόσβαση στο χάρτη και σε αναφορές, τον εγγεγραμμένο χρήστη που μπορεί επιπλέον να τροποποιήσει κάποια δεδομένα με το περιορισμό ότι ελέγχονται στη συνέχεια και επαληθεύονται από το διαχειριστή και επίσης μπορεί να έχει πρόσβαση και σε ιδιωτικές αναφορές και το διαχειριστή ο οποίος ελέγχει ολόκληρο το σύστημα, επαληθεύει και δημοσιοποιεί δεδομένα.

Στην παρούσα εργασία η διεπαφή έχει υλοποιηθεί με HTML/CSS, jQuery και Bootstrap και συνδέεται με Google Maps για τη παροχή χάρτη ενώ στην υπό σύγκριση εργασία υλοποιείται με Chameleon και συνδέεται με τον MapServer για τη παροχή του χάρτη αντίστοιχα. Η πρόσβαση στις υπηρεσίες της εφαρμογής της παρούσας εργασίας είναι ανοιχτή για όλους τους χρήστες σε αντίθεση με την εφαρμογή της εργασίας που συγκρίνεται, στην οποία οι χρήστες κατηγοριοποιούνται με κατάλληλα δικαιώματα ανά κατηγορία.

Ομοιότητα παρατηρείται στο ότι και στις δύο εφαρμογές έχει χρησιμοποιηθεί συνδυασμός Apache/PHP για επιλογή εξυπηρετητή web και γλώσσας προγραμματισμού. Επίσης, χρησιμοποιείται κοινό σχεδιαστικό πρότυπο, το MVC. Επιπλέον, και στις δύο εφαρμογές χρησιμοποιείται η βάση δεδομένων PostgreSQL/PostGIS. Τέλος, υποστηρίζεται η λειτουργία της γεωκωδκοποίησης εξίσου και στις δύο εργασίες.

Στην παρούσα εργασία έχει γίνει βελτίωση στο ότι δίνεται η δυνατότητα στο χρήστη να αλληλεπιδρά κατευθείαν με τη διεπαφή του χάρτη επιτελώντας με αυτό τον τρόπο αντίστροφη γεωκωδικοποίηση. Επίσης, στη παρούσα εργασία χρησιμοποιείται το μοντέλο οντοτήτων – συσχετίσεων (ER) για το σχεδιασμό της βάσης δεδομένων ενώ στην άλλη εργασία αυτό επιτυγχάνεται με ένα απλό εννοιολογικό σχήμα οντοτήτων και χωρίς την εισαγωγή συσχετίσεων ανάμεσά τους.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση των δεδομένων στη βάση από αρχεία shape και αντ' αυτού τα δεδομένα έχουν εισαχθεί με πολλαπλές SQL εντολές INSERT και με τη βοήθεια του Quantum GIS.

Στην εργασία των Ligęza et al. (2011) αναπτύχθηκαν τρία πρωτότυπα συστήματα Web για απόκτηση, διαχείριση και επεξεργασία πληροφοριών σχετικά με διάφορους υφιστάμενους κινδύνους και απειλές για τη δημόσια ασφάλεια σε αστικά περιβάλλοντα και για αυτόματη εξαγωγή γνώσης και διαχείριση αυτής της γνώσης με χωρικές (GIS) τεχνικές.

Οι λειτουργικές απαιτήσεις του συστήματος περιλαμβάνουν: υποβολή και διαχείριση των σχετιζόμενων με απειλές δεδομένων, απεικόνιση πληροφοριών επάνω σε χάρτη, ανάλυση των δεδομένων με προκαθορισμένες αναφορές, ανεπτυγμένες ικανότητες αναζήτησης, κατηγοριοποίηση των χρηστών, πλαίσιο για την εκτίμηση της αξιοπιστίας των υποβληθέντων απειλών, σύστημα ειδοποίησης, ενημερωτικά δελτία, μια μηχανή κανόνων για τη διευκόλυνση της διαχείρισης των δεδομένων και της κατάρτισης των αναφορών, ενσωμάτωση με γνωστές εφαρμογές κοινωνικής δικτύωσης, αυθεντικοποίηση του χρήστη και υποστήριξη για πλατφόρμες κινητών.

Αρχικά, παρουσιάζονται τα τρία πρωτότυπα συστήματα (INDECT Threat Monitor, Threat Radar, Safety Protect) τα οποία παρουσιάζουν στο χρήστη πιθανούς κινδύνους – απειλές και τη τοποθεσία τους στο χάρτη. Οι εγγεγραμμένοι χρήστες μπορούν να εισάγουν νέους κινδύνους και τα αντίστοιχα χαρακτηριστικά τους (γεωμετρικά και μη) τα οποία αποθηκεύονται σε χωρική βάση δεδομένων.

Περιγράφεται η λειτουργία τους και αξιολογούνται οι διάφορες τεχνολογίες που χρησιμοποιήθηκαν για κάθε ένα από αυτά. Με βάση τα αποτελέσματα επιχειρείται να υλοποιηθεί το τελικό σύστημα.

Για το INDECT Threat Monitor χρησιμοποιήθηκαν: το πλαίσιο εργασίας Django, τα Google Maps και η PostgreSQL/PostGIS. Για το Threat Radar χρησιμοποιήθηκαν τόσο τεχνολογίες από τη μεριά του πελάτη όπως, XHTML, CSS, JavaScript, jQuery, jQuery UI και η προγραμματιστική διεπαφή των Google Maps όσο και τεχνολογίες από τη μεριά του εξυπηρετητή όπως, ο

εξυπηρετητής Apache, το πλαίσιο εργασίας Zend (βασισμένο σε PHP), το Maestro και η PostgreSQL/PostGIS. Το Safety Protect βασίζεται στο Ruby on Rails πλαίσιο εργασίας και χρησιμοποιεί τη τεχνολογία Phusion Passenger.

Στην παρούσα εργασία η ανάπτυξη της εφαρμογής βασίζεται σε ένα ενιαίο πρωτότυπο και όχι σε τρία όπως συμβαίνει στην άλλη εργασία. Επίσης, στη παρούσα εργασία έχει χρησιμοποιηθεί το πλαίσιο εργασίας CodeIgniter σε αντίθεση με το Django , Zend και Ruby on Rails του πρώτου, του δεύτερου και του τρίτου πρωτοτύπου αντίστοιχα της άλλης εργασίας. Η πρόσβαση στις υπηρεσίες της εφαρμογής της παρούσας εργασίας είναι ανοιχτή για όλους τους χρήστες σε αντίθεση με την άλλη εργασία σύμφωνα με την οποία οι χρήστες κατηγοριοποιούνται με κατάλληλα δικαιώματα ανά κατηγορία.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει χρησιμοποιηθεί συνδυασμός Apache/PHP για επιλογή εξυπηρετητή web και γλώσσας προγραμματισμού. Επίσης, χρησιμοποιείται το σύστημα διαχείρισης χωρικών βάσεων δεδομένων PostgreSQL/PostGIS και έχουν υλοποιηθεί χωρικά ευρετήρια στις γεωμετρικές στήλες των πινάκων των βάσεων. Επιπλέον χρησιμοποιούνται τα Google Maps ως υπηρεσία εμφάνισης χαρτών στο web. Υποστηρίζονται οι λειτουργίες γεωκωδικοποίησης και αντίστροφης γεωκωδικοποίησης εξίσου και στις δύο εργασίες. Παρέχονται πληροφοριακά παράθυρα και στις δύο εργασίες.

Στην παρούσα εργασία έχει βελτιωθεί η αρχιτεκτονική του συστήματος με σκοπό η εφαρμογή να είναι επεκτάσιμη. Επίσης, χρησιμοποιείται το Bootstrap το οποίο περιλαμβάνει, μεταξύ άλλων, εκτεταμένο CSS προσφέροντας μία ελκυστική και διαισθητική διεπαφή στο τελικό χρήστη.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση λειτουργίας σχολιασμού και αξιολόγησης των δεδομένων όπως επίσης και η χρήση SSL για ασφαλή πλοήγηση των χρηστών μέσω https. Επίσης, έχει παραληφθεί η χρήση της βιβλιοθήκης jQuery UI.

Στην επιστημονική εργασία (Anderson & Moreno-Sanchez, 2003) αναπτύχθηκε ένα γεωγραφικό πληροφοριακό σύστημα Web με χρήση ελεύθερου λογισμικού και λογισμικού ανοικτού κώδικα για την υποστήριξη και ανάλυση της χωροταξίας στο κεντρικό Μεξικό. Στόχος του συστήματος είναι να επιτρέψει στους τελικούς χρήστες την εκτέλεση ερωτημάτων (queries) με επιθυμητές τιμές διάφορων περιβαλλοντολογικών παραγόντων για την αναγνώριση των περιοχών που τηρούν τα παραπάνω κριτήρια.

Οι απαιτήσεις του περιλαμβάνουν: την εύκολη ενσωμάτωσή του με την υπόλοιπη υπάρχουσα υποδομή, να είναι επεκτάσιμο και χαμηλού κόστους, την ελαχιστοποίηση των εξειδικευμένων απαιτήσεων του, την ευκολία εκμάθησής του και τη βελτίωση της αποδοτικότητάς του όσον αφορά στην επέκταση, τη συντήρηση και τον ποιοτικό έλεγχο της εθνικής βάσης δεδομένων με τη συγκέντρωση αυτών των λειτουργιών σε ένα μέρος.

Η διεπαφή του χωρίζεται σε δύο μέρη, όπου το αριστερό περιέχει τη φόρμα όπου ο χρήστης εισάγει τους επιθυμητούς περιβαλλοντολογικούς παράγοντες και εκτελεί το ερώτημα και το δεξί όπου εμφανίζονται τα αποτελέσματα επάνω σε χάρτη. Ο χάρτης δημιουργείται από τον MapServer και περιλαμβάνει διάφορα στρώματα (επίπεδα), κάθε ένα από τα οποία περιέχει τα χαρακτηριστικά που επιλέχθηκαν στο ερώτημα. Ο χρήστης μπορεί να επιλέξει ποιά στρώματα και ποια χαρακτηριστικά επιθυμεί να εμφανίσει ή να αποκρύψει. Οπτικά μπορεί να προσδιοριστεί αν υπάρχει μια περιοχή όπου τέμνονται όλα τα χαρακτηριστικά. Σε μια τέτοια περίπτωση, δημιουργούνται έγγραφα GML που αντιπροσωπεύουν τα επιλεγμένα χαρακτηριστικά του κάθε στρώματος. Τελικά, υπολογίζεται η τομή των επιλεγμένων περιοχών και η περιοχή που σχηματίζεται εξάγεται σε ένα αρχείο SVG και σε ένα αρχείο GML. Το αρχείο SVG μπορεί να εμφανιστεί αυτούσιο ή και με άλλα στρώματα χρησιμοποιώντας τον SVGeoprocessor.

Η λειτουργία έχει ως εξής: Τα δεδομένα εισάγονται από αρχεία shape ESRI στη χωρική βάση δεδομένων, που έχει υλοποιηθεί στο σύστημα PostgreSQL/PostGIS, όπου προκύπτουν οι αντίστοιχοι πίνακες. Ο χρήστης μέσω της φόρμας της διεπαφής επιλέγει τις παραμέτρους που τον ενδιαφέρουν για κάθε στρώμα και εκτελεί το ερώτημα στη βάση επιλέγοντας το αντίστοιχο πλήκτρο. Αν βρεθεί κάποια περιοχή που να τηρεί τις επιλεγμένες παραμέτρους εμφανίζεται σε ένα SVG χάρτη αλλιώς ο χρήστης λαμβάνει κατάλληλο μήνυμα. Οι παράμετροι που στάλθηκαν

στον εξυπηρετητή από το χρήστη μετασχηματίζονται σε ερωτήματα SQL με τη βοήθεια κώδικα PHP και εξυπηρετούνται από τη βάση η οποία επιστρέφει σχετικές πληροφορίες και τις συντεταγμένες οι οποίες με τη σειρά τους μετασχηματίζονται κατάλληλα, με τη βοήθεια κώδικα PHP, σε πολυγωνικές οντότητες GML που αποθηκεύονται σε έγγραφα GML (ένα για κάθε χαρακτηριστικό). Τα έγγραφα GML εισάγονται σε κατάλληλο πρόγραμμα Java το οποίο υπολογίζει τη τομή των χαρακτηριστικών. Συγκεκριμένα, ορίζει μια κλάση που χρησιμοποιεί τη μέθοδο SAX για τη μετατροπή των πολυγώνων GML σε αντικείμενα Java2D και αφού υπολογίσει τη τομή τους εξάγει το αποτέλεσμά της σε ένα νέο έγγραφο GML. Ένα αρχείο XSLT περιέχει οδηγίες για τη μετατροπή του κώδικα GML του προηγούμενου αρχείου σε κώδικα SVG για την γραφική αναπαράσταση των δεδομένων στο χάρτη. Τελικά δημιουργείται ο χάρτης SVG όπου φαίνονται τα αποτελέσματα. Το σύστημα βασίστηκε στο λειτουργικό σύστημα Red Hat Linux και στον εξυπηρετητή web Apache.

Στην παρούσα εργασία η ανάπτυξη της εφαρμογής βασίζεται στο πλαίσιο εργασίας CodeIgniter σε αντίθεση με την εφαρμογή της υπό σύγκρισης εργασίας που βασίζεται σε συνδυασμό PHP και Java. Επίσης, στη παρούσα εργασία χρησιμοποιείται η προγραμματιστική διεπαφή των Google Maps και τα Google Maps ως υπηρεσία εμφάνισης χαρτών στο web ενώ στην άλλη εργασία χρησιμοποιείται ο MapServer. Τέλος, στη παρούσα εργασία τα γεωμετρικά δεδομένα μορφοποιούνται σε GeoJSON μορφή ενώ στην άλλη εργασία χρησιμοποιείται η μορφή GML για τη περιγραφή τους και, γενικά, ενώ στη παρούσα εργασία το πρότυπο μορφοποίησης που ακολουθείται για τη περιγραφή των δεδομένων είναι το JSON στην άλλη είναι η XML.

Ομοιότητα παρατηρείται στο ότι και στις δύο εργασίες έχει χρησιμοποιηθεί ο εξυπηρετητής web Apache. Επίσης, χρησιμοποιείται το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL/PostGIS για την υλοποίηση των βάσεων δεδομένων των υπό σύγκριση εφαρμογών. Επιπλέον, και στις δύο εφαρμογές έχει χρησιμοποιηθεί HTML για τη δημιουργία των φορμών.

Στην παρούσα εργασία έχει γίνει βελτίωση ως προς τη ταχύτητα ανάλυσης και επεξεργασίας των χωρικών δεδομένων λόγω του ότι υποστηρίζονται και χρησιμοποιούνται όλοι οι απαραίτητοι τελεστές και συναρτήσεις του PostGIS ενώ στην άλλη εργασία λόγω παλαιότερης έκδοσης του PostGIS απαιτείται επιπλέον χωρική επεξεργασία που επιτυγχάνεται με κώδικα Java.

Στην παρούσα εργασία έχει παραληφθεί η ενσωμάτωση των δεδομένων στη βάση από αρχεία shape και αντ' αυτού τα δεδομένα έχουν εισαχθεί με πολλαπλές εντολές SQL INSERT και με τη βοήθεια του Quantum GIS. Επίσης, έχει παραληφθεί η χρήση κάποιου εργαλείου παρουσίασης της αναπαράστασης των δεδομένων, όπως το XSLT, λόγω του ότι δε χρησιμοποιείται XML στην παρούσα εργασία.

2.2 Ανασκόπηση των συγκριτικών αποτελεσμάτων

Στην εικόνα 4 πραγματοποιείται μία ανασκόπηση των εργαλείων που χρησιμοποίησαν οι επιστημονικές εργασίες που παρουσιάστηκαν προηγουμένως.

Boondao, Esichaikul & *Tripathi*, 2003	MapServer PostgreSQL/PostGIS Apache Web Server PHP

Piarsa, Wiranatha & Putra, 2012	Google Maps, Google Maps Server Ruby On Rails MVC AJAX, JavaScript JSON Responsive web design
Oussalah et al., 2012	Google Maps PosgreSQL/PostGIS , MySQL Apache Lucene, Apache SOLR Django / GeoDjango Python Java Twitter Streaming API MTV WordNet JavaScript JSON, GeoJSON, XML
Singh, Chutia & Sudhakar, 2012	MapServer PostgreSQL/PostGIS Apache Web Server PHP Chameleon CWC2 HTML
Yuan & Cheng, 2007	Google Maps, IHACRES JSP AJAX, JavaScript XML
Wood et al., 2007	Google Earth, LandSerf GIS MySQL PHP KML Tag map, Tag cloud

Rao, Govardhan & Rao, 2012	OpenJump GIS PostgreSQL/PostGIS Java MVC
Serrano et al., 2008	MapServer PostgreSQL/PostGIS Apache Web Server PHP MVC Chameleon
Ligęza et al., 2011	Google Maps PostgreSQL/PostGIS Apache Web Server Django Web framework, Zend framework, Ruby on Rails PHP JavaScript, jQuery, XHTML, CSS
Anderson & Moreno-Sanchez, 2003	MapServer PostgreSQL/PostGIS Apache Web Server PHP Java GML, SVG XSLT HTML

Εικόνα 4 - Ανασκόπηση των εργαλείων που χρησιμοποίησαν οι επιστημονικές εργασίες

2.3 Σύνοψη των χαρακτηριστικών της πλατφόρμας

Λαμβάνοντας υπόψη τις τεχνολογίες που χρησιμοποιήθηκαν στις παραπάνω επιστημονικές εργασίες και τις απαιτήσεις της παρούσας εργασίας καθορίστηκαν τα θεωρητικά χαρακτηριστικά που έχει η πλατφόρμα τα οποία παρουσιάζονται στην εικόνα 5.

Υπηρεσία εμφάνισης χαρτών στο Web	Google Maps
Σύστημα διαχείρισης βάσης δεδομένων	PostgreSQL/PostGIS
Web εξυπηρετητής	Apache
Πλαίσιο εργασίας (framework)	CodeIgniter
Σχεδιαστικά πρότυπα	MVC, Front Controller

Γλώσσα προγραμματισμού από τη μεριά του εξυπηρετητή	PHP
Γλώσσες προγραμματισμού από τη μεριά του πελάτη	JavaScript, jQuery, AJAX, HTML, CSS, Twitter Bootstrap
Πρότυπο μορφοποίησης δεδομένων	JSON, GeoJSON
GIS εργαλείο	Quantum GIS

Εικόνα 5 - Σύνοψη των θεωρητικών χαρακτηριστικών της πλατφόρμας

Με την αναφορά των θεωρητικών χαρακτηριστικών της πλατφόρμας κλείνει το παρόν κεφάλαιο. Στο επόμενο κεφάλαιο παρουσιάζονται πλήρως όλες οι τεχνολογίες, οι αρχιτεκτονικές και τα σχεδιαστικά πρότυπα (design patterns) που χρησιμοποιήθηκαν τόσο για το σχεδιασμό όσο και για την υλοποίηση της εφαρμογής.

ΚΕΦΑΛΑΙΟ 3: ΤΕΧΝΟΛΟΓΙΕΣ

3.1 Η γλώσσα HTML

Η HTML (HyperText Markup Language) χρησιμοποιείται για τη δημιουργία ιστοσελίδων στατικού περιεχομένου. Η HTML δεν είναι μια πραγματική γλώσσα προγραμματισμού καθόσον δε διαθέτει τις συνήθεις εντολές που χαρακτηρίζουν τις γλώσσες προγραμματισμού. Χρησιμοποιείται κυρίως για τον προσδιορισμό του περιεχόμενου ενός εγγράφου όπως επίσης και για την περιγραφή του τρόπου με τον οποίο το έγγραφο αυτό θα παρουσιαστεί σε ένα web πελάτη (φυλλομετρητής). Ο φυλλομετρητής ιστοσελίδων είναι ένα πρόγραμμα το οποίο χρησιμεύει για την εξερεύνηση του διαδικτύου. Το πρόγραμμα αυτό μεταφράζει τον κώδικα ο οποίος είναι γραμμένος σύμφωνα με τους κανόνες της HTML σε μια οπτική παρουσίαση στην οθόνη. Από τεχνικής πλευράς το έγγραφο είναι ο κώδικας της HTML και η σελίδα είναι ο τρόπος με τον οποίο παρουσιάζεται το αποτέλεσμα του κώδικα αυτού στην οθόνη. Με την HTML μπορούν να δημιουργηθούν υπέρ-έγγραφα. Ένα υπέρ-έγγραφο περιέχει συνδέσμους (links) με άλλα σημεία τα οποία μπορεί να βρίσκονται στο ίδιο ή σε άλλα έγγραφα.

Μια ιστοσελίδα προσδιορίζεται από τη διεύθυνση URL (Uniform Resource Locator) η οποία περιγράφει τη θέση ή την διεύθυνση στο διαδίκτυο ενός εγγράφου. Ο γενικός τύπος μιας διεύθυνσης URL είναι ο ακόλουθος: **Πρωτόκολλο://όνομα domain/διαδρομή/όνομα αρχείου**

Ο κώδικας HTML αποτελείται από κάποια δομικά στοιχεία τα οποία είναι οι ετικέτες (tags) και οι ιδιότητες (attributes). Οι ετικέτες είναι κώδικας HTML που περικλείεται ανάμεσα στα σύμβολα μικρότερο (<) και μεγαλύτερο (>). Οι ετικέτες αυτές χρησιμοποιούνται για να περιγράψουν το περιεχόμενο μιας ιστοσελίδας. Μπορούμε να πούμε ότι οι ετικέτες είναι ένα είδος εντολής που καθορίζουν την εμφάνιση και τις ιδιότητες του κειμένου. Για παράδειγμα, η ετικέτα <p> καθορίζει την αρχή μιας νέας παραγράφου ενώ η ετικέτα
 εισάγει νέα γραμμή στο κείμενο. Η γενική σύνταξη μιας εντολής HTML έχει τον ακόλουθο τύπο: **<Ετικέτα έναρξης ιδιότητα=”τιμή”> Περιεχόμενο ετικέτας <Ετικέτα τερματισμού>**

Το W3C (World Wide Web Consortium) έχει καθορίσει τις ετικέτες και τα πρωτόκολλα της HTML. Ιδρύθηκε το 1994 προκειμένου να προάγει την εξέλιξη του web. Ελέγχεται από το Massachusetts Institute of Technology των ΗΠΑ, το Institut National de Recherché en Informatique et en Automatique της Γαλλίας και το Keio University της Ιαπωνίας.

Η HTML έχει επεκταθεί με την XHTML (Extensible HyperText Markup Language) η οποία είναι σχεδόν ταυτόσημη με την HTML 4.01 αλλά έχει πιο αυστηρή σύνταξη. Ένας από τους λόγους για τους οποίους αναπτύχθηκε είναι η ύπαρξη πολλών σελίδων στο web με «κακό» κώδικα. Στην XHTML όλες οι ετικέτες και τα χαρακτηριστικά τους γράφονται με μικρά γράμματα, οι ετικέτες κλείνουν πάντα (ακόμα και αυτές χωρίς περιεχόμενο) και οι τιμές των χαρακτηριστικών περικλείονται πάντα σε εισαγωγικά.

3.2 Κανόνες στυλ CSS

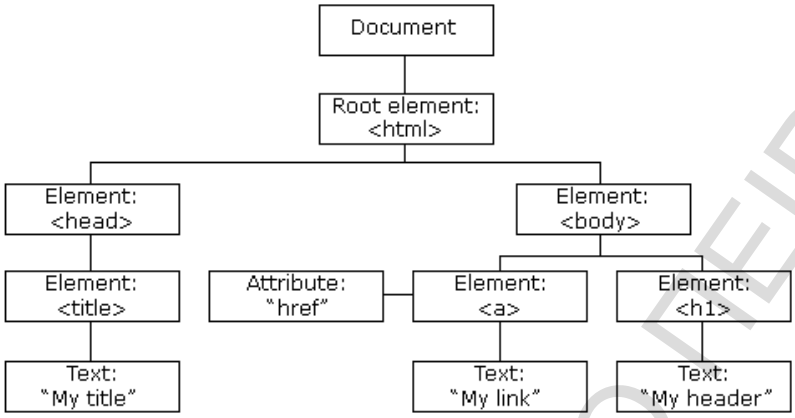
Η χρήση κανόνων στυλ (Cascading Style Sheets) για τη διαμόρφωση της σελίδας στην οθόνη παρέχουν στον προγραμματιστή τη δυνατότητα ομοιόμορφης εμφάνισης των ιστοσελίδων με ένα γενικό και συγκεντρωτικό τρόπο. Οι κανόνες στυλ μπορούν να διαμορφωθούν μέσα στο ίδιο το έγγραφο με τη χρήση της ετικέτας <style> και </style> ή μπορεί να δημιουργηθούν σε ένα εξωτερικό αρχείο και να καλούνται στη συνέχεια από το έγγραφο (εξωτερικοί κανόνες στυλ).

Οι κανόνες στυλ δημιουργούνται με τη γλώσσα CSS και τα αντίστοιχα αρχεία έχουν την κατάληξη “.css”. Έχουν το ίδιο συντακτικό με τους κανόνες που δημιουργούνται μέσα στις ετικέτες <style> και </style>. Το όνομα της ετικέτας ακολουθείται από μια αριστερή αγκύλη ({). Οι ιδιότητες τοποθετούνται μετά την αγκύλη, έχουν τη μορφή **ιδιότητα: τιμή** και διαχωρίζονται με το σύμβολο (;). Μια δεξιά αγκύλη (}) κλείνει τις αλλαγές στο στυλ που αφορούν τη συγκεκριμένη ετικέτα. Εκτός από κανόνες στυλ για ετικέτες δίνεται η δυνατότητα ορισμού κανόνων στυλ για κλάσεις όπου ομαδοποιούνται ιδιότητες που θα εφαρμοσθούν στο έγγραφο ή σε πολλαπλά έγγραφα κάτω από ένα όνομα που αρχίζει απαραίτητα με τελεία (.) Αντίστοιχα

δίνεται η δυνατότητα ορισμού κανόνων στυλ για id μόνο που εδώ το όνομα αρχίζει απαραίτητα με το σύμβολο (#).

3.3 Μοντέλο αντικειμένου εγγράφου (DOM)

Το μοντέλο αντικειμένου εγγράφου είναι μια προγραμματιστική διεπαφή (API) που επιτρέπει στα προγράμματα να αλληλεπιδράσουν με έγγραφα τύπου HTML (ή XML). Αποτελεί πρότυπο του W3C και βασίζεται σε μια αντικειμενοστραφή δομή που αντιστοιχεί στη δομή του εγγράφου που μοντελοποιεί.



Εικόνα 6 - DOM

Στο DOM, τα έγγραφα έχουν μια ιεραρχική δενδρική δομή. Περιλαμβάνει πάντα ως κόμβο – ρίζα, το έγγραφο από το οποίο προκύπτει το στοιχείο html. Χωρίζεται σε 3 επίπεδα, το βασικό (core) DOM που αποτελεί το καθιερωμένο μοντέλο για κάθε δομημένο έγγραφο, το XML DOM που είναι το καθιερωμένο μοντέλο για έγγραφα XML και το HTML DOM που είναι το καθιερωμένο μοντέλο για έγγραφα HTML.

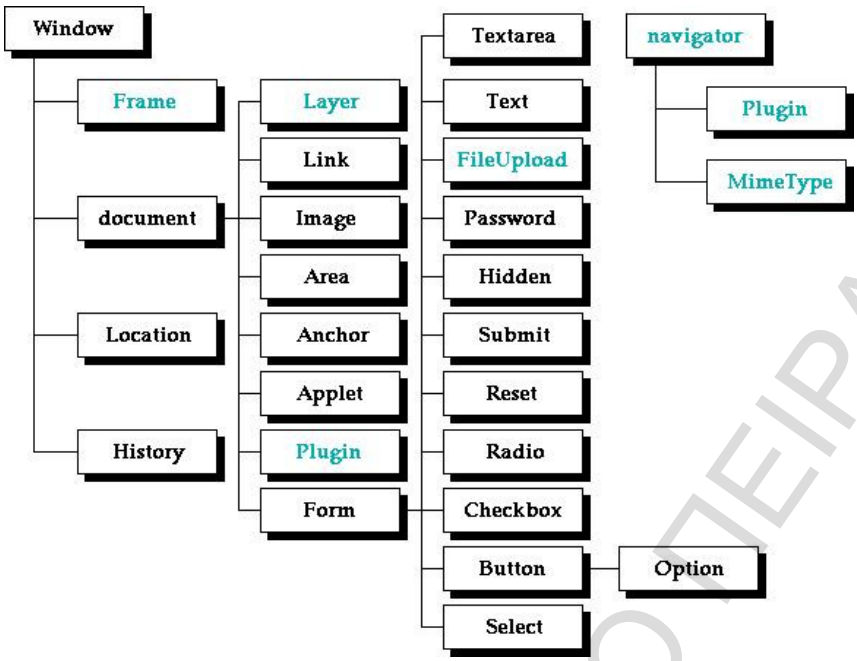
Το DOM ορίζει τα αντικείμενα και τις ιδιότητες όλων των στοιχείων του εγγράφου και τις μεθόδους για την πρόσβαση σε αυτά. Επιτρέπει στα σενάρια να προσπελάσουν και να τροποποιήσουν το δέντρο του εγγράφου.

3.4 Η γλώσσα σεναρίων JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού σεναρίων (scripting) για την εκτέλεση προγραμμάτων στην πλευρά του πελάτη (client-side) που χρησιμοποιείται ευρέως, λόγω των πολλών δυνατοτήτων και της απλότητας που την χαρακτηρίζει, σε web εφαρμογές. Για παράδειγμα, η εμφάνιση κάποιας αλλαγής (στο κείμενο, στο χρώμα του φόντου, στις εικόνες κτλ.) κατόπιν αλληλεπίδρασης του χρήστη με κάποιο αντικείμενο της ιστοσελίδας ή η δημιουργία εντυπωσιακών μενού και ελκυστικών γκαλερί είναι κάποιες από τις δυνατότητες που σχετίζονται με την JavaScript. Αναπτύχθηκε αρχικά από τον Brendan Eich της Netscape το 1995 και πρωτοεμφανίστηκε στον Netscape Navigator.

Εκτελείται στο φυλλομετρητή και έχει πρόσβαση σε όλα τα στοιχεία του εγγράφου HTML μέσω του DOM. Θα μπορούσαμε να πούμε ότι είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού λόγω του ότι τα βασικότερα στοιχεία της υλοποιούνται ως αντίγραφα κλάσεων, δηλαδή αντικείμενα. Τα πιο βασικά αντικείμενά της είναι: το αντικείμενο του πίνακα, της συμβολοσειράς, της ημερομηνίας, του παραθύρου, του εγγράφου, της φόρμας και των στοιχείων της φόρμας (πεδίο κειμένου, πλήκτρο, λίστα επιλογής κτλ.). Τα αντικείμενα έχουν ιδιότητες και μεθόδους όπου οι ιδιότητες απλά αλλάζουν τιμές και οι μέθοδοι εκτελούν ενέργειες.

Για παράδειγμα το αντικείμενο του εγγράφου (document) έχει ιδιότητες όπως τη bgcolor που προσδιορίζει το χρώμα του φόντου της σελίδας και μεθόδους όπως την open η οποία ανοίγει ένα νέο έγγραφο και τη write που γράφει μέσα στο έγγραφο.



Εικόνα 7 - Ιεραρχική δομή αντικειμένων στη JavaScript

Για η παράδειγμα προσπέλαση της τιμής του πεδίου κειμένου text1 γίνεται ως ακολούθως: window.document.form.text1.value

Βασίζεται στη δημιουργία συναρτήσεων οι οποίες σχεδιάζονται με στόχο να επιτελούν κάποια συγκεκριμένη λειτουργία και καλούνται μέσα από HTML έγγραφα κατόπιν πραγματοποίησης γεγονότων, πχ. onLoad, onClick κτλ., σε αντικείμενα της σελίδας. Επιπλέον, η JavaScript υποστηρίζει και δικές της συναρτήσεις όπως η alert.

Η JavaScript δεν απαιτεί κάποιο εξειδικευμένο λογισμικό και το μόνο που χρειάζεται είναι ένας επεξεργαστής κειμένου για την δημιουργία του κώδικα και ένας φυλλομετρητής για την εκτέλεση του κώδικα.

Εισαγωγή JavaScript κώδικα μπορεί να επιτευχθεί με δύο τρόπους. (α) Απευθείας εισαγωγή στο HTML έγγραφο μεταξύ των ετικετών <script></script>. Για παράδειγμα:

```
<script type="text/javascript">
    document.write("Hello World");
</script>
```

(β) Δημιουργία ξεχωριστού αρχείου με κατάληξη ".js" το οποίο μπορεί να συμπεριληφθεί μέσα από το HTML έγγραφο. Για παράδειγμα:

myfile.js:

```
function popup()
{
    alert("Hello World");
}
<script type="text/javascript" src="myfile.js"></script>
```

Ο κώδικας JavaScript και κατ' επέκταση οι ετικέτες <script></script> που τον περικλείουν εισάγονται μεταξύ των ετικετών <head></head> ή εναλλακτικά μεταξύ των ετικετών <body></body>.

Μερικά από τα χαρακτηριστικά της δομής της JavaScript είναι ότι υπάρχει διάκριση ανάμεσα στα πεζά και τα κεφαλαία γράμματα (case sensitive), υπάρχει ευελιξία στις δηλώσεις των μεταβλητών, τα άγκιστρα ({ και }) χρησιμοποιούνται για να ομαδοποιούν εντολές, το ερωτηματικό στο τέλος κάθε εντολής δεν είναι απαραίτητο και οι μεταβλητές ορίζονται με χρήση της δήλωσης var ή και χωρίς.

3.4.1 Η βιβλιοθήκη jQuery

Το jQuery είναι μια βιβλιοθήκη που περιλαμβάνει καλογραμμένο και συμπαγή κώδικα JavaScript. Αποτελεί ελεύθερο λογισμικό/ανοικτού κώδικα και μπορεί ο καθένας να κατεβάσει το αρχείο από την επίσημη ιστοσελίδα (http://jquery.com). Μεταξύ των χρηστών του jQuery συγκαταλέγονται και γνωστές επιφανείς εταιρίες όπως οι IBM, Netflix, Amazon, Dell, Twitter και Bank of America.

Με το jQuery απλοποιείται η διαχείριση εγγράφων HTML, η διαχείριση γεγονότων, η εισαγωγή κίνησης (animation) και οι αλληλεπιδράσεις AJAX (Asynchronous JavaScript and XML) με απώτερο στόχο την ταχεία ανάπτυξη πλούσιων αλληλεπιδραστικών εφαρμογών στο web χωρίς τη χρήση εκτεταμένου κώδικα.

Μερικά επιπλέον πλεονεκτήματα που προσφέρει είναι ότι βοηθά στην βελτίωση της απόδοσης της εφαρμογής, βοηθά στην ανάπτυξη ιστοσελίδων με συμβατότητα ως προς τους φυλλομετρητές, βοηθά στην ανάπτυξη διαδραστικών εφαρμογών χωρίς την απαίτηση για συγγραφή μεγάλου όγκου κώδικα, είναι γρήγορο, είναι επεκτάσιμο, βασίζεται στο συντακτικό της JavaScript και περιέχει απλό και καθαρό κώδικα.

Για τη ενσωμάτωση και χρήση του jQuery σε μια εφαρμογή το μόνο που απαιτείται είναι εισαγωγή του αντίστοιχου ".js" αρχείου στο HTML έγγραφο σύμφωνα με τον παρακάτω κώδικα:

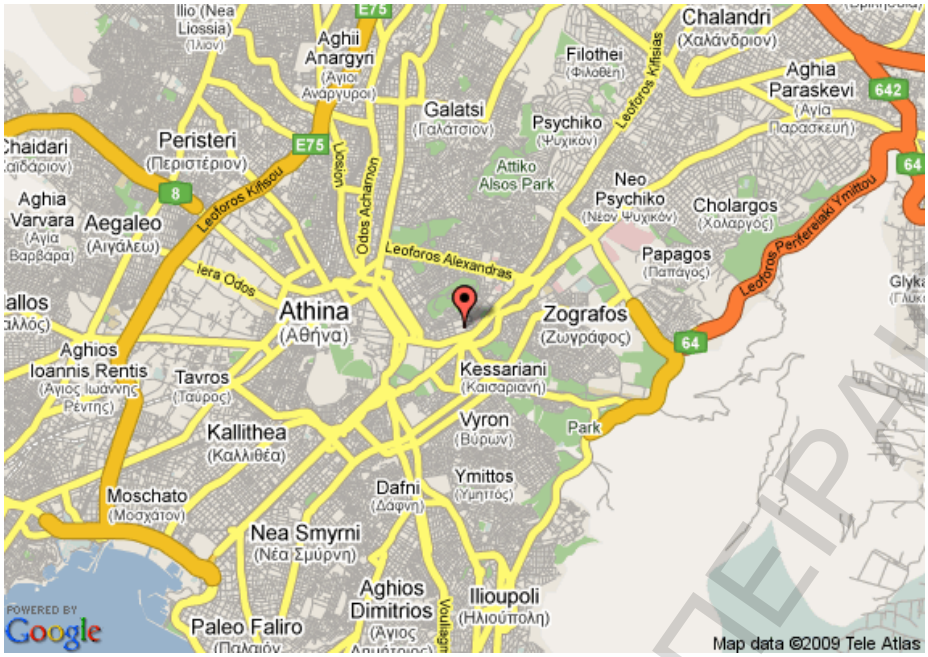
```
<script type="text/javascript" src="jquery-1.7.2.min.js"></script>
```

Πέρα από τη βασική (core) βιβλιοθήκη έχει επεκταθεί με μια συνοδευτική βιβλιοθήκη, το jQuery UI, το οποίο παρέχει τα απαραίτητα εργαλεία για την κατασκευή γραφικών διεπαφών για χρήστες (user interfaces). Επίσης, έχει πλαισιωθεί από ένα μεγάλο πλήθος επεκτάσεων (extensions) και προσθέτων (plugins) τόσο επίσημα όσο και ανεπίσημα. Ένα επίσημο πρόσθετο είναι το jQuery form plugin το οποίο προσφέρει εύκολη διαχείριση των φορμών HTML κατά τη χρήση AJAX. Οι κύριες μέθοδοι που περιλαμβάνει, ajaxForm και ajaxSubmit, συλλέγουν πληροφορίες από το στοιχείο form ώστε να προσδιορίσουν το πώς θα διαχειριστούν την διαδικασία υποβολής της φόρμας. Και οι δύο αυτές μέθοδοι υποστηρίζουν ένα μεγάλο πλήθος επιλογών που επιτρέπουν τον πλήρη έλεγχο στο πώς υποβάλλονται τα δεδομένα. Μπορεί κάποιος να το κατεβάσει από την παρακάτω ιστοσελίδα (http://jquery.malsup.com/form/#getting-started) και να το εισαγάγει με τον παρακάτω κώδικα

```
<script type="text/javascript" src="jquery.form.js"></script>
```

3.4.2 Η προγραμματιστική διεπαφή Google Maps JavaScript v3

Το Google Maps είναι εργαλείο που προσφέρει υπηρεσίες εμφάνισης χαρτών στο web και παρέχεται από την Google. Μπορεί να ενσωματωθεί σε μία ιστοσελίδα με χρήση της προγραμματιστικής διεπαφής των Google Maps. Ένα Google Map μπορεί να χρησιμοποιηθεί για παράδειγμα ως εργαλείο πλοήγησης για την εύρεση και εμφάνιση κάποιας τοποθεσίας ενδιαφέροντος ή για την εύρεση κάποιας διαδρομής με το λιγότερο δυνατό κόστος.



Εικόνα 8 - Google Map

Τα Google Maps συνίστανται από δύο μέρη:

- **Το Google Map**: ουσιαστικά πρόκειται για τον ίδιο το χάρτη, ο οποίος μπορεί να ρυθμιστεί κατάλληλα για την εμφάνιση οποιασδήποτε τοποθεσίας στη γη σε μια πληθώρα τύπων χάρτη.

- **Τη τοποθεσία στο Google Map**: ουσιαστικά παρέχεται η δυνατότητα δημιουργίας σημείου (marker) επάνω στην επιθυμητή τοποθεσία στο χάρτη.

Τα Google Maps προσφέρουν πλήθος χαρακτηριστικών όπως πολλαπλούς τύπους χαρτών και πλήκτρων ελέγχου, έλεγχο κλίμακας, εμφάνιση οδών (street view), το Google Earth, υπηρεσία γεωκωδικοποίησης, σημεία τοποθεσίας και παραμετροποιήσιμα εικονίδια τοποθεσίας.

Για τη χρήση και εκμετάλλευση των δυνατοτήτων των Google Maps απαραίτητη προϋπόθεση είναι η ενσωμάτωσή τους σε κάποια ιστοσελίδα. Αυτό μπορεί να επιτευχθεί μέσω της προγραμματιστικής διεπαφής, Google Maps JavaScript API. Η τελευταία επίσημη έκδοση είναι η version 3 η οποία είναι ειδικά σχεδιασμένη για να προσφέρει μεγαλύτερη ταχύτητα και να είναι πιο εφαρμόσιμη τόσο σε εφαρμογές κινητών συσκευών όσο και σε παραδοσιακές εφαρμογές γραφείου. Προσφέρει μια γκάμα από υπηρεσίες για το χειρισμό χαρτών και την προσθήκη περιεχομένου στο χάρτη, επιτρέποντας τη δημιουργία εύρωστων εφαρμογών με χάρτες στο web.

Είναι προφανές ότι αυτή η προγραμματιστική διεπαφή απαιτεί από τους προγραμματιστές μια οικειότητα με την JavaScript και με θέματα που άπτονται του αντικειμενοστρεφούς προγραμματισμού. Ωστόσο, προσφέρει πλήρη τεκμηρίωση στην παρακάτω ιστοσελίδα (https://developers.google.com/maps/documentation/javascript/)

Τα βήματα που απαιτούνται για την εμφάνιση του Google Map στην ιστοσελίδα είναι η παραλαβή ενός Google Maps API κλειδιού, η δημιουργία μιας σωστά δομημένης HTML σελίδας, η φόρτωση των απαραίτητων JavaScript βιβλιοθηκών (*), η συγγραφή JavaScript

κώδικα για την εμφάνιση του χάρτη και η συγγραφή επιπρόσθετου κώδικα ώστε ο χάρτης να προσφέρει κάτι χρήσιμο προς τους επισκέπτες.

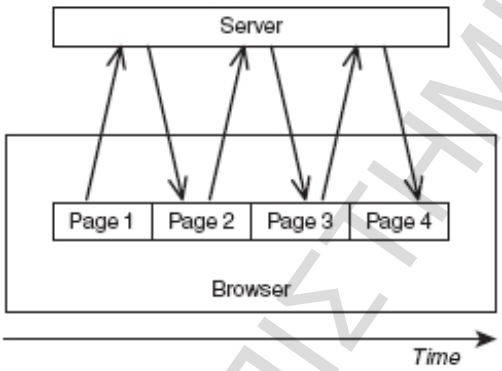
(*) Για την φόρτωση του Google Maps API απαιτείται η εισαγωγή του παρακάτω κώδικα στην ετικέτα <head> του HTML εγγράφου:

```
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=SE
T_TO_TRUE_OR_FALSE"></script>
```

3.4.3 Η τεχνική Ajax

Η τεχνική AJAX (Asynchronous JavaScript and XML) είναι ένας συνδυασμός από σενάρια κώδικα τόσο από τη μεριά του πελάτη (client-side scripting) όσο και από τη μεριά του εξυπηρετητή (server-side scripting) συνδυάζοντας τεχνολογίες όπως JavaScript, DHTML και ένα αντικείμενο που ονομάζεται XmlHttpRequest. Αποτελεί πλέον τη ραχοκοκαλιά του Web 2.0 και αποτελεί το μέσο παραγωγής πλούσιων διαδικτυακών εφαρμογών (Rich Internet Applications - RIAs). Επιπρόσθετα, βελτιώνει σημαντικά την εμπειρία των χρηστών προσφέροντας δυναμικές διεπαφές χρηστών και μειώνει το χρόνο φόρτωσης των σελίδων. Όλοι οι φυλλομετρητές στις τελευταίες τους εκδόσεις υποστηρίζουν AJAX.

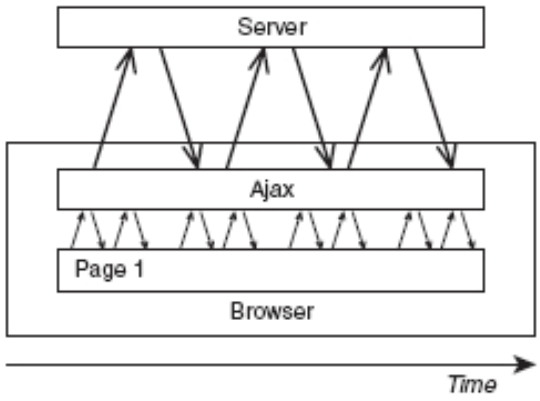
Στο παραδοσιακό μοντέλο πελάτη – εξυπηρετητή, ο πελάτης εκτελεί κάποια ενέργεια (για παράδειγμα υποβολή μιας φόρμας) και περιμένει, καθώς ο φυλλομετρητής ανανεώνεται, μια νέα ή αναθεωρημένη σελίδα που στέλνει ο εξυπηρετητής.



Εικόνα 9 - Παραδοσιακή επικοινωνία πελάτη - εξυπηρετητή

Στο νέο όμως μοντέλο που βασίζεται σε AJAX, δημιουργείται ένα επιπρόσθετο επίπεδο επεξεργασίας μεταξύ του πελάτη και του εξυπηρετητή. Αυτό το επίπεδο καλείται ως μηχανή Ajax (Ajax engine) ή πλαίσιο εργασίας Ajax (Ajax framework) και λαμβάνει αιτήσεις από τον πελάτη καθώς στο παρασκήνιο χειρίζεται τις αλληλεπιδράσεις (επικοινωνία) με τον εξυπηρετητή με διακριτικό τρόπο και ασύγχρονα. Αποτελεί τον συνδετικό κρίκο μεταξύ πελάτη και εξυπηρετητή.

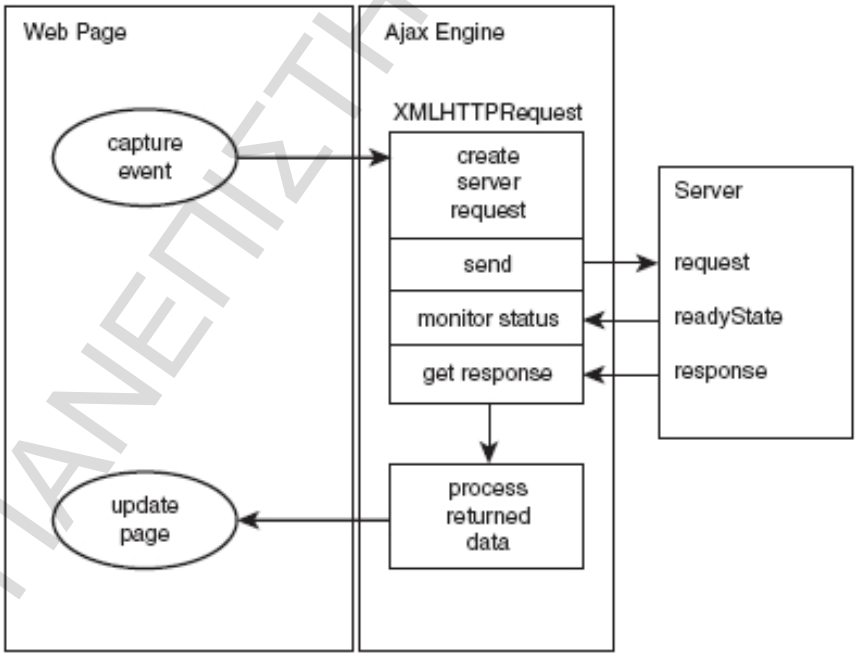
Μέσω της τεχνικής AJAX η επικοινωνία με τον εξυπηρετητή (αίτηση - απάντηση) γίνεται ασύγχρονα που σημαίνει ότι μέχρι να απαντήσει ο εξυπηρετητής ο χρήστης μπορεί να συνεχίσει την πλοήγησή του, επιτελώντας και άλλες ενέργειες, κανονικά και χωρίς διακοπές. Επιπλέον, με το που θα απαντήσει ο εξυπηρετητής η σελίδα ενημερώνεται κατάλληλα χωρίς επαναφόρτωσή της.



Εικόνα 10 - Επικοινωνία πελάτη – εξυπηρετητή μέσω AJAX

Η καρδιά του AJAX βρίσκεται στο αντικείμενο XmlHttpRequest (XHR). Η απάντηση (response) έρχεται με δύο τρόπους, ως responseXML όπου επιστρέφεται ένα έγγραφο DOM (μπορούν να χρησιμοποιηθούν συναρτήσεις όπως η getElementById) και ως responseText όπου επιστρέφεται μία ακολουθία χαρακτήρων (HTML ή JavaScript κώδικας). Δεν χρειάζεται να χρησιμοποιήσουμε οπωσδήποτε XML, πιο αποδοτικό είναι πλέον το JSON.

Τελικά, ο κύκλος ζωής μιας εφαρμογής που χρησιμοποιεί AJAX έχει ως εξής. Αρχικά ο χρήστης επισκέπτεται την ιστοσελίδα, περιμένει να φορτώσει η σελίδα και πραγματοποιεί στη συνέχεια κάποιο συμβάν. Με την ενεργοποίηση του συμβάντος δημιουργείται αίτηση από τον πελάτη προς τον εξυπηρετητή και κατόπιν επεξεργασίας της αίτησης έρχεται η απάντηση από τον εξυπηρετητή προς τον πελάτη και ενημερώνεται η ιστοσελίδα.



Εικόνα 11 - Διάγραμμα του πώς λειτουργεί μια εφαρμογή ιστού με AJAX

Γνωστές εφαρμογές που κάνουν χρήση AJAX είναι το Gmail, τα Google Maps, το Google suggest, τα Google News / igoogle / netvibes και το web chat.

Πλέον, είναι διαδεδομένα αρκετά πλαίσια εργασίας (frameworks) προγραμματισμού με AJAX που χρησιμοποιούνται για τη διευκόλυνση χρήσης του AJAX. Μερικά από αυτά είναι τα jQuery, DOJO, Scriptaculous, Prototype, κ.α. Εστιάζουν στη μεριά του πελάτη, παρέχοντας εύκολους τρόπους προσθήκης οπτικών εφέ στις ιστοσελίδες και μια υψηλού επιπέδου προσέγγιση στο AJAX. Για παράδειγμα η βιβλιοθήκη jQuery προσφέρει μια πλήρη σουίτα από δυνατότητες AJAX. Οι συναρτήσεις και οι μέθοδοι που περιλαμβάνει επιτρέπουν την φόρτωση δεδομένων από τον εξυπηρετητή χωρίς να γίνεται ανανέωση σελίδας. Η ασύγχρονη HTTP (Ajax) αίτηση επιτυγχάνεται μέσω της jQuery.ajax(url[, settings]). Για παράδειγμα:

```
$.ajax({
    url: “test.php”,
    type: “POST”,
    data: { name: “John”, location: “Boston”}
}).done(function(msg) {
    alert(“Data saved: ” + msg);
});
```

Στο παραπάνω παράδειγμα, στέλνεται μια αίτηση AJAX με κάποια δεδομένα στο αρχείο test.php μέσω της μεθόδου POST και όταν εξυπηρετηθεί επιτυχώς από τον εξυπηρετητή και φτάσει πίσω η απάντηση τότε φροντίζει η συνάρτηση επιστροφής κλήσης (callback) η οποία εκτελείται και έχει ως αποτέλεσμα την εμφάνιση στον χρήστη ενός πλαισίου με κάποιο μήνυμα το οποίο περιλαμβάνει και δεδομένα που επιστράφηκαν.

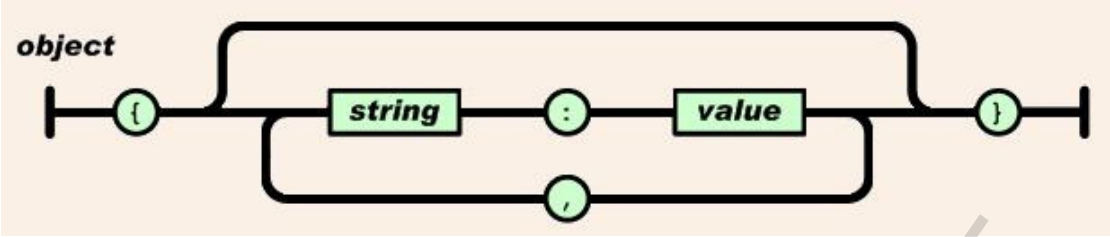
3.4.4 Το πρότυπο JSON

Η JSON (JavaScript Object Notation) αποτελεί ένα πρότυπο ανταλλαγής δεδομένων και μορφοποίησης δεδομένων σε JavaScript. Η μορφή της είναι εξαιρετικά απλή, τυποποιημένη, ιεραρχική και ανεξάρτητη από τη γλώσσα προγραμματισμού ή την πλατφόρμα. Η JSON αποτελεί μια καλή εναλλακτική λύση σε σχεση με την XML παρόλο που δε μπορεί να την αντικαταστήσει πλήρως. Αυτό συμβαίνει επειδή δεν υποστηρίζει επικύρωση σχήματος (schema validation), δεν μπορεί από μόνη της να ενημερώσει σχετικά με την κωδικοποίησή της και δε χρησιμοποιεί την έννοια των ιδιοτήτων. Όπως και η XML, η JSON έχει αυτό-τεκμηριωμένη (self-documented) μορφή που περιγράφει την δομή των δεδομένων και δεν ασχολείται με την παρουσίασή τους.

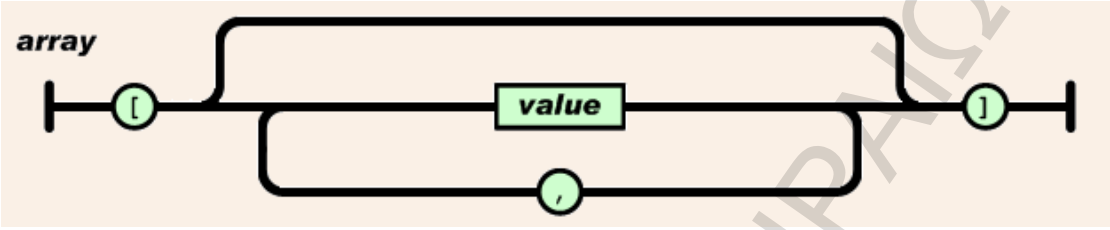
Το κύριο πλεονέκτημα της JSON σε σχέση με την XML είναι ότι είναι συμπαγής. Επιπλέον, στα πλαίσια του web έχει ένα ακόμη πλεονέκτημα: έχει 100% έγκυρο JavaScript κώδικα και είναι πολύ εύκολη η μετατροπή κειμένου σε δεδομένα JavaScript. Από την άλλη είναι ευανάγνωστη, κατατοπιστική με μια ματιά και εν τέλει κατανοητή από τους ανθρώπους.

Η JSON βασίζεται σε δύο δομές, (α) μία συλλογή με ζεύγη ονομάτων/τιμών (name/value). Σε διάφορες γλώσσες προγραμματισμού αυτό συνδέεται με έννοιες όπως, αντικείμενο, εγγραφή, δομή, λεξικό, συσχετιζόμενος πίνακας, κτλ. και (β) μια ταξινομημένη λίστα με τιμές. Στις περισσότερες γλώσσες προγραμματισμού αυτό συνδέεται με έννοιες όπως, πίνακας, λίστα, διάνυσμα, κτλ.

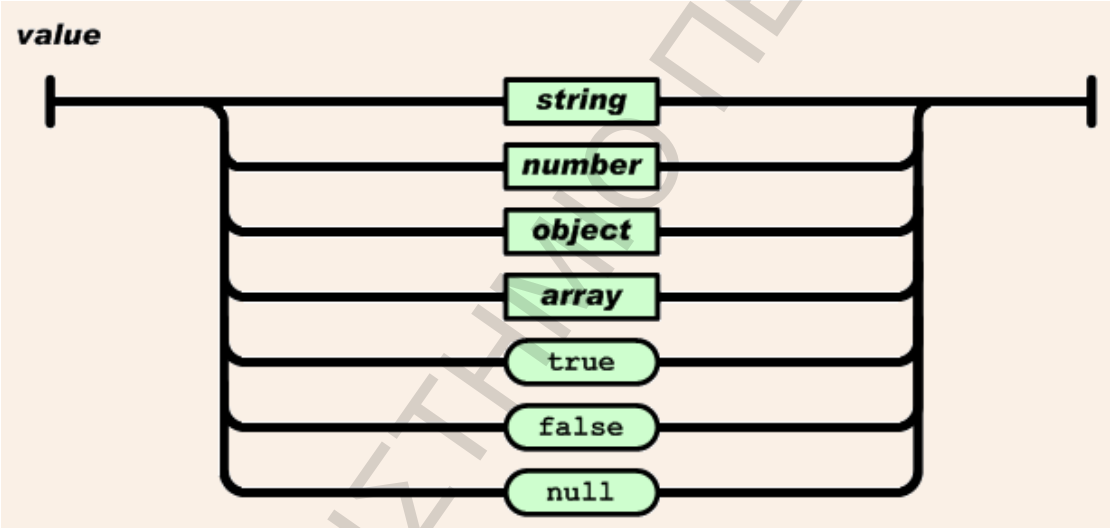
Υποστηρίζεται από όλες τις σύγχρονες γλώσσες προγραμματισμού όπως είναι η C/C++, C#, Java, Perl, PHP, Python, κ.α. Για παράδειγμα στην PHP υπάρχει η συνάρτηση json_encode η οποία επιστρέφει την αναπαράσταση σε JSON της τιμής που λαμβάνει ως παράμετρο.



Εικόνα 12 - Δομή αντικειμένου σε JSON



Εικόνα 13 - Δομή πίνακα σε JSON



Εικόνα 14 - Δομή τιμής σε JSON

3.5 Η συλλογή Twitter Bootstrap

Το Twitter Bootstrap είναι μία συλλογή (front-end) εργαλειοθηκών (toolkit) για γρήγορη ανάπτυξη web εφαρμογών. Περιλαμβάνει CSS και HTML για τη δημιουργία πλεγμάτων (grids), σχεδίων (HTML templates), τυπογραφιών (typography), πινάκων (tables), φορμών (forms), μενού (για πλοήγηση), πλαισίων (για λάθη, προειδοποιήσεις, κτλ.) και άλλων εξαρτημάτων για διεπαφές (interfaces). Όλα αυτά προσφέρονται μέσα από ένα πολύ μικρό αρχείο το οποίο μπορεί κάποιος να το κατεβάσει από την επίσημη ιστοσελίδα (http://twitter.github.com/bootstrap). Το αρχείο αυτό περιλαμβάνει αρχεία CSS (bootstrap.css, bootstrap.min.css, bootstrap-responsive.css, bootstrap-responsive.min.css), JavaScript αρχεία (bootstrap.js, bootstrap.min.js) και κάποιες εικόνες.

Το Bootstrap αναπτύχθηκε από τον Mark Otto και τον Jacob Thornton ως πλαίσιο εργασίας με στόχο την προώθηση της συμβατότητας μεταξύ των διαδικτυακών εργαλείων. Πριν το Bootstrap, διάφορες βιβλιοθήκες χρησιμοποιούνταν για ανάπτυξη διεπαφών, γεγονός που οδήγησε γρήγορα σε ασυμβατότητες με ένα επακόλουθο υψηλό φορτίο συντήρησης. Τον

Αύγουστο του 2011 η Twitter κυκλοφόρησε το Bootstrap ως ελεύθερο και ανοικτού κώδικα λογισμικό. Από τον Φεβρουάριο του 2012 αποτελεί το πιο δημοφιλές πρότζεκτ στην ομάδα GitHub. Μεταξύ άλλων, χρησιμοποιείται από τη NASA και την MSNBC.

Ο πυρήνας του Twitter Bootstrap παρόλο που είναι CSS έχει δομηθεί με κανόνες στύλ Less, έναν ευέλικτο προ-επεξεργαστή που προσφέρει περισσότερη δύναμη και ευελιξία από ό,τι το CSS. Με το Less προκύπτει ένα ευρύ πλήθος χαρακτηριστικών όπως ένθετες δηλώσεις, μεταβλητές, «mixins» και διάφορες λειτουργίες-συναρτήσεις.

```
@base: #f938ab;

.box-shadow(@style, @c) when (iscolor(@c)) {
  box-shadow:         @style @c;
  -webkit-box-shadow: @style @c;
  -moz-box-shadow:    @style @c;
}
.box-shadow(@style, @alpha: 50%) when (isnumber(@alpha)) {
  .box-shadow(@style, rgba(0, 0, 0, @alpha));
}
.box {
  color: saturate(@base, 5%);
  border-color: lighten(@base, 30%);
  div { .box-shadow(0 0 5px, 30%) }
}
```

Εικόνα 15 - Κανόνες στυλ Less

Το Twitter Bootstrap έχει δημιουργηθεί ώστε να έχει πλήρη συμβατότητα με όλους τους μοντέρνους περιηγητές παρόλο που μέχρι στιγμής έχει περιορισμένη υποστήριξη για HTML 5 και CSS 3. Από την version 2 και μετά υποστηρίζει τη δυναμική διάταξη των ιστοσελίδων που σημαίνει ότι το στατικό – γραφικό μέρος των ιστοσελίδων προσαρμόζεται δυναμικά λαμβάνοντας υπόψη τα χαρακτηριστικά της συσκευής που το προβάλλει (PC, tablet, smart phone).

Για την χρήση του Twitter Bootstrap το μόνο που χρειάζεται είναι απλά η εισαγωγή στο <head></head> μέρος της HTML ιστοσελίδας του παρακάτω κώδικα:

```
<link href="bootstrap/css/bootstrap.css" rel="stylesheet"
type="text/css" />
```

3.6 Αντικειμενοστρεφής προσέγγιση

Οι άνθρωποι στην καθημερινότητά τους σκέφτονται με βάση τα αντικείμενα. Όπου και αν κοιτάξει κανείς βλέπει αντικείμενα όπως για παράδειγμα αυτοκίνητα, κτίρια, υπολογιστές, τηλέφωνα, σπίτια κτλ. Έτσι λοιπόν και τα αντικειμενοστρεφή προγράμματα συνίστανται από δομικά στοιχεία, τα αντικείμενα.

Τα αντικείμενα χαρακτηρίζονται από κάποια κοινά πράγματα. Όλα έχουν χαρακτηριστικά (attributes) όπως για παράδειγμα σχήμα, μέγεθος, χρώμα, βάρος κτλ. και επιδεικνύουν διάφορες συμπεριφορές (behaviors) όπως για παράδειγμα αναπήδηση μπάλας, επιτάχυνση αυτοκινήτου, κλήση τηλεφώνου κτλ. Το ίδιο συμβαίνει αντίστοιχα και με τα αντικειμενοστρεφή προγράμματα.

3.6.1 Αντικειμενοστρεφής ανάλυση & σχεδιασμός

Η αντικειμενοστρεφής ανάλυση και σχεδιασμός (Object Oriented Analysis & Design – OOAD) ξεκινάει αρχικά με την ανάλυση των απαιτήσεων ενός έργου καθορίζοντας το «τι» θα κάνει το

σύστημα και συνεχίζει με το σχεδιασμό που ικανοποιεί τις απαιτήσεις, αποφασίζοντας το «πώς» θα το κάνει το σύστημα. (Πολλές φορές πραγματοποιείται αναθεώρηση του σχεδιασμού με σκοπό το βέλτιστο αποτέλεσμα και μετά ακολουθεί η υλοποίηση με κώδικα).

Η γλώσσα μοντελοποίησης (με γραφικό τρόπο) που χρησιμοποιείται ευρέως για αυτό τον σκοπό είναι η UML (Unified Modeling Language) η οποία αναπτύχθηκε στα μέσα της δεκαετίας του 1990 από τους Grady Booch, James Rumbaugh και Ivar Jacobson. Αποτελεί το βιομηχανικό πρότυπο που μας επιτρέπει να συνδέουμε αρμονικά και καθαρά τις απαιτήσεις, τις αρχιτεκτονικές και τον σχεδιασμό.

Η αντικειμενοστρεφής σχεδίαση παρέχει ένα φυσικό και διαισθητικό τρόπο σε κάποιον που θέλει να δει την διαδικασία σχεδίασης, μοντελοποιώντας τα χαρακτηριστικά και τις συμπεριφορές και ενθυλακώνοντάς τα σε αντικείμενα. Επίσης μοντελοποιεί την επικοινωνία μεταξύ των αντικειμένων.

3.6.2 Αντικειμενοστρεφής προγραμματισμός

Ο αντικειμενοστρεφής προγραμματισμός (Object Oriented Programming - OOP) αποτελεί μια μοντέρνα και συγκεκριμένη προσέγγιση ανάπτυξης και υλοποίησης προγραμμάτων, όπου μπορούμε να θεωρήσουμε τα στοιχεία του προγράμματος ως αντικείμενα. Οι έννοιες κλάση και αντικείμενο είναι βασικές στον αντικειμενοστρεφή προγραμματισμό. Η κλάση είναι μια γενική φόρμα για την κατασκευή αντικειμένων που έχουν παρόμοια χαρακτηριστικά. Θα μπορούσαμε να πούμε ότι είναι μια «συνταγή» δημιουργίας αντικειμένων. Αφού ορίσουμε μια κλάση, στη συνέχεια μπορούμε να χρησιμοποιήσουμε τον ορισμό της (τη «συνταγή») για να δημιουργήσουμε αντικείμενα. Κάθε αντικείμενο λέγεται και στιγμιότυπο (instance) μιας κλάσης. Παίρνοντας το πρώτο παράδειγμα από τη ζωή μας, θα μπορούσαμε να θεωρήσουμε ως κλάση την έννοια «δένδρο» και ως αντικείμενα της κλάσης τη «λεμονιά», τη «μηλιά», το «έλατο» κτλ. Όλα τα αντικείμενα έχουν τα γενικά χαρακτηριστικά της κλάσης «δένδρο» (ρίζα, κορμό, κλαδιά, φύλλα), αλλά το καθένα έχει και κάποιες ιδιαίτερες ιδιότητες, όπως το είδος του καρπού, το σχήμα, το αν είναι φυλλοβόλο ή όχι, κτλ.

Δύο είναι τα στοιχεία που συμπεριλαμβάνονται στον ορισμό μιας κλάσης, οι μεταβλητές (variables) οι οποίες διαφοροποιούν το ένα αντικείμενο της κλάσης από το άλλο και οι μέθοδοι (methods), οι οποίες καθορίζουν τι μπορούμε να κάνουμε στα (ή με τα) αντικείμενα μιας κλάσης. Η μέθοδος είναι μια ομάδα από δηλώσεις και εντολές, η οποία εκτελεί μια εξειδικευμένη λειτουργία του αντικειμένου.

Μία κλάση παρουσιάζει μια συγκεκριμένη εικόνα «προς τα έξω». Αυτή η εικόνα εμφανίζει τις λειτουργίες (μεθόδους) που επιτελεί η κλάση καθώς και τα δεδομένα που χρησιμοποιεί. Οτιδήποτε άλλο συμβαίνει μέσα στην κλάση, όπως για παράδειγμα οι λεπτομερείς ενέργειες που γίνονται μέσα στις μεθόδους, παραμένει κρυφό.

Τα σημαντικότερα χαρακτηριστικά που διέπουν τον αντικειμενοστρεφή προγραμματισμό αποτελούν η αφαίρεση (abstraction), η ενθυλάκωση (encapsulation) και απόκρυψη δεδομένων (data hiding), ο πολυμορφισμός (polymorphism), η κληρονομικότητα (inheritance) και η επαναχρησιμοποίηση του κώδικα (reusability of code).

Μετά από την ανάλυση και τον σχεδιασμό, ακολουθεί η υλοποίηση της εφαρμογής με κώδικα (εδώ εμπίπτει ο αντικειμενοστρεφής προγραμματισμός). Μερικές αντικειμενοστρεφείς γλώσσες προγραμματισμού (ή που υποστηρίζουν την αντικειμενοστραφή προσέγγιση) είναι οι Java, C++, C#, PHP.

3.7 Η γλώσσα PHP

Η PHP (Hypertext Preprocessor) είναι μια γενικού σκοπού ισχυρή γλώσσα προγραμματισμού που εκτελείται από τη πλευρά του εξυπηρετητή (server-side) και σχεδιάστηκε για την ανάπτυξη εφαρμογών web δυναμικού περιεχομένου. Η PHP πλεονεκτεί έναντι ανταγωνιστών της όπως είναι η ASP της Microsoft λόγω του ότι είναι ισχυρό και ελεύθερο λογισμικό και επιπλέον είναι

εύκολη στην εκμάθηση της. Η PHP αρχικά χρησιμοποιήθηκε σε κεντρικούς υπολογιστές με περιβάλλον UNIX, αλλά τώρα χρησιμοποιείται και σε άλλες πλατφόρμες λειτουργικών.

Οι δυναμικές ιστοσελίδες, που χρησιμοποιούν προγραμματισμό από τη πλευρά του εξυπηρετητή, λαμβάνουν τα δεδομένα που αποστέλλει ο χρήστης στον εξυπηρετητή, τα επεξεργάζονται ή ανατρέχουν σε κάποια βάση δεδομένων και ως αποτέλεσμα δημιουργούν μια προσαρμοσμένη ιστοσελίδα την οποία αποστέλλουν στο χρήστη. Για παράδειγμα, όταν ο χρήστης επισκεφτεί μια ιστοσελίδα που περιέχει κώδικα PHP, ο εξυπηρετητής εκτελεί τον κώδικα και αυτό που επιστρέφει είναι μια σελίδα HTML. Ο φυλλομετρητής δε χρειάζεται κάποιο πρόσθετο πρόγραμμα ή εργαλείο για να εμφανίσει το αποτέλεσμα της PHP.

Η PHP προσφέρει πλήθος δυνατοτήτων. Πιο συγκεκριμένα, υποστηρίζει την επικοινωνία με πολλές βάσεις δεδομένων, όπως MySQL, PostgreSQL, Oracle και άλλες, παίρνει πληροφορίες από φόρμες και τις επεξεργάζεται ή τις αποθηκεύει σε βάσεις δεδομένων, ελέγχει την πρόσβαση των χρηστών στις ιστοσελίδες, φιλοξενεί συζητήσεις (forums) και διαχειρίζεται σελίδες XML.

Το όνομα του αρχείου ενός προγράμματος PHP όπως επίσης και το όνομα μιας σελίδας HTML που περιλαμβάνει κώδικα PHP τελειώνουν υποχρεωτικά με την κατάληξη ".php". Κάθε τμήμα κώδικα σε PHP ξεκινά με <?php και τελειώνει με ?>. Για παράδειγμα:

```
<?php
echo "Hello World!";
?>
```

Η PHP παρέχει μια βιβλιοθήκη με έτοιμες συναρτήσεις και επιπλέον δίνει τη δυνατότητα για δημιουργία νέων συναρτήσεων. Επιπλέον, υποστηρίζει τη δημιουργία κλάσεων λόγω του ότι από την PHP 5 και μετά έχει δοθεί τεράστια βαρύτητα στην αντικειμενοστραφή προσέγγιση της γλώσσας. Επίσης, αποτελεί ένα εύχρηστο εργαλείο για επικοινωνία των ιστοσελίδων με βάσεις δεδομένων. Για το σκοπό αυτό παρέχονται έτοιμες συναρτήσεις για τη σύνδεση στον εξυπηρετητή της βάσης, την επιλογή μιας βάσης δεδομένων, την εκτέλεση ενός SQL ερωτήματος στη βάση που συνδέθηκε και πολλές άλλες ακόμα.

Κυκλοφορεί ένα μεγάλο πλήθος από πλαίσια εργασίας PHP (frameworks) με τα CodeIgniter, Zend, Yii και CakePHP να ξεχωρίζουν.

3.8 Το σχεδιαστικό πρότυπο MVC

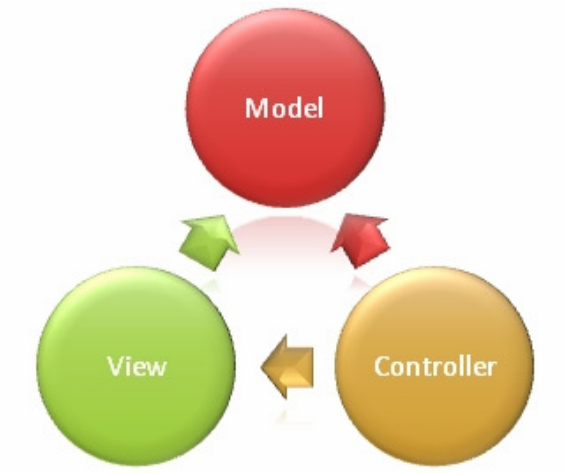
Το MVC (Model-View-Controller) είναι ένα σχεδιαστικό πρότυπο (design pattern) που διαχωρίζει την λογική της εφαρμογής από την παρουσίαση. Ορίζει τρία λογικά επίπεδα, το επιχειρηματικό επίπεδο (model logic), το επίπεδο εμφάνισης (view logic) και το επίπεδο ελέγχου της εισόδου (controller logic).

Το **μοντέλο (model)** είναι το κομμάτι της εφαρμογής που είναι υπεύθυνο με τη λογική των δεδομένων της εφαρμογής. Συχνά, μέσα από τα μοντέλα ανακτούνται και αποθηκεύονται δεδομένα στη βάση δεδομένων.

Η **όψη (view)** είναι το κομμάτι της εφαρμογής που είναι υπεύθυνο με την εμφάνιση/παρουσίαση των δεδομένων. Συχνά, οι όψεις δημιουργούνται από τα δεδομένα του μοντέλου.

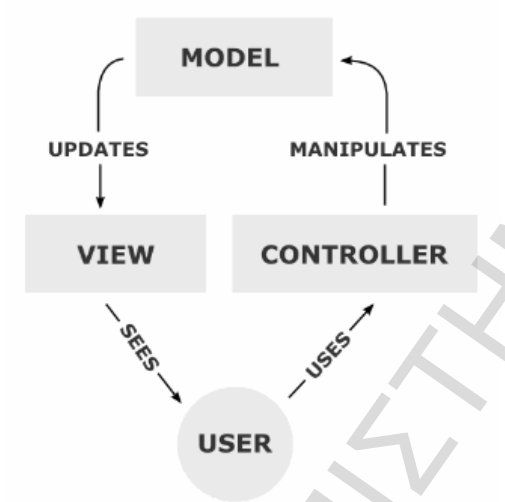
Ο **ελεγκτής (controller)** είναι το κομμάτι της εφαρμογής που είναι υπεύθυνο για τη διαχείριση της αλληλεπίδρασης του χρήστη. Τυπικά, οι ελεγκτές διαβάζουν δεδομένα από τις όψεις, ελέγχουν την είσοδο του χρήστη και στέλνουν τα δεδομένα εισόδου στο μοντέλο. Κάποιες φορές όμως, σε πιο απλοποιημένες προσεγγίσεις, στέλνουν τα δεδομένα κατευθείαν πίσω στην όψη αλλάζοντας τη. Συνεπώς, ο ελεγκτής αποτελεί το συνδετικό κρίκο μεταξύ του μοντέλου, της όψης και οποιουδήποτε άλλου πόρου απαιτείται για την επεξεργασία της αίτησης HTTP και την παραγωγή της σελίδας.

Λόγω αυτής της οργάνωσης των τριών επιπέδων, πολλές όψεις και πολλοί ελεγκτές μπορούν να αλληλεπιδρούν με το ίδιο το μοντέλο.



Εικόνα 16 - MVC

Ο τρόπος λειτουργίας του MVC φαίνεται στην εικόνα 17.



Εικόνα 17 - Τρόπος λειτουργίας του MVC

Ο τελικός χρήστης αλληλεπιδρά με τον ελεγκτή ο οποίος διαχειρίζεται το μοντέλο στέλνοντάς του δεδομένα. Το μοντέλο με τη σειρά του ενημερώνει την όψη τροφοδοτώντας τη με δεδομένα και η όψη στη συνέχεια αφού μορφοποιήσει κατάλληλα τα δεδομένα, με κανόνες στυλ, τα εμφανίζει στον τελικό χρήστη.

Η κεντρική ιδέα πίσω από το πρότυπο MVC είναι η επαναχρησιμοποίηση του κώδικα (reusability) και η τμηματοποίηση-διαχωρισμός που συντελείται με έναν ορθολογικό καταμερισμό των διάφορων λειτουργιών που λαμβάνουν μέρος σε μια εφαρμογή. Το MVC παρέχει πλήρη έλεγχο σε τεχνολογίες όπως HTML, CSS, JavaScript, PHP κ.α. Επίσης, έχει πλέον υιοθετηθεί ως αρχιτεκτονική για εφαρμογές web. Αρκετά εμπορικά και μη εμπορικά πλαίσια εργασίας (όπως για παράδειγμα το CodeIgniter) έχουν βασιστεί στο σχεδιαστικό πρότυπο MVC.

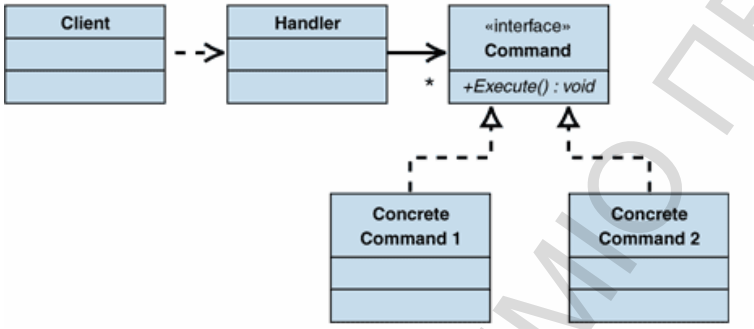
Μερικά από τα πλεονεκτήματα που προσφέρει το σχεδιαστικό πρότυπο MVC είναι ο αποδοτικός σχεδιασμός, η υποστήριξη πολλαπλών όψεων, η ανεξαρτησία του μοντέλου από

την όψη, η ευκολία ανάπτυξης, η παράλληλη ανάπτυξη από διαφορετικές ομάδες, η διευκόλυνση στη συντήρηση του κώδικα και ο καλύτερος έλεγχος (testability).

3.9 Ο Εμπρόσθιος ελεγκτής (Front controller)

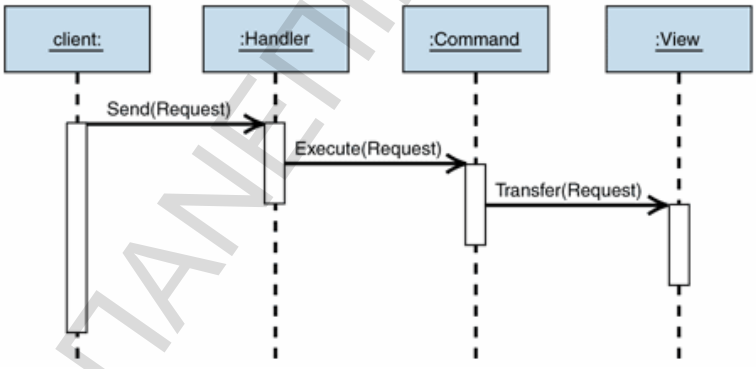
Ο εμπρόσθιος ελεγκτής (front controller) περιγράφει ουσιαστικά στρατηγικές υλοποίησης για το κομμάτι του ελεγκτή του προτύπου MVC. Ο εμπρόσθιος ελεγκτής είναι ένα σχεδιαστικό πρότυπο το οποίο παρέχει ένα κεντρικό σημείο εισόδου για την διαχείριση όλων των αιτήσεων του χρήστη σε μια web εφαρμογή. Με αυτόν τον τρόπο αποτελεί ένα κεντρικό εξάρτημα (component) που προσφέρεται για την ενσωμάτωση της κοινής λογικής που διέπει την εφαρμογή και ταυτόχρονα μειώνει σημαντικά τις εξαρτήσεις μεταξύ των άλλων εξαρτημάτων. Βοηθά έτσι στην εξάλειψη του επαναλαμβανόμενου κώδικα προσφέροντας τελικά ευκολία συντήρησης και ανάπτυξης. Ουσιαστικά παρεμβάλλεται μεταξύ της εφαρμογής web και του εξυπηρετητή. Βοηθά στην ομαδοποίηση (ενθυλάκωση) των διαφορετικών τρόπων που η εφαρμογή εγγράφεται στον web εξυπηρετητή με το να εγγράφεται μόνο ο εμπρόσθιος ελεγκτής.

Ένας εμπρόσθιος ελεγκτής συνήθως αποτελείται από ένα διαχειριστή αιτήσεων (request handler) και ένα λεξιλόγιο εντολών (command dictionary).



Εικόνα 18 - Δομή του εμπρόσθιου ελεγκτή (front controller)

Ο διαχειριστής αιτήσεων λαμβάνει την αίτηση του πελάτη και την αποστέλλει στο κατάλληλο command του λεξικού που θα την εξυπηρετήσει, αυτό μπορεί να είναι ο κατάλληλος ελεγκτής εφαρμογής.



Εικόνα 19 - Τυπικό σενάριο του εμπρόσθιου ελεγκτή

Προσφέρει αρκετά πλεονεκτήματα όπως κεντρικό έλεγχο (centralized control), ασφάλεια, ασφάλεια νημάτων (thread safety), ικανότητα παραμετροποίησης (configurability) και ευκολία

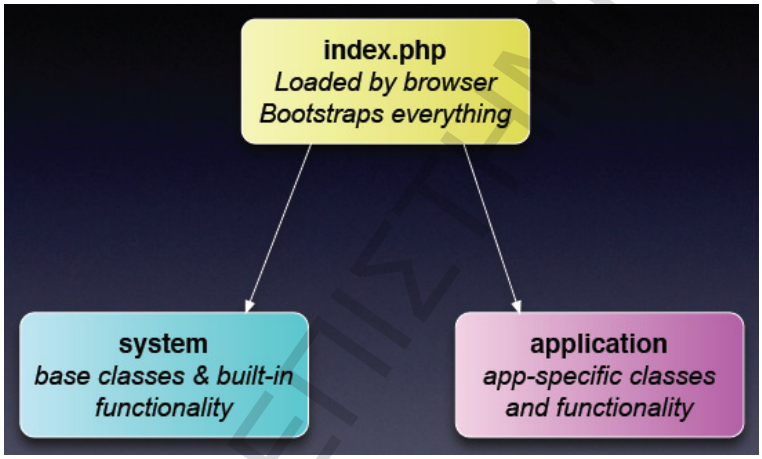
συντήρησης. Βέβαια είναι φανερό ότι έχει αντίκτυπο στην απόδοση από τη στιγμή που πολλές και διαφορετικές αιτήσεις περνάνε διαμέσου αυτού.

Το σχεδιαστικό πρότυπο του εμπρόσθιου ελεγκτή είναι διαθέσιμο για χρήση σε μια σειρά από δυναμικές γλώσσες προγραμματισμού από τη μεριά του εξυπηρετητή όπως PHP, Java servlets/JSP, ASP. Επίσης χρησιμοποιείται και σε γνωστά πλαίσια εργασίας όπως είναι το CodeIgniter και το Zend.

3.10 Το πλαίσιο εργασίας CodeIgniter

Το CodeIgniter είναι ένα ισχυρό πλαίσιο ανάπτυξης web εφαρμογών που βασίζεται στην PHP και στο σχεδιαστικό πρότυπο MVC. Βοηθά στην γρήγορη ανάπτυξη των έργων (projects) παρέχοντας ένα μεγάλο πλήθος πλούσιων βιβλιοθηκών για συνηθισμένες εργασίες και μία απλή διεπαφή με μια λογική δομή για την προσπέλαση των παρεχόμενων βιβλιοθηκών. Το CodeIgniter είναι ελεύθερο λογισμικό και ανοικτού κώδικα το οποίο μπορεί κάποιος να το κατεβάσει από την επίσημη ιστοσελίδα, στον ακόλουθο σύνδεσμο (http://codeigniter.com/downloads/).

Όπως ήδη αναφέρθηκε το CodeIgniter βασίζεται στο πρότυπο MVC. Τυπικά, οι κλάσεις του μοντέλου περιέχουν συναρτήσεις που βοηθούν στην ανάκτηση, εισαγωγή και ενημέρωση πληροφοριών στη βάση δεδομένων. Μία όψη κανονικά είναι μια σελίδα web, αλλά όμως στο CodeIgniter, μπορεί κάλλιστα να είναι ένα κομμάτι της σελίδας όπως πχ. το πάνω τμήμα (header) ή το κάτω τμήμα (footer). Μπορεί επίσης να είναι μια σελίδα RSS ή οποιουδήποτε άλλου τύπου ιστοσελίδα. Ο ελεγκτής δρα ως ο μεσολαβητής μεταξύ του μοντέλου, της όψης και οποιουδήποτε άλλου πόρου χρειάζεται για την επεξεργασία της αίτησης HTTP και την εμφάνιση της σελίδας. Το CodeIgniter παρέχει μια αφαιρετική προσέγγιση στο πρότυπο MVC μιας και το μοντέλο δεν είναι υποχρεωτικό.



Εικόνα 20 - Δομή του CodeIgniter (Διαθέσιμο σε: http://funkatron.com/content/EdFinkler-Introduction%20to%20CodeIgniter.pdf)

Το CodeIgniter δίνει τη δυνατότητα ενσωμάτωσης κώδικα από τον προγραμματιστή, ή ακόμα και της ανάπτυξης νέων βιβλιοθηκών για το σύστημα, καθιστώντας δυνατή την εργασία με ένα τρόπο που βολεύει τον κάθε προγραμματιστή. Επίσης, παρέχει συνοπτικά URL τα οποία συνίστανται από τέσσερα τμήματα (τα τρία πρώτα είναι υποχρεωτικά). Το πρώτο τμήμα αποτελεί το domain (ακολουθούμενο από τη διαδρομή αν υπάρχει), το δεύτερο τμήμα αποτελεί τη κλάση του ελεγκτή που θα ζητηθεί, το τρίτο τμήμα αποτελεί τη συνάρτηση (μέθοδο) της κλάσης που θα κληθεί και το τέταρτο τμήμα αποτελεί τη παράμετρο (ή παραμέτρους) που θα περαστεί στον ελεγκτή.

Το CodeIgniter παρέχει αρκετά πλεονεκτήματα όπως εξαιρετική απόδοση, ασφάλεια, συνοπτικά URL, συμβατότητα με πακέτα φιλοξενίας ιστοσελίδων (hosting) που τρέχουν

διάφορες εκδόσεις της PHP, σχεδόν μηδαμινή ρύθμιση (configuration) κατά την εισαγωγή, μηδενική απαίτηση για χρήση γραμμών εντολών (command line), δεν απαιτεί την υιοθέτηση περιοριστικών κανόνων κώδικα, δεν απαιτεί την εκμάθηση γλώσσας για templates, βοηθά στην αποφυγή πολυπλοκότητας ευνοώντας απλές λύσεις και παρέχει καθαρή, αναλυτική και ολοκληρωμένη τεκμηρίωση (documentation).

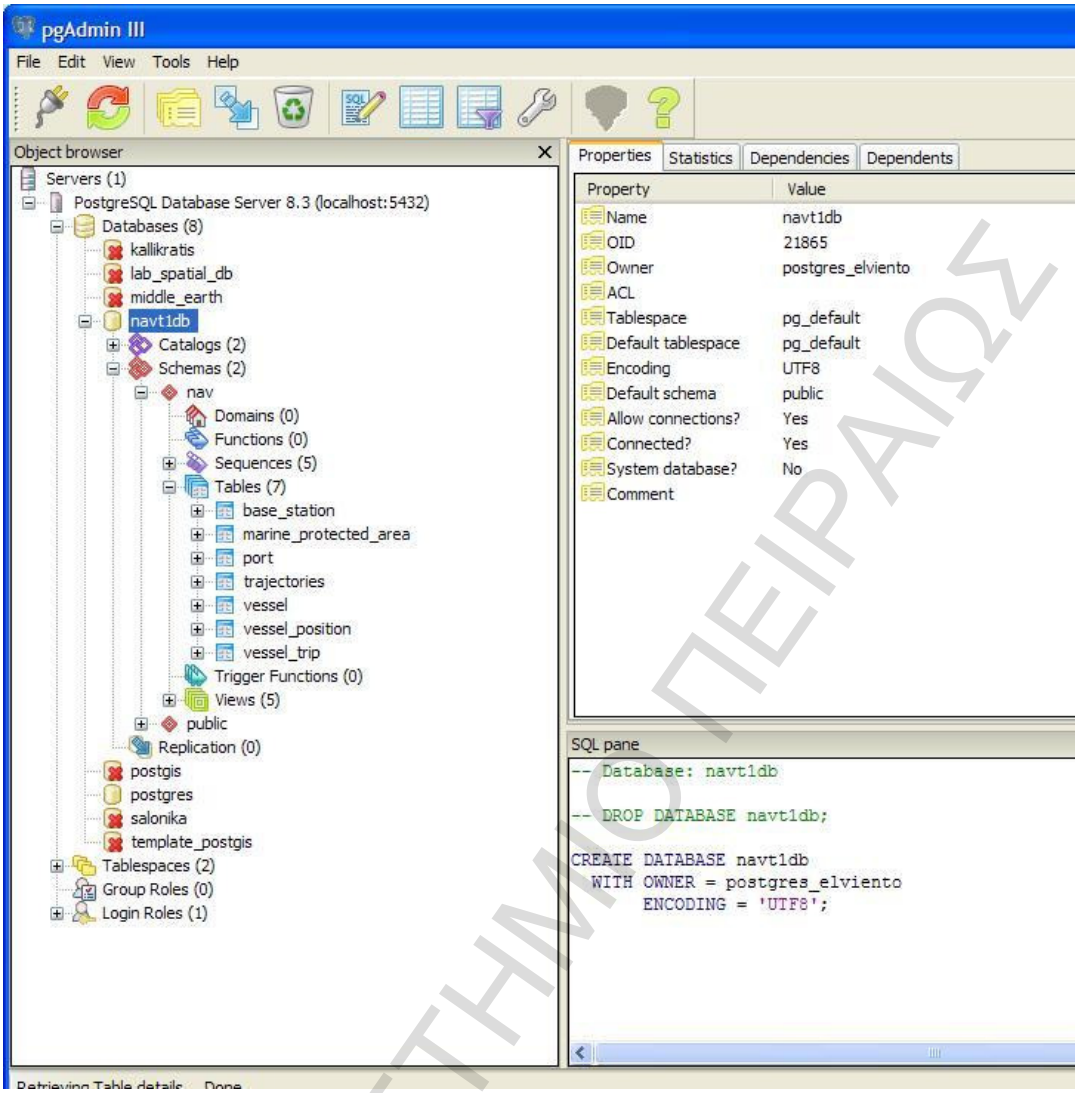
3.11 Η βάση δεδομένων PostgreSQL

Η PostgreSQL είναι ένα ολοκληρωμένο, αντικείμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (ORDBMS) με γεωγραφική υποστήριξη χάρη στην επέκταση της, PostGIS. Η PostgreSQL υποστηρίζει λειτουργίες όπως συναρτήσεις, ευρετήρια (B+-tree, hash, GiST και GiN), σκανδαλιστές (triggers), κανόνες και ένα ευρύ φάσμα από προκαθορισμένους και ορισμένους από τον χρήστη τύπους δεδομένων και αντικείμενα. Επιπλέον υποστηρίζονται λειτουργίες κληρονομικότητας χαρακτηριστικών πινάκων, περιορισμοί, όψεις (views), συναλλαγές, λειτουργίες κρυπτογράφησης, αποθήκευσης μεγάλων αντικειμένων κ.α. Παρέχει την δυνατότητα αποτελεσματικής διαχείρισης μεγάλου όγκου δεδομένων και εκτέλεσης ευέλικτων και πολύπλοκων ερωτημάτων με χρήση της γλώσσας SQL (Structured Query Language).

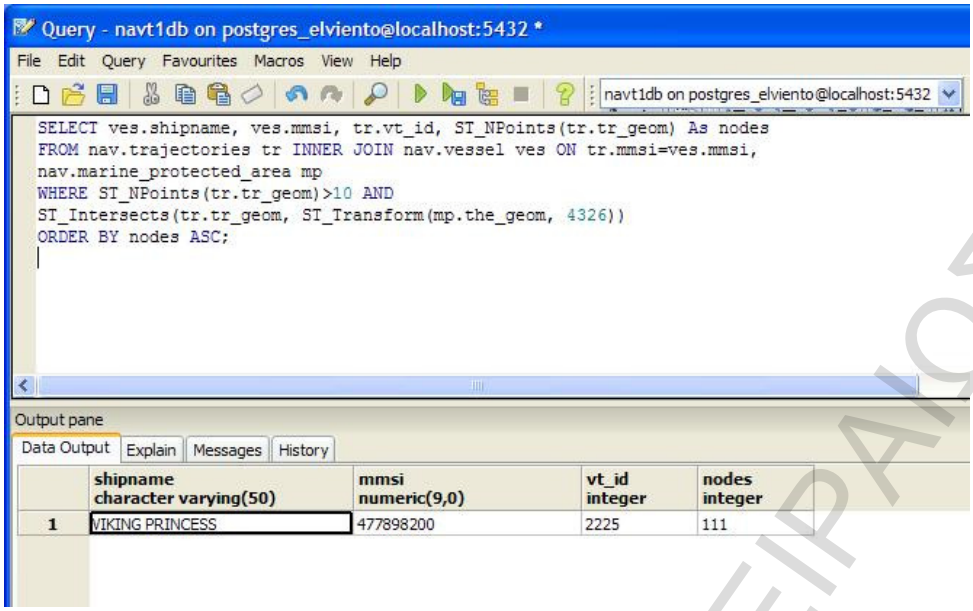
Είναι ελεύθερο λογισμικό/ανοικτού κώδικα το οποίο αναπτύσσεται συνεχώς και παρέχει εκδόσεις για Windows, Linux και Mac OS X. Μπορεί να ληφθεί από την επίσημη ιστοσελίδα στον παρακάτω σύνδεσμο (http://www.enterprisedb.com/products-services-training/pgdownload). Η PostgreSQL συνοδεύεται με το εργαλείο pgAdmin III το οποίο αποτελεί το προκαθορισμένο (default) γραφικό περιβάλλον εκτέλεσης ερωτημάτων και εγκαθίσταται αυτόματα μαζί με την PostgreSQL.

Υποστηρίζει χωρικά στοιχεία όπως γεωμετρικούς τύπους δεδομένων (πχ. point, linestring, polygon), χωρικά ευρετήρια, γεωμετρικούς τελεστές, γεωμετρικές συναρτήσεις και μετατροπές γεωμετρικών στοιχείων.

Τα χωρικά ευρετήρια μειώνουν σημαντικά το χρόνο αναζήτησης σε ένα χωρικό πίνακα. Για να αξιοποιήσουμε το ευρετήριο χρησιμοποιούμε συναρτήσεις ή τελεστές, όπως την ST_Overlaps ή τον τελεστή overlap (&&) αντίστοιχα (και στα αριστερά του τελεστή βάζουμε την στήλη στην οποία είναι χτισμένο το ευρετήριο). Υποστηρίζονται R-δένδρα για πολυγωνικά δεδομένα (μόνο) και γενικευμένα δένδρα αναζήτησης (GiST) τα οποία χρησιμοποιούνται για δεικτοδότηση όλων των χωρικών τύπων και για παραμετροποίηση των χωρικών λειτουργιών.



Εικόνα 21 - Το περιβάλλον του εργαλείου pgAdmin III



Εικόνα 22 - Σύνταξη και εκτέλεση ερωτήματος (query) στο pgAdmin III

3.11.1 PostGIS

Η PostGIS αποτελεί επέκταση της PostgreSQL προσθέτοντάς της χωρικές λειτουργίες. Είναι επίσης λογισμικό ανοικτού κώδικα και ακολουθεί το πρότυπο OGC για τον ορισμό γεωγραφικών δεδομένων σε περιβάλλον SQL. Συγκεκριμένα η PostGIS υποστηρίζει τύπους δεδομένων όπως σημεία (points), γραμμές (linestrings), πολύγωνα (polygons), πολυσημεία (multipoints), πολυγραμμές (multilinestrings) και πολύ-πολύγωνα (multipolygons). Επίσης, υποστηρίζει συλλογές γεωμετρικών δεδομένων, χωρικά κατηγορήματα (spatial predicates) δηλαδή συνθήκες για τον προσδιορισμό των αλληλεπιδράσεων ανάμεσα σε γεωμετρικά δεδομένα με χρήση του πίνακα Egenhofer 3x3, γεωμετρικούς τελεστές όπως τους area, distance, length, perimeter, γεωμετρικές λειτουργίες όπως τις union, difference, buffer και χωρικά ευρετήρια όπως τα R-δένδρα και γενικευμένα δένδρα αναζήτησης όπως τα GiST.

Αναπτύχθηκε το 2005 οπότε και προέκυψε η πρώτη έκδοση. Πιο πρόσφατη έκδοση είναι η 2.0 η οποία υποστηρίζει διανυσματικά (vector) και ψηφιογραφικά (raster) δεδομένα, χωρικά ευρετήρια τύπου R-δένδρα και μετατροπή δεδομένων μεταξύ διάφορων συστημάτων γεωγραφικών συντεταγμένων. Εξειδικεύεται στην αποθήκευση γεωμετρικών δεδομένων και στην εκτέλεση γεωμετρικών υπολογισμών και ερωτημάτων. Για το λόγο αυτό περιλαμβάνει ένα μεγάλο πλήθος από συναρτήσεις.

Μεταξύ των γεωδαιτικών συστημάτων αναφοράς (Spatial Reference Systems - SRID) που υποστηρίζονται είναι το ελληνικό γεωδαιτικό σύστημα GGRS87 με EPSG κωδικό: 2100 και μονάδα μέτρησης τα μέτρα και το παγκόσμιο γεωδαιτικό σύστημα WGS84 με EPSG κωδικό: 4326 και μονάδα μέτρησης τις μοίρες που χρησιμοποιείται στα Google Maps. Χαρακτηρίζεται από το γεωγραφικό μήκος (longitude μεταξύ -180 και 180) και το γεωγραφικό πλάτος (latitude μεταξύ -90 και 90). Υπάρχουν διαθέσιμα και άλλα συστήματα τα οποία χαρακτηρίζονται μοναδικά από έναν συγκεκριμένο κωδικό EPSG.

Το PostGIS πλέον υποστηρίζει ένα νέο δυναμικό γεωμετρικό τύπο δεδομένων, τον geometry. Οπότε οι στήλες που περιέχουν γεωμετρικά δεδομένα μπορούν να οριστούν και με αυτό τον τύπο. Επιπλέον, υποστηρίζει WKT (Well Known Text) μορφή των γεωμετρικών δεδομένων. Για να δούμε το γεωμετρικό αντικείμενο στη μορφή WKT δίνουμε το αντικείμενο ως παράμετρο στη συνάρτηση St_AsText.

Η αρχιτεκτονική της PostGIS στοχεύει στην ελαχιστοποίηση των απαιτούμενων πόρων υπολογιστικής ισχύος και μνήμης. Η χρήση γεωμετρικών στοιχείων με χαμηλές απαιτήσεις

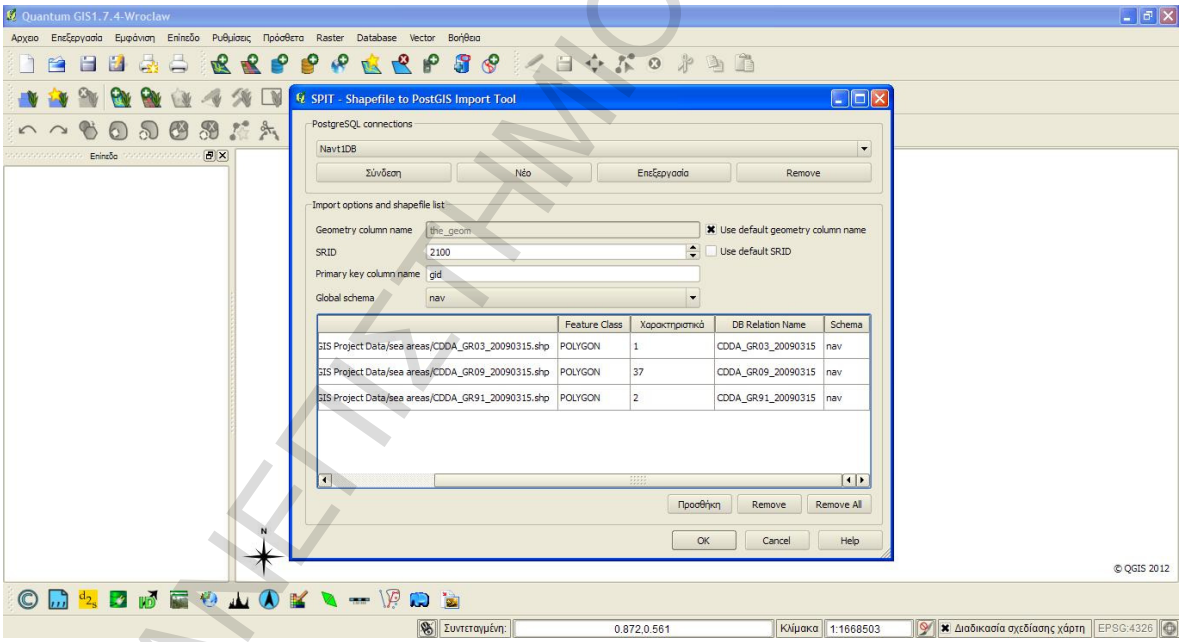
φυσικής μνήμης επιτρέπει την διατήρηση μεγάλου όγκου δεδομένων από την φυσική μνήμη στην υπολογιστική μνήμη (RAM) με αποτέλεσμα την ταχύτερη εκτέλεση των ερωτημάτων.

3.12 Το εργαλείο Quantum GIS

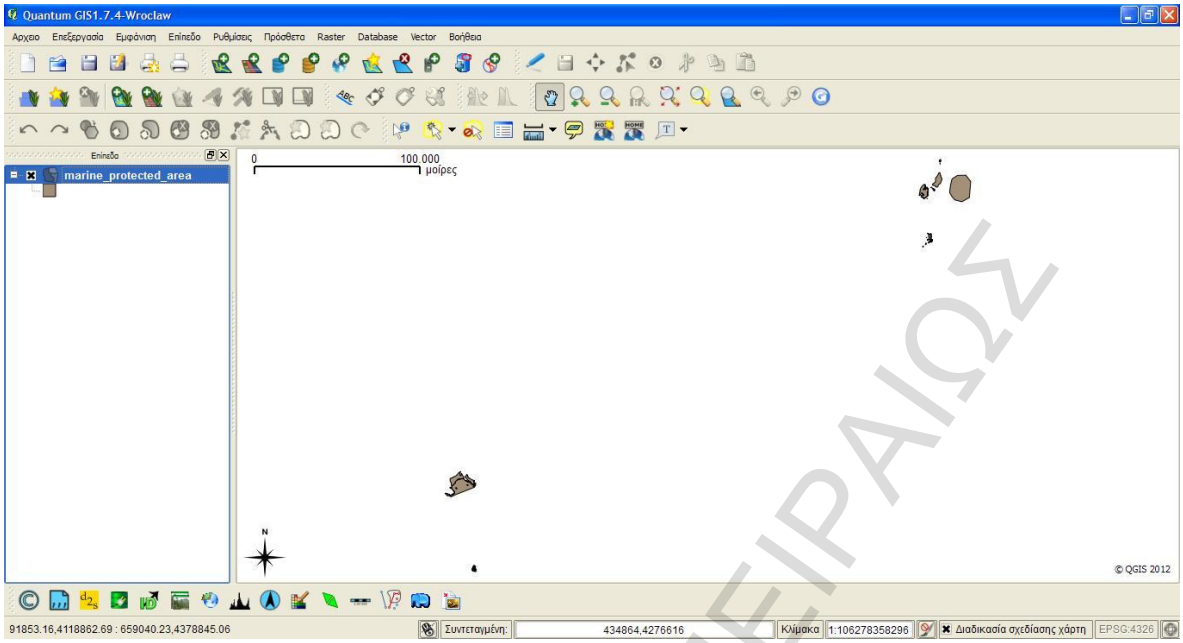
Το Quantum GIS είναι ένα GIS εργαλείο γραφείου που επιτρέπει την οπτικοποίηση των δεδομένων μιας χωρικής βάσης δεδομένων όπως είναι η PostgreSQL και την επέμβαση σε αυτά σε ένα εύχρηστο και φιλικό προς τον χρήστη γραφικό περιβάλλον. Πιο συγκεκριμένα, επιτρέπει στον χρήστη την ερώτηση, την αναζήτηση και την ανάλυση χωρικών δεδομένων όπως επίσης και την μετατροπή, εισαγωγή και τροποποίηση χωρικών δεδομένων καθώς και τη δημιουργία χαρτών από αυτά.

Πρόκειται για ελεύθερο λογισμικό/ανοικτού κώδικα και μπορεί κάποιος να το κατεβάσει από τον ακόλουθο ιστότοπο (http://hub.qgis.org/projects/quantum-gis/wiki/Download). Η ανάπτυξη του λογισμικού ξεκίνησε από τον Gary Sherman το 2002 και σύντομα γύρω από αυτό αναπτύχθηκε το Open Source Geospatial Foundation. Το πρόγραμμα συντηρείται από μια ενεργή κοινότητα χρηστών και προγραμματιστών που σε τακτά χρονικά διαστήματα κυκλοφορούν νέες εκδόσεις επεκτείνοντας τις δυνατότητες των παλιών και διορθώνοντας σφάλματα.

Στα πλεονοκτήματά του συγκαταλέγονται η συμβατότητα του με Windows/Unix/Linux/Mac OS X, η δυνατότητα πολλαπλής εισαγωγής αρχείων απλά πατώντας το πλήκτρο Add προσέχοντας ωστόσο όλα τα αρχεία να έχουν το ίδιο SRID και η παροχή σύντομης σύνοψης για το κάθε αρχείο προτού φορτωθεί, όπως αριθμό και τύπο γεωμετρίας.



Εικόνα 23 - Εργαλείο εισαγωγής αρχείου shape σε PostGIS



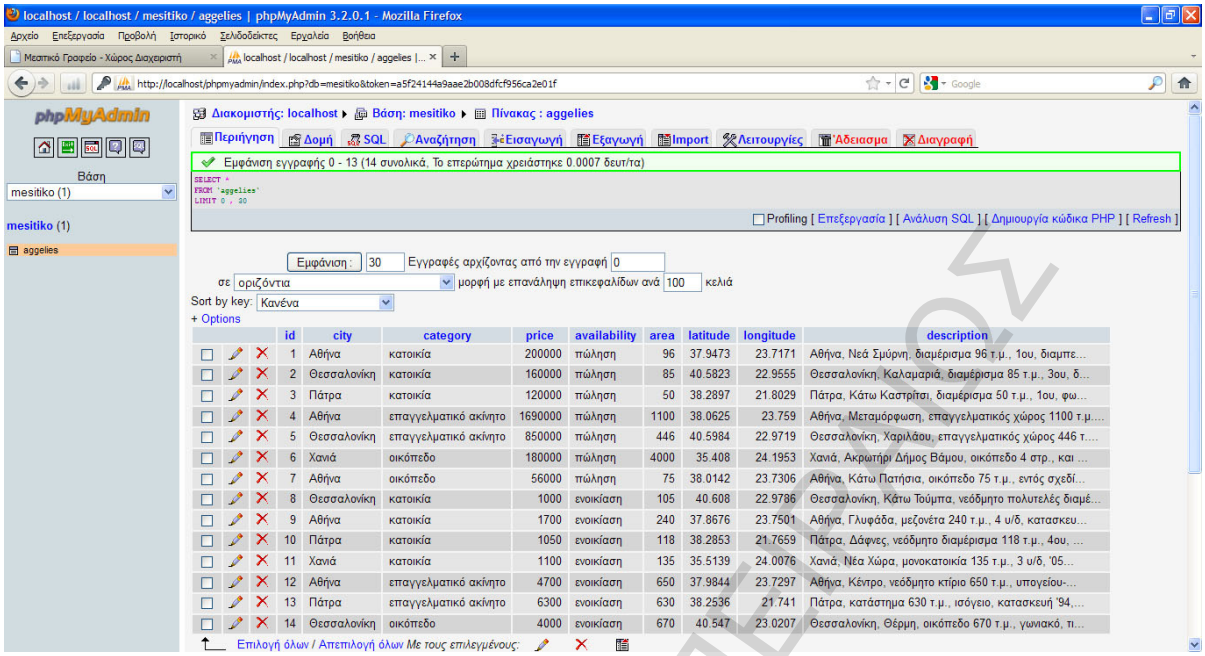
Εικόνα 24 - Οπτικοποίηση των δεδομένων ενός πίνακα από μια χωρική βάση δεδομένων

3.13 Το πακέτο XAMPP

Το XAMPP (cross-platform (X), Apache, MySQL, PHP, Perl) είναι ένα πακέτο προγραμμάτων ελεύθερου λογισμικού/ανοικτού κώδικα και ανεξάρτητο πλατφόρμας το οποίο περιέχει τον εξυπηρετητή Apache, τη βάση δεδομένων MySQL και ένα διερμηνέα για προγράμματα γραμμένα σε PHP και Perl. Τρέχει σε Windows, Linux, Solaris και Mac OS X και μπορεί κάποιος να το κατεβάσει από την επίσημη ιστοσελίδα (http://www.apachefriends.org/en/xampp.html). Χρησιμοποιείται ως πλατφόρμα για τη σχεδίαση, ανάπτυξη και δοκιμή ιστοσελίδων με τεχνολογίες από τη μεριά του εξυπηρετητή όπως PHP τοπικά στον υπολογιστή (localhost).

Για την πρόσβαση τοπικά στον εξυπηρετητή (localhost) αρκεί να εκκινήσουμε, μέσα από το διαχειριστικό περιβάλλον του XAMPP, τον Apache και την MySQL και ύστερα να εισαγάγουμε το παρακάτω URL στο φυλλομετρητή: http://localhost/

Το XAMPP συμπεριλαμβάνει επίσης τα πακέτα OpenSSL και phpMyAdmin. Το phpMyAdmin είναι ένα πρόγραμμα ελεύθερου λογισμικού/ανοικτού κώδικα γραμμένο σε PHP που χρησιμοποιείται για την διαχείριση MySQL βάσεων δεδομένων μέσω ενός φυλλομετρητή. Επιτελεί διάφορες εργασίες όπως δημιουργία, τροποποίηση ή διαγραφή βάσεων, πινάκων, πεδίων ή εγγραφών μέσα από μία απλή διεπαφή και δίνει παράλληλα τη δυνατότητα εκτέλεσης εντολών SQL. Επίσης, χειρίζεται τα δικαιώματα των χρηστών. Για την πρόσβαση στο phpMyAdmin εισάγουμε το παρακάτω URL στο φυλλομετρητή: http://localhost/phpmyadmin/



Εικόνα 25 - Περιβάλλον phpMyAdmin

Έχοντας στο οπλοστάσιο το θεωρητικό υπόβαθρο των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής μπορούμε να προχωρήσουμε παρακάτω στο επόμενο κεφάλαιο όπου παρουσιάζεται σταδιακά η υλοποίησή της. Αρχικά παρουσιάζεται το περίγραμμα της εφαρμογής και στη συνέχεια επιδεικνύεται ο σχεδιασμός της βάσης δεδομένων της εφαρμογής και η υλοποίησή της. Στη συνέχεια, αναλύονται αρχικά οι απαιτήσεις της εφαρμογής, ακολουθεί ο σχεδιασμός με διαγράμματα UML και τέλος περιγράφεται η διαδικασία υλοποίησης της εφαρμογής.

ΚΕΦΑΛΑΙΟ 4: ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΛΕΙΤΟΥΡΓΙΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

4.1 Περίγραμμα της εφαρμογής

Αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη μιας εφαρμογής ιστού η οποία είναι βασισμένη στην αντικειμενοστρεφή αρχιτεκτονική (Object Oriented Architecture - OOA) και εξυπηρετείται από ένα γεωγραφικό πληροφοριακό σύστημα (GIS).

Το θέμα σχετίζεται με το φανταστικό κόσμο της Μέσης Γης (Middle Earth) και επικεντρώνεται στα βασίλειά της, τα κάστρα που κάθε βασίλειο περιλαμβάνει και τα μονοπάτια που διασχίζουν τα βασίλεια. Τα βασίλεια ουσιαστικά μοντελοποιούνται ως πολύγωνα δηλαδή οριοθετημένες χωρικές εκτάσεις, τα κάστρα ως σημεία ενδιαφέροντος και τα μονοπάτια ως τεθλασμένες γραμμές.

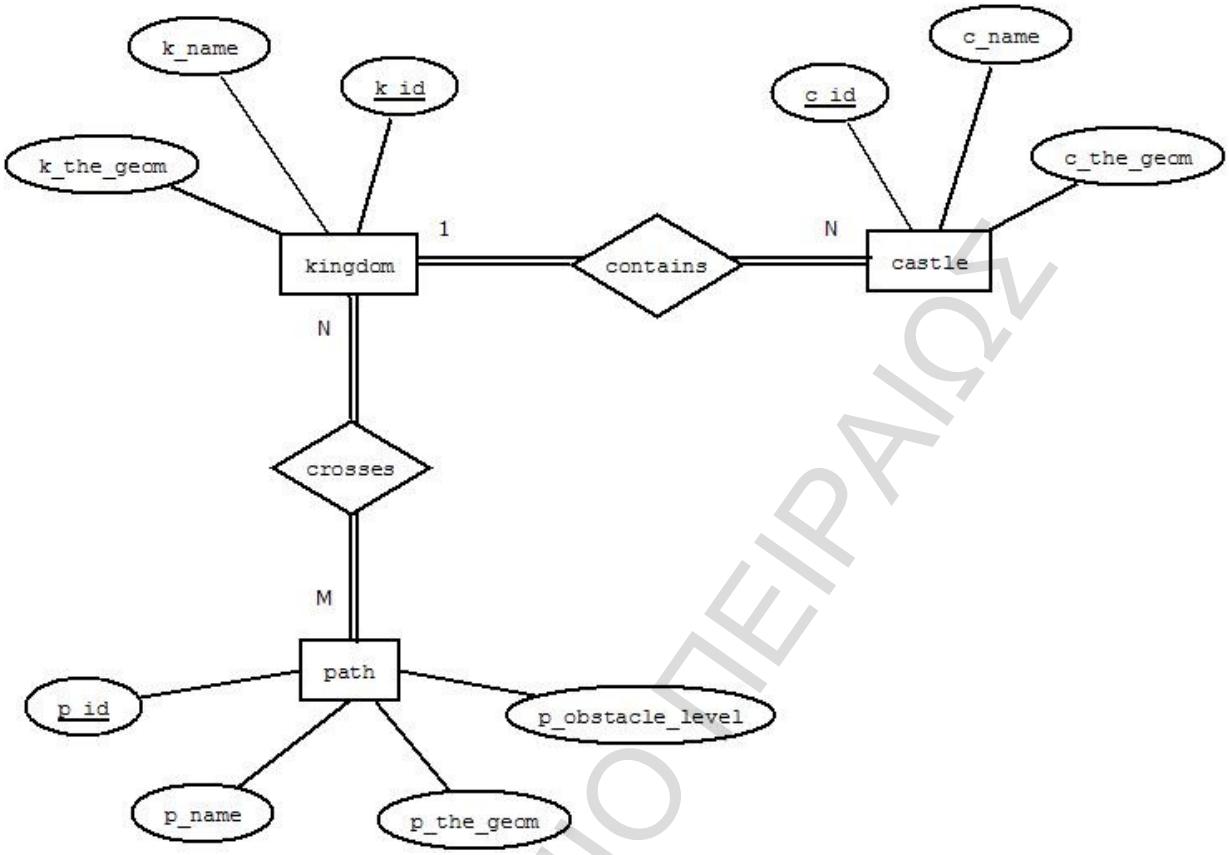
Βασίζεται σε δύο κύριους άξονες λειτουργικότητας:

- **Γεωκωδικοποίηση (Geocoding)**
- **Αντίστροφη γεωκωδικοποίηση (Reverse geocoding)**

Η ραχοκοκαλιά της εφαρμογής είναι η βάση δεδομένων και συνεπώς πραγματοποιείται αρχικά ο εννοιολογικός σχεδιασμός και στη συνέχεια η υλοποίηση της βάσης δεδομένων. Επόμενο βήμα είναι η ανάλυση της εφαρμογής όσον αφορά τις λειτουργικές της απαιτήσεις, δηλαδή το «τι κάνει» η εφαρμογή και έπειτα ακολουθεί ο σχεδιασμός της εφαρμογής, δηλαδή το «πώς το κάνει». Τέλος, γίνεται η υλοποίηση της εφαρμογής.

4.2 Εννοιολογικός σχεδιασμός της βάσης δεδομένων

Αρχικά δημιουργήθηκε το εννοιολογικό σχήμα της βάσης δεδομένων με διάγραμμα ER, ώστε να αποτυπώσουμε τον «μικρόκοσμο» της εφαρμογής.



Εικόνα 26 - Διάγραμμα ER

Ορίζονται οι παρακάτω οντότητες και τα χαρακτηριστικά τους:

- **kingdom**: k_id (κλειδί), k_name, k_the_geom
- **castle**: c_id (κλειδί), c_name, c_the_geom
- **path**: p_id (κλειδί), p_name, p_the_geom, p_obstacle_level

Καθώς και οι συσχετίσεις μεταξύ των οντοτήτων:

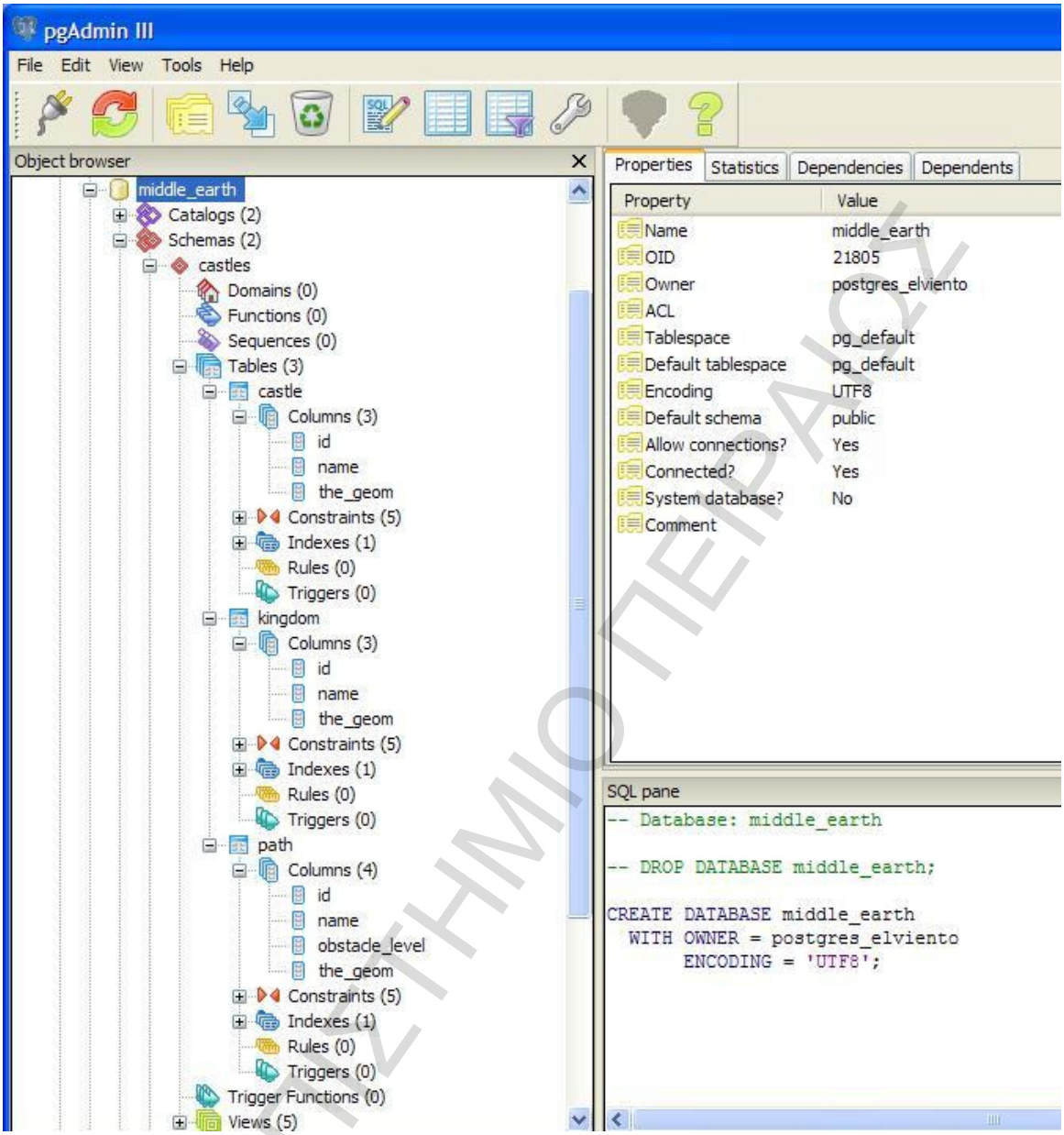
- **contains**: Ένα βασίλειο περιλαμβάνει πολλά κάστρα (ολική συμμετοχή) και ένα κάστρο ανήκει σε ένα βασίλειο υποχρεωτικά (ολική συμμετοχή). Επομένως, η συσχέτιση contains είναι 1 προς N.
- **crosses**: Ένα μονοπάτι διασχίζει πολλά βασίλεια (ολική συμμετοχή) και ένα βασίλειο διασχίζεται από πολλά μονοπάτια (ολική συμμετοχή). Επομένως, η συσχέτιση crosses είναι N προς M.

4.3 Υλοποίηση της βάσης δεδομένων

Οι οντότητες (όλες είναι ισχυρές) θα γίνουν πίνακες με πεδία τα χαρακτηριστικά τους. Ο κάθε πίνακας θα έχει ως πρωτεύον κλειδί το κλειδί της αντίστοιχης οντότητας. Δεν υπάρχει ανάγκη για τη δημιουργία ξένων κλειδιών (foreign keys) από τη στιγμή που οι συσχετίσεις μεταξύ των πινάκων θα υποδηλώνονται πρακτικά μέσω των χωρικών πληροφοριών που θα χαρακτηρίζουν τις εγγραφές του κάθε πίνακα.

Η υλοποίηση της βάσης και των πινάκων που την απαρτίζουν πραγματοποιήθηκε στη PostgreSQL με χρήση της απαραίτητης επέκτασής της, της PostGIS, για την υποστήριξη των χωρικών δεδομένων και του προκαθορισμένου γραφικού περιβάλλοντος εκτέλεσης ερωτημάτων pgAdmin III. Αρχικά, δημιουργήθηκε η βάση δεδομένων με όνομα middle_earth ενισχυμένη με χωρικές λειτουργίες (TEMPLATE = template_postgis) και στη συνέχεια το σχήμα (schema) με όνομα castles όπου θα εισαχθούν οι πίνακες. Το επόμενο βήμα ήταν η δημιουργία των τριών πινάκων (castle, kingdom, path) και η εισαγωγή δεδομένων σε αυτούς. Για κάθε πίνακα το πεδίο id δηλώθηκε ως integer και το πεδίο name ως varchar. Το πεδίο obstacle_level του πίνακα path δηλώθηκε ως integer. Επίσης, για κάθε έναν πίνακα ξεχωριστά, χρησιμοποιήθηκε η συνάρτηση AddGeometryColumn για να εισαχθεί η γεωμετρία του, ως πεδίο (the_geom). Η γεωμετρία του πίνακα castle ορίσθηκε ως σημείο (point) με SRID ίσο με 4326, η γεωμετρία του πίνακα kingdom ορίσθηκε ως πολύγωνο (polygon) με SRID ίσο με 4326 και η γεωμετρία του πίνακα path ορίσθηκε ως γραμμή (linestring) με SRID ίσο με 4326. Επιπρόσθετα, ενεργοποιήθηκε ο έλεγχος εγκυρότητας της γεωμετρίας για κάθε έναν πίνακα ξεχωριστά.

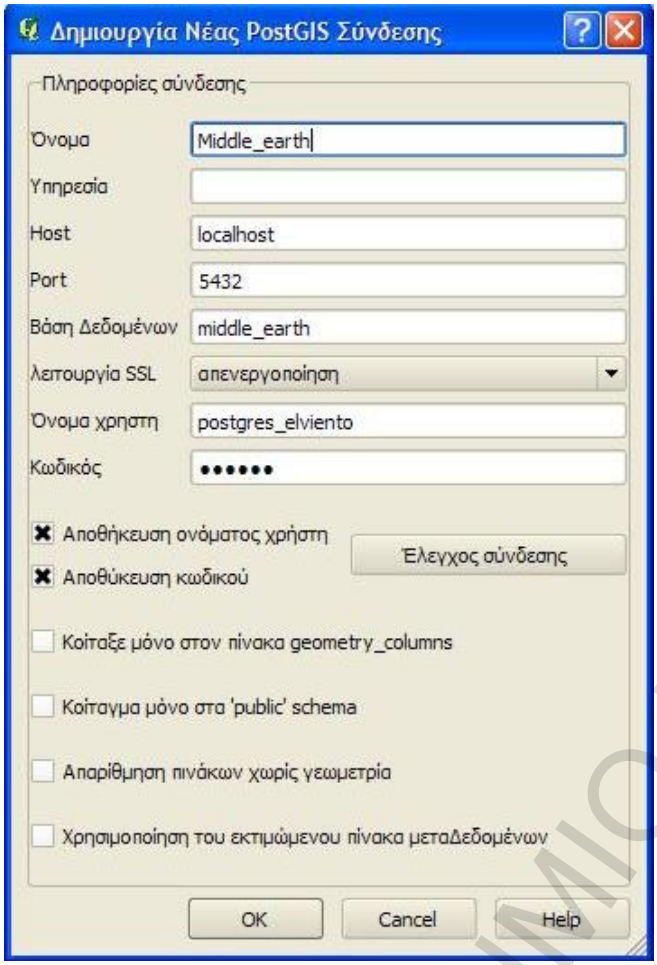
Δόθηκε μεγάλη βαρύτητα στην απόδοση και κατ' επέκταση στην ελαχιστοποίηση του χρόνου αναζήτησης σε ερωτήματα (queries) στους πίνακες της βάσης με τη δημιουργία κατάλληλων χωρικών ευρετηρίων. Έτσι, δημιουργήθηκε ένα χωρικό ευρετήριο γενικευμένου δέντρου αναζήτησης στη γεωμετρική στήλη the_geom για κάθε έναν από τους πίνακες.



Εικόνα 27 - Οι πίνακες της ΒΔ middle_earth εντός του σχήματος castles

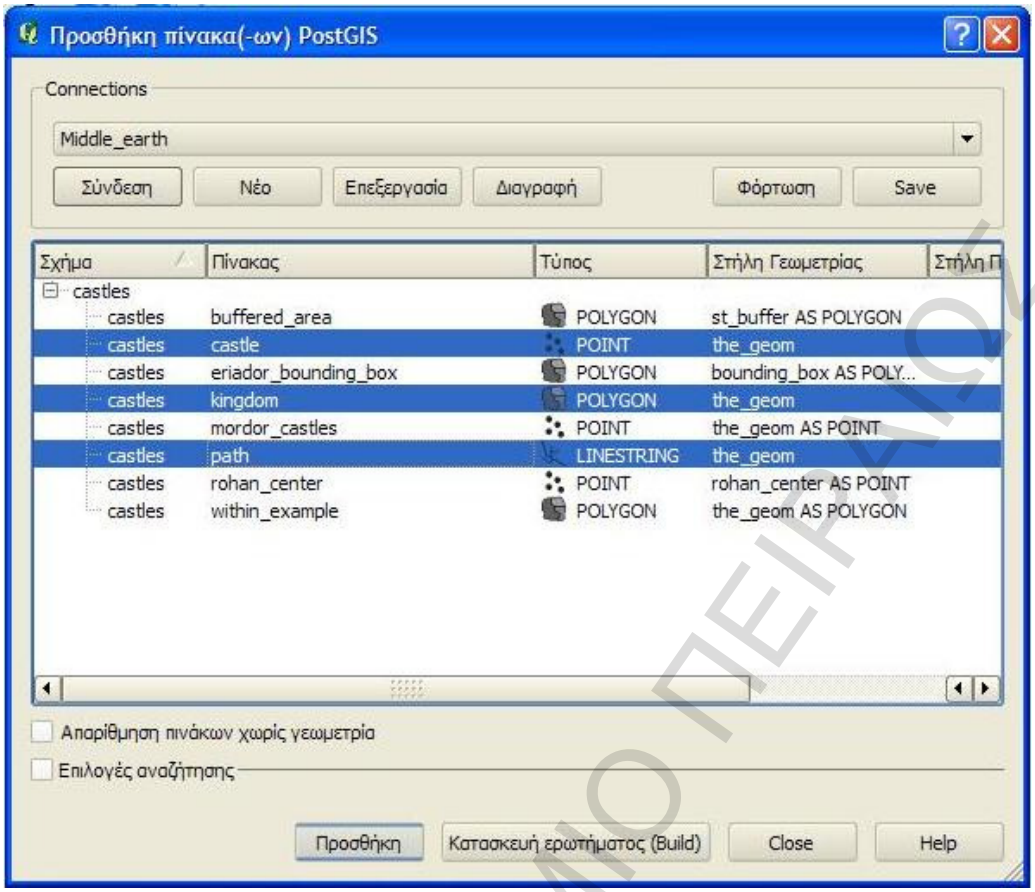
Στη συνέχεια περάστηκαν τα δεδομένα του φανταστικού κόσμου της Μέσης Γης στους πίνακες με διαδοχικές εντολές INSERT και επιπλέον με τη βοήθεια του Quantum GIS όπου με την απεικόνιση των θεματικών επιπέδων (πινάκων) ήταν πιο εύκολη η ακριβής και κατάλληλη τοποθέτηση των γεωμετρικών δεδομένων και η επακόλουθη εισαγωγή των εγγραφών στην βάση.

Για το σκοπό αυτό, προηγήθηκε η δημιουργία σύνδεσης με τη βάση middle_earth στο Quantum GIS όπως φαίνεται στην εικόνα 28.



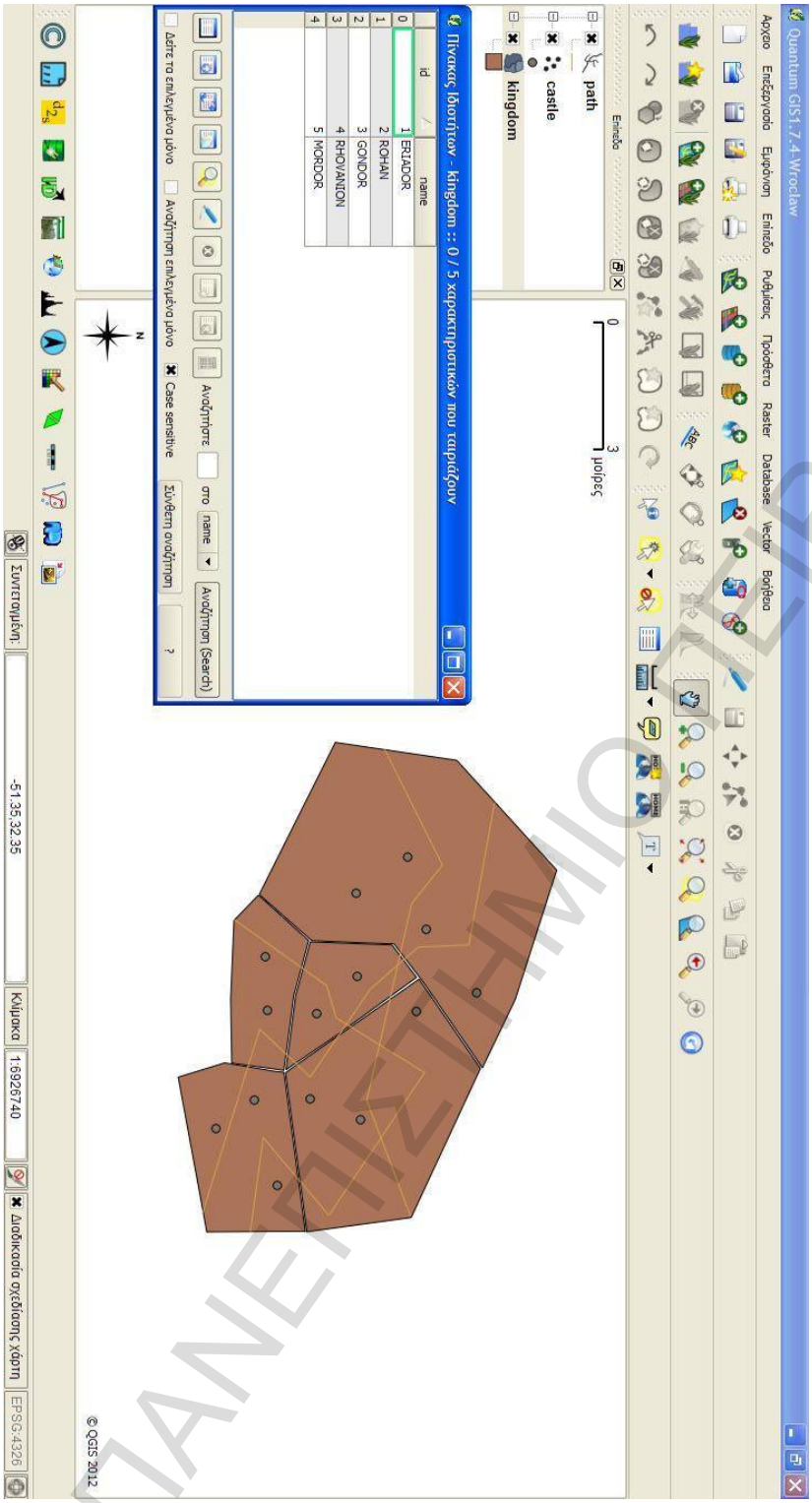
Εικόνα 28 - Δημιουργία σύνδεσης με την βάση middle_earth

Στη συνέχεια έγινε σύνδεση στη βάση middle_earth και επιλέχθηκαν οι πίνακές της ώστε να πραγματοποιηθεί η προσθήκη τους ως θεματικά επίπεδα στο Quantum GIS.



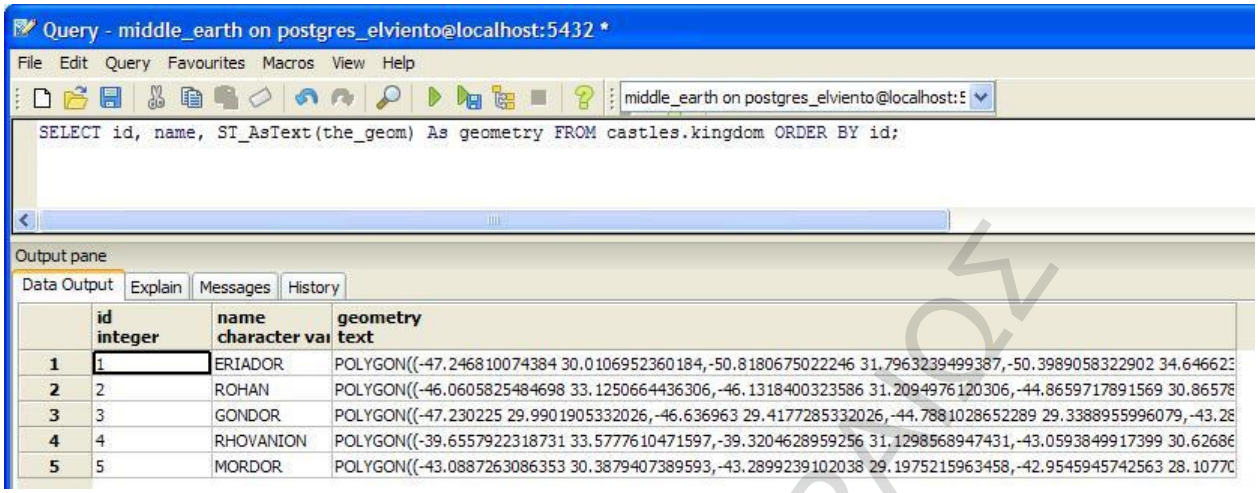
Εικόνα 29 - Σύνδεση με τη βάση middle_earth, επιλογή και προσθήκη των πινάκων της

Τελικά, οι πίνακες απεικονίζονται στο Quantum GIS ως θεματικά επίπεδα (postgis layers). Σε αυτό το σημείο μπορεί να γίνει εισαγωγή, επεξεργασία, τροποποίηση και διαγραφή δεδομένων.



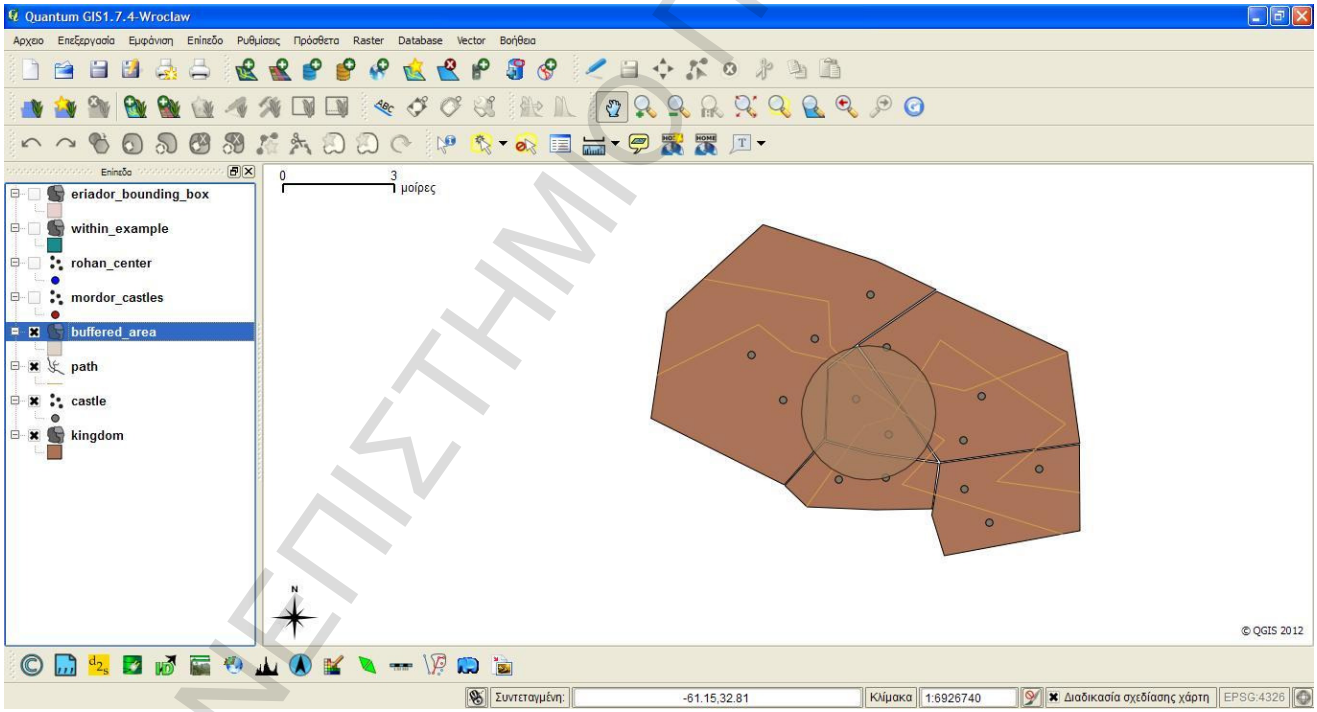
Εικόνα 30 - Τα 3 θεματικά επίπεδα (πίνακες) της ΒΔ όπως φαίνονται στο Quantum GIS

Στην εικόνα 31 φαίνονται οι εγγραφές για τον πίνακα kingdom στο pgAdmin III.



Εικόνα 31 - Οι εγγραφές του πίνακα kingdom

Δημιουργήθηκαν και κάποιες όψεις (views) για λόγους επαλήθευσης της ορθής απεικόνισης των χωρικών δεδομένων στο Quantum GIS.



Εικόνα 32 - Όψη buffered_area – Δείχνει τα κάστρα που περιλαμβάνονται στη περιοχή που δημιουργείται με κέντρο ένα δοθέν σημείο και ακτίνα 1.8 μοίρες

4.4 Ανάλυση της εφαρμογής

Η εφαρμογή μέσω μιας εύχρηστης και φιλικής προς το χρήστη διεπαφής, προσφέρει τρείς βασικές υπηρεσίες προς τους χρήστες.

- **Εύρεση γεωγραφικής τοποθεσίας (Geocoding)**: Ο χρήστης επιλέγει μέσα από κατάλληλο μενού επιλογών κάποιο από τα διαθέσιμα βασίλεια και στη συνέχεια αφενός

κεντράρεται ο χάρτης της μέσης γης στο κέντρο του επιλεγμένου βασιλείου και αφετέρου γεμίζει ένα δεύτερο μενού επιλογών με όλα τα κάστρα που περιλαμβάνει το επιλεγμένο βασίλειο. Με την επιλογή κάποιου κάστρου εμφανίζεται στο χάρτη το εικονίδιό του στην τοποθεσία στο χάρτη που βρίσκεται το κάστρο.

- **Εύρεση πληροφοριών τοποθεσίας (Reverse geocoding)**: Ο χρήστης επιλέγει κάποιο σημείο πάνω στο χάρτη της Μέσης Γης και παίρνει πληροφορίες σχετικά με το βασίλειο όπου το σημείο αυτό ανήκει και το γεωγραφικό μήκος και πλάτος του κέντρου του βασιλείου. Επιπρόσθετα, ο χάρτης κεντράρεται στο κέντρο του βασιλείου.

- **Εύρεση σημείων ενδιαφέροντος σε συγκεκριμένη ακτίνα (Points of interest (POI) detection in a specific range)**: Αυτή η λειτουργικότητα αποτελεί συνδυασμό των δύο προηγούμενων. Ο χρήστης επιλέγει κάποιο σημείο πάνω στο χάρτη της Μέσης Γης αφού αρχικά έχει επιλέξει την επιθυμητή ακτίνα μέσα από κατάλληλο μενού επιλογών και παίρνει αφενός πληροφορίες για το γεωγραφικό μήκος και πλάτος του επιλεγμένου σημείου καθώς και του μήκους της επιλεγμένης ακτίνας και αφετέρου εμφανίζονται στο χάρτη τα κάστρα (σημεία ενδιαφέροντος) που βρίσκονται γύρω από το σημείο του χάρτη που επιλέχθηκε και περιέχονται σε αυτή την ακτίνα.

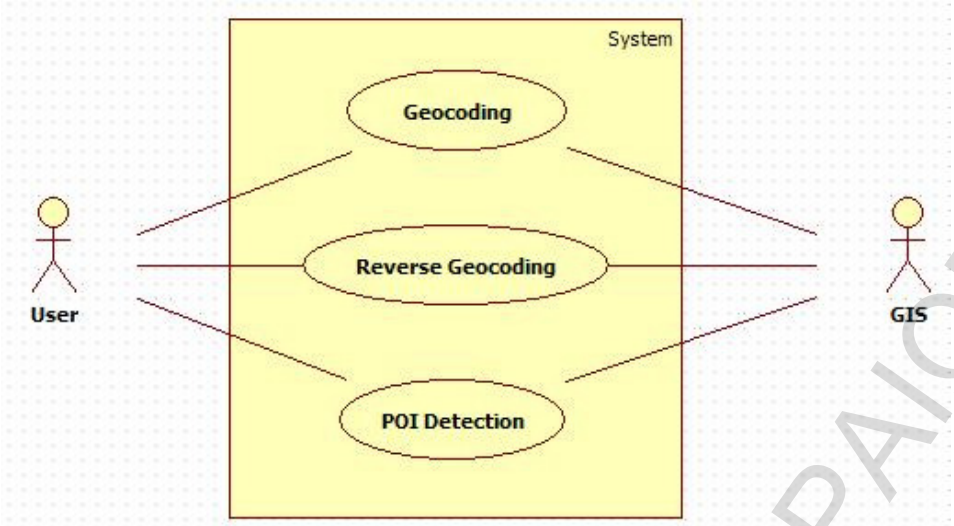
4.5 Σχεδιασμός της εφαρμογής

Για το σχεδιασμό της εφαρμογής χρησιμοποιήθηκε η γλώσσα μοντελοποίησης UML και δημιουργήθηκαν τα διαγράμματα περιπτώσεων χρήσης, τα διαγράμματα κλάσεων, τα διαγράμματα συνεργασίας, τα διαγράμματα σειράς, τα διαγράμματα δραστηριοτήτων, τα διαγράμματα καταστάσεων, τα διαγράμματα εξαρτημάτων και τα διαγράμματα διανομής.

4.5.1 Διαγράμματα Περιπτώσεων Χρήσης (Use Case Diagrams)

Τα διαγράμματα περιπτώσεων χρήσης χρησιμοποιούνται για τη μοντελοποίηση της συμπεριφοράς της πλατφόρμας, όπως αυτή γίνεται αντιληπτή από τους εξωτερικούς χρήστες. Μας δείχνουν ποιοι συμμετέχουν στην εφαρμογή και τι μπορεί να κάνει ο καθένας τους. Συνεπώς, τα διαγράμματα αυτά αποτελούνται από τους δράστες (actors) και τις επιμέρους περιπτώσεις χρήσεις (use cases). Οι δράστες είναι 2 ειδών: (α) φυσικά πρόσωπα, (β) συστήματα που δρουν στην εφαρμογή. Οι περιπτώσεις χρήσεις αποτελούν ένα υψηλού επιπέδου κομμάτι λειτουργικότητας που η εφαρμογή μπορεί να προσφέρει. Με άλλα λόγια, μια περίπτωση χρήσης απεικονίζει με ποιον τρόπο μπορεί κάποιος να χρησιμοποιήσει το σύστημα.

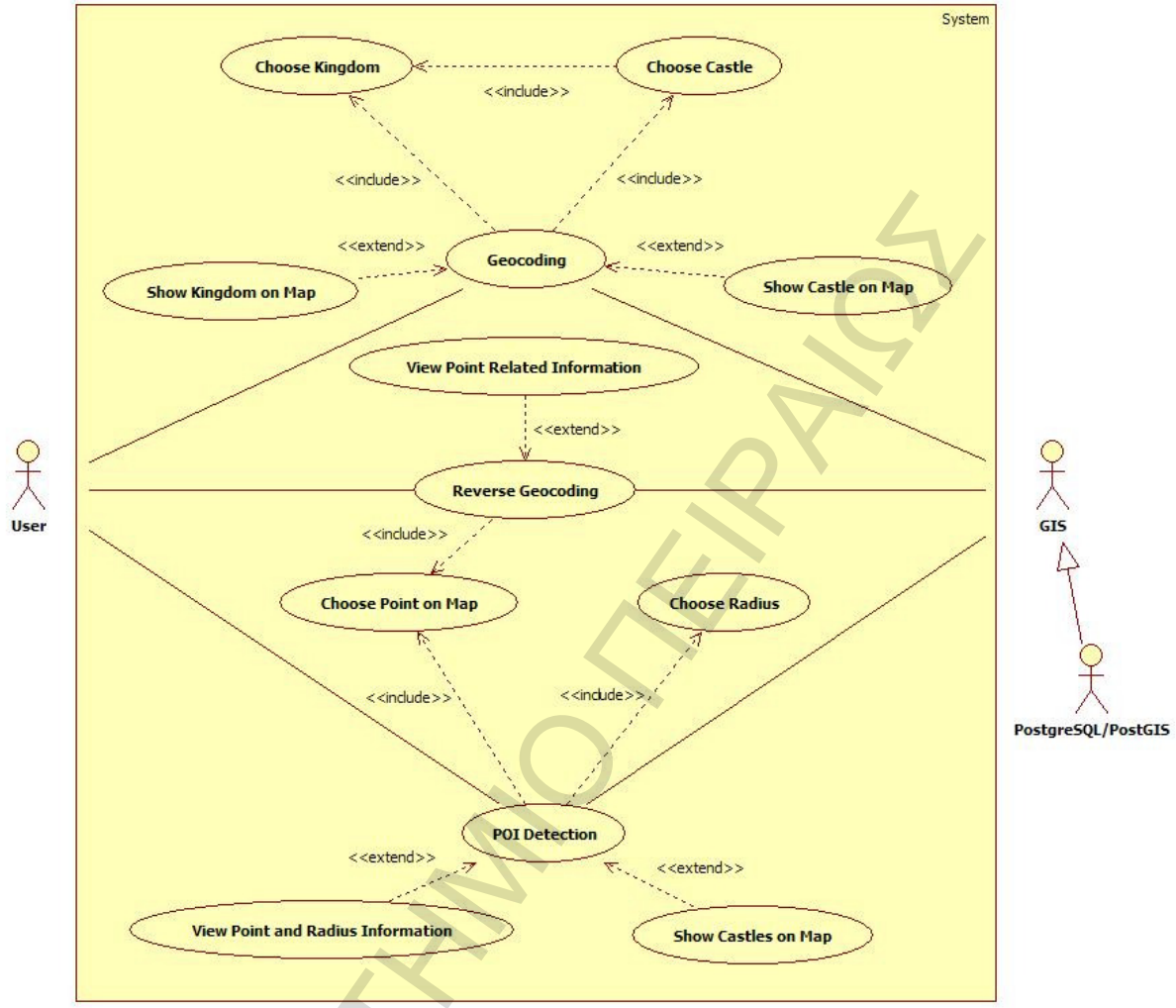
Στην εικόνα 33 (πρώτη έκδοση) φαίνεται αφενός ο δράστης – φυσικό πρόσωπο που είναι ο χρήστης της εφαρμογής και οι τρόποι με τους οποίους μπορεί να χρησιμοποιήσει το σύστημα αντίστοιχα και αφετέρου ο δράστης – σύστημα που είναι το γεωγραφικό πληροφοριακό σύστημα που υποστηρίζει την πλατφόρμα και τις λειτουργίες της.



Εικόνα 33 - Διάγραμμα περιπτώσεων χρήσης (πρώτη έκδοση)

Στην εικόνα 34 (δεύτερη και τελική έκδοση) έχει γίνει βελτίωση και επέκταση του προηγούμενου με μεγαλύτερη λεπτομέρεια. Στη πρώτη βασική περίπτωση χρήσης, «Geocoding», έχει εισαχθεί η σχέση extend ώστε να μπορεί να επεκτείνεται με την εμφάνιση του βασιλείου στο χάρτη και την εμφάνιση του κάστρου στο χάρτη. Επίσης, έχει εισαχθεί η σχέση include ώστε να απαιτεί υποχρεωτικά τη λειτουργία των περιπτώσεων χρήσης «Choose Kingdom» (πρώτα) και στη συνέχεια «Choose Castle». Στη δεύτερη βασική περίπτωση χρήσης, «Reverse Geocoding», έχει εισαχθεί η σχέση extend ώστε να μπορεί να επεκτείνεται με την εμφάνιση πληροφοριών που να σχετίζονται με το σημείο που επιλέχθηκε πάνω στο χάρτη. Η σχέση include έχει εισαχθεί διότι απαιτείται υποχρεωτικά η λειτουργία της περίπτωσης χρήσης «Choose Point on Map». Στην τρίτη βασική περίπτωση χρήσης, «POI Detection», έχει εισαχθεί η σχέση extend ώστε να μπορεί να επεκτείνεται με την εμφάνιση πληροφοριών σχετικών με το σημείο που επιλέχθηκε πάνω στο χάρτη και την επιλεγμένη ακτίνα δράσης αλλά και την εμφάνιση των κάστρων στο χάρτη. Επίσης, έχει εισαχθεί η σχέση include ώστε να απαιτεί υποχρεωτικά τη λειτουργία των περιπτώσεων χρήσης, «Choose Radius» και στη συνέχεια «Choose Point on Map».

Επιπρόσθετα, όπως φαίνεται υπάρχει σχέση κληρονομικότητας μεταξύ του γενικευμένου δράστη «GIS» και του εξειδικευμένου δράστη «PostgreSQL/PostGIS» κάτι που δείχνει ότι η PostgreSQL/PostGIS είναι ουσιαστικά το τελικό τμήμα του γεωγραφικού πληροφοριακού συστήματος.

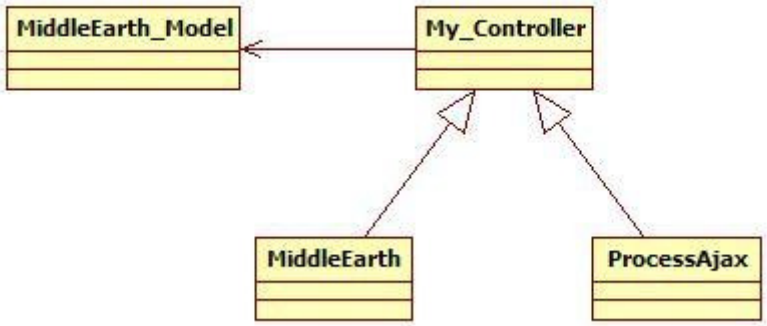


Εικόνα 34 - Διάγραμμα περιπτώσεων χρήσης (δεύτερη τελική έκδοση)

4.5.2 Διαγράμματα Κλάσεων (Class Diagrams)

Τα διαγράμματα κλάσεων μοντελοποιούν τη στατική δομή του συστήματος και περιλαμβάνουν κλάσεις (classes) που αντιπροσωπεύουν οντότητες με τα χαρακτηριστικά (attributes) και τις λειτουργίες (operations) τους και φυσικά συσχετισμούς μεταξύ των κλάσεων.

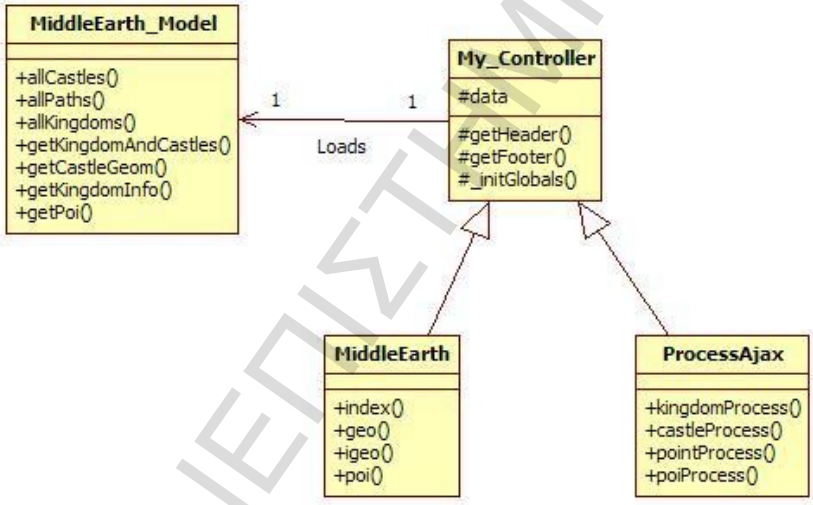
Οι κλάσεις της εφαρμογής είναι οι MiddleEarth_Model (το μοντέλο), MY_Controller (ο εμπρόσθιος ελεγκτής), MiddleEarth (ελεγκτής), ProcessAjax (ελεγκτής). Ακολουθεί το αρχικό διάγραμμα κλάσεων όπου ορίζεται σχέση γενίκευσης μεταξύ της κλάσης MY_Controller και των κλάσεων MiddleEarth και ProcessAjax οι οποίες κληρονομούν από αυτή. Η κλάση MY_Controller αλληλεπιδρά με την κλάση MiddleEarth_Model όπως και οι υποκλάσεις MiddleEarth και ProcessAjax λόγω κληρονομικότητας.



Εικόνα 35 - Διάγραμμα κλάσεων (πρώτη έκδοση)

Στην εικόνα 36 (δεύτερη και τελική έκδοση) έχουν εισαχθεί τα χαρακτηριστικά και οι λειτουργίες των κλάσεων καθώς και οι ορατότητες αυτών λαμβάνοντας υπόψη τη βασική αρχή της ενθυλάκωσης που διέπει την αντικειμενοστρεφή ανάπτυξη. Το χαρακτηριστικό data της κλάσης MY_Controller ορίστηκε ως προστατευόμενο (protected) ώστε να μπορεί να προσπελαστεί και από τις υποκλάσεις της. Οι λειτουργίες της ορίστηκαν, για τον ίδιο ακριβώς λόγο, ως προστατευόμενες. Όλες οι λειτουργίες της κλάσης MiddleEarth_Model και των MiddleEarth και ProcessAjax ορίστηκαν ως δημόσιες (public).

Στο συσχετισμό μεταξύ των κλάσεων MY_Controller και MiddleEarth_Model έχει οριστεί η πληθικότητα, 1 προς 1, και έχει δοθεί το όνομα «Loads» που σημαίνει ότι η κλάση (ελεγκτής) MY_Controller αναλαμβάνει να φορτώσει τη κλάση (μοντέλο) MiddleEarth_Model.



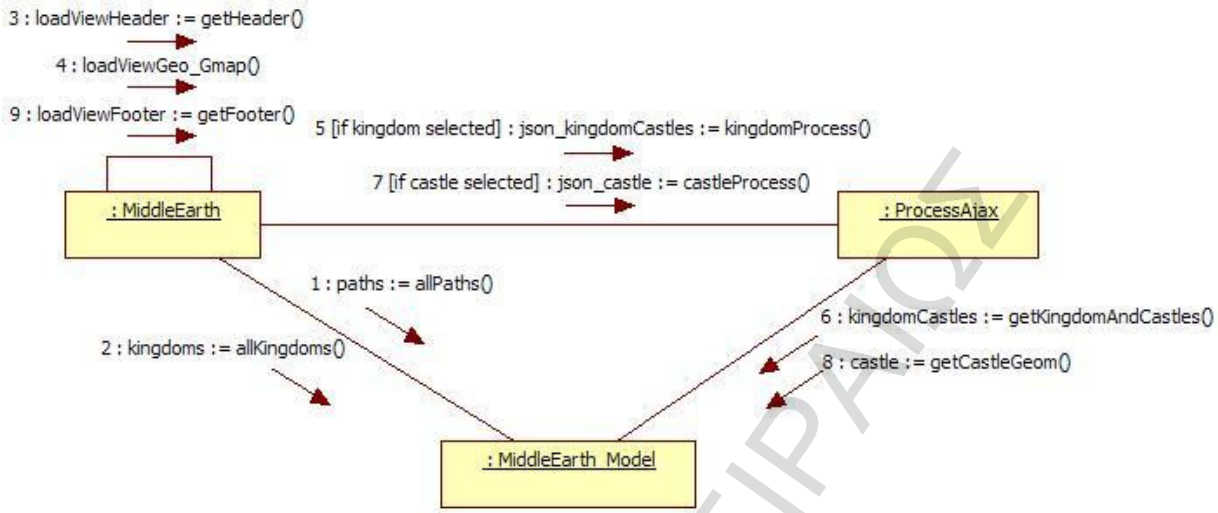
Εικόνα 36 - Διάγραμμα κλάσεων (δεύτερη τελική έκδοση)

4.5.3 Διαγράμματα Συνεργασίας (Collaboration Diagrams)

Τα διαγράμματα συνεργασίας απεικονίζουν τις αλληλεπιδράσεις ανάμεσα στα αντικείμενα και συνεπώς μοντελοποιούν τα δυναμικά χαρακτηριστικά του συστήματος.

Ακολουθούν τα διαγράμματα συνεργασίας για κάθε μια από τις τρείς βασικές λειτουργίες που προσφέρει η εφαρμογή.

Α) Γεωκωδικοποίηση (Geocoding)



Εικόνα 37 - Διάγραμμα συνεργασίας – Γεωκωδικοποίηση

1. Το στιγμιότυπο του MiddleEarth (ελεγκτής) ανακτά από το στιγμιότυπο του MiddleEarth_Model (μοντέλο – διαχειρίζεται τη βάση middle_earth) όλα τα μονοπάτια εκτελώντας τη μέθοδο allPaths.

2. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα βασίλεια εκτελώντας τη μέθοδο allKingdoms.

3. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getHeader ώστε να φορτωθεί το πάνω μέρος της ιστοσελίδας (η όψη header).

4. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο loadViewGeo_Gmap ώστε να φορτωθεί το κεντρικό μέρος της ιστοσελίδας (η όψη geo_Gmap).

5. Το στιγμιότυπο του MiddleEarth, σε περίπτωση που επιλέχθηκε κάποιο βασίλειο, εκτελεί τη μέθοδο kingdomProcess του στιγμιότυπου του ProcessAjax (ελεγκτής) με αποτέλεσμα την επιστροφή του επιλεγμένου βασιλείου και των κάστρων που αυτό περιλαμβάνει σε μορφή JSON.

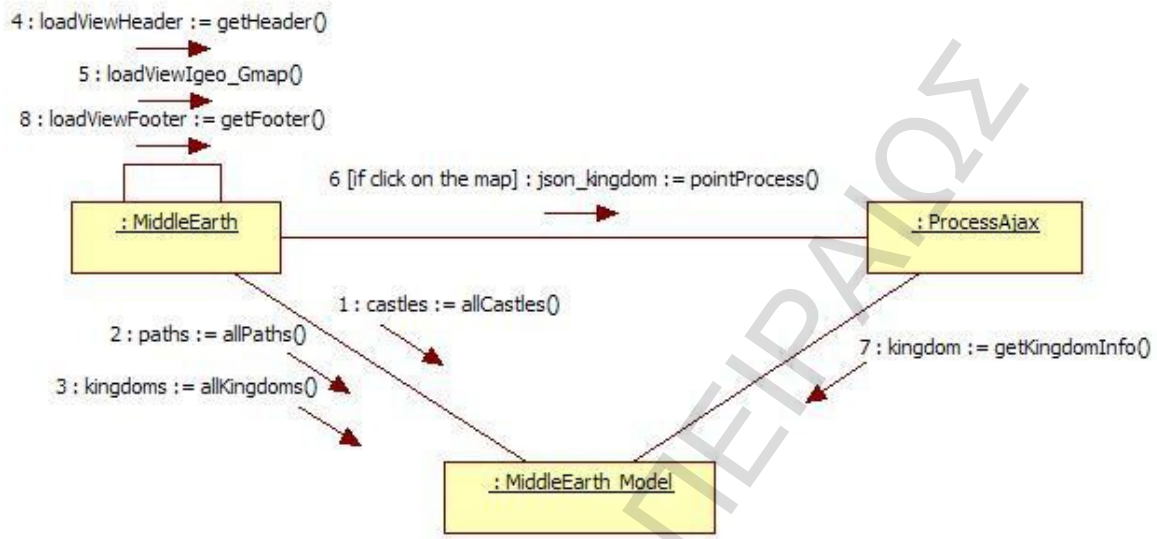
6. Το στιγμιότυπο του ProcessAjax εκτελεί με τη σειρά του τη μέθοδο getKingdomAndCastles του στιγμιότυπου του MiddleEarth_Model με αποτέλεσμα να λάβει το βασίλειο και τα κάστρα που αυτό περιλαμβάνει.

7. Το στιγμιότυπο του MiddleEarth, σε περίπτωση που επιλέχθηκε κάποιο κάστρο, εκτελεί τη μέθοδο castleProcess του στιγμιότυπου του ProcessAjax με αποτέλεσμα την επιστροφή του επιλεγμένου κάστρου σε μορφή JSON.

8. Το στιγμιότυπο του ProcessAjax εκτελεί με τη σειρά του τη μέθοδο getCastleGeom του στιγμιότυπου του MiddleEarth_Model με αποτέλεσμα να λάβει το κάστρο.

9. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getFooter ώστε να φορτωθεί το κάτω μέρος της ιστοσελίδας (η όψη footer).

Β) Αντίστροφη γεωκωδικοποίηση (Reverse geocoding)



Εικόνα 38 - Διάγραμμα συνεργασίας – Αντίστροφη γεωκωδικοποίηση

1. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα κάστρα εκτελώντας τη μέθοδο allCastles.

2. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα μονοπάτια εκτελώντας τη μέθοδο allPaths.

3. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα βασίλεια εκτελώντας τη μέθοδο allKingdoms.

4. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getHeader ώστε να φορτωθεί το πάνω μέρος της ιστοσελίδας (η όψη header).

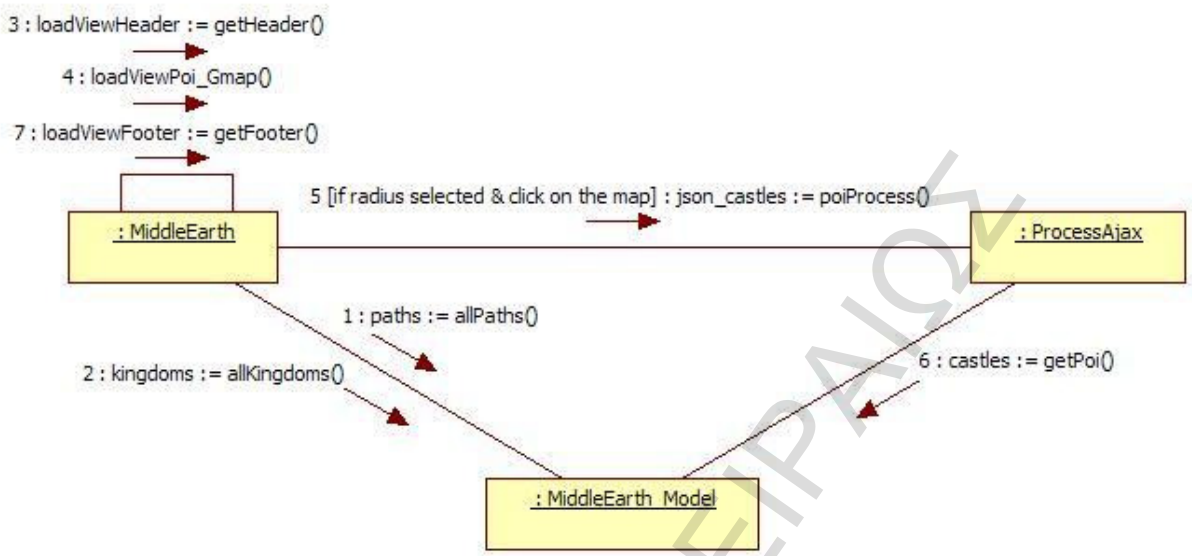
5. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο loadViewIgeo_Gmap ώστε να φορτωθεί το κεντρικό μέρος της ιστοσελίδας (η όψη igeo_Gmap).

6. Το στιγμιότυπο του MiddleEarth, σε περίπτωση που επιλέχθηκε κάποιο σημείο πάνω στο χάρτη, εκτελεί τη μέθοδο pointProcess του στιγμιότυπου του ProcessAjax με αποτέλεσμα την επιστροφή του επιλεγμένου βασιλείου σε μορφή JSON.

7. Το στιγμιότυπο του ProcessAjax εκτελεί με τη σειρά του τη μέθοδο getKingdomInfo του στιγμιότυπου του MiddleEarth_Model με αποτέλεσμα να λάβει το βασίλειο.

8. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getFooter ώστε να φορτωθεί το κάτω μέρος της ιστοσελίδας (η όψη footer).

Γ) Εύρεση σημείων ενδιαφέροντος (POI detection)



Εικόνα 39 - Διάγραμμα συνεργασίας – Εύρεση σημείων ενδιαφέροντος

1. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα μονοπάτια εκτελώντας τη μέθοδο allPaths.

2. Το στιγμιότυπο του MiddleEarth ανακτά από το στιγμιότυπο του MiddleEarth_Model όλα τα βασίλεια εκτελώντας τη μέθοδο allKingdoms.

3. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getHeader ώστε να φορτωθεί το πάνω μέρος της ιστοσελίδας (η όψη header).

4. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο loadViewPoi_Gmap ώστε να φορτωθεί το κεντρικό μέρος της ιστοσελίδας (η όψη poi_Gmap).

5. Το στιγμιότυπο του MiddleEarth, σε περίπτωση που επιλέχθηκε ακτίνα και σημείο πάνω στο χάρτη, εκτελεί τη μέθοδο poiProcess του στιγμιότυπου του ProcessAjax με αποτέλεσμα την επιστροφή των κάστρων σε μορφή JSON.

6. Το στιγμιότυπο του ProcessAjax εκτελεί με τη σειρά του τη μέθοδο getPoi του στιγμιότυπου του MiddleEarth_Model με αποτέλεσμα να λάβει τα κάστρα που βρίσκονται γύρω από το σημείο του χάρτη που επιλέχθηκε και εμπίπτουν σε αυτή την ακτίνα.

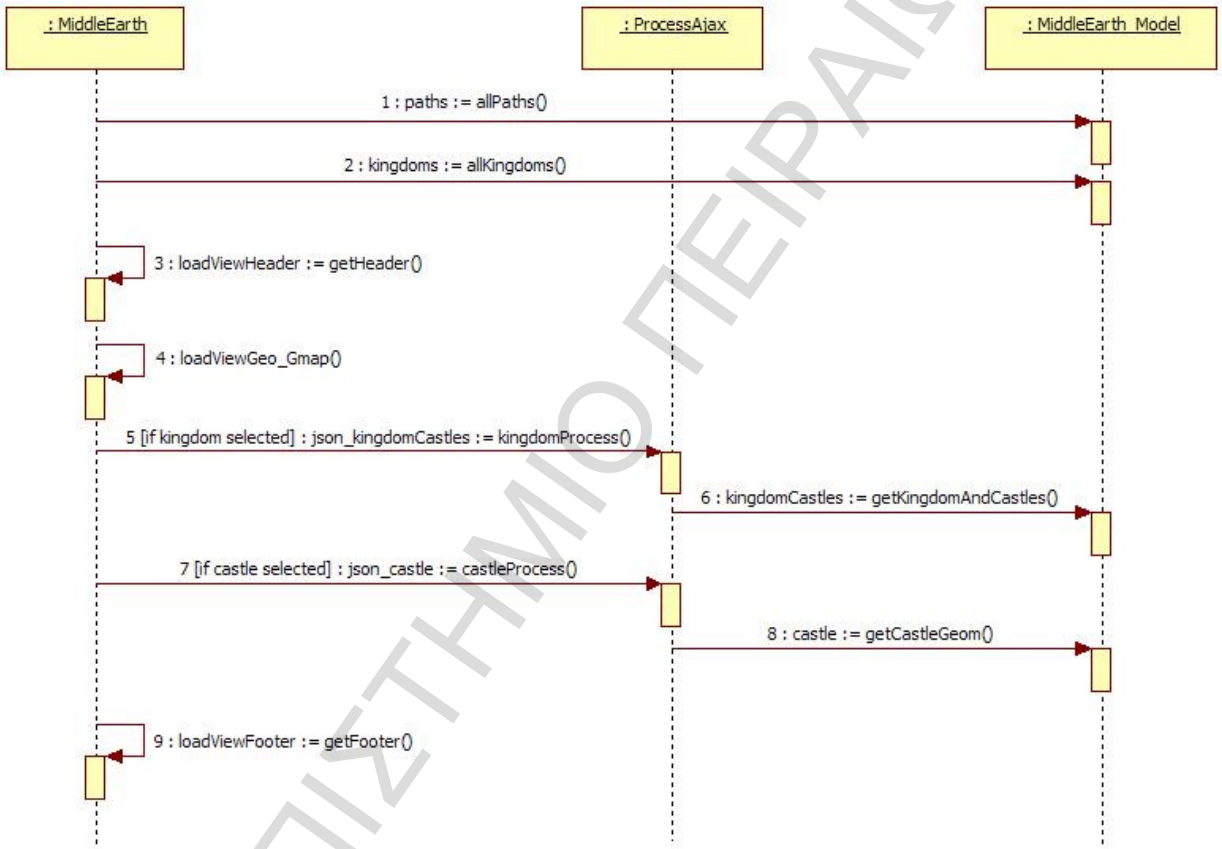
7. Το στιγμιότυπο του MiddleEarth εκτελεί τη δική του μέθοδο getFooter ώστε να φορτωθεί το κάτω μέρος της ιστοσελίδας (η όψη footer).

4.5.4 Διαγράμματα Σειράς (Sequence Diagrams)

Τα διαγράμματα σειράς απεικονίζουν τις αλληλεπιδράσεις ανάμεσα στα αντικείμενα από μια χρονική άποψη. Η αναπαράσταση επικεντρώνεται στην έκφραση των αλληλεπιδράσεων. Συνεπώς μοντελοποιούν τα δυναμικά χαρακτηριστικά του συστήματος.

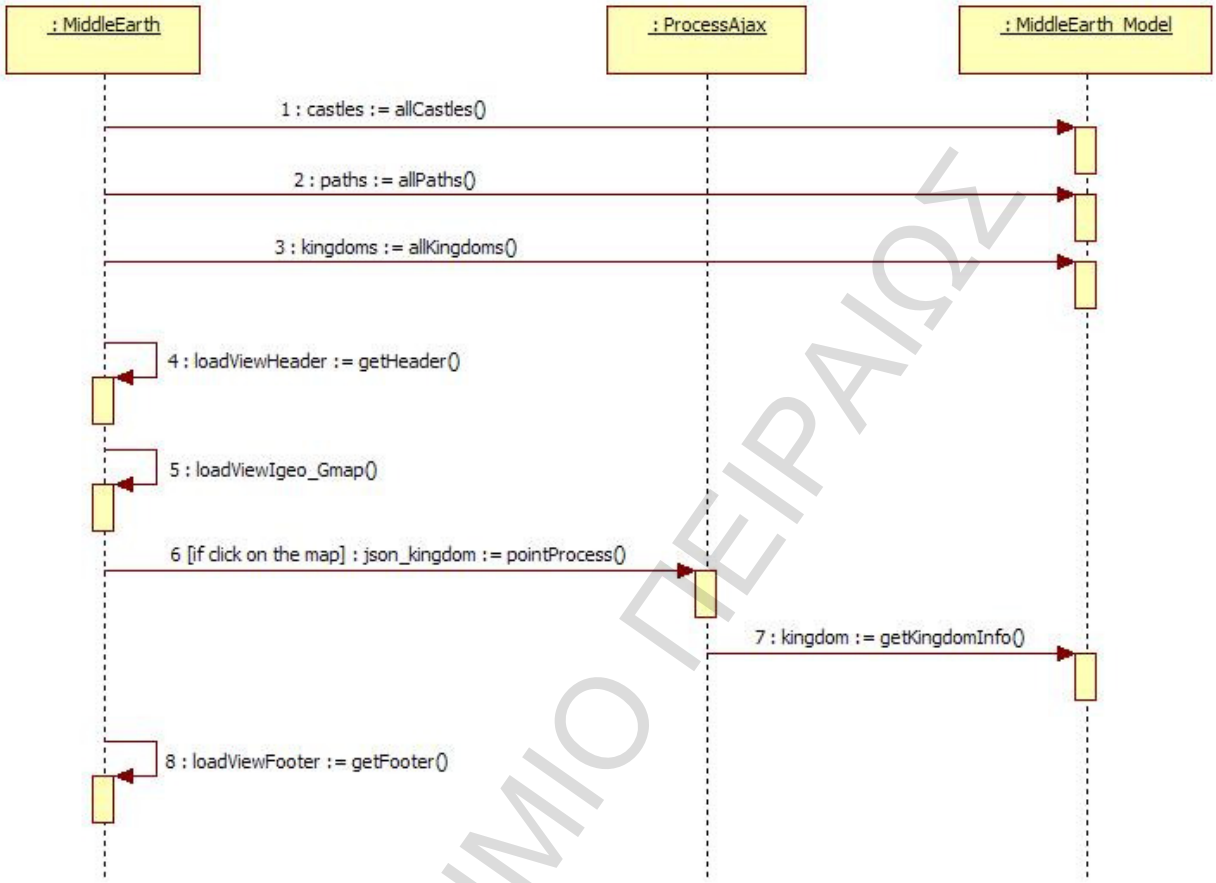
Ακολουθούν τα διαγράμματα σειράς για κάθε μία από τις τρείς βασικές λειτουργίες που προσφέρει η εφαρμογή. Η λογική είναι η ίδια με αυτή των αντίστοιχων διαγραμμάτων συνεργασίας.

Α) Γεωκωδικοποίηση (Geocoding)



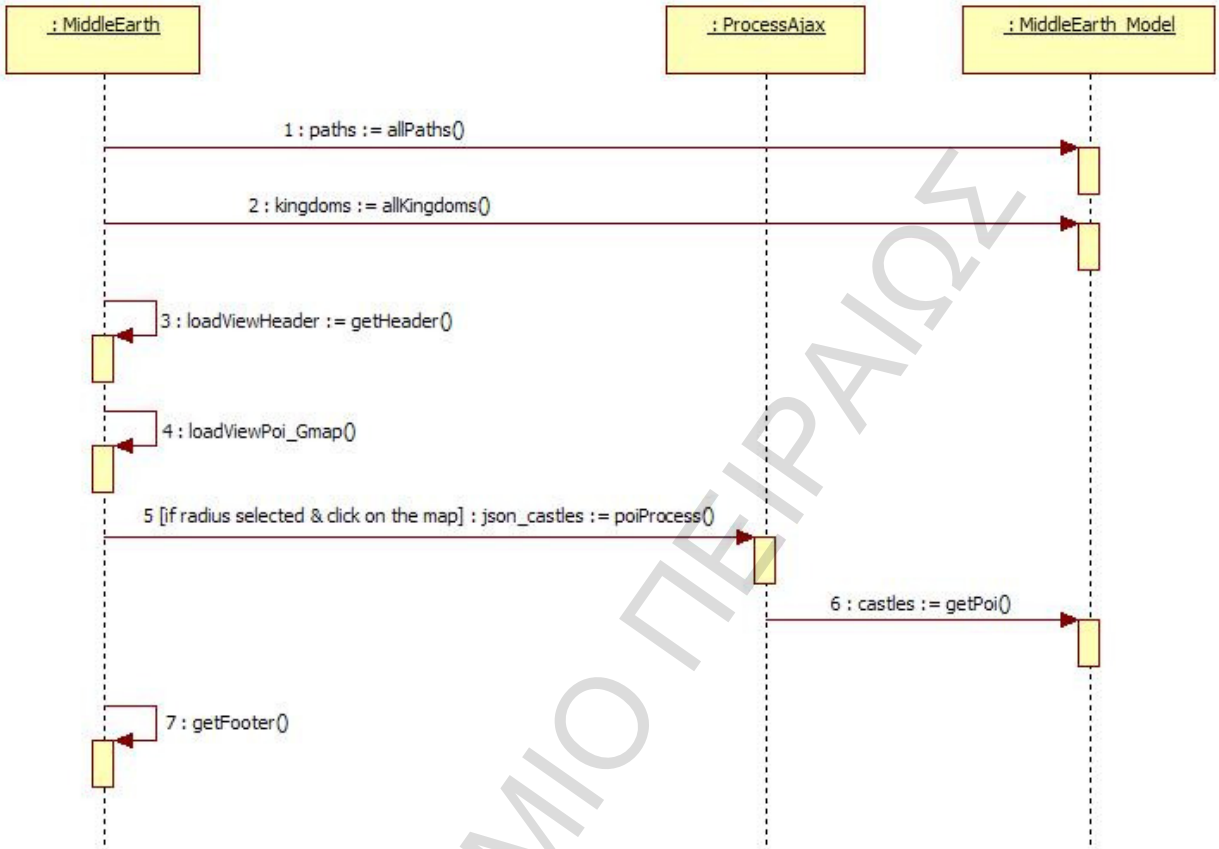
Εικόνα 40 - Διάγραμμα σειράς – Γεωκωδικοποίηση

Β) Αντίστροφη γεωκωδικοποίηση (Reverse geocoding)



Εικόνα 41 - Διάγραμμα σειράς – Αντίστροφη γεωκωδικοποίηση

Γ) Εύρεση σημείων ενδιαφέροντος (POI detection)

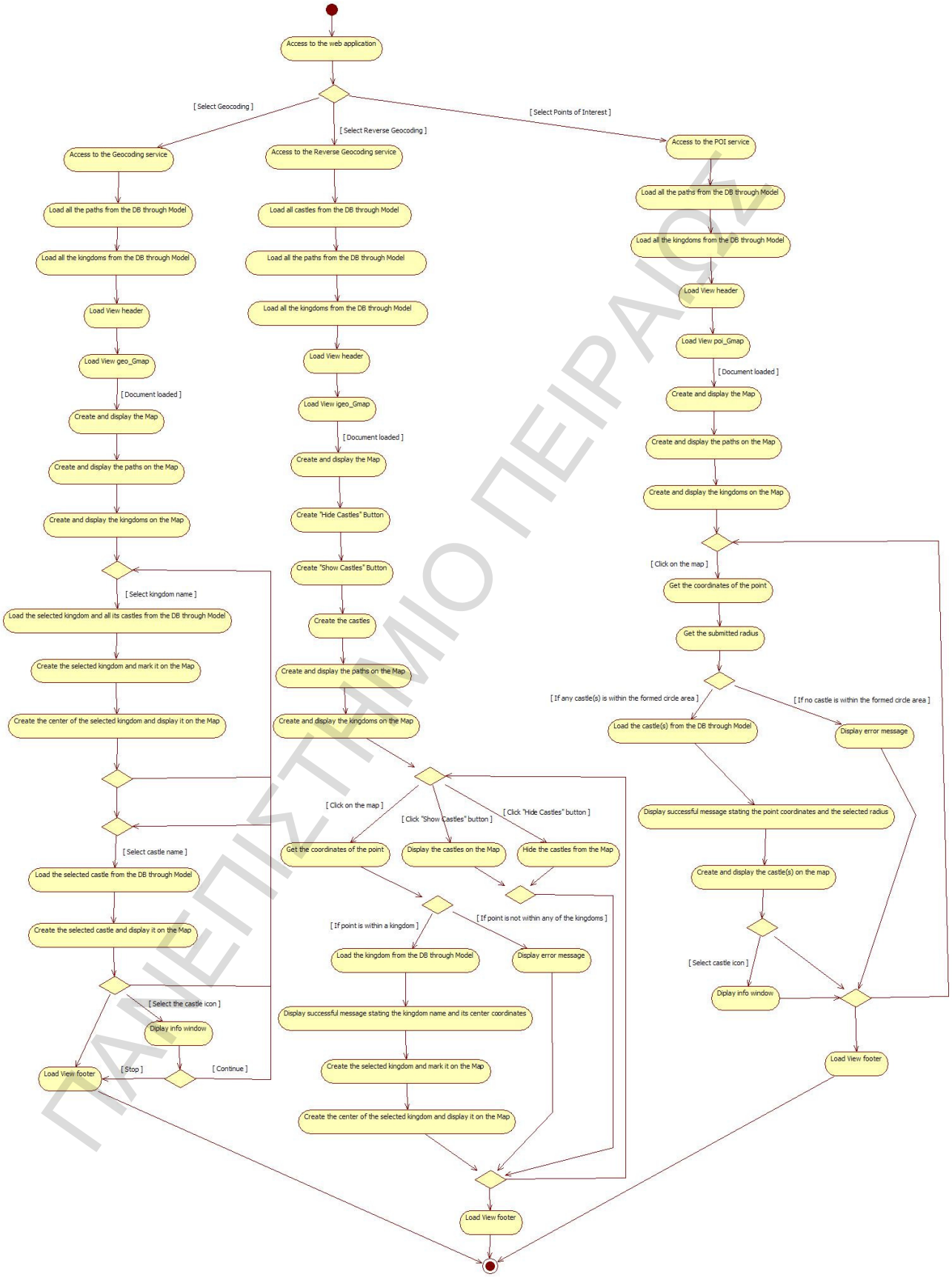


Εικόνα 42 - Διάγραμμα σειράς – Εύρεση σημείων ενδιαφέροντος

4.5.5 Διαγράμματα Δραστηριοτήτων (Activity Diagrams)

Τα διαγράμματα δραστηριοτήτων μοντελοποιούν τα βήματα εκτέλεσης (ενέργειες) μιας διαδικασίας. Στοχεύουν κυρίως στην αναπαράσταση της εσωτερικής συμπεριφοράς των διαδικασιών της εφαρμογής.

Ακολουθεί το διάγραμμα δραστηριοτήτων (εικόνα 43) το οποίο είναι απόλυτα αναλυτικό και μοντελοποιεί τα βήματα εκτέλεσης των τριών βασικών λειτουργιών που η εφαρμογή προσφέρει.

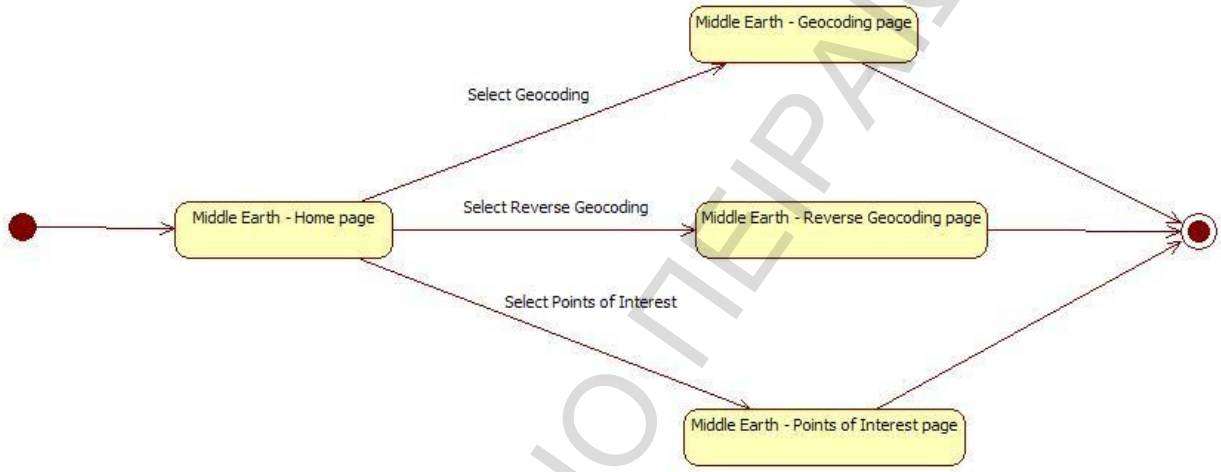


Εικόνα 43 - Διάγραμμα δραστηριοτήτων

4.5.6 Διαγράμματα Καταστάσεων (Statechart Diagrams)

Τα διαγράμματα καταστάσεων μοντελοποιούν τη συμπεριφορά συγκεκριμένων αντικειμένων αναπαριστώντας μηχανές καταστάσεων από την άποψη των καταστάσεων και των μεταβάσεων. Ένα ή περισσότερα γεγονότα ενεργοποιούν μια μετάβαση από μια κατάσταση ενός αντικειμένου σε μια άλλη κατάσταση.

Ακολουθεί το διάγραμμα καταστάσεων (εικόνα 44).



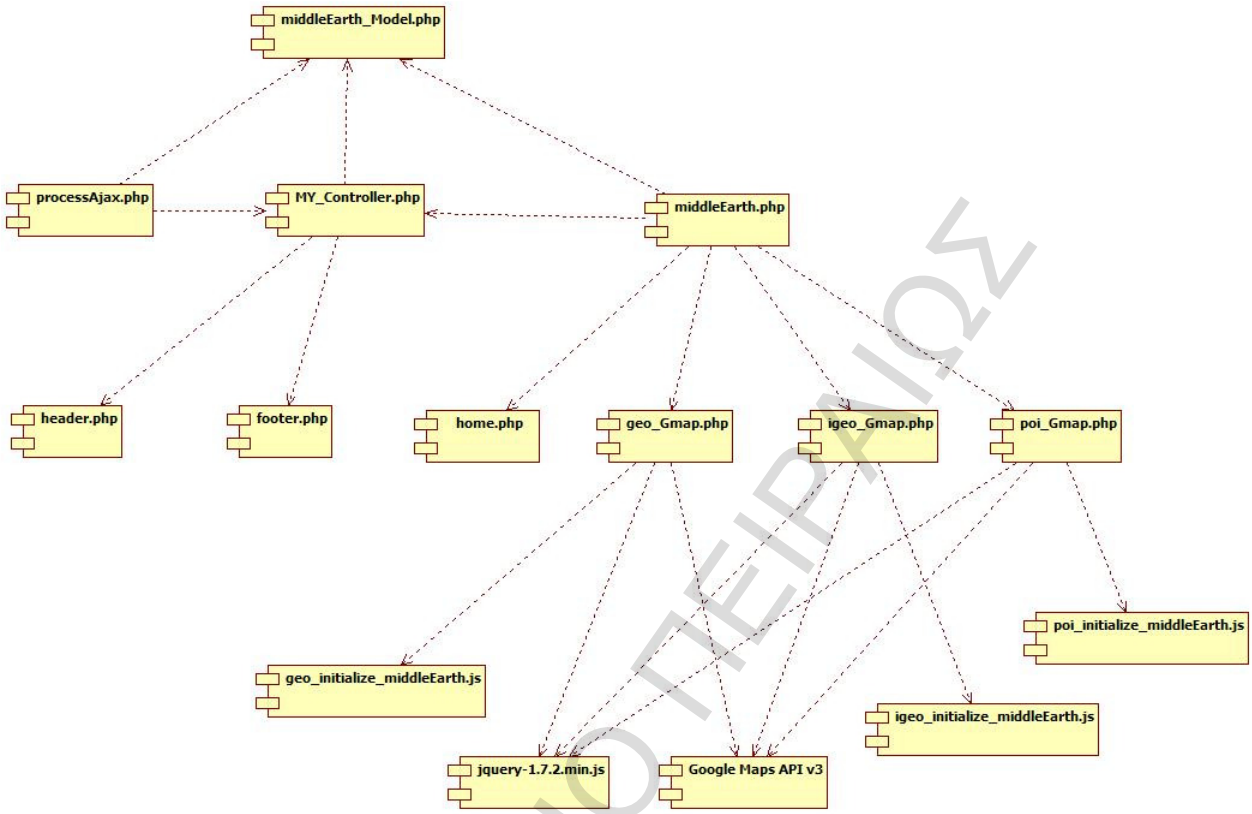
Εικόνα 44 - Διάγραμμα καταστάσεων

Αρχικά η εφαρμογή βρίσκεται στην αρχική κατάσταση που είναι η «Home page». Όταν επιλεγεί η λειτουργία «Geocoding» τότε η εφαρμογή αλλάζει κατάσταση και εισέρχεται στη «Geocoding page». Όταν επιλεγεί η λειτουργία «Reverse Geocoding» τότε η εφαρμογή αλλάζει κατάσταση και εισέρχεται στη «Reverse Geocoding page». Όταν επιλεγεί η λειτουργία «Points of Interest» τότε η εφαρμογή αλλάζει κατάσταση και εισέρχεται στη «Points of Interest page».

4.5.7 Διαγράμματα Εξαρτημάτων (Component Diagrams)

Τα διαγράμματα εξαρτημάτων περιγράφουν τμήματα της εφαρμογής και τη σχέση τους με το περιβάλλον υλοποίησης. Υποδηλώνουν τις επιλογές που γίνονται κατά τη χρονική στιγμή της υλοποίησης. Οι σχέσεις εξάρτησης χρησιμοποιούνται στα διαγράμματα εξαρτημάτων για να δηλώσουν ότι ένα εξάρτημα αναφέρεται σε υπηρεσίες που προσφέρονται από άλλα εξαρτήματα. Αυτός ο τύπος εξάρτησης ανακλά επιλογές υλοποίησης. Μία σχέση εξάρτησης αναπαρίσταται με ένα βέλος με διακεκομμένη γραμμή σχεδιασμένο από τον πελάτη προς τον προμηθευτή.

Ακολουθεί το διάγραμμα εξαρτημάτων (εικόνα 45).



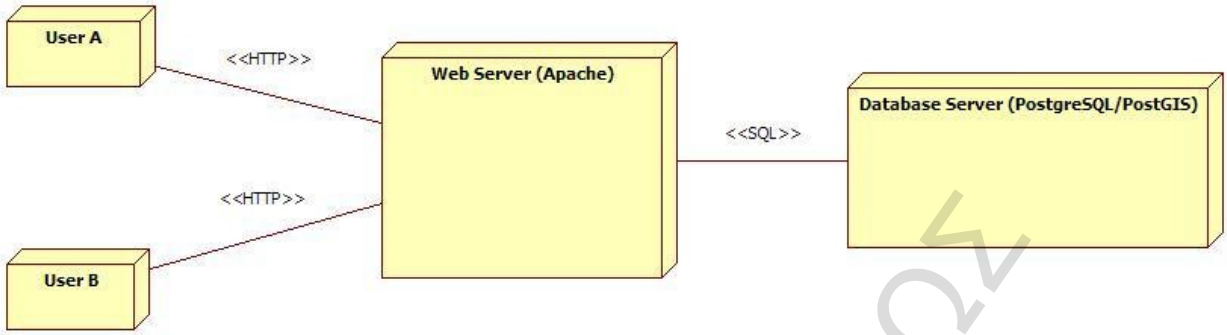
Εικόνα 45 - Διάγραμμα εξαρτημάτων

Το αρχείο MY_Controller.php (εμπρόσθιος ελεγκτής) όπως και τα αρχεία processAjax.php και middleEarth.php, λαμβάνουν υπηρεσίες από το αρχείο middleEarth_Model.php (μοντέλο) το οποίο προσφέρει υπηρεσίες ανάκτησης δεδομένων από τη βάση. Τα αρχεία processAjax.php και middleEarth.php λαμβάνουν υπηρεσίες από την υπερκλάση τους, MY_Controller.php, λόγω κληρονομικότητας. Το αρχείο MY_Controller.php λαμβάνει υπηρεσίες εμφάνισης περιεχομένου του πάνω και κάτω μέρους των ιστοσελίδων από τα αντίστοιχα αρχεία header.php και footer.php που είναι όψεις. Το αρχείο middleEarth.php λαμβάνει υπηρεσίες εμφάνισης περιεχομένου και χάρτη από τα αρχεία home.php, geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php που είναι όψεις και τα οποία λαμβάνουν υπηρεσίες από τα αντίστοιχα τους αρχεία javascript.

4.5.8 Διαγράμματα Διανομής (Deployment Diagrams)

Τα διαγράμματα διανομής δείχνουν τη φυσική διάταξη των διαφόρων εξαρτημάτων (κόμβων) υλικού που αποτελούν το σύστημα όπως και τη διανομή των εκτελέσιμων προγραμμάτων σε αυτό το υλικό. Κάθε πόρος υλικού αναπαρίσταται με κύβο.

Ακολουθεί το διάγραμμα διανομής.



Εικόνα 46 - Διάγραμμα διανομής

Όπως φαίνεται στην εικόνα 46, οι χρήστες αποκτούν πρόσβαση στην εφαρμογή, η οποία φιλοξενείται στον εξυπηρετητή Apache, μέσω του πρωτοκόλλου HTTP. Η εφαρμογή επικοινωνεί με τη βάση δεδομένων, η οποία φιλοξενείται στον εξυπηρετητή βάσης PostgreSQL/PostGIS, μέσω SQL ερωτημάτων.

4.6 Υλοποίηση της εφαρμογής

Η διαδικτυακή εφαρμογή υλοποιήθηκε με χρήση του πλαισίου εργασίας CodeIgniter και για τη φιλοξενία της χρησιμοποιήθηκε ο εξυπηρετητής Apache για τρείς βασικούς λόγους. Υποστηρίζει PHP, είναι δημοφιλής, περισσότερο από το 50% των ιστοχώρων του παγκόσμιου ιστού χρησιμοποιούν τον Apache ως εξυπηρετητή και χρησιμοποιείται σε διάφορα λειτουργικά συστήματα όπως Linux, Unix, Microsoft Windows, Mac Os X.

Έγινε χρήση της προγραμματιστικής διεπαφής Google Maps Javascript API v3 για την απεικόνιση/οπτικοποίηση των δεδομένων της βάσης σε χάρτη. Για την ανάπτυξη του στατικού (markup) μέρους της εφαρμογής χρησιμοποιήθηκε το Twitter Bootstrap.

4.6.1 Εγκατάσταση του XAMPP και του CodeIgniter

Αρχικά εγκαταστάθηκε το XAMPP παρέχοντας έτσι μία ολοκληρωμένη πλατφόρμα με Apache και PHP για την ανάπτυξη της διαδικτυακής εφαρμογής.

Εγκαταστάθηκε στη συνέχεια το CodeIgniter στο κύριο (web root) κατάλογο του XAMPP, το htdocs. Ο βασικός (root) κατάλογος του CodeIgniter, που περιέχει το βασικό αρχείο index.php και τους καταλόγους application, system και user_guide, μετονομάστηκε σε ptyxiaki. Μέσα σε αυτό τον κατάλογο, δημιουργήθηκε ένας νέος κατάλογος με όνομα public_html και μεταφέρθηκε εκεί το βασικό αρχείο index.php. Για το λόγο αυτό έγιναν οι κατάλληλες αλλαγές στο index.php όσον αφορά τον επαναπροσδιορισμό των μονοπατιών των καταλόγων system και application. Επιπλέον, στο κατάλογο public_html δημιουργήθηκε το αρχείο .htaccess το οποίο περιέχει κανόνες που εξαλείφουν την ανάγκη για εισαγωγή του index.php στο URL όπως θα συνέβαινε κανονικά. Ακολούθησε στη συνέχεια η σύνδεση του CodeIgniter με τη βάση δεδομένων middle_earth με τη κατάλληλη συμπλήρωση των απαραίτητων πληροφοριών στο αρχείο database.php Στο αρχείο config.php καθορίστηκε το βασικό (root) μονοπάτι της εφαρμογής με την ακόλουθη τιμή

```
$config['base_url'] = 'http://localhost/ptyxiaki/public_html/';
```

Για την πρόσβαση σε αυτό μέσα από τις όψεις της εφαρμογής, αρκεί μόνο η μεταβλητή $baseUrl (θα εξηγηθεί στη συνέχεια, στην ενότητα με τους ελεγκτές).

4.6.2 Δημιουργία του μοντέλου (model)

Επόμενο βήμα ήταν η δημιουργία της κλάσης για το μοντέλο. Έτσι λοιπόν, δημιουργήθηκε το αρχείο middleEarth_Model.php, στο μονοπάτι application/models/, το οποίο περιέχει την κλάση MiddleEarth_Model (η οποία επεκτείνει την υπερκλάση CI_Model) και τις μεθόδους της για την ανάκτηση γεωγραφικών και μη δεδομένων από τη βάση middle_earth.

Για παράδειγμα η μέθοδος allCastles εκτελεί ένα ερώτημα στη βάση δεδομένων της εφαρμογής ανακτώντας τα πεδία όλων των κάστρων και έπειτα επιστρέφεται το αποτέλεσμα σε μορφή πίνακα. Με την ίδια λογική έχουν υλοποιηθεί και οι υπόλοιπες μέθοδοι της κλάσης MiddleEarth_Model.

4.6.3 Δημιουργία των όψεων (views)

Επόμενο βήμα ήταν η δημιουργία των όψεων για τον προσδιορισμό του περιεχομένου του πάνω (header) και κάτω (footer) μέρους των ιστοσελίδων της εφαρμογής. Για το λόγο αυτό δημιουργήθηκαν τα αντίστοιχα αρχεία header.php και footer.php, στο μονοπάτι application/views/templates/, τα οποία περιλαμβάνουν HTML κώδικα. Το πάνω μέρος (header) περιλαμβάνει το κεντρικό μενού για την πλοήγηση του χρήστη σε κάθε λειτουργία που η εφαρμογή υποστηρίζει. Το κάτω μέρος (footer) περιέχει πληροφορίες σχετικά με τον δημιουργό της εφαρμογής.

Εκτός από κώδικα HTML, υπάρχει και κώδικας PHP ο οποίος εκτυπώνει το κύριο μονοπάτι (root path) της εφαρμογής, που όπως αναφέρθηκε προηγουμένως έχει ανατεθεί στη μεταβλητή $baseUrl και επιπλέον τις αντίστοιχες τιμές των μεταβλητών $layout_normalActive, $layout_inverseActive, $layout_poiActive (μόνο μία από αυτές περιέχει κάθε φορά τη τιμή 'active' όπως θα εξηγηθεί στη συνέχεια, στην ενότητα με τους ελεγκτές).

Στη συνέχεια δημιουργήθηκαν οι απαραίτητες όψεις για τον προσδιορισμό του περιεχομένου του κυρίως μέρους των ιστοσελίδων της εφαρμογής. Για το λόγο αυτό δημιουργήθηκαν τα αντίστοιχα αρχεία home.php, geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php στο μονοπάτι application/views/middle_earth/, τα οποία περιλαμβάνουν κώδικα HTML.

4.6.4 Δημιουργία των ελεγκτών (controllers)

Στη συνέχεια, δημιουργήθηκε ο εμπρόσθιος ελεγκτής (front controller), MY_Controller ο οποίος συνιστά μια κλάση η οποία επεκτείνει την υπερκλάση CI_Controller. Το αντίστοιχο αρχείο, MY_Controller.php αποθηκεύτηκε στο μονοπάτι application/core/.

Ο MY_Controller συνίσταται από:

- Ένα πεδίο $data, τύπου πίνακα με προστατευόμενη ορατότητα (protected), για την αποθήκευση δεδομένων που πρέπει να περαστούν με δυναμικό τρόπο στις όψεις. Στην εφαρμογή χρησιμοποιείται ως συσχετιζόμενος πίνακας με τη μορφή ζευγαριών κλειδιού (key) – τιμής (value). Με τη δυναμική εισαγωγή του σε κάποια όψη, κάθε τιμή ανακτάται με βάση το αντίστοιχο κλειδί μετασχηματισμένο ως απλή μεταβλητή php. Για παράδειγμα, αν στον ελεγκτή υπάρχει η δήλωση `$this->data['title'] = 'Paradeigma'` και περαστεί ο πίνακας δυναμικά σε κάποια όψη, τότε στην όψη θα ανακτηθεί η τιμή ως εξής `<?php echo $title ?>`

- Ένα κατασκευαστή (constructor), ο οποίος εκτελείται αυτόματα κατά την αρχικοποίηση της κλάσης επιτελώντας κάποιες βασικές λειτουργίες. Πρώτα απ' όλα εκτελεί τον κατασκευαστή της υπερκλάσης CI_Controller για τη σωστή αρχικοποίησή της. Έπειτα, συνδέεται στη βάση δεδομένων, φορτώνει το middleEarth_Model και το helper url το οποίο περιέχει συναρτήσεις για τη διαχείριση των URLs. Τέλος, εκτελεί τη μέθοδο _initGlobals

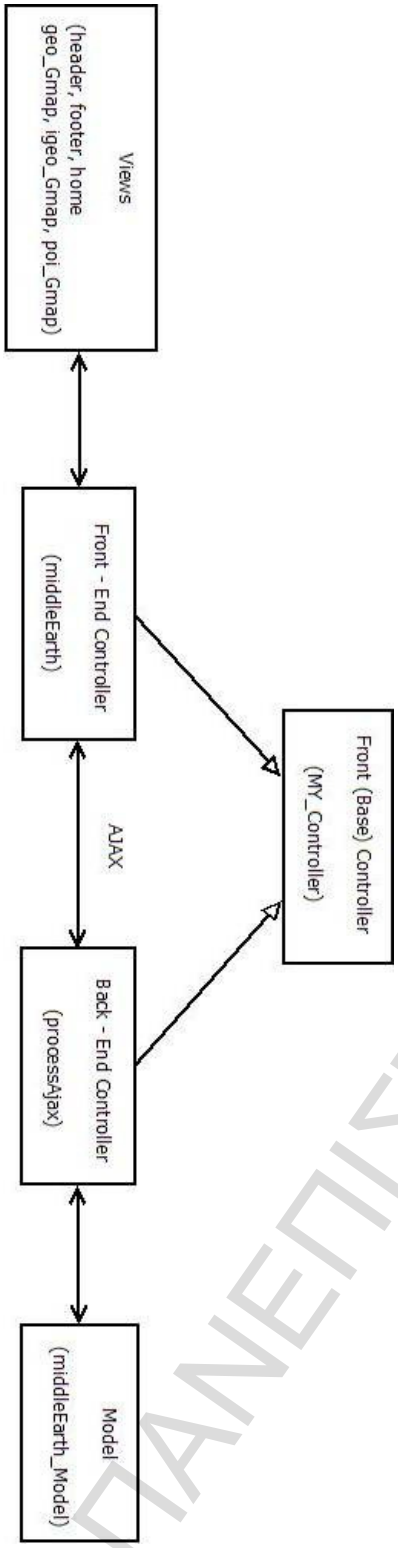
- Τις δημόσιες (public) μεθόδους getHeader και getFooter οι οποίες απλώς φορτώνουν τις όψεις header και footer αντίστοιχα περνώντας τους επίσης τον πίνακα $data δυναμικά.

- Τη δημόσια μέθοδο _initGlobals η οποία απλώς αποθηκεύει το κύριο μονοπάτι (root path) της εφαρμογής στον πίνακα $data με κλειδί 'baseUrl'. Ουσιαστικά, γίνεται χρήση της συνάρτησης base_url (βασική συνάρτηση του helper url) η οποία επιστρέφει το κύριο μονοπάτι όπως έχει καθοριστεί στο αρχείο config.php.

Ακολούθησε η δημιουργία των ελεγκτών MiddleEarth και ProcessAjax, οι οποίοι συνιστούν κλάσεις που επεκτείνουν τον MY_Controller και ως εκ τούτου κληρονομούν το πεδίο και τις μεθόδους του. Τα αντίστοιχα αρχεία middleEarth.php και processAjax.php αποθηκεύτηκαν στο μονοπάτι application/controllers/.

Γνωρίζοντας ότι ένας ελεγκτής αποτελεί το μεσολαβητή μεταξύ των όψεων και του μοντέλου αποφασίστηκε η δημιουργία των δύο παραπάνω ελεγκτών για την αλληλεπίδραση του ενός (MiddleEarth) με τις όψεις και την αλληλεπίδραση του δεύτερου (ProcessAjax) με το μοντέλο. Έτσι, ο MiddleEarth αποτελεί τον αρχικό (front-end) ελεγκτή και ο ProcessAjax τον τελικό (back-end) ελεγκτή. Η επικοινωνία μεταξύ τους, επιτυγχάνεται με χρήση της τεχνολογίας AJAX.

Η δομή της αρχιτεκτονικής που περιγράφηκε απεικονίζεται στο σχήμα της εικόνας 47.



Εικόνα 47 - Δομή αρχιτεκτονικής της εφαρμογής

Ο MiddleEarth συνίσταται από:

- Ένα κατασκευαστή (constructor), ο οποίος εκτελείται αυτόματα κατά την αρχικοποίηση της κλάσης. Πρώτα απ' όλα εκτελεί τον κατασκευαστή της υπερκλάσης MY_Controller για τη σωστή αρχικοποίησή της. Έπειτα, αρχικοποιεί με κενές τιμές τα δεδομένα του συσχετιζόμενου πίνακα, layout_normalActive, layout_inverseActive και layout_poiActive.

- Τέσσερις δημόσιες μεθόδους, index, geo, igeo και poi οι οποίες προσφέρουν την εμπρόσθια (front-end) λειτουργικότητα της εφαρμογής.
    - Η index παρέχει την αρχική σελίδα και συνίσταται από τις όψεις header, home και footer.
    - Η geo παρέχει τη σελίδα με τη λειτουργικότητα της γεωκωδικοποίησης και συνίσταται από τις όψεις header, geo_Gmap και footer. Παράλληλα, περνάει τη τιμή 'active' στο layout_normalActive του πίνακα $data και εκτελεί τις μεθόδους allPaths και allKingdoms του middleEarth_Model αποθηκεύοντας τα αποτελέσματα που επιστρέφουν στο πίνακα $data. Όλα τα δεδομένα γίνονται διαθέσιμα στις όψεις της με δυναμική φόρτωση. Στην όψη header η μεταβλητή $layout_normalActive έχει τη τιμή 'active'. Στην όψη geo_Gmap πραγματοποιείται τροποποίηση στον κώδικα HTML με την εισαγωγή PHP για το γέμισμα του πρώτου μενού επιλογών με όλα τα βασίλεια. Επίσης, εισάγεται κώδικας JavaScript σε συνδυασμό με PHP για την ανάθεση των γεωμετριών των μονοπατιών και των βασιλείων σε αντίστοιχους πίνακες JavaScript για τον κατάλληλο χειρισμό τους αργότερα.
    - Η igeo παρέχει τη σελίδα με τη λειτουργικότητα της αντίστροφης γεωκωδικοποίησης και συνίσταται από τις όψεις header, igeo_Gmap και footer. Παράλληλα, περνάει τη τιμή 'active' στο layout_inverseActive του πίνακα $data και εκτελεί τις μεθόδους allCastles, allPaths και allKingdoms του middleEarth_Model αποθηκεύοντας τα αποτελέσματα που επιστρέφουν στο πίνακα $data. Όλα τα δεδομένα γίνονται διαθέσιμα στις όψεις της με δυναμική φόρτωση. Στην όψη header η μεταβλητή $layout_inverseActive έχει την τιμή 'active'. Στην όψη igeo_Gmap εισάγεται JavaScript κώδικας σε συνδυασμό με PHP για την ανάθεση των γεωμετριών των κάστρων, των μονοπατιών και των βασιλείων σε αντίστοιχους πίνακες JavaScript για τον κατάλληλο χειρισμό τους αργότερα.
    - Η poi παρέχει τη σελίδα με τη λειτουργικότητα της εύρεσης σημείων ενδιαφέροντος και συνίσταται από τις όψεις header, poi_Gmap και footer. Παράλληλα, περνάει τη τιμή 'active' στο layout_poiActive του πίνακα $data και εκτελεί τις μεθόδους allPaths και allKingdoms του middleEarth_Model αποθηκεύοντας τα αποτελέσματα που επιστρέφουν στο πίνακα $data. Όλα τα δεδομένα γίνονται διαθέσιμα στις όψεις της με δυναμική φόρτωση. Στην όψη header η μεταβλητή $layout_poiActive έχει τη τιμή 'active'. Στην όψη poi_Gmap εισάγεται JavaScript κώδικας σε συνδυασμό με PHP για την ανάθεση των γεωμετριών των μονοπατιών και των βασιλείων σε αντίστοιχους πίνακες JavaScript για τον κατάλληλο χειρισμό τους αργότερα.

Ο ProcessAjax συνίσταται από:

- Τέσσερις δημόσιες μεθόδους, kingdomProcess, castleProcess, pointProcess και poiProcess οι οποίες προσφέρουν την τελική (back-end) λειτουργικότητα της εφαρμογής.

- Η kingdomProcess εξυπηρετεί τις αιτήσεις που προέρχονται από τη σελίδα με τη λειτουργικότητα της γεωκωδικοποίησης. Δέχεται την τιμή (id) του βασιλείου που στάλθηκε (μέσω post array) και στη συνέχεια εκτελεί τη συνάρτηση getKingdomAndCastles του middleEarth_Model με βάση αυτή την τιμή αποθηκεύοντας τα αποτελέσματα που επιστρέφει στο πίνακα $data. Στη συνέχεια και μετά από τον κατάλληλο έλεγχο τα περνάει σε ένα διδιάστατο πίνακα ο οποίος τελικά μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται.
- Η castleProcess εξυπηρετεί τις αιτήσεις που προέρχονται από τη σελίδα με τη λειτουργικότητα της γεωκωδικοποίησης. Δέχεται την τιμή (id) του κάστρου που στάλθηκε (μέσω post array) και στη συνέχεια εκτελεί τη συνάρτηση getCastleGeom του middleEarth_Model με βάση αυτή την τιμή αποθηκεύοντας τα αποτελέσματα που επιστρέφει στο πίνακα $data. Στη συνέχεια και μετά από τον κατάλληλο έλεγχο τα περνάει σε κατάλληλες μεταβλητές και με βάση αυτές δημιουργείται ένας νέος συσχετιζόμενος πίνακας ο οποίος τελικά μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται.
- Η pointProcess εξυπηρετεί τις αιτήσεις που προέρχονται από τη σελίδα με τη λειτουργικότητα της αντίστροφης γεωκωδικοποίησης. Δέχεται τις συντεταγμένες (γεωγραφικό μήκος και πλάτος) που στάλθηκαν (μέσω post array) και στη συνέχεια εκτελεί τη συνάρτηση getKingdomInfo του middleEarth_Model με βάση τις συντεταγμένες αποθηκεύοντας τα αποτελέσματα που επιστρέφει στο πίνακα $data. Στη συνέχεια και μετά από κατάλληλο έλεγχο τα περνάει σε απλές μεταβλητές και με βάση αυτές δημιουργείται ένας νέος συσχετιζόμενος πίνακας ο οποίος τελικά μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται. Αν δεν επιστραφούν αποτελέσματα (δηλαδή δεν επιστράφηκε κανένα βασίλειο) δημιουργείται ένας νέος συσχετιζόμενος πίνακας με τιμή κατάλληλο ενημερωτικό μήνυμα ο οποίος μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται.
- Η poiProcess εξυπηρετεί τις αιτήσεις που προέρχονται από τη σελίδα με τη λειτουργικότητα της εύρεσης σημείων ενδιαφέροντος. Δέχεται τις συντεταγμένες (γεωγραφικό μήκος και πλάτος) και την ακτίνα που στάλθηκαν (μέσω post array) και στη συνέχεια εκτελεί τη συνάρτηση getPoi του middleEarth_Model με βάση τις συντεταγμένες και την ακτίνα αποθηκεύοντας τα αποτελέσματα που επιστρέφει στο πίνακα $data. Στη συνέχεια και μετά από τον κατάλληλο έλεγχο τα περνάει σε ένα διδιάστατο πίνακα ο οποίος τελικά μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται. Αν δεν επιστραφούν αποτελέσματα (δηλαδή δεν επιστράφηκε κανένα κάστρο) δημιουργείται ένας νέος συσχετιζόμενος πίνακας με τιμή κατάλληλο ενημερωτικό μήνυμα ο οποίος μετασχηματίζεται σε μορφή JSON με τη συνάρτηση json_encode και στη συνέχεια εκτυπώνεται.

4.6.5 Ενσωμάτωση του Twitter Bootstrap

Για την κατάλληλη μορφοποίηση των όψεων και κατ' επέκταση της εμφάνισης του περιεχομένου των ιστοσελίδων χρησιμοποιήθηκε το Bootstrap το οποίο εγκαταστάθηκε στο μονοπάτι, public_html/css/. Για τη χρήση του έγινε εισαγωγή, στο <head></head> μέρος της όψης header, του παρακάτω κώδικα με τον οποίο καλείται το αρχείο bootstrap.css

```
<link href="<?php echo $baseUrl; ?>css/bootstrap/css/bootstrap.css"
rel="stylesheet" type="text/css" />
```

Έτσι λοιπόν, όλες οι όψεις τροποποιήθηκαν στη συνέχεια κατάλληλα, ώστε να χρησιμοποιούν κανόνες στυλ από το αρχείο bootstrap.css το οποίο περιλαμβάνει μεγάλο πλήθος κανόνων στυλ για ετικέτες και κλάσεις. Για κάποιους επιπλέον κανόνες στυλ, δημιουργήθηκε συμπληρωματικά το αρχείο styles.css για τη χρήση του οποίου εισήχθη στο <head></head> μέρος της όψης header ο παρακάτω κώδικας με τον οποίο καλείτε:

```
<link href="<?php echo $baseUrl; ?>css/styles.css" rel="stylesheet"
type="text/css" />
```

4.6.6 Ενσωμάτωση της βιβλιοθήκης jQuery και της προγραμματιστικής διεπαφής Google Maps JavaScript v3

Για τη διαχείριση αφενός των γεγονότων των όψεων της εφαρμογής και αφετέρου των αλληλεπιδράσεων AJAX χρησιμοποιήθηκε η βιβλιοθήκη jQuery. Το αρχείο jquery-1.7.2.min.js αποθηκεύτηκε στο μονοπάτι, public_html/js/. Για τη χρήση του έγινε εισαγωγή, στον κώδικα HTML των όψεων geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php, του παρακάτω κώδικα με τον οποίο καλείται το αρχείο jquery-1.7.2.min.js.

```
<script type="text/javascript" src="<?php echo $baseUrl; ?>js/jquery-
1.7.2.min.js"></script>
```

Η προγραμματιστική διεπαφή των Google Maps JavaScript API v3 χρησιμοποιήθηκε εκτενώς για την ενσωμάτωση χαρτών και συγκεκριμένα Google Maps στις όψεις της εφαρμογής. Μέσω των Google Maps προσφέρονται οι υπηρεσίες εμφάνισης χαρτών και χαρτογραφικών υπηρεσιών στο web που παρέχει η εφαρμογή. Για τη χρήση του, λήφθηκε ένα προσωπικό κλειδί για το API και στη συνέχεια έγινε εισαγωγή, στον κώδικα HTML των όψεων geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php, του παρακάτω κώδικα με τον οποίο καλείται διαδικτυακά:

```
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCxknxFfQBJpjt4Nk
eMv2O0q-Sld1KC5vs&sensor=true"></script>
```

4.6.7 Δημιουργία κώδικα JavaScript

Για κάθε μία από τις όψεις, geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php, που προσφέρουν χωρικές λειτουργίες, δημιουργήθηκε αντίστοιχο συμπληρωματικό αρχείο με κώδικα JavaScript. Έτσι λοιπόν, προέκυψαν τα ακόλουθα αρχεία, geo_initialize_middleEarth.js, igeo_initialize_middleEarth.js και poi_initialize_middleEarth.js, τα οποία αποθηκεύτηκαν στο μονοπάτι, public_html/js/ και αρχικοποιούν τα Google Maps μεταξύ άλλων. Τα αρχεία αυτά εκμεταλλεύονται πολλές από τις δυνατότητες που παρέχουν τόσο η βιβλιοθήκη jQuery όσο και η προγραμματιστική διεπαφή των Google Maps. Για τη χρήση τους έγινε εισαγωγή, στον κώδικα HTML των αντίστοιχων όψεων geo_Gmap.php, igeo_Gmap.php και poi_Gmap.php, του κώδικα με τον οποίο καλούνται:

```
<script type="text/javascript" src="<?php echo $baseUrl;
?>js/το_αντίστοιχο_js_αρχείο"></script>
```

Ο κώδικας για κάθε JavaScript αρχείο έχει ενσωματωθεί μεταξύ των εντολών:

```
$(document).ready(function(){  και  });
```

Δηλαδή με το που θα φορτωθεί πλήρως το DOM, θα εκτελεστεί η ανώνυμη συνάρτηση επιστροφής κλήσης (callback συνάρτηση) και ως εκ τούτου όλος ο κώδικας που περιλαμβάνει.

Και τα τρία αρχεία περιέχουν κώδικα JavaScript:

- Για τη δημιουργία και εμφάνιση των Google Maps.

- Για τη διαχείριση της εμφάνισης/απόκρυψης πληροφοριών, των αντίστοιχων όψεων, σχετικά με τις χωρικές υπηρεσίες που προσφέρουν. Αυτό επιτυγχάνεται με τη διαχείριση του γεγονότος επιλογής του πλήκτρου από τη συνάρτηση επιστροφής κλήσης η οποία εκτελείται προσφέροντας την εναλλαγή εμφάνισης/απόκρυψης των πληροφοριών.

- Για τη δημιουργία και εμφάνιση των μονοπατιών και των βασιλείων στα Google Maps (επιπλέον το αρχείο igeo_initialize_middleEarth.js δημιουργεί και τα κάστρα). Έχοντας από τη μία τους πίνακες JavaScript που περιέχουν τις γεωμετρίες (σε μορφή GeoJSON) των μονοπατιών και των βασιλείων και που δημιουργήθηκαν στην αντίστοιχη όψη και από την άλλη τις κατάλληλες επιλογές μορφοποίησης τους δημιουργούνται αντικείμενα καλώντας κάθε φορά τον κατασκευαστή GeoJSON.

Το αρχείο geo_initialize_middleEarth.js περιλαμβάνει, επιπλέον, κώδικα JavaScript για:

- Την περίπτωση που επιλεχθεί κάποιο βασίλειο από το μενού επιλογών. Τότε το γεγονός το διαχειρίζεται η συνάρτηση επιστροφής κλήσης η οποία εκτελείται και στέλνει μία αίτηση AJAX με το βασίλειο (id) προς τον ελεγκτή processAjax (συγκεκριμένα στη μέθοδο kingdomProcess) μέσω της μεθόδου POST. Όταν εξυπηρετηθεί επιτυχώς από τον εξυπηρετητή και φτάσει πίσω η απάντηση σε μορφή JSON τότε φροντίζει η συνάρτηση επιστροφής κλήσης (.done(function(data)) η οποία εκτελείται και έχει ως αποτέλεσμα την ενημέρωση του δεύτερου μενού επιλογών με τα κάστρα του βασιλείου, τη δημιουργία του βασιλείου ως αντικείμενο καλώντας τον κατασκευαστή GeoJSON, την επισήμανση του στο χάρτη και την εμφάνιση του κέντρου του ως σημείο (marker). Επίσης, κεντράρεται ο χάρτης στο κέντρο του βασιλείου.

- Την περίπτωση που επιλεχθεί κάποιο κάστρο από το μενού επιλογών. Τότε το γεγονός το διαχειρίζεται η συνάρτηση επιστροφής κλήσης η οποία εκτελείται και στέλνει μία αίτηση AJAX με το κάστρο (id) προς τον ελεγκτή processAjax (συγκεκριμένα στη μέθοδο castleProcess) μέσω της μεθόδου POST. Όταν εξυπηρετηθεί επιτυχώς από τον εξυπηρετητή και φτάσει πίσω η απάντηση σε μορφή JSON τότε φροντίζει η συνάρτηση επιστροφής κλήσης (.done(function(data)) η οποία εκτελείται και έχει ως αποτέλεσμα τη δημιουργία του κάστρου ως αντικείμενο καλώντας τον κατασκευαστή GeoJSON, την επισήμανση του στο χάρτη και την εμφάνιση πληροφοριακού παραθύρου κατόπιν επιλογής του κάστρου.

Το αρχείο igeo_initialize_middleEarth.js περιλαμβάνει, επιπλέον, κώδικα JavaScript για:

- Τη δημιουργία των πλήκτρων «Hide Castles» και «Show Castles» με τα οποία αποκρύβονται και εμφανίζονται τα κάστρα αντίστοιχα. Αυτό επιτυγχάνεται μέσω των κατασκευαστών
    - HideCastlesControl – Με τον οποίο δημιουργείται το πλήκτρο «Hide Castles» και του προστίθεται ακροατής (listener) ώστε σε περίπτωση εμφάνισης γεγονότος επιλογής του πλήκτρου να αποκρύπτονται τα κάστρα.

- ShowCastlesControl - Με τον οποίο δημιουργείται το πλήκτρο «Show Castles» και του προστίθεται ακροατής (listener) ώστε σε περίπτωση εμφάνισης γεγονότος επιλογής του πλήκτρου να εμφανίζονται τα κάστρα.

- Την περίπτωση που επιλεχθεί κάποιο σημείο επάνω στο Google Map. Έχουν προστεθεί ακροατές (listeners) σε όλη την έκταση της διεπαφής του Google Map, οπότε σε περίπτωση εμφάνισης γεγονότος επιλογής σημείου επάνω στο χάρτη εκτελείται η συνάρτηση pointRequest η οποία με τη σειρά της στέλνει μία αίτηση AJAX με τις συντεταγμένες (γεωγραφικό μήκος και πλάτος) προς τον ελεγκτή processAjax (συγκεκριμένα στη μέθοδο pointProcess) μέσω της μεθόδου POST. Όταν εξυπηρετηθεί επιτυχώς από τον εξυπηρετητή και φτάσει πίσω η απάντηση σε μορφή JSON τότε φροντίζει η συνάρτηση επιστροφής κλήσης (.done(function(data)) η οποία εκτελείται και έχει ως αποτέλεσμα, σε περίπτωση που το σημείο βρίσκεται εντός κάποιου βασιλείου, αφενός την εμφάνιση μηνύματος (επιτυχίας) με το όνομα του βασιλείου και τις συντεταγμένες του κέντρου του και αφετέρου τη δημιουργία του βασιλείου ως αντικείμενο καλώντας τον κατασκευαστή GeoJSON, την επισήμανση του στο χάρτη και την εμφάνιση του κέντρου του ως σημείο (marker). Επίσης, κεντράρεται ο χάρτης στο κέντρο του βασιλείου. Σε περίπτωση που το σημείο δεν ανήκει σε κάποιο βασίλειο, τότε το αποτέλεσμα είναι η εμφάνιση κατάλληλου μηνύματος (αποτυχίας).

Το αρχείο poi_initialize_middleEarth.js περιλαμβάνει, επιπλέον, κώδικα JavaScript για:

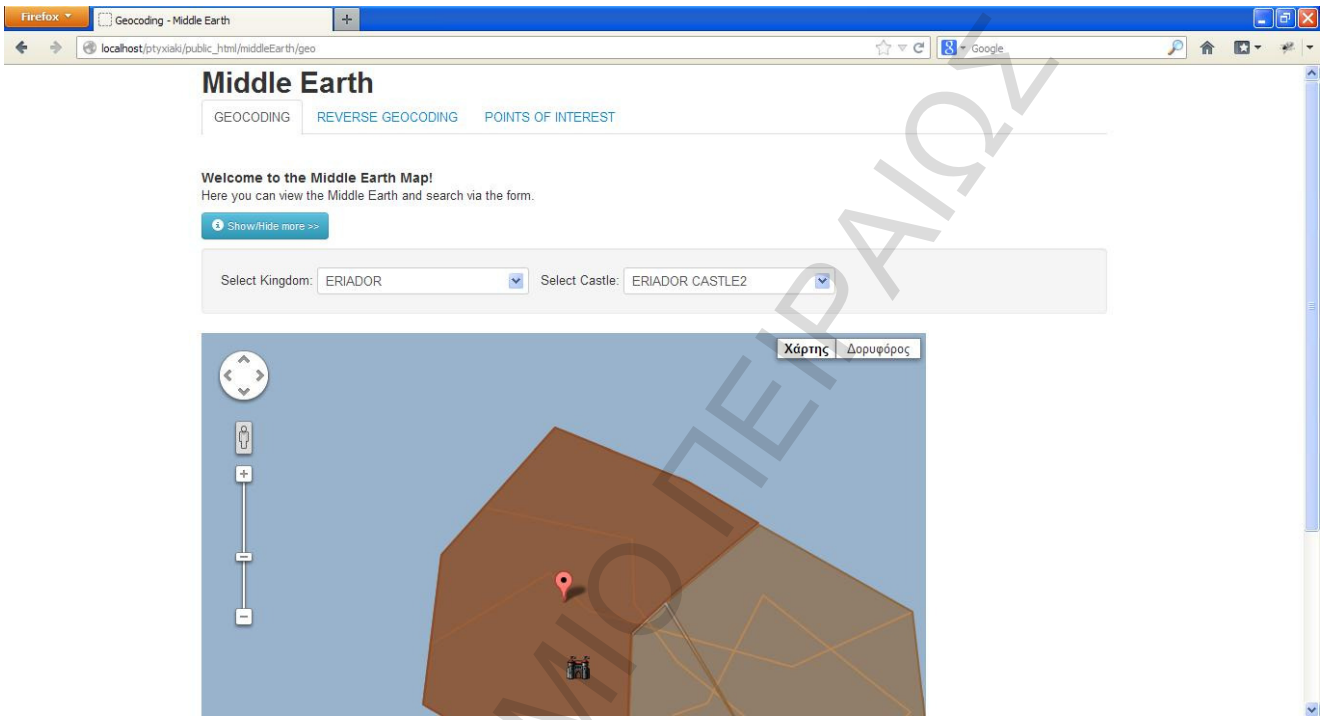
- Την περίπτωση που επιλεχθεί κάποιο σημείο επάνω στο Google Map. Έχουν προστεθεί ακροατές (listeners) σε όλη την έκταση της διεπαφής του Google Map, οπότε σε περίπτωση εμφάνισης γεγονότος επιλογής σημείου επάνω στο χάρτη εκτελείται η συνάρτηση poiRequest η οποία με τη σειρά της στέλνει μία αίτηση AJAX με τις συντεταγμένες (γεωγραφικό μήκος και πλάτος) και την επιλεγμένη ακτίνα προς τον ελεγκτή processAjax (συγκεκριμένα στη μέθοδο poiProcess) μέσω της μεθόδου POST. Όταν εξυπηρετηθεί επιτυχώς από τον εξυπηρετητή και φτάσει πίσω η απάντηση σε μορφή JSON τότε φροντίζει η συνάρτηση επιστροφής κλήσης (.done(function(data)) η οποία εκτελείται και έχει ως αποτέλεσμα, σε περίπτωση που βρεθεί κάποιο/α κάστρο/α εντός της περιοχής που σχηματίζεται από το σημείο και την ακτίνα, αφενός την εμφάνιση μηνύματος (επιτυχίας) με τις συντεταγμένες του επιλεγμένου σημείου και του μήκους της ακτίνας και αφετέρου τη δημιουργία του/των κάστρου/ων ως αντικείμενο/α καλώντας τον κατασκευαστή GeoJSON και την εμφάνιση του/ς επάνω στο χάρτη. Επίσης, από τη στιγμή που έχει προστεθεί ακροατής σε κάθε κάστρο δίνεται η δυνατότητα εμφάνισης πληροφοριακού παραθύρου με την απόσταση του κάθε κάστρου από το επιλεγμένο σημείο κατόπιν επιλογής του κάστρου. Σε περίπτωση που δεν υπάρχει κάποιο κάστρο εντός της περιοχής δράσης, τότε το αποτέλεσμα είναι η εμφάνιση κατάλληλου μηνύματος (αποτυχίας).

4.6.8 Υπηρεσίες της εφαρμογής

Στη συνέχεια γίνεται μία μικρή επίδειξη της εφαρμογής και των υπηρεσιών που προσφέρει προς τους χρήστες.

Εύρεση γεωγραφικής τοποθεσίας (Geocoding)

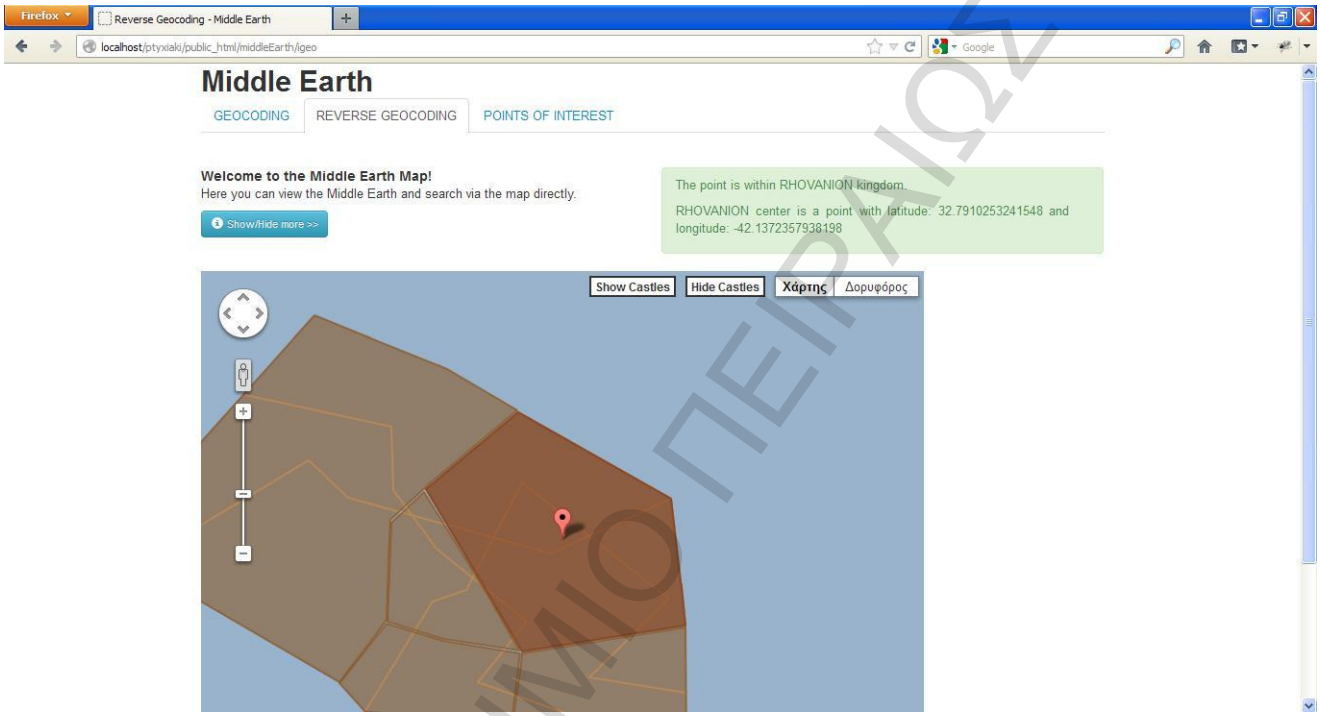
Στη σελίδα γεωκωδικοποίησης (geocoding), από το πρώτο μενού επιλογών έχει επιλεχθεί το ζητούμενο βασίλειο και ο χάρτης έχει κεντραριστεί στο κέντρο του βασιλείου (που εμφανίζεται ως σημείο) και το βασίλειο έχει επισημανθεί. Στη συνέχεια έχει επιλεχθεί από το δεύτερο μενού επιλογών το ζητούμενο κάστρο το οποίο εμφανίζεται στο χάρτη στη τοποθεσία που ανήκει.



Εικόνα 48 - Υπηρεσία γεωκωδικοποίησης

Εύρεση πληροφοριών τοποθεσίας (Reverse Geocoding)

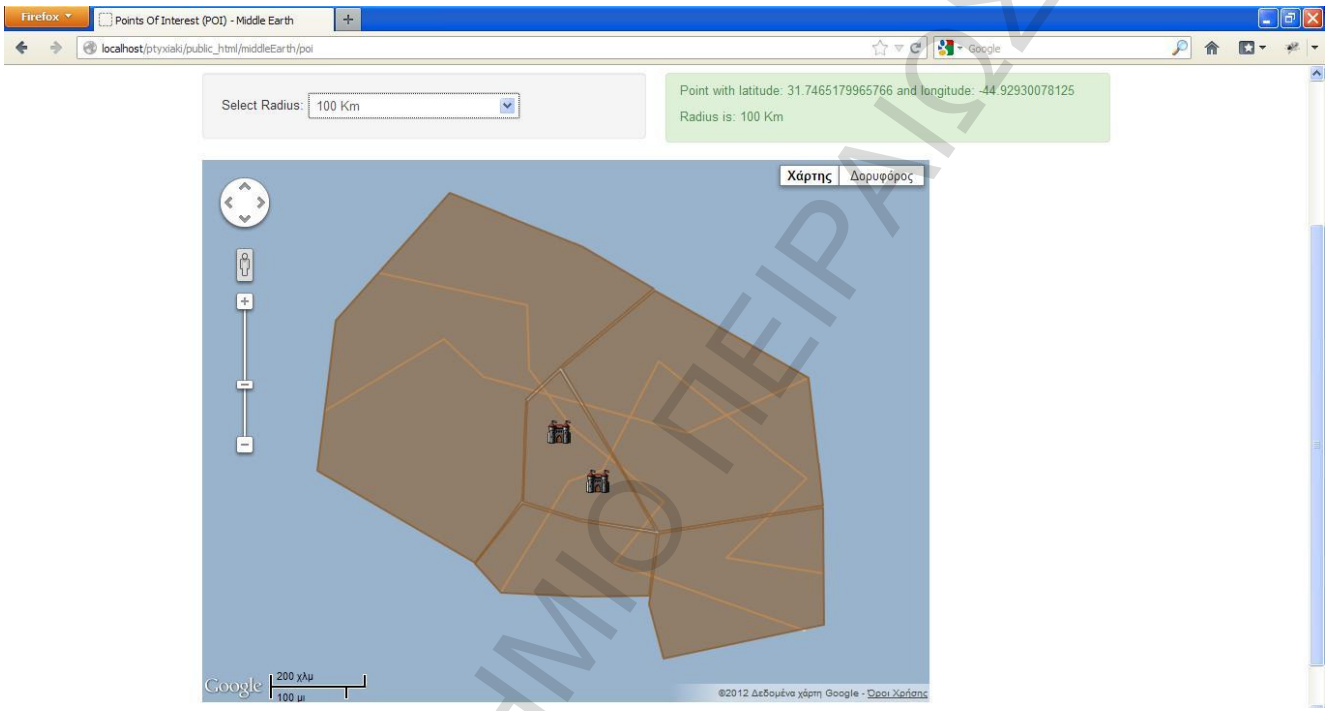
Στη σελίδα αντίστροφης γεωκωδικοίησης (reverse geocoding), έχει επιλεχθεί ένα σημείο επάνω στο χάρτη (εντός του βασιλείου RHOVANION) και ο χάρτης κεντράρεται στο κέντρο του βασιλείου (που εμφανίζεται ως σημείο) που περιλαμβάνει το σημείο που επιλέχθηκε. Το βασίλειο έχει επισημανθεί και έχει εμφανιστεί επιτυχές μήνυμα με το όνομα του βασιλείου και τις συντεταγμένες του κέντρου του.



Εικόνα 49 - Υπηρεσία αντίστροφης γεωκωδικοποίησης

Εύρεση σημείων ενδιαφέροντος σε συγκεκριμένη ακτίνα (Points of interest (POI) detection in a specific range)

Στη σελίδα σημείων ενδιαφέροντος (points of interest), από το μενού επιλογών έχει επιλεχθεί ακτίνα ίση με 100 Km και στη συνέχεια έχει επιλεχθεί ένα σημείο επάνω στο χάρτη και εμφανίζονται τα κάστρα που βρίσκονται εντός της περιοχής με βάση το επιλεχθέν σημείο και την ακτίνα. Επίσης, έχει εμφανιστεί επιτυχές μήνυμα με τις συντεταγμένες του σημείου και το μήκος της ακτίνας.



Εικόνα 50 - Υπηρεσία εύρεσης σημείων ενδιαφέροντος

Έχοντας παρουσιάσει με αναλυτικό τρόπο τον σχεδιασμό και την υλοποίηση της εφαρμογής μπορούμε να προχωρήσουμε στο επόμενο κεφάλαιο όπου γίνεται παρουσίαση των συμπερασμάτων αναφορικά στις τεχνολογίες και στη διαδικασία που ακολουθήθηκε για τη διεκπεραίωση της εργασίας (συγγραφή, σχεδιασμός της εφαρμογής, υλοποίηση, κτλ.), των περιορισμών που τέθηκαν και τέλος των μελλοντικών επεκτάσεων και προκλήσεων.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 Συμπεράσματα για την εφαρμογή

Η διαδικτυακή εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας βασίστηκε σε δύο κύριους άξονες, (α) στην αντικειμενοστρεφή αρχιτεκτονική και (β) σε ένα γεωγραφικό πληροφοριακό σύστημα (GIS). Οι γεωγραφικές/χωρικές απαιτήσεις της παρούσας εργασίας καλύφθηκαν με συνδυασμό της προγραμματιστικής διεπαφής (API) των Google Maps και της PostgreSQL/PostGIS με αποτέλεσμα την εξάλειψη της ανάγκης για χρήση κάποιου μεμονωμένου και εμπορικού λογισμικού για γεωγραφικά πληροφοριακά συστήματα. Εξάλλου, η PostgreSQL/PostGIS παρέχει όλες τις απαιτούμενες χωροαναλυτικές δυνατότητες ενός τυπικού γεωγραφικού πληροφοριακού συστήματος.

Η υλοποίηση της εφαρμογής πραγματοποιήθηκε στο πλαίσιο εργασίας CodeIgniter λόγω της εξαιρετικής του απόδοσης αλλά και της αναλυτικής και ολοκληρωμένης τεκμηρίωσης που παρέχει. Συνεπώς, χρησιμοποιήθηκε αντικειμενοστρεφής PHP με τη λογική του σχεδιαστικού προτύπου MVC. Από τη μεριά του πελάτη έγινε χρήση διάφορων διαδεδομένων Web τεχνολογιών όπως είναι η JavaScript, η βιβλιοθήκη jQuery, η τεχνική AJAX, η HTML και το Twitter Bootstrap. Ως πρότυπο μορφοποίησης των δεδομένων, χρησιμοποιήθηκε το JSON.

Όσο αφορά στη διαδικασία για τη διεκπεραίωση της παρούσας εργασίας, ακολουθήθηκε μία ορθολογική σειρά και για αυτό κρίθηκε αναγκαίο να σχεδιαστεί και να δημιουργηθεί αρχικά η βάση δεδομένων και στη συνέχεια να γίνει η ανάλυση, ο σχεδιασμός και η υλοποίηση της εφαρμογής. Στο τέλος, πραγματοποιήθηκε η συγγραφή της εργασίας.

5.2 Περιορισμοί που τέθηκαν

Στην παρούσα εργασία παραλείφθηκαν θέματα που σχετίζονται με την ασφάλεια της εφαρμογής λόγω του ότι η βαρύτητα δόθηκε στις λειτουργίες και υπηρεσίες που προσφέρει προς τους χρήστες. Για παράδειγμα, τα δεδομένα που υποβάλλονται στις φόρμες δεν ελέγχονται ούτε φιλτράρονται με αποτέλεσμα να ελλοχεύει ο κίνδυνος από επιθέσεις, τύπου SQL εγχύσεων (SQL injections), που μπορούν να προκαλέσουν απρόσμενες και ανυπολόγιστες συνέπειες στη βάση δεδομένων.

Επίσης, πρέπει να σημειωθεί το γεγονός ότι στη παρούσα εργασία παραλήφθηκε η ενσωμάτωση των δεδομένων στη βάση από αρχεία .shp (shape files) και αντ' αυτού τα δεδομένα εισήχθησαν με πολλαπλές εντολές SQL INSERT και με τη βοήθεια του Quantum GIS. Αυτό από τη μία, δημιούργησε σημαντική καθυστέρηση ως προς την φόρτωση όλων των δεδομένων αλλά από την άλλη αποφεύχθηκε η πιθανότητα παρουσίας διαφορετικών SRIDS (συστήματα συντεταγμένων) στις γεωμετρίες των δεδομένων, που θα δημιουργούσαν πρόβλημα ως προς την ομαλή απεικόνιση των δεδομένων μέσα από τα Google Maps.

5.3 Μελλοντικές επεκτάσεις

Μία από τις βασικές υπηρεσίες που προσφέρει η εφαρμογή είναι η εύρεση σημείων ενδιαφέροντος σε συγκεκριμένη ακτίνα. Ωστόσο, θα μπορούσε να βελτιωθεί δίνοντας τη δυνατότητα στο χρήστη να ορίζει τη περιοχή δράσης όπως ακριβώς επιθυμεί, σχεδιάζοντας απευθείας επάνω στο χάρτη το περικλείων πολύγωνο.

Μία άλλη πρόκληση αφορά την επέκταση του φανταστικού κόσμου της Μέσης Γής (Middle Earth) με την προσθήκη περισσότερων βασιλείων, κάστρων και μονοπατιών στα ήδη υπάρχοντα. Αυτό βέβαια θα δημιουργούσε μεγαλύτερο χρονικό κόστος σε ερωτήματα αναζήτησης με αποτέλεσμα την απαίτηση για ένα μηχανισμό που θα μπορούσε να εξισορροπήσει αυτή τη κατάσταση αυξάνοντας την απόδοση και ελαχιστοποιώντας το φόρτο εργασίας στη βάση δεδομένων. Ένας τέτοιος μηχανισμός είναι ο ElasticSearch

(http://www.elasticsearch.org) ο οποίος είναι ένας εξυπηρετητής αναζήτησης και βασίζεται στη βιβλιοθήκη ευρετηρίων και ανάκτησης του Apache Lucene (http://lucene.apache.org/core).

Εκτός από τις παραπάνω επεκτάσεις, μία επίσης πολύ σημαντική βελτίωση περιλαμβάνει την εισαγωγή πολιτικών ασφαλείας στο επίπεδο της εφαρμογής με τη κατάλληλη κατηγοριοποίηση των χρηστών σε εγγεγραμμένους και μη και την ενσωμάτωση SSL (Secure Socket Layer) συστήματος κρυπτογράφησης για την προστασία της επικοινωνίας μεταξύ του πελάτη και του εξυπηρετητή μέσω του Παγκόσμιου Ιστού.

Τέλος, θα πρέπει να υποστηριχθούν εξίσου οι χρήστες που αποκτούν πρόσβαση στην εφαρμογή μέσω κινητού τηλεφώνου. Για το λόγο αυτό η ενσωμάτωση τεχνικών δυναμικής διάταξης για την ορθή και συνεπή απεικόνιση της εφαρμογής, ανεξαρτήτου οθόνης και συσκευής, αποτελεί σημαντική πρόκληση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Boondao, R., Esichaikul, V., & Tripathi, N. K. (2003). A Model of Location Based Services for Crime Control. *Map Asia Conference.* Διαθέσιμο σε: http://www.gisdevelopment.net/technology/lbs/pdf/ma03217.pdf [Προσπελάστηκε 12 Οκτ 2012]

Piarsa, I. N., Wiranatha, A. C., & Putra, Y. P. (2012). Land Sale Geographic Information System based on Web and Web Mobile using Google Map. *International Journal of Modern Engineering Research (IJMER)*, 2(4), 2500-2503. Διαθέσιμο σε: http://www.ijmer.com/papers/Vol2_Issue4/CZ2425002503.pdf [Προσπελάστηκε 13 Οκτ 2012]

Oussalah, M., Bhat, F., Challis, K., & Schnier, T. (2012). A software architecture for Twitter collection, search and geolocation services. *In proceeding of the 10th IEEE International Conference on Cybernetics and Intelligent Systems (CIS)*, 139-144. Διαθέσιμο σε: http://s3.maghalam.com/As2012121319351.pdf [Προσπελάστηκε 13 Οκτ 2012]

Singh, P. S., Chutia, D., & Sudhakar, S. (2012). Development of a Web Based GIS Application for Spatial Natural Resources Information System Using Effective Open Source Software and Standards. *Journal of Geographic Information System*, 4, 261-266. Διαθέσιμο σε: http://www.scirp.org/journal/PaperInformation.aspx?paperID=19600 [Προσπελάστηκε 14 Οκτ 2012]

Yuan, Y., & Cheng, Q. (2007). Integrating Web-GIS and Hydrological Model: a Case Study with Google Maps and IHACRES in the Oak Ridges Moraine area, Southern Ontario, Canada. *In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 4574-4577. Διαθέσιμο σε: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4423875&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F4422707%2F4422708%2F04423875.pdf%3Farnumber%3D4423875 [Προσπελάστηκε 11 Οκτ 2012]

Wood, J., Dykes, J., Slingsby, A., & Clarke, K. (2007). Interactive Visual Exploration of a Large Spatio-Temporal Dataset: Reflections on a Geovisualization Mashup. *IEEE Transactions on Visualization and Computer Graphics*, 13(6), 1176-1183. Διαθέσιμο σε: http://openaccess.city.ac.uk/176/2/Interactive%20visual%20exploration%20of%20a%20large%20spatio-temporal%20dataset.pdf [Προσπελάστηκε 15 Οκτ 2012]

Rao, K. V., Govardhan, A., & Rao, C. (2012). Mining Topological Relationship Patterns from Spatiotemporal Databases. *International Journal of Data Mining & Knowledge Management Process (IJDKP), 2(2)*, 47-54. Διαθέσιμο σε: http://airccse.org/journal/ijdkp/papers/2212ijdkp05.pdf [Προσπελάστηκε 14 Οκτ 2012]

Serrano, S., Jiménez-Hornero, F. J., Gutiérrez de Ravé, E., & Jodral, M. L. (2008). GIS design application for "Sierra Morena Honey" designation of origin. *Journal of Computers and Electronics in Agriculture, 64(2)*, 307-317. Διαθέσιμο σε: http://www.sciencedirect.com/science/article/pii/S0168169908001701 [Προσπελάστηκε 16 Οκτ 2012]

Ligęza, A., Adrian, W. T., Ernst, S., Nalepa, G. J., Szpyrka, M., Czapko, M., Grzesiak, P., & Krzych, M. (2011). Prototypes of a Web System for Citizen Provided Information, Automatic Knowledge Extraction, Knowledge Management and GIS Integration. *In Proceedings of the 4th International Conference on Multimedia Communications, Services and Security (MCSS)*, 268-276. Διαθέσιμο σε: http://link.springer.com/chapter/10.1007%2F978-3-642-21512-4_32?LI=true#page-1 [Προσπελάστηκε 17 Οκτ 2012]

Anderson, G., & Moreno-Sanchez, R. (2003). Building Web-based Spatial Information Solutions around Open Specifications and Open Source Software. *Transactions in GIS, 7(4)*, 447-466. Διαθέσιμο σε: http://onlinelibrary.wiley.com/doi/10.1111/1467-9671.00158/pdf [Προσπελάστηκε 19 Οκτ 2012]

GIS Applications, [Online], Διαθέσιμο σε: http://www.supergeotek.com/Library_GISApplication.aspx [Προσπελάστηκε 3 Σεπ 2012]

Abukhater, A. (2011). *GIS for Planning and Community Development: Solving Global Challenges*, [Online], Διαθέσιμο σε: http://www.directionsmag.com/articles/gis-for-planning-and-community-development-solving-global-challenges/149245 [Προσπελάστηκε 10 Σεπ 2012]

Dangermond, J. (2009). *The Work of GIS Professionals and Evolving GIS Technology GIS - Geography in Action*, [Online], Διαθέσιμο σε: http://www.esri.com/news/arcnews/winter0809articles/gis-geography-in-action.html [Προσπελάστηκε 10 Σεπ 2012]

PostGIS, *Documentation,* [Online], Διαθέσιμο σε: http://postgis.refractions.net/documentation/ [Προσπελάστηκε 11 Μαρ 2012]

PostgreSQL, *PostgreSQL 8.3.23 Documentation*, [Online], Διαθέσιμο σε: http://www.postgresql.org/docs/8.3/static/index.html [Προσπελάστηκε 11 Μαρ 2012]

Quantum GIS, *QGIS Features*, [Online], Διαθέσιμο σε: http://qgis.org/en/about-qgis/features.html [Προσπελάστηκε 25 Μαρ 2012]

Obe, R., & Hsu, L. (2011). *PostGIS in Action*, Manning Publications Co.

Κολιός, Ν. (2009). *Χωρική Βάση Δεδομένων PostgreSQL/PostGIS και Σύστημα Γεωγραφικών Πληροφοριών QuantumGIS Οδηγός Χρήσης,* [pdf], Διαθέσιμο σε: http://old.ellak.gr/index.php?option=com_openwiki&Itemid=103&id=ellak:postgresql_postgis_quantumgis [Προσπελάστηκε 12 Μαρ 2012]

Θεωδορίδης, Ι., & Πελέκης, Ν. (2012). *Εισαγωγή στη Γεωπληροφορική*, [pdf], Διαθέσιμο σε: http://gunet2.cs.unipi.gr/eclass/modules/document/document.php?course=TME133&openDir=/20100928232959ibolfyds [Προσπελάστηκε 10 Απρ 2012]

Βόντας, Μ., & Ταμπάκης, Π. (2012). *Εργαστηριακή Διάλεξη στα Γεωγραφικά Πληροφοριακά Συστήματα – PostGIS*, [Online], Διαθέσιμο σε: http://gunet2.cs.unipi.gr/eclass/modules/document/file.php/TME133/Υλικό Εργαστηριων/PostGIS/GIS Lab.pdf [Προσπελάστηκε 5 Ιουν 2012]

Φρέτζος, Η. (2010). *Εισαγωγή στη PostgreSQL – PostGIS*, [Online], Διαθέσιμο σε: http://infolab.cs.unipi.gr/pre-eclass/courses/gis/lab/Lab-PostGIS-doc.pdf [Προσπελάστηκε 11 Απρ 2012]

The PHP Documentation Group, *PHP Manual*, [Online], Διαθέσιμο σε: http://php.net/manual/en/index.php [Προσπελάστηκε 5 Απρ 2012]

CodeIgniter, *User Guide,* [Online], Διαθέσιμο σε: http://codeigniter.com/user_guide/ [Προσπελάστηκε 25 Μαρ 2012]

W3Schools, *ASP.NET MVC Tutorial*, [Online], Διαθέσιμο σε: http://www.w3schools.com/aspnet/mvc_intro.asp [Προσπελάστηκε 27 Μαρ 2012]

MSDN, *Model-View-Controller*, [Online], Διαθέσιμο σε: http://msdn.microsoft.com/en-us/library/ff649643.aspx [Προσπελάστηκε 5 Σεπ 2012]

MSDN, *Front Controller*, [Online], Διαθέσιμο σε: http://msdn.microsoft.com/en-us/library/ff648617.aspx [Προσπελάστηκε 9 Σεπ 2012]

McCallum, E. (2004). *Building a PHP Front Controller*, [Online], Διαθέσιμο σε: http://onlamp.com/pub/a/php/2004/07/08/front_controller.html [Προσπελάστηκε 12 Σεπ 2012]

Δουληγέρης, Χ., Μαυροπόδη, Ρ., & Κοπανάκη, Ε. (2004). *Τεχνολογίες Διαδικτύου - Αρχές Λειτουργίας και Προγραμματισμός Εφαρμογών στο Διαδίκτυο*, Αθήνα: Νηρηίδες

Αγγελή, Χ. (2005). *Προγραμματισμός Web HTML4 & ASP*, Αθήνα: Σύγχρονη Εκδοτική

Le Hégaret, P., Wood, L., & Robie, J. (2000). *What is the Document Object Model,* [Online], Διαθέσιμο σε: http://www.w3.org/TR/DOM-Level-2-Core/introduction.html [Προσπελάστηκε 18 Σεπ 2012]

W3Schools, *HTML DOM Tutorial*, [Online], Διαθέσιμο σε: http://www.w3schools.com/htmldom/default.asp [Προσπελάστηκε 15 Σεπ 2012]

CERN, *Introduction to JavaScript*, [Online], Διαθέσιμο σε: http://enacit1.epfl.ch/tutorials/JavaScript/hierarchy.html [Προσπελάστηκε 21 Σεπ 2012]

Suehring, S. (2010). *JavaScript Step by Step*, 2nd Edition, Microsoft Press

Ballard, P., & Moncur, M. (2008). *Teach Yourself Ajax, JavaScript, and PHP All in One*, SAMS

Nixon, R. (2009). *Learning PHP, MySQL, and JavaScript*, O'Reilly Media

jQuery, *Documentation*, [Online], Διαθέσιμο σε: http://docs.jquery.com/ [Προσπελάστηκε 5 Οκτ 2012]

W3Schools, *jQuery Tutorial*, [Online], Διαθέσιμο σε: http://www.w3schools.com/jquery/default.asp [Προσπελάστηκε 5 Οκτ 2012]

Google Developers, *Google Maps JavaScript API v3*, [Online], Διαθέσιμο σε: https://developers.google.com/maps/documentation/javascript/tutorial [Προσπελάστηκε 15 Μαΐου 2012]

Squiz Matrix, *What is Google Maps*, [Online], Διαθέσιμο σε: http://manuals.matrix.squizsuite.net/google-maps/chapters/what-is-google-maps [Προσπελάστηκε 15 Μαΐου 2012]

JSON, *Introducing JSON*, [Online], Διαθέσιμο σε: http://www.json.org/ [Προσπελάστηκε 10 Ιουν 2012]

W3Schools, *AJAX Tutorial*, [Online], Διαθέσιμο σε: http://www.w3schools.com/ajax/default.asp [Προσπελάστηκε 10 Απρ 2012]

jQuery, *AJAX Documentation*, [Online], Διαθέσιμο σε: http://api.jquery.com/jQuery.ajax/ [Προσπελάστηκε 14 Απρ 2012]

Bootstrap, *Getting started*, [Online], Διαθέσιμο σε: http://twitter.github.com/bootstrap/getting-started.html [Προσπελάστηκε 2 Ιουν 2012]

Otto, M. (2011). *Bootstrap from Twitter*, [Online], Διαθέσιμο σε: https://dev.twitter.com/blog/bootstrap-twitter [Προσπελάστηκε 2 Ιουν 2012]

W3Resource, *Twitter Bootstrap tutorial*, [Online], Διαθέσιμο σε: http://www.w3resource.com/twitter-bootstrap/tutorial.php [Προσπελάστηκε 2 Ιουν 2012]

Λιακέας, Γ. (2003). *Εισαγωγή στην Java 2*, Αθήνα: Κλειδάριθμος

Prata, S. (2004). *C++ Primer Plus*, 5th Edition, SAMS

O'Docherty, M. (2005). *Object-Oriented Analysis and Design Understanding System Development with UML 2.0*, West Sussex: John Wiley & Sons, Ltd

ApacheFriends, *XAMPP*, [Online], Διαθέσιμο σε: http://www.apachefriends.org/en/xampp.html [Προσπελάστηκε 20 Μαρ 2012]

ΠΑΡΑΡΤΗΜΑ

Κώδικας εφαρμογής

middleEarth_Model.php

```
<?php

class MiddleEarth_Model extends CI_Model {

	public function allCastles()
	{
		$queryCastles = $this->db->query("SELECT id, name,
ST_AsGeoJSON(the_geom) FROM castles.castle;");
		return $queryCastles->result_array();
	}

	public function allPaths()
	{
		$queryPaths = $this->db->query("SELECT id, name,
ST_AsGeoJSON(the_geom) FROM castles.path;");
		return $queryPaths->result_array();
	}

	public function allKingdoms()
	{
		$queryKingdoms = $this->db->query("SELECT id, name,
ST_AsGeoJSON(the_geom) FROM castles.kingdom;");
		return $queryKingdoms->result_array();
	}

	public function getKingdomAndCastles($kingdom_id)
	{
	    $queryKingdom = $this->db->query("SELECT c.id as c_id,
c.name as c_name, k.name as k_name,
		ST_X(ST_Centroid(k.the_geom)) as k_centerx,
ST_Y(ST_Centroid(k.the_geom)) as k_centery,
		ST_AsGeoJSON(k.the_geom) FROM castles.castle c,
castles.kingdom k
		WHERE WITHIN(c.the_geom, k.the_geom) AND
k.id=$kingdom_id;");
	    return $queryKingdom->result_array();
	}

	public function getCastleGeom($castle_id)
	{
		$queryCastleGeom = $this->db->query("SELECT name,
ST_AsGeoJSON(the_geom) FROM castles.castle WHERE id=$castle_id;");
```

```
            return $queryCastleGeom->row_array();
      }

      public function getKingdomInfo($lon, $lat)
      {
            $queryKingdomInfo = $this->db->query("SELECT name,
ST_X(ST_Centroid(the_geom)), ST_Y(ST_Centroid(the_geom)),
            ST_AsGeoJSON(the_geom) FROM castles.kingdom WHERE
ST_Within(GeomFromText('POINT($lon $lat)', 4326),
kingdom.the_geom);");
            return $queryKingdomInfo->row_array();
      }

      public function getPoi($lon, $lat, $radius)
      {
            $queryPoi = $this->db->query("SELECT name,
ST_AsGeoJSON(the_geom),
            ST_Distance(the_geom, GeomFromText('POINT($lon
$lat)',4326)) AS distance,
            ST_X(GeomFromText('POINT($lon $lat)',4326)),
            ST_Y(GeomFromText('POINT($lon $lat)',4326))
            FROM castles.castle
            WHERE ST_Within(the_geom,
ST_Buffer(GeomFromText('POINT($lon $lat)',4326), $radius));");
            return $queryPoi->result_array();
      }
}
```

header.php

```
<html>
<head>
<title><?php echo $title ?> - Middle Earth</title>
<link href="<?php echo $baseUrl; ?>css/styles.css" rel="stylesheet"
type="text/css" />
<link href="<?php echo $baseUrl; ?>css/bootstrap/css/bootstrap.css"
rel="stylesheet" type="text/css" />
</head>
<body>
<div class="container">
<h1>Middle Earth</h1>
<ul class="nav nav-tabs">
<li class="<?php echo $layout_normalActive; ?>"><a href="<?php echo
$baseUrl; ?>middleEarth/geo">GEOCODING</a></li>
<li class="<?php echo $layout_inverseActive; ?>"><a href="<?php echo
$baseUrl; ?>middleEarth/igeo">REVERSE GEOCODING</a></li>
<li class="<?php echo $layout_poiActive; ?>"><a href="<?php echo
$baseUrl; ?>middleEarth/poi">POINTS OF INTEREST</a></li>
```

```
</ul>
```

footer.php

```
<br/><br/>
<i>Copyright &copy; 2012. All Rights Reserved.<br/>
Developed by Apostolos Christodoulou.</i>
</div><!-- container div -->
</body>
</html>
```

home.php

```
<br/>
<div class="row-fluid">
<div class="span6">
<h4>Welcome to the Middle Earth Application!</h4>
<br/>
<p>Here you can have access to:
<ul>
<li>Geocoding service</li>
<li>Reverse Geocoding service</li>
<li>Points of Interest (POI) service</li>
</ul>
</p>
</div>
</div>
```

geo_Gmap.php

```
<br/>
<div class="row-fluid">
<div class="span6">
<h4>Welcome to the Middle Earth Map!</h4>
<p>Here you can view the Middle Earth and search via the form.<br/>
<span class="hide">Particularly, you can view the kingdoms and the
paths crossing the kingdoms.
Select a kingdom via the first select box in order to select any of
the castles that the selected kingdom
contains via the second select box. In addition, the kingdom is marked
upon the map showing also the center of it.
Afterwards, since the castle selection is done you see it on the map
in the right location.<br/><br/>
</span>
</p>
<p>
<a id="slider" class="btn btn-info btn-small" href="#"><i class="icon-
info-sign icon-white"></i> Show/Hide more >></a>
</p>
</div>
```

```
</div>
<form id="middleEarthForm" class="well form-inline">
<label class="control-label" for="select01">Select Kingdom: </label>
<select id="select01">
<option id="intro" value="kingdom">Kingdom</option>
<?php foreach ($myKingdoms as $kingdom): ?>
<option value="<?php echo $kingdom['id']; ?>">
<?php echo $kingdom['name']; ?>
</option>
<?php endforeach ?>
</select>
 
<label class="control-label" for="select02">Select Castle: </label>
<select id="select02">
<option id="intro2" value="castle">Castle</option>
</select>
</form>
<div id="map_canvas"></div>
<script type="text/javascript" src="<?php echo $baseUrl; ?>js/jquery-
1.7.2.min.js">
</script>
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCxknxFfQBJpjt4Nk
eMv2O0q-Sld1KC5vs&sensor=true">
</script>
<script type="text/javascript">
var i=0;
var j=0;
var paths = new Array();
var kingdoms = new Array();

<?php foreach ($myPaths as $path): ?>
paths[i] = <?php echo $path['st_asgeojson']; ?>;
i++;
<?php endforeach ?>

<?php foreach ($myKingdoms as $kingdom): ?>
kingdoms[j] = <?php echo $kingdom['st_asgeojson']; ?>;
j++;
<?php endforeach ?>
</script>
<script type="text/javascript" src="<?php echo $baseUrl;
?>js/geo_initialize_middleEarth.js">
</script>
```

igeo_Gmap.php

```
<br/>
<div class="row-fluid">
<div class="span6">
<h4>Welcome to the Middle Earth Map!</h4>
<p>Here you can view the Middle Earth and search via the map
directly.<br/>
<span class="hide">Particularly, you can view the kingdoms, the
castles of every kingdom
and the paths crossing the kingdoms. You can click anywhere on the map
and you directly get
information related to the kingdom that the point belongs. In
addition, the kingdom
is marked upon the map showing also the center of it.<br/><br/>
<u>Note</u>: You can show/hide the castles by pressing the "Show
Castles" and the "Hide Castles"
respectively.
</span>
</p>
<p>
<a id="slider" class="btn btn-info btn-small" href="#"><i class="icon-
info-sign icon-white"></i> Show/Hide more >></a>
</p>
</div>
<div id="info" class="alert span6"></div>
</div>
<div id="map_canvas"></div>
<script type="text/javascript" src="<?php echo $baseUrl; ?>js/jquery-
1.7.2.min.js">
</script>
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCxknxFfQBJpjt4Nk
eMv2O0q-Sld1KC5vs&sensor=true">
</script>
<script type="text/javascript">
var i=0;
var j=0;
var k=0;
var castles = new Array();
var paths = new Array();
var kingdoms = new Array();

<?php foreach ($myCastles as $castle): ?>
castles[i] = <?php echo $castle['st_asgeojson']; ?>;
i++;
<?php endforeach ?>

<?php foreach ($myPaths as $path): ?>
paths[j] = <?php echo $path['st_asgeojson']; ?>;
j++;
<?php endforeach ?>

<?php foreach ($myKingdoms as $kingdom): ?>
kingdoms[k] = <?php echo $kingdom['st_asgeojson']; ?>;
k++;
<?php endforeach ?>
</script>
```

```
<script type="text/javascript" src="<?php echo $baseUrl;
?>js/igeo_initialize_middleEarth.js">
</script>
```

poi_Gmap.php

```
<br/>
<div class="row-fluid">
<div class="span6">
<h4>Welcome to the Middle Earth Map!</h4>
<p>Here you can view the Middle Earth and the castles around a point
that you clicked on the map.<br/>
<span class="hide">Particularly, you can view the kingdoms and the
paths crossing the kingdoms.
Select a radius via the select box, then click anywhere on the map and
you directly
see the castles within the circle area (formed by the point and the
radius) and get
information related to the point latitude, longitude.
In addition, you get information related to the castle distance from
the point
that you clicked via an info window above every castle.<br/><br/>
</span>
</p>
<p>
<a id="slider" class="btn btn-info btn-small" href="#"><i class="icon-
info-sign icon-white"></i> Show/Hide more >></a>
</p>
</div>
</div>
<div class="row-fluid">
<div class="span6">
<form id="middleEarthForm" class="well form-inline">
<label class="control-label" for="select03">Select Radius: </label>
<select id="select03">
<option value="0.9">100 Km</option>
<option selected="selected" value="1.8">200 Km</option>
<option value="2.7">300 Km</option>
</select>
</form>
</div>
<div id="info" class="alert span6"></div>
</div>
<div id="map_canvas"></div>
<script type="text/javascript" src="<?php echo $baseUrl; ?>js/jquery-
1.7.2.min.js">
</script>
```

```
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCxknxFfQBJpjt4Nk
eMv2O0q-S1d1KC5vs&sensor=true">
</script>
<script type="text/javascript">
var i=0;
var j=0;
var paths = new Array();
var kingdoms = new Array();

<?php foreach ($myPaths as $path): ?>
paths[i] = <?php echo $path['st_asgeojson']; ?>;
i++;
<?php endforeach ?>

<?php foreach ($myKingdoms as $kingdom): ?>
kingdoms[j] = <?php echo $kingdom['st_asgeojson']; ?>;
j++;
<?php endforeach ?>
</script>
<script type="text/javascript" src="<?php echo $baseUrl;
?>js/poi_initialize_middleEarth.js">
</script>
```

MY_Controller.php

```
<?php

class MY_Controller extends CI_Controller {

      protected $data = array();

      public function __construct()
      {
            parent::__construct();

            $this->load->database();

            $this->load->model('middleEarth_Model');

            $this->load->helper('url');

            $this->_initGlobals();
      }

      protected function getHeader()
      {
```

```
            $this->load->view('templates/header', $this->data);
      }

    protected function getFooter()
      {
            $this->load->view('templates/footer', $this->data);
      }

      protected function _initGlobals()
      {
            $this->data['baseUrl'] = base_url();
      }
}
```

middleEarth.php

```
<?php

class MiddleEarth extends MY_Controller {

      public function __construct()
      {
            parent::__construct();

            $this->data['layout_normalActive'] = '';
            $this->data['layout_inverseActive'] = '';
            $this->data['layout_poiActive'] = '';
      }

      public function index()
      {
          $this->data['title'] = 'Home';
            $this->getHeader();
            $this->load->view('middle_earth/home', $this->data);
            $this->getFooter();
      }

      public function geo()
      {
            $this->data['layout_normalActive'] = 'active';

            $this->data['myPaths'] = $this->middleEarth_Model-
>allPaths();
            $this->data['myKingdoms'] = $this->middleEarth_Model-
>allKingdoms();
```

```
            $this->data['title'] = 'Geocoding';
              $this->getHeader();
              $this->load->view('middle_earth/geo_Gmap', $this->data);
              $this->getFooter();
      }

      public function igeo()
      {
            $this->data['layout_inverseActive'] = 'active';

            $this->data['myCastles'] = $this->middleEarth_Model-
>allCastles();
            $this->data['myPaths'] = $this->middleEarth_Model-
>allPaths();
            $this->data['myKingdoms'] = $this->middleEarth_Model-
>allKingdoms();

            $this->data['title'] = 'Reverse Geocoding';
              $this->getHeader();
              $this->load->view('middle_earth/igeo_Gmap', $this->data);
              $this->getFooter();
      }

      public function poi()
      {
            $this->data['layout_poiActive'] = 'active';

              $this->data['myPaths'] = $this->middleEarth_Model-
>allPaths();
              $this->data['myKingdoms'] = $this->middleEarth_Model-
>allKingdoms();

            $this->data['title'] = 'Points Of Interest (POI)';
              $this->getHeader();
              $this->load->view('middle_earth/poi_Gmap', $this->data);
              $this->getFooter();
      }
}
```

processAjax.php

```
<?php

class ProcessAjax extends MY_Controller {

      public function kingdomProcess()
      {
```

```
            $kingdom_id = $this->input->post('kingdom');
            $this->data['kingdomAndCastles'] = $this-
>middleEarth_Model->getKingdomAndCastles($kingdom_id);
            $kingdomAndCastles;
            $i=0;

            if ($this->data['kingdomAndCastles'])
            {
                  foreach ($this->data['kingdomAndCastles'] as $row)
                  {

                        $kingdomAndCastles[$i]['c_id'] = $row['c_id'];
                        $kingdomAndCastles[$i]['c_name'] =
$row['c_name'];
                        $kingdomAndCastles[$i]['k_name'] =
$row['k_name'];
                        $kingdomAndCastles[$i]['k_centerX'] =
$row['k_centerx'];
                        $kingdomAndCastles[$i]['k_centerY'] =
$row['k_centery'];
                        $kingdomAndCastles[$i]['st_asgeojson'] =
$row['st_asgeojson'];
                        $i++;
                  }
                  echo json_encode($kingdomAndCastles);
            }
      }

      public function castleProcess()
      {
            $castle_id = $this->input->post('castle');
            $this->data['castleGeom'] = $this->middleEarth_Model-
>getCastleGeom($castle_id);

            if ($this->data['castleGeom'])
            {
                  $castleGeom = $this-
>data['castleGeom']['st_asgeojson'];
                  $castleName = $this->data['castleGeom']['name'];
                  $theCastle_prefixed = array('name' => $castleName,
'geom' => $castleGeom);
                  echo json_encode($theCastle_prefixed);
            }
      }

      public function pointProcess()
      {
```

```
            $lon = $this->input->post('lon');
            $lat = $this->input->post('lat');
            $this->data['theKingdom'] = $this->middleEarth_Model-
>getKingdomInfo($lon, $lat);

            if ($this->data['theKingdom'])
            {
                  $kingdomName = $this->data['theKingdom']['name'];
                  $kingdomCenterX = $this->data['theKingdom']['st_x'];
                  $kingdomCenterY = $this->data['theKingdom']['st_y'];
                  $kingdomGeom = $this-
>data['theKingdom']['st_asgeojson'];
                  $theKingdom_prefixed = array('name' => $kingdomName,
'centerX' => $kingdomCenterX, 'centerY' => $kingdomCenterY, 'geom' =>
$kingdomGeom);
                  echo json_encode($theKingdom_prefixed);
            }

            else
            {
                  $errorMessage = array('wrong' => 'This point is not
within any of the kingdoms...');
                  echo json_encode($errorMessage);
            }
      }

      public function poiProcess()
      {
            $lon = $this->input->post('lon');
            $lat = $this->input->post('lat');
            $radius = $this->input->post('radius');
            $this->data['castles'] = $this->middleEarth_Model-
>getPoi($lon, $lat, $radius);
            $castles;
            $i=0;

            if ($this->data['castles'])
            {
                  foreach ($this->data['castles'] as $row)
                  {
                        $castles[$i]['name'] = $row['name'];
                        $castles[$i]['st_asgeojson'] =
$row['st_asgeojson'];
                        $castles[$i]['distance'] = $row['distance'];
                        $castles[$i]['st_x'] = $row['st_x'];
                        $castles[$i]['st_y'] = $row['st_y'];
                        $i++;
```

```
                  }
                  echo json_encode($castles);
            }

            else
            {
                  $errorMessage = array('wrong' => 'No castles within
the circle area...');
                  echo json_encode($errorMessage);
            }
      }
}
```

geo_initialize_middleEarth.js

```
$(document).ready(function()
{
      $("a#slider").click(function(event)
      {
            $("p span").slideToggle();
            event.preventDefault();
      });

      var myPoint = new google.maps.LatLng(32.565, -45.171);
      var container = $("#map_canvas")[0];
      var map;

      var myOptions = {
    center: myPoint,
    zoom: 6,
      scaleControl: true,
    mapTypeId: google.maps.MapTypeId.TERRAIN
    };

      map = new google.maps.Map(container, myOptions);

      var pathsOptions = {
            strokeColor: "#FFCC99",
            strokeWeight: 2,
            strokeOpacity: 1.0
      };

      var kingdomsOptions = {
            strokeColor: "#8B5A2B",
            strokeWeight: 2,
            strokeOpacity: 0.8,
```

```
            fillColor: "#8B5A2B",
            fillOpacity: 0.6
      };

      var pathsVector = new Array();
      var kingdomsVector = new Array();

      for (var i=0; i<paths.length; i++)
      {
            pathsVector[i] = new GeoJSON(paths[i], pathsOptions);
            pathsVector[i].setMap(map);
      }

      for (var i=0; i<kingdoms.length; i++)
      {
            kingdomsVector[i] = new GeoJSON(kingdoms[i],
kingdomsOptions);
            kingdomsVector[i].setMap(map);
      }

      function GeoJSON(geojson, options)
      {
            var googleObj;

            if (geojson.type == "Point")
            {
                  options.position = new
google.maps.LatLng(geojson.coordinates[1], geojson.coordinates[0]);
                  googleObj = new google.maps.Marker(options);
            }

            else if (geojson.type == "LineString")
            {
                  var path = new Array();
                  for (var i=0; i<geojson.coordinates.length; i++)
                  {
                        var coord = geojson.coordinates[i];
                        var lpoint = new google.maps.LatLng(coord[1],
coord[0]);
                        path.push(lpoint);
                  }
                  options.path = path;
                  googleObj = new google.maps.Polyline(options);
            }

            else // Polygon
```

```
            {
                  var paths = new Array();
                  for (var i=0; i<geojson.coordinates.length; i++)
                  {
                        for (var j=0; j<geojson.coordinates[i].length;
j++)
                        {
                              var lpoint = new
google.maps.LatLng(geojson.coordinates[i][j][1],
geojson.coordinates[i][j][0]);
                              paths.push(lpoint);
                        }
                  }
                  options.paths = paths;
                  googleObj = new google.maps.Polygon(options);
            }

            return googleObj;
      }

      var centerMarker;
      var selectedKingdom;
      var opts = {
            strokeColor: "#8B4726",
            strokeWeight: 2,
            strokeOpacity: 0.8,
            fillColor: "#8B4726",
            fillOpacity: 0.6
      };

      $("#select01").change(function()
      {
            if ($("#select01 option#intro").length > 0)
            {
                  $("#select01 option#intro").remove();
            }

            if (castleMarker)
            {
                  castleMarker.setMap(null);
            }

            var str = $("#select01 option:selected").val();

            $.ajax({
            data: {kingdom: str},
```

```
            dataType: 'json',
            url:
'http://localhost/ptyxiaki/public_html/processAjax/kingdomProcess',
            type: 'POST'
            }).done(function(data)
            {
                  $("#select02").children().remove();
                  $("#select02").append("<option id='intro2'
value='castle'>Castle</option>");

                  for (var i=0; i<data.length; i++)
                  {
                        $("#select02").append("<option value='" +
data[i].c_id + "'>" + data[i].c_name + "</option>");
                  }

            if (centerMarker)
                {
                        centerMarker.setMap(null);
                  }

                  if (selectedKingdom)
                  {
                        selectedKingdom.setMap(null);
                  }

                  parsedKingdom = JSON.parse(data[0].st_asgeojson);

                  selectedKingdom = new GeoJSON(parsedKingdom, opts);
                  selectedKingdom.setMap(map);

                  var center = new
google.maps.LatLng(data[0].k_centerY, data[0].k_centerX);
                  centerMarker = new google.maps.Marker({
                        position: center,
                        map: map,
                        title: "Center of " + data[0].k_name
                  });
                  map.setCenter(center);
            });
      });

      var castleMarker;
      var infowindow = new google.maps.InfoWindow();
      var castleOptions = {
```

```
            icon:
"http://localhost/ptyxiaki/public_html/images/castle.png",
            animation: google.maps.Animation.DROP
      };

      $("#select02").change(function()
      {
            var str = $("#select02 option:selected").val();

            if (str != $("#select02 option#intro2").val())
            {
                  $.ajax({
                  data: {castle: str},
                  dataType: 'json',
                  url:
'http://localhost/ptyxiaki/public_html/processAjax/castleProcess',
                  type: 'POST'
                  }).done(function(data)
                  {
                        if (castleMarker)
                        {
                              castleMarker.setMap(null);
                        }

                        parsedCastle = JSON.parse(data.geom);

                        castleMarker = new GeoJSON(parsedCastle,
castleOptions);
                        castleMarker.setMap(map);

                        google.maps.event.addListener(castleMarker,
'click', function() {
                              infowindow.setContent("This is the
<b>"+data.name+"</b>");
                              infowindow.open(map, castleMarker);
                        });
                  });
            }
      });
});
```

igeo_initialize_middleEarth.js

```
$(document).ready(function()
{
      $("a#slider").click(function(event)
      {
            $("p span").slideToggle();
```

```
            event.preventDefault();
      });

      var myPoint = new google.maps.LatLng(32.565, -45.171);
      var container = $("#map_canvas")[0];
      var map;

      var myOptions = {
    center: myPoint,
    zoom: 6,
      scaleControl: true,
    mapTypeId: google.maps.MapTypeId.TERRAIN
    };

      map = new google.maps.Map(container, myOptions);

      var hideCastlesControlDiv = document.createElement('div');
      var hideCastlesControl = new
HideCastlesControl(hideCastlesControlDiv, map);
      hideCastlesControlDiv.index = 1;
      map.controls[google.maps.ControlPosition.TOP_RIGHT].push(hideCas
tlesControlDiv);
      var showCastlesControlDiv = document.createElement('div');
      var showCastlesControl = new
ShowCastlesControl(showCastlesControlDiv, map);
      showCastlesControlDiv.index = 1;
      map.controls[google.maps.ControlPosition.TOP_RIGHT].push(showCas
tlesControlDiv);

      var castlesOptions = {
            icon:
"http://localhost/ptyxiaki/public_html/images/castle.png",
            animation: google.maps.Animation.DROP
      };

      var pathsOptions = {
            strokeColor: "#FFCC99",
            strokeWeight: 2,
            strokeOpacity: 1.0
      };

      var kingdomsOptions = {
            strokeColor: "#8B5A2B",
            strokeWeight: 2,
            strokeOpacity: 0.8,
            fillColor: "#8B5A2B",
            fillOpacity: 0.6
      };

      var castlesVector = new Array();
      var pathsVector = new Array();
      var kingdomsVector = new Array();

      for (var i=0; i<castles.length; i++)
      {
            castlesVector[i] = new GeoJSON(castles[i],
castlesOptions);
```

```
	}

	for (var i=0; i<paths.length; i++)
	{
		pathsVector[i] = new GeoJSON(paths[i], pathsOptions);
		pathsVector[i].setMap(map);
	}

	for (var i=0; i<kingdoms.length; i++)
	{
		kingdomsVector[i] = new GeoJSON(kingdoms[i],
kingdomsOptions);
		kingdomsVector[i].setMap(map);
	}

	function GeoJSON(geojson, options)
	{
		var googleObj;

		if (geojson.type == "Point")
		{
			options.position = new
google.maps.LatLng(geojson.coordinates[1], geojson.coordinates[0]);
			googleObj = new google.maps.Marker(options);
		}

		else if (geojson.type == "LineString")
		{
			var path = new Array();
			for (var i=0; i<geojson.coordinates.length; i++)
			{
				var coord = geojson.coordinates[i];
				var lpoint = new google.maps.LatLng(coord[1],
coord[0]);
				path.push(lpoint);
			}
			options.path = path;
			googleObj = new google.maps.Polyline(options);
		}

		else // Polygon
		{
			var paths = new Array();
			for (var i=0; i<geojson.coordinates.length; i++)
			{
				for (var j=0; j<geojson.coordinates[i].length;
j++)
				{
					var lpoint = new
google.maps.LatLng(geojson.coordinates[i][j][1],
geojson.coordinates[i][j][0]);
					paths.push(lpoint);
				}
			}
			options.paths = paths;
			googleObj = new google.maps.Polygon(options);
		}
```

```
            return googleObj;
      }

      var centerMarker;
      var selectedKingdom;
      var opts = {
            strokeColor: "#8B4726",
            strokeWeight: 2,
            strokeOpacity: 0.8,
            fillColor: "#8B4726",
            fillOpacity: 0.6
      };

      function pointRequest(longitude, latitude)
      {
            $.ajax({
            data: {lon: longitude, lat: latitude},
            dataType: 'json',
            url:
'http://localhost/ptyxiaki/public_html/processAjax/pointProcess',
            type: 'POST'
            }).done(function(data)
            {
                  if (data.name)
                  {
                        $("div#info").addClass("alert-success");
                    $("div#info").removeClass("alert-error");
                        $("div#info").html("<p>The point is within " +
data.name + " kingdom.</p><p>" + data.name +
                        " center is a point with latitude: " +
data.centerY + " and longitude: " + data.centerX + "</p>");
                        $("div#info").show();

                        if (centerMarker)
                        {
                              centerMarker.setMap(null);
                        }

                        if (selectedKingdom)
                        {
                              selectedKingdom.setMap(null);
                        }

                        parsedKingdom = JSON.parse(data.geom);

                        selectedKingdom = new GeoJSON(parsedKingdom,
opts);
                        selectedKingdom.setMap(map);

                        var center = new
google.maps.LatLng(data.centerY, data.centerX);
                        centerMarker = new google.maps.Marker({
                        position: center,
                        map: map,
                        title: "Center of " + data.name
                        });
```

```
                        map.setCenter(center);
                  }

                  else
                  {
                        if (centerMarker)
                        {
                              centerMarker.setMap(null);
                        }

                        if (selectedKingdom)
                        {
                              selectedKingdom.setMap(null);
                        }

                        $("div#info").addClass("alert-error");
                    $("div#info").removeClass("alert-success");
                        $("div#info").text(data.wrong);
                        $("div#info").show();
                  }
            });
      }

      google.maps.event.addListener(map, 'click', function(event)
      {
            pointRequest(event.latLng.lng(), event.latLng.lat());
      });

      for (var i=0; i<castlesVector.length; i++)
      {
            google.maps.event.addListener(castlesVector[i], 'click',
function(event)
            {
                  pointRequest(event.latLng.lng(),
event.latLng.lat());
            });
      }

      for (var i=0; i<pathsVector.length; i++)
      {
            google.maps.event.addListener(pathsVector[i], 'click',
function(event)
            {
                  pointRequest(event.latLng.lng(),
event.latLng.lat());
            });
      }

      for (var i=0; i<kingdomsVector.length; i++)
      {
            google.maps.event.addListener(kingdomsVector[i], 'click',
function(event)
            {
                  pointRequest(event.latLng.lng(),
event.latLng.lat());
            });
      }
```

```
	function ShowCastlesControl(controlDiv, map)
	{
		controlDiv.style.padding = '5px';

		var controlUI = document.createElement('div');
		controlUI.style.backgroundColor = '#ffffff';
		controlUI.style.borderStyle = 'solid';
		controlUI.style.borderWidth = '2px';
		controlUI.style.cursor = 'pointer';
		controlUI.style.textAlign = 'center';
		controlUI.title = 'Click to show all castles';
		controlDiv.appendChild(controlUI);

		var controlText = document.createElement('div');
		controlText.style.fontFamily = 'Arial, sans-serif';
		controlText.style.fontSize = '12px';
		controlText.style.paddingLeft = '4px';
		controlText.style.paddingRight = '4px';
		controlText.innerHTML = '<strong>Show Castles</strong>';
		controlUI.appendChild(controlText);

		google.maps.event.addDomListener(controlUI, 'click',
function()
		{
			if (castlesVector)
			{
				for (i in castlesVector)
				{
					castlesVector[i].setMap(map);
				}
			}
		});
	}

	function HideCastlesControl(controlDiv, map)
	{
		controlDiv.style.padding = '5px';

		var controlUI = document.createElement('div');
		controlUI.style.backgroundColor = '#ffffff';
		controlUI.style.borderStyle = 'solid';
		controlUI.style.borderWidth = '2px';
		controlUI.style.cursor = 'pointer';
		controlUI.style.textAlign = 'center';
		controlUI.title = 'Click to hide all castles';
		controlDiv.appendChild(controlUI);

		var controlText = document.createElement('div');
		controlText.style.fontFamily = 'Arial, sans-serif';
		controlText.style.fontSize = '12px';
		controlText.style.paddingLeft = '4px';
		controlText.style.paddingRight = '4px';
		controlText.innerHTML = '<strong>Hide Castles</strong>';
		controlUI.appendChild(controlText);

		google.maps.event.addDomListener(controlUI, 'click',
```

```
function()
            {
                  if (castlesVector)
                  {
                        for (i in castlesVector)
                        {
                              castlesVector[i].setMap(null);
                        }
                  }
            });
      }
});
```

poi_initialize_middleEarth.js

```
$(document).ready(function()
{
      $("a#slider").click(function(event)
      {
            $("p span").slideToggle();
            event.preventDefault();
      });

      var myPoint = new google.maps.LatLng(32.565, -45.171);
      var container = $("#map_canvas")[0];
      var map;

      var myOptions = {
    center: myPoint,
    zoom: 6,
      scaleControl: true,
    mapTypeId: google.maps.MapTypeId.TERRAIN
    };

      map = new google.maps.Map(container, myOptions);

      var pathsOptions = {
            strokeColor: "#FFCC99",
            strokeWeight: 2,
            strokeOpacity: 1.0
      };

      var kingdomsOptions = {
            strokeColor: "#8B5A2B",
            strokeWeight: 2,
            strokeOpacity: 0.8,
            fillColor: "#8B5A2B",
            fillOpacity: 0.6
```

```
	};

	var pathsVector = new Array();
	var kingdomsVector = new Array();

	for (var i=0; i<paths.length; i++)
	{
		pathsVector[i] = new GeoJSON(paths[i], pathsOptions);
		pathsVector[i].setMap(map);
	}

	for (var i=0; i<kingdoms.length; i++)
	{
		kingdomsVector[i] = new GeoJSON(kingdoms[i],
kingdomsOptions);
		kingdomsVector[i].setMap(map);
	}

	function GeoJSON(geojson, options)
	{
		var googleObj;

		if (geojson.type == "Point")
		{
			options.position = new
google.maps.LatLng(geojson.coordinates[1], geojson.coordinates[0]);
			googleObj = new google.maps.Marker(options);
		}

		else if (geojson.type == "LineString")
		{
			var path = new Array();
			for (var i=0; i<geojson.coordinates.length; i++)
			{
				var coord = geojson.coordinates[i];
				var lpoint = new google.maps.LatLng(coord[1],
coord[0]);
				path.push(lpoint);
			}
			options.path = path;
			googleObj = new google.maps.Polyline(options);
		}

		else // Polygon
		{
			var paths = new Array();
```

```
                  for (var i=0; i<geojson.coordinates.length; i++)
                  {
                        for (var j=0; j<geojson.coordinates[i].length;
j++)
                        {
                              var lpoint = new
google.maps.LatLng(geojson.coordinates[i][j][1],
geojson.coordinates[i][j][0]);
                              paths.push(lpoint);
                        }
                  }
                  options.paths = paths;
                  googleObj = new google.maps.Polygon(options);
            }

            return googleObj;
      }

      var castlesVector = new Array();
      var castlesOptions = {
            icon:
"http://localhost/ptyxiaki/public_html/images/castle.png",
            animation: google.maps.Animation.DROP
      };

      function poiRequest(longitude, latitude)
      {
            var strVal = $("#select03 option:selected").val();
            var strTxt = $("#select03 option:selected").text();
            var parsedCastles = new Array();
            $.ajax({
            data: {lon: longitude, lat: latitude, radius: strVal},
            dataType: 'json',
            url:
'http://localhost/ptyxiaki/public_html/processAjax/poiProcess',
            type: 'POST'
            }).done(function(data)
            {
                  if (castlesVector)
                  {
                        for (i in castlesVector)
                        {
                              castlesVector[i].setMap(null);
                        }
                  }
```

```
                  if (!data.wrong)
                  {
                        for (var i=0; i<data.length; i++)
                        {
                              parsedCastles[i] =
JSON.parse(data[i].st_asgeojson);
                        }

                        $("div#info").addClass("alert-success");
                    $("div#info").removeClass("alert-error");
                        $("div#info").html("<p>Point with latitude: "
+ data[0].st_y + " and longitude: " + data[0].st_x + "</p>" +
                        "<p>Radius is: " + strTxt + "</p>");
                        $("div#info").show();

                        var infowindow = new google.maps.InfoWindow();

                        for (var i=0; i<parsedCastles.length; i++)
                        {
                              castlesVector[i] = new
GeoJSON(parsedCastles[i], castlesOptions);
                              var kmDist =
parseFloat((data[i].distance * 40000)/360).toFixed(2);
                              var castleName = data[i].name;
                              castlesVector[i].setMap(map);


      google.maps.event.addListener(castlesVector[i], 'click',
(function(kmDist, castleName) {
                                    return function() {
                                    infowindow.setContent("This is the
<b>"+castleName+
                                    "</b><br/>It is located
<b>"+kmDist+" Km</b> far from the point that you clicked.");
                                    infowindow.open(map, this);
                                    }
                              })(kmDist, castleName));
                        }
                  }

                  else
                  {
                        $("div#info").addClass("alert-error");
                    $("div#info").removeClass("alert-success");
                        $("div#info").text(data.wrong);
                        $("div#info").show();
                  }
```

```
		});
	}

	google.maps.event.addListener(map, 'click', function(event)
	{
		poiRequest(event.latLng.lng(), event.latLng.lat());
	});

	for (var i=0; i<pathsVector.length; i++)
	{
		google.maps.event.addListener(pathsVector[i], 'click',
function(event)
		{
			poiRequest(event.latLng.lng(), event.latLng.lat());
		});
	}

	for (var i=0; i<kingdomsVector.length; i++)
	{
		google.maps.event.addListener(kingdomsVector[i], 'click',
function(event)
		{
			poiRequest(event.latLng.lng(), event.latLng.lat());
		});
	}
});
```