ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής-Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**«Ανάπτυξη εφαρμογής με εξατομίκευση και δημιουργία διαγραμμάτων UML πάνω σε αυτήν»** **«Development of an application with personalization and creation of UML diagrams on it»**
Ονοματεπώνυμο Φοιτητή	**Τσίτσος Γεώργιος**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ21057**
Επιβλέπουσα	**Βίρβου Μαρία, Καθηγήτρια**

Ημερομηνία Παράδοσης, **Φεβρουάριος 2024**

Τριμελής Εξεταστική Επιτροπή

Βίρβου Μαρία	Αλέπης Ευθύμιος	Χρυσαφιάδη Κωνσταντίνα
Καθηγήτρια	Αναπληρωτής Καθηγητής	Επίκουρη Καθηγήτρια

.

Περιεχόμενα

Περίληψη 6
Abstract 7
1. Εισαγωγή 8
2. UML και ανάπτυξη με βάση το μοντέλο 9
2.1 Τι είναι η UML 9
2.2 Ιστορία της UML 9
2.3 Επισκόπηση της UML 10
2.3.1 Προβολές 10
2.3.2 Διαγράμματα 12
2.3.3 Στοιχεία μοντέλου 15
2.3.4 Γενικοί μηχανισμοί 15
2.4 Χρήση της UML 16
2.5 Φάσεις ανάπτυξης συστήματος 17
2.6 Διαγράμματα καταστάσεων 18
2.7 Μέθοδοι της UML 19
3. Δοκιμές λογισμικού 21
3.1 Μέθοδοι δοκιμής 21
3.2 Δοκιμές βασισμένες σε μοντέλα 22
3.3 Διαδικασία ανάλυσης μετάλλαξης 23
4. Το πλαίσιο της κινητής τηλεφωνίας 25
4.1 Μοντελοποίηση εφαρμογών για κινητά τηλέφωνα: κατάσταση προόδου 25
4.1.1 Σχεδιασμός βάσει μοντέλου για την υποστήριξη της ανάπτυξης εφαρμογών 25
4.1.2 Προσεγγίσεις επεκτάσεων της UML 26
4.1.3 Επεκτάσεις της UML για κινητές εφαρμογές 27
4.1.4 Εμπειρικά πειράματα για την αξιολόγηση της χρησιμότητας των συμβολισμών 27
5. Κινητές τεχνολογίες και ανάπτυξη εφαρμογών 29
5.1 Τα κινητά λειτουργικά συστήματα 30
5.1.1 Λειτουργικό σύστημα Android 31
5.1.2 Λειτουργικό σύστημα iPhone 35
5.2 Ανάπτυξη εφαρμογών Android 39

5.2.1 Kotlin 39
5.2.2 Java 40
5.2.3 Android Studio 40
5.2.4 Άλλα εργαλεία 40
5.3 Ανάπτυξη εφαρμογών iOS 41
5.3.1 Swift 42
5.3.2 Objective C 42
5.3.3 Xcode 42
5.4 Ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες 43
5.5 Javascript 44
5.6 MongoDB 45
6. Δημιουργία εξατομικευμένης εφαρμογής 46
6.1 Διαγράμματα UML (usecase, activity, class, sequence) 55
6.1.1 Use Case Diagram 55
6.1.2 Class Diagram 56
6.1.3 Sequence Diagram 57
6.1.4 Activity Diagram 58
7. Συμπεράσματα 59
7.1 Use Case Diagrams 59
7.2 Class Diagrams 59
7.3 Sequence Diagrams 60
7.4 Activity diagrams 60
7.5 Σύγκριση της χρησιμότητας των διαγραμμάτων 61
8. Βιβλιογραφία 63

Κατάλογος Εικόνων

Εικόνα 1: Η προβολή της οθόνης σύνδεσης και ο κώδικας του front end. 50
Εικόνα 2: Σελίδα της εφαρμογής. 51
Εικόνα 3: Σελίδα της εφαρμογής. 51
Εικόνα 4: Σελίδα της εφαρμογής. 52
Εικόνα 5: Σελίδα της εφαρμογής. 52
Εικόνα 6: Use Case Diagram – Παράδειγμα Α. 55
Εικόνα 7: Use Case Diagram – Παράδειγμα Β. 56
Εικόνα 8: Class Diagram. 57
Εικόνα 9: Sequence Diagram: Παράδειγμα διαγράμματος ακολουθίας για την εγγραφή χρήστη. 57
Εικόνα 10: Activity Diagram. 58

- **Περίληψη**

Ο βασικός σκοπός της εργασίας είναι η ανάλυση και η παρουσίαση της διαδικασίας ανάπτυξης λογισμικού με χρήση της γλώσσας μοντελοποίησης UML (Unified Modeling Language), με έμφαση στην εφαρμογή της στον τομέα της κινητής τηλεφωνίας. Η εργασία καλύπτει επίσης θέματα δοκιμής λογισμικού, το πλαίσιο της κινητής τηλεφωνίας και τις τεχνολογίες ανάπτυξης εφαρμογών για κινητές συσκευές, συμπεριλαμβανομένων των λειτουργικών συστημάτων Android και iOS, καθώς και την ανάπτυξη εφαρμογών πολλαπλών πλατφορμών. Το πρακτικό μέρος της εργασίας περιλαμβάνει τη δημιουργία διαγραμμάτων UML, όπως τα διαγράμματα χρήσης, κλάσης, ακολουθίας και δραστηριότητας.

Λέξεις – κλειδιά: ***UML - Ανάπτυξη λογισμικού - Δοκιμές λογισμικού - Κινητή τηλεφωνία - Σχεδιασμός εφαρμογών - Επεκτάσεις της UML***

- **Abstract**

The main purpose of the work is the analysis and presentation of the software development process using the UML (Unified Modeling Language) modeling language, with an emphasis on its application in the field of mobile telephony. The work also covers software testing topics, the mobile framework and mobile application development technologies, including Android and iOS operating systems, as well as cross-platform application development. The practical part of the work involves creating UML diagrams, such as usage, class, sequence and activity diagrams.

Keywords: ***UML - Software Development - Software Testing - Mobile Telephony - Application Design - Extensions of UML***

1. Εισαγωγή

Η ανάπτυξη εφαρμογής είναι μια σύνθετη και συναρπαστική διαδικασία που απαιτεί τη συνεργασία πολλών στοιχείων, από τον σχεδιασμό και την υλοποίηση του λογισμικού έως τη διαχείριση των αναγκών των χρηστών. Ένα από τα βασικά εργαλεία που χρησιμοποιούν οι προγραμματιστές και οι μηχανικοί λογισμικού κατά την ανάπτυξη μιας εφαρμογής είναι τα διαγράμματα UML (Unified Modeling Language), τα οποία αποτελούν ισχυρό εργαλείο για την οπτικοποίηση και τον σχεδιασμό του λογισμικού (Narawita & Vidanage, 2016).

Τα διαγράμματα UML προσφέρουν μια γλώσσα που επιτρέπει στους προγραμματιστές να αναπαραστήσουν τη δομή, τη συμπεριφορά και τις σχέσεις μεταξύ των στοιχείων μιας εφαρμογής. Αυτό βοηθά στον καλύτερο σχεδιασμό του λογισμικού, την αναγνώριση προβλημάτων και την προώθηση της συνεργασίας μεταξύ των μελών της ομάδας ανάπτυξης.

Από την άλλη πλευρά, η εξατομίκευση είναι ένας σημαντικός παράγοντας που συμβάλλει στην επιτυχία μιας εφαρμογής, καθώς οι χρήστες έχουν διάφορες ανάγκες και προτιμήσεις. Η διαδικασία εξατομίκευσης προϋποθέτει την συλλογή αναλυτικών προδιαγραφών και τη δημιουργία προτύπων που προσαρμόζονται σε κάθε χρήστη. Αυτό δεν αφορά μόνο την εμφάνιση της εφαρμογής, αλλά και τη λειτουργικότητα, εξασφαλίζοντας ότι η εφαρμογή ανταποκρίνεται ακριβώς στις ανάγκες κάθε χρήστη.

Οι διαγραμματικές αναπαραστάσεις μπορούν να βοηθήσουν στην ανίχνευση τυχόν αντιφάσεων ή προβλημάτων στον σχεδιασμό και να επιτρέψουν τη διόρθωσή τους προτού το λογισμικό φτάσει στη φάση της υλοποίησης. Αυτό έχει ως αποτέλεσμα την εξοικονόμηση χρόνου και πόρων και τη βελτίωση της ποιότητας του τελικού προϊόντος.

Συνοψίζοντας, η ανάπτυξη εφαρμογής με εξατομίκευση και τη χρήση διαγραμμάτων UML αποτελούν έναν σημαντικό συνδυασμό που συμβάλλει στη δημιουργία υψηλής ποιότητας λογισμικού. Οι προγραμματιστές και οι μηχανικοί λογισμικού μπορούν να επωφεληθούν από την συνδυασμένη χρήση αυτών των δύο στοιχείων για τη δημιουργία προσαρμοσμένων εφαρμογών που ανταποκρίνονται ακριβώς στις ανάγκες των χρηστών και των επιχειρηματικών στόχων (Narawita & Vidanage, 2016).

2. UML και ανάπτυξη με βάση το μοντέλο

2.1 Τι είναι η UML

Η ενοποιημένη γλώσσα μοντελοποίησης (Unified Modelling Language - UML) είναι μια γενική γλώσσα οπτικής μοντελοποίησης συστημάτων. Αν και η UML συνδέεται συχνότερα με τη μοντελοποίηση συστημάτων αντικειμενοστραφούς λογισμικού, έχει πολύ ευρύτερη εφαρμογή από αυτή λόγω των ενσωματωμένων μηχανισμών επεκτασιμότητάς της (Arlow & Neustadt, 2005).

Η UML σχεδιάστηκε για να ενσωματώσει τις τρέχουσες βέλτιστες πρακτικές στις τεχνολογίες μοντελοποίησης και στη μηχανική λογισμικού. Είναι σημαντικό να γίνει κατανοητό ότι η UML δεν προσφέρει κανενός είδους μεθοδολογία μοντελοποίησης. Η ενοποιημένη διαδικασία (Unified Process - UP) είναι μια μεθοδολογία που υποδεικνύει τους εργαζόμενους, τις δραστηριότητες και τα τεχνουργήματα που πρέπει να χρησιμοποιηθούν, να εκτελεστούν ή να δημιουργηθούν προκειμένου να μοντελοποιηθεί ένα σύστημα λογισμικού (Arlow & Neustadt, 2005).

Η UML δεν συνδέεται με κάποια συγκεκριμένη μεθοδολογία ή κύκλο ζωής και μάλιστα είναι ικανή να χρησιμοποιηθεί με τις υπάρχουσες μεθοδολογίες. Η UP χρησιμοποιεί την UML ως το υποκείμενο συντακτικό οπτικής μοντελοποίησης και επομένως είναι δυνατό να θεωρηθεί ότι η UP είναι η προτιμώμενη μέθοδος για την UML, καθώς είναι η καλύτερα προσαρμοσμένη σε αυτήν, αλλά η ίδια η UML μπορεί να παρέχει την υποστήριξη οπτικής μοντελοποίησης για άλλες μεθόδους (Arlow & Neustadt, 2005).

2.2 Ιστορία της UML

Πριν από το 1994, ο κόσμος των αντικειμενοστραφών μεθόδων (Object Oriented methods) ήταν λίγο ακατάστατος. Υπήρχαν διάφορες ανταγωνιστικές γλώσσες και μεθοδολογίες οπτικής μοντελοποίησης, όλες με τα δυνατά και τα αδύνατα σημεία τους και όλες με τα πλεονεκτήματα και τα μειονεκτήματά τους. Όσον αφορά τις γλώσσες οπτικής μοντελοποίησης, οι ξεκάθαροι ηγέτες ήταν ο Booch (μέθοδος Booch) και ο Rumbaugh (Object Modelling Technique - OMT), οι οποίοι μεταξύ τους είχαν πάνω από τη μισή αγορά. Από την πλευρά των μεθοδολογιών, ο Jacobson είχε μακράν την ισχυρότερη υπόθεση, καθώς παρόλο που πολλοί συγγραφείς ισχυρίζονταν ότι είχαν μια "μέθοδο", το μόνο που πολλοί από αυτούς είχαν στην πραγματικότητα ήταν μια σύνταξη οπτικής μοντελοποίησης και μια συλλογή από περισσότερο ή λιγότερο χρήσιμες οδηγίες (Tufail et al., 2018).

Το 1996, η Ομάδα Διαχείρισης Αντικειμένων (Object Management Group - OMG) παρήγαγε ένα αίτημα για πρόταση (Request-For-Proposal - RFP) για μια αντικειμενοστραφή γλώσσα οπτικής μοντελοποίησης και υποβλήθηκε η UML. Το 1997, η OMG αποδέχθηκε την UML και γεννήθηκε η πρώτη ανοικτή, βιομηχανικά τυποποιημένη γλώσσα μοντελοποίησης αντικειμενοστραφούς προσανατολισμού. Έκτοτε, όλες οι ανταγωνιστικές μέθοδοι έχουν ξεθωριάσει και η UML παραμένει αδιαμφισβήτητη ως η τυποποιημένη γλώσσα μοντελοποίησης αντικειμενοστραφούς μοντέλου της βιομηχανίας (Tufail et al., 2018).

Το 2000, η UML 1.4 εισήγαγε μια σημαντική επέκταση της UML με την προσθήκη της σημασιολογίας δράσης. Αυτές περιγράφουν τη συμπεριφορά ενός συνόλου πρωτόγονων ενεργειών που μπορούν να υλοποιηθούν από συγκεκριμένες γλώσσες ενεργειών. Η σημασιολογία ενεργειών και μια γλώσσα ενεργειών επιτρέπουν τον λεπτομερή προσδιορισμό των στοιχείων συμπεριφοράς των μοντέλων της UML (όπως οι πράξεις κλάσεων) απευθείας στο μοντέλο της UML. Αυτή ήταν μια πολύ σημαντική εξέλιξη, καθώς κατέστησε την προδιαγραφή της UML υπολογιστικά πλήρη και έτσι κατέστη δυνατό να γίνουν τα μοντέλα της UML εκτελέσιμα (Narawita & Vidanage, 2016).

Το 2005 ολοκληρώθηκε η προδιαγραφή UML 2.0. Η UML είναι πλέον μια πολύ ώριμη γλώσσα μοντελοποίησης. Είχε αποδείξει την αξία της σε χιλιάδες έργα ανάπτυξης λογισμικού παγκοσμίως. Η UML 2.0 εισήγαγε αρκετά νέα οπτική σύνταξη. Κάποια από αυτά αντικαθιστούν και αποσαφηνίζουν την υπάρχουσα σύνταξη της 1.x, ενώ κάποια από αυτά είναι εντελώς νέα και αντιπροσωπεύουν νέα σημασιολογία που προστέθηκε στη γλώσσα. Η UML παρείχε πάντα πολλές επιλογές σχετικά με τον τρόπο με τον οποίο μπορεί να απεικονιστεί ένα συγκεκριμένο στοιχείο μοντέλου και δεν θα υποστηρίζονται όλες αυτές από κάθε γλώσσα μοντελοποίησης (Narawita & Vidanage, 2016).

2.3 Επισκόπηση της UML

Η UML έχει ευρύ φάσμα χρήσης. Μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση επιχειρήσεων, τη μοντελοποίηση λογισμικού σε όλες τις φάσεις ανάπτυξης και για όλους τους τύπους συστημάτων, καθώς και για τη γενική μοντελοποίηση οποιασδήποτε κατασκευής που έχει στατική δομή και δυναμική συμπεριφορά. Προκειμένου να επιτευχθούν αυτές οι ευρείες δυνατότητες, η γλώσσα ορίζεται να είναι αρκετά εκτεταμένη και γενική ώστε να επιτρέπει τη μοντελοποίηση τόσο διαφορετικών συστημάτων, αποφεύγοντας τα πολύ εξειδικευμένα και πολύ σύνθετα (Nguyen et al., 2015).

Η επισκόπηση της UML έχει τα ακόλουθα διαφορετικά μέρη (Nguyen et al., 2015):

- **Προβολές:** Οι προβολές δείχνουν διάφορες πτυχές του συστήματος που μοντελοποιείται. Μια όψη δεν είναι γράφημα, αλλά μια αφαίρεση που αποτελείται από έναν αριθμό διαγραμμάτων. Μόνο με τον ορισμό ενός αριθμού από όψεις, καθεμία από τις οποίες δείχνει μια συγκεκριμένη πτυχή του συστήματος, μπορεί να κατασκευαστεί μια πλήρης εικόνα του συστήματος. Οι όψεις συνδέουν επίσης τη γλώσσα μοντελοποίησης με τη μέθοδο/διαδικασία που επιλέγεται για την ανάπτυξη.
- **Διαγράμματα:** Τα διαγράμματα είναι οι γραφικές παραστάσεις που περιγράφουν το περιεχόμενο μιας όψης. Η UML διαθέτει εννέα διαφορετικούς τύπους διαγραμμάτων που χρησιμοποιούνται σε συνδυασμό για την παροχή όλων των όψεων του συστήματος.
- **Στοιχεία μοντέλου:** Οι έννοιες που χρησιμοποιούνται στα διαγράμματα είναι στοιχεία μοντέλου που αναπαριστούν κοινές αντικειμενοστραφείς έννοιες, όπως κλάσεις, αντικείμενα και μηνύματα, και τις σχέσεις μεταξύ αυτών των εννοιών, συμπεριλαμβανομένων των συσχετίσεων, των εξαρτήσεων και της γενίκευσης. Ένα στοιχείο μοντέλου χρησιμοποιείται σε πολλά διαφορετικά διαγράμματα, αλλά έχει πάντα την ίδια έννοια και το ίδιο σύμβολο.
- **Γενικοί μηχανισμοί:** Οι γενικοί μηχανισμοί παρέχουν επιπλέον σχόλια, πληροφορίες ή σημασιολογία για ένα στοιχείο μοντέλου- παρέχουν επίσης μηχανισμούς επέκτασης για την προσαρμογή ή την επέκταση της UML σε μια συγκεκριμένη μέθοδο/διαδικασία, οργανισμό ή χρήστη.

2.3.1 Προβολές

Η μοντελοποίηση ενός πολύπλοκου συστήματος είναι ένα εκτεταμένο έργο. Ιδανικά, ολόκληρο το σύστημα περιγράφεται σε ένα ενιαίο γράφημα που ορίζει όλο το σύστημα με σαφήνεια και είναι εύκολο να κατανοηθεί. Ωστόσο, αυτό είναι συνήθως αδύνατο. Ένας ενιαίος γράφος δεν μπορεί να συλλάβει όλες τις πληροφορίες που απαιτούνται για την περιγραφή ενός συστήματος. Ένα σύστημα περιγράφεται με διάφορες πτυχές: λειτουργικές (η στατική δομή και οι δυναμικές αλληλεπιδράσεις του), μη λειτουργικές (απαιτήσεις χρονισμού, αξιοπιστία, ανάπτυξη κ.λπ.) και οργανωτικές πτυχές (οργάνωση εργασιών, αντιστοίχιση σε μονάδες κώδικα κ.λπ.). Έτσι, ένα σύστημα περιγράφεται σε έναν αριθμό από όψεις, όπου κάθε όψη αντιπροσωπεύει μια προβολή της πλήρους περιγραφής του συστήματος, που δείχνει μια συγκεκριμένη πτυχή του συστήματος (Chansuwath & Senivongse, 2016).

Οι διάφορες προβολές είναι οι εξής (Chansuwath & Senivongse, 2016):

- **Προβολή περίπτωσης χρήσης:** Μια όψη που δείχνει τη λειτουργικότητα του συστήματος όπως την αντιλαμβάνονται οι εξωτερικοί παράγοντες.
- **Λογική προβολή:** Μια άποψη που δείχνει πώς η λειτουργικότητα σχεδιάζεται στο εσωτερικό του συστήματος, από την άποψη της στατικής δομής και της δυναμικής συμπεριφοράς του συστήματος.
- **Προβολή στοιχείων:** Μια προβολή που δείχνει την οργάνωση των στοιχείων του κώδικα.
- **Προβολή ταυτόχρονης λειτουργίας:** Μια προβολή που δείχνει την ταυτόχρονη λειτουργία του συστήματος, αντιμετωπίζοντας τα προβλήματα επικοινωνίας και συγχρονισμού που υπάρχουν σε ένα ταυτόχρονο σύστημα.
- **Προβολή ανάπτυξης:** Μια προβολή που δείχνει την ανάπτυξη του συστήματος στη φυσική αρχιτεκτονική με υπολογιστές και συσκευές που ονομάζονται κόμβοι.

Προβολή περίπτωσης χρήσης

Η προβολή περίπτωσης χρήσης περιγράφει τη λειτουργικότητα που πρέπει να παρέχει το σύστημα, όπως την αντιλαμβάνονται οι εξωτερικοί φορείς. Ένας φορέας αλληλεπιδρά με το σύστημα. Μπορεί να είναι ένας χρήστης ή ένα άλλο σύστημα. Η προβολή περίπτωσης χρήστη απευθύνεται σε πελάτες, σχεδιαστές, προγραμματιστές και δοκιμαστές. Περιγράφεται σε διαγράμματα περίπτωσης χρήστη και περιστασιακά σε διαγράμματα δραστηριοτήτων. Η επιθυμητή χρήση του συστήματος περιγράφεται ως ένας αριθμός περιπτώσεων χρήσης στην άποψη περίπτωσης χρήσης, όπου μια περίπτωση χρήσης είναι μια γενική περιγραφή μιας χρήσης του συστήματος (μια ζητούμενη λειτουργία) (Koç et al., 2021).

Η προβολή περίπτωσης χρήσης είναι κεντρική, δεδομένου ότι το περιεχόμενό της οδηγεί την ανάπτυξη των άλλων απόψεων. Ο τελικός στόχος του συστήματος είναι να παρέχει τη λειτουργικότητα που περιγράφεται σε αυτή την άποψη - μαζί με ορισμένες μη λειτουργικές ιδιότητες - επομένως αυτή η άποψη επηρεάζει όλες τις άλλες.

Λογική προβολή

Η λογική προβολή περιγράφει τον τρόπο με τον οποίο παρέχεται η λειτουργικότητα του συστήματος. Απευθύνεται κυρίως σε σχεδιαστές και προγραμματιστές. Σε αντίθεση με την άποψη περιπτώσεων χρήσης, η λογική άποψη εξετάζει το εσωτερικό του συστήματος. Περιγράφει τόσο τις στατικές (κλάσεις, αντικείμενα και σχέσεις) όσο και τις δυναμικές συνεργασίες που συμβαίνουν όταν τα αντικείμενα στέλνουν μηνύματα το ένα στο άλλο για να παρέχουν μια δεδομένη λειτουργία. Ορίζονται επίσης ιδιότητες όπως η συνέπεια και η ταυτόχρονη λειτουργία, καθώς και οι διεπαφές και οι εσωτερικές δομές των κλάσεων.

Η στατική δομή περιγράφεται σε διαγράμματα κλάσεων και αντικειμένων. Η δυναμική μοντελοποίηση περιγράφεται σε διαγράμματα κατάστασης, ακολουθίας, συνεργασίας και δραστηριότητας (Koç et al., 2021).

Προβολή στοιχείων

Η προβολή στοιχείων είναι μια περιγραφή των ενοτήτων υλοποίησης και των εξαρτήσεών τους. Απευθύνεται κυρίως στους προγραμματιστές και αποτελείται από το διάγραμμα συστατικών. Τα συστατικά, τα οποία είναι διαφορετικοί τύποι ενοτήτων κώδικα, εμφανίζονται με τη δομή και τις εξαρτήσεις τους. Μπορούν επίσης να προστεθούν πρόσθετες πληροφορίες σχετικά με τα συστατικά στοιχεία, όπως η κατανομή πόρων (ευθύνη για ένα συστατικό στοιχείο) ή άλλες διοικητικές πληροφορίες, όπως έκθεση προόδου για τις εργασίες ανάπτυξης (Rumpe, 2016).

Προβολή ταυτόχρονης λειτουργίας

Η προβολή ταυτόχρονης λειτουργίας ασχολείται με τον διαχωρισμό του συστήματος σε διεργασίες και επεξεργαστές. Αυτή η πτυχή, η οποία αποτελεί μη λειτουργική ιδιότητα του συστήματος, επιτρέπει την αποτελεσματική χρήση των πόρων, την παράλληλη εκτέλεση και τον χειρισμό ασύγχρονων συμβάντων από το περιβάλλον. Εκτός από το διαχωρισμό του συστήματος σε νήματα ελέγχου που εκτελούνται ταυτόχρονα, η όψη πρέπει επίσης να ασχολείται με την επικοινωνία και το συγχρονισμό αυτών των νημάτων.

Η προβολή ταυτόχρονης λειτουργίας απευθύνεται στους προγραμματιστές και τους ενοποιητές του συστήματος και αποτελείται από δυναμικά διαγράμματα (διαγράμματα κατάστασης, ακολουθίας, συνεργασίας και δραστηριότητας) και διαγράμματα υλοποίησης (διαγράμματα συνιστωσών και ανάπτυξης) (Rumpe, 2016).

Προβολή ανάπτυξης

Η προβολή ανάπτυξης δείχνει τη φυσική ανάπτυξη του συστήματος, όπως οι υπολογιστές και οι συσκευές (κόμβοι) και πώς συνδέονται μεταξύ τους. Η προβολή ανάπτυξης απευθύνεται σε προγραμματιστές, ενοποιητές και αναπαρίσταται από το διάγραμμα ανάπτυξης. Αυτή η προβολή περιλαμβάνει επίσης μια χαρτογράφηση που δείχνει πώς αναπτύσσονται τα στοιχεία στη φυσική αρχιτεκτονική, για παράδειγμα, ποιο πρόγραμμα ή αντικείμενα εκτελούνται σε κάθε αντίστοιχο υπολογιστή (Rumpe, 2016).

2.3.2 Διαγράμματα

Τα διαγράμματα είναι οι πραγματικές γραφικές παραστάσεις που παρουσιάζουν τα σύμβολα των στοιχείων του μοντέλου τοποθετημένα έτσι ώστε να απεικονίζουν ένα συγκεκριμένο μέρος ή πτυχή του συστήματος. Ένα μοντέλο συστήματος έχει συνήθως πολλά διαγράμματα κάθε τύπου. Ένα διάγραμμα αποτελεί μέρος συγκεκριμένης προβολής- και όταν σχεδιάζεται, συνήθως κατανέμεται σε μια προβολή. Ορισμένοι τύποι διαγραμμάτων μπορούν να αποτελούν μέρος πολλών όψεων, ανάλογα με το περιεχόμενο του διαγράμματος (Ciccozzi et al., 2019).

Διάγραμμα περίπτωσης χρήσης

Ένα διάγραμμα περίπτωσης χρήσης δείχνει έναν αριθμό εξωτερικών παραγόντων και τη σύνδεσή τους με τις περιπτώσεις χρήσης που παρέχει το σύστημα (Σχήμα 2). Μια περίπτωση χρήσης είναι η περιγραφή μιας λειτουργικότητας (μια συγκεκριμένη χρήση του συστήματος) που παρέχει το σύστημα. Η περιγραφή της πραγματικής περίπτωσης χρήσης γίνεται συνήθως σε απλό κείμενο ως ιδιότητα τεκμηρίωσης του συμβόλου περίπτωσης χρήσης, αλλά μπορεί επίσης να περιγραφεί με τη χρήση ενός διαγράμματος δραστηριοτήτων. Οι περιπτώσεις χρήσης περιγράφονται μόνο όπως τις βλέπει εξωτερικά ο δράστης (η συμπεριφορά του συστήματος όπως την αντιλαμβάνεται ο χρήστης) και δεν περιγράφουν τον τρόπο με τον οποίο παρέχεται η λειτουργικότητα στο εσωτερικό του συστήματος. Οι περιπτώσεις χρήσης καθορίζουν τις απαιτήσεις λειτουργικότητας του συστήματος (Ciccozzi et al., 2019).

Διάγραμμα κλάσεων

Ένα διάγραμμα κλάσεων δείχνει τη στατική δομή των κλάσεων του συστήματος. Οι κλάσεις αντιπροσωπεύουν τα "πράγματα" που χειρίζεται το σύστημα. Οι κλάσεις μπορούν να σχετίζονται μεταξύ τους με διάφορους τρόπους: συσχετίζονται (συνδέονται μεταξύ τους), εξαρτώνται (μια κλάση εξαρτάται/χρησιμοποιεί μια άλλη κλάση), εξειδικεύονται (μια κλάση είναι εξειδίκευση μιας άλλης κλάσης) ή συσκευάζονται (ομαδοποιούνται ως μονάδα). Όλες αυτές οι σχέσεις απεικονίζονται σε ένα διάγραμμα κλάσεων μαζί με την εσωτερική δομή των κλάσεων όσον αφορά τα χαρακτηριστικά και τις λειτουργίες. Το διάγραμμα θεωρείται στατικό, δεδομένου ότι η δομή που περιγράφεται ισχύει πάντα σε οποιοδήποτε σημείο του κύκλου ζωής του συστήματος.

Ένα σύστημα έχει συνήθως έναν αριθμό διαγραμμάτων κλάσεων. Δεν εισάγονται όλες οι κλάσεις σε ένα μόνο διάγραμμα κλάσεων και μια κλάση μπορεί να συμμετέχει σε πολλά διαγράμματα κλάσεων (Rumpe, 2016).

Διάγραμμα αντικειμένων

Το διάγραμμα αντικειμένων είναι μια παραλλαγή του διαγράμματος κλάσεων και χρησιμοποιεί σχεδόν πανομοιότυπο συμβολισμό. Η διαφορά μεταξύ των δύο είναι ότι το διάγραμμα αντικειμένου δείχνει έναν αριθμό αντικειμενικών περιπτώσεων κλάσεων, αντί για πραγματικές κλάσεις. Ένα διάγραμμα αντικειμένου είναι επομένως ένα παράδειγμα διαγράμματος κλάσεων που δείχνει ένα πιθανό στιγμιότυπο της εκτέλεσης του συστήματος: πώς θα μπορούσε να μοιάζει το σύστημα σε κάποια χρονική στιγμή. Χρησιμοποιείται ο ίδιος συμβολισμός με το

διάγραμμα κλάσεων, με δύο εξαιρέσεις: τα αντικείμενα γράφονται με υπογραμμισμένα τα ονόματά τους και όλες οι περιπτώσεις σε μια σχέση.

Τα διαγράμματα αντικειμένων δεν είναι τόσο σημαντικά όσο τα διαγράμματα κλάσεων, αλλά μπορούν να χρησιμοποιηθούν για να παραδειγματίσουν ένα σύνθετο διάγραμμα κλάσεων, δείχνοντας πώς θα μπορούσαν να μοιάζουν τα πραγματικά στιγμιότυπα και οι σχέσεις. Τα διαγράμματα αντικειμένων χρησιμοποιούνται επίσης ως μέρος των διαγραμμάτων συνεργασίας, στα οποία παρουσιάζεται η δυναμική συνεργασία μεταξύ ενός συνόλου αντικειμένων (Ciccozzi et al., 2019).

Διάγραμμα κατάστασης

Ένα διάγραμμα κατάστασης αποτελεί συνήθως συμπλήρωμα της περιγραφής μιας κλάσης. Δείχνει όλες τις πιθανές καταστάσεις που μπορούν να έχουν τα αντικείμενα της κλάσης και ποια γεγονότα προκαλούν την αλλαγή της κατάστασης. Ένα γεγονός μπορεί να είναι ένα άλλο αντικείμενο που στέλνει ένα μήνυμα σε αυτό, για παράδειγμα, ότι έχει παρέλθει ένας συγκεκριμένος χρόνος, ή ότι έχει εκπληρωθεί κάποια συνθήκη. Η αλλαγή της κατάστασης ονομάζεται μετάβαση. Μια μετάβαση μπορεί επίσης να συνδέεται με μια ενέργεια που καθορίζει τι πρέπει να γίνει σε σχέση με τη μετάβαση κατάστασης.

Τα διαγράμματα καταστάσεων δεν σχεδιάζονται για όλες τις κλάσεις, αλλά μόνο για εκείνες που έχουν έναν αριθμό καλά καθορισμένων καταστάσεων και όπου η συμπεριφορά της κλάσης επηρεάζεται και μεταβάλλεται από τις διάφορες καταστάσεις. Τα διαγράμματα καταστάσεων μπορούν επίσης να σχεδιαστούν για το σύστημα στο σύνολό του (Unhelkar, 2017).

Διαγράμματα ακολουθίας

Ένα διάγραμμα ακολουθίας δείχνει μια δυναμική συνεργασία μεταξύ ενός αριθμού αντικειμένων. Η σημαντική πτυχή αυτού του διαγράμματος είναι να δείχνει μια ακολουθία μηνυμάτων που αποστέλλονται μεταξύ των αντικειμένων. Δείχνει επίσης μια αλληλεπίδραση μεταξύ των αντικειμένων, κάτι που θα συμβεί σε ένα συγκεκριμένο σημείο της εκτέλεσης του συστήματος. Το διάγραμμα αποτελείται από έναν αριθμό αντικειμένων που απεικονίζονται με κάθετες γραμμές. Ο χρόνος περνάει προς τα κάτω στο διάγραμμα και το διάγραμμα δείχνει την ανταλλαγή μηνυμάτων μεταξύ των αντικειμένων καθώς περνάει ο χρόνος στην ακολουθία ή τη λειτουργία. Τα μηνύματα απεικονίζονται ως γραμμές με βέλη μηνυμάτων μεταξύ των κάθετων γραμμών των αντικειμένων. Οι χρονικές προδιαγραφές και άλλα σχόλια προστίθενται σε ένα σενάριο στο περιθώριο του διαγράμματος (Unhelkar, 2017).

Διάγραμμα συνεργασίας

Ένα διάγραμμα συνεργασίας δείχνει μια δυναμική συνεργασία, όπως ακριβώς και το διάγραμμα ακολουθίας. Συχνά υπάρχει η δυνατότητα επιλογής μεταξύ της παρουσίασης της συνεργασίας είτε ως διάγραμμα ακολουθίας είτε ως διάγραμμα συνεργασίας. Εκτός από την παρουσίαση της ανταλλαγής μηνυμάτων, που ονομάζεται αλληλεπίδραση, το διάγραμμα συνεργασίας παρουσιάζει τα αντικείμενα και τις σχέσεις τους, το οποίο μερικές φορές αναφέρεται ως πλαίσιο. Το αν θα χρησιμοποιηθεί ένα διάγραμμα ακολουθίας ή ένα διάγραμμα συνεργασίας μπορεί συχνά να αποφασιστεί από: Εάν ο χρόνος ή η ακολουθία είναι η πιο σημαντική πτυχή που πρέπει να τονιστεί, επιλέξτε το διάγραμμα ακολουθίας- εάν το πλαίσιο είναι σημαντικό να τονιστεί, επιλέξτε το διάγραμμα συνεργασίας. Η αλληλεπίδραση μεταξύ των αντικειμένων παρουσιάζεται και στα δύο διαγράμματα (Chansuwath & Senivongse, 2016).

Το διάγραμμα συνεργασίας σχεδιάζεται ως διάγραμμα αντικειμένων, όπου παρουσιάζεται ένας αριθμός αντικειμένων μαζί με τις σχέσεις τους, χρησιμοποιώντας τον συμβολισμό του διαγράμματος κλάσεων/αντικειμένων. Τα βέλη μηνυμάτων σχεδιάζονται μεταξύ των αντικειμένων για να δείξουν τη ροή των μηνυμάτων μεταξύ των αντικειμένων. Στα μηνύματα τοποθετούνται ετικέτες, οι οποίες, μεταξύ άλλων, δείχνουν τη σειρά με την οποία αποστέλλονται τα μηνύματα. Μπορούν επίσης να δείξουν συνεργασίες, επαναλήψεις, τιμές επιστροφής κ.ο.κ.

Όταν εξοικειωθεί με τη σύνταξη των ετικετών μηνυμάτων, ένας προγραμματιστής μπορεί να διαβάσει τη συνεργασία και να παρακολουθήσει τη ροή εκτέλεσης και την ανταλλαγή μηνυμάτων. Ένα διάγραμμα συνεργασίας μπορεί επίσης να περιέχει ενεργά αντικείμενα, αυτά που εκτελούνται ταυτόχρονα με άλλα ενεργά αντικείμενα (Chansuwath & Senivongse, 2016).

Διάγραμμα δραστηριότητας

Ένα διάγραμμα δραστηριότητας δείχνει μια διαδοχική ροή δραστηριοτήτων. Το διάγραμμα δραστηριοτήτων χρησιμοποιείται συνήθως για την περιγραφή των δραστηριοτήτων που εκτελούνται σε μια λειτουργία, αν και μπορεί επίσης να χρησιμοποιηθεί για την περιγραφή άλλων ροών δραστηριοτήτων, όπως μια περίπτωση χρήσης ή μια αλληλεπίδραση. Το διάγραμμα δραστηριότητας αποτελείται από καταστάσεις δράσης, οι οποίες περιέχουν μια προδιαγραφή μιας δραστηριότητας που πρέπει να εκτελεστεί μια ενέργεια. Μια κατάσταση δράσης θα εγκαταλείψει την κατάσταση όταν η ενέργεια έχει εκτελεστεί. Μια κατάσταση σε ένα διάγραμμα καταστάσεων χρειάζεται ένα ρητό γεγονός πριν εγκαταλείψει την κατάσταση. Έτσι, ο έλεγχος ρέει μεταξύ των καταστάσεων δράσης, οι οποίες συνδέονται μεταξύ τους. Οι αποφάσεις και οι συνθήκες, καθώς και η παράλληλη εκτέλεση καταστάσεων δράσης, μπορούν επίσης να απεικονιστούν στο διάγραμμα. Το διάγραμμα μπορεί επίσης να περιέχει προδιαγραφές των μηνυμάτων που αποστέλλονται ή λαμβάνονται ως μέρος των ενεργειών που εκτελούνται (Bowman et al., 2016).

Διάγραμμα συνιστωσών

Ένα διάγραμμα συνιστωσών δείχνει τη φυσική δομή του κώδικα από την άποψη των στοιχείων του κώδικα. Ένα στοιχείο μπορεί να είναι ένα στοιχείο πηγαίου κώδικα, ένα δυαδικό στοιχείο ή ένα εκτελέσιμο στοιχείο. Ένα στοιχείο περιέχει πληροφορίες για τη λογική κλάση ή τις λογικές κλάσεις που υλοποιεί, δημιουργώντας έτσι μια αντιστοίχιση από τη λογική προβολή στην προβολή στοιχείων. Οι εξαρτήσεις μεταξύ των στοιχείων εμφανίζονται, καθιστώντας εύκολη την ανάλυση του τρόπου με τον οποίο άλλα στοιχεία επηρεάζονται από μια αλλαγή σε ένα στοιχείο. Τα στοιχεία μπορούν επίσης να εμφανίζονται με οποιαδήποτε από τις διεπαφές που εκθέτουν, όπως οι διεπαφές OLE/COM και μπορούν να ομαδοποιηθούν σε πακέτα. Το διάγραμμα συνιστωσών χρησιμοποιείται στην πρακτική προγραμματιστική εργασία (Bowman et al., 2016).

Διάγραμμα ανάπτυξης

Το διάγραμμα ανάπτυξης δείχνει τη φυσική αρχιτεκτονική του υλικού και του λογισμικού του συστήματος. Σε αυτό το διάγραμμα, οι πραγματικοί υπολογιστές και οι συσκευές (κόμβοι) παρουσιάζονται μαζί με τις συνδέσεις μεταξύ τους, συμπεριλαμβανομένου του τύπου των συνδέσεων. Επίσης, στο εσωτερικό των κόμβων απεικονίζονται τα εκτελέσιμα στοιχεία και τα αντικείμενα για να παρουσιαστούν ποιες μονάδες λογισμικού εκτελούνται σε ποιους κόμβους. Επίσης, υπάρχει η δυνατότητα παρουσίασης των εξαρτήσεων μεταξύ των στοιχείων στο διάγραμμα ανάπτυξης, προσδιορίζοντας ποια στοιχεία εξαρτώνται από άλλα (Bowman et al., 2016).

Το διάγραμμα ανάπτυξης περιγράφει την πραγματική αρχιτεκτονική του συστήματος. Αυτό απέχει πολύ από τη λειτουργική περιγραφή της άποψης περίπτωσης χρήσης. Ωστόσο, με ένα καλά καθορισμένο μοντέλο, είναι δυνατή η πλοήγηση σε όλη τη διαδρομή από έναν κόμβο της φυσικής αρχιτεκτονικής στα συστατικά του στην κλάση που υλοποιεί στην οποία συμμετέχουν τα αντικείμενα της κλάσης και τέλος σε μια περίπτωση χρήσης. Οι διαφορετικές όψεις του συστήματος χρησιμοποιούνται για να δώσουν μια συνεκτική περιγραφή του συστήματος στο σύνολό του (Bowman et al., 2016).

2.3.3 Στοιχεία μοντέλου

Οι έννοιες που χρησιμοποιούνται στα διαγράμματα ονομάζονται στοιχεία μοντέλου. Ένα στοιχείο μοντέλου ορίζεται με σημασιολογία, έναν επίσημο ορισμό των στοιχείων ή την ακριβή έννοια αυτού που αντιπροσωπεύει σε αδιαμφισβήτητες δηλώσεις. Ένα στοιχείο μοντέλου έχει επίσης ένα αντίστοιχο στοιχείο προβολής, το οποίο είναι η γραφική αναπαράσταση του στοιχείου ή το γραφικό σύμβολο που χρησιμοποιείται για την αναπαράσταση του στοιχείου στα διαγράμματα. Ένα στοιχείο μπορεί να υπάρχει σε πολλούς διαφορετικούς τύπους διαγραμμάτων, αλλά υπάρχουν κανόνες για το ποια στοιχεία μπορούν να εμφανίζονται σε κάθε τύπο διαγράμματος. Ορισμένα παραδείγματα στοιχείων μοντέλου είναι η κλάση, το αντικείμενο, η κατάσταση, ο κόμβος, το πακέτο και το συστατικό (Koç et al., 2021).

Ορισμένες διαφορετικές σχέσεις είναι οι εξής (Koç et al., 2021):

- **Συσχέτιση:** Συνδέει στοιχεία και συνδέει περιπτώσεις
- **Γενίκευση:** Σημαίνει ότι ένα στοιχείο μπορεί να είναι εξειδίκευση ενός άλλου στοιχείου.
- **Εξάρτηση:** Δείχνει ότι ένα στοιχείο εξαρτάται με κάποιο τρόπο από ένα άλλο στοιχείο.
- **Συγκέντρωση:** Μια μορφή συσχέτισης στην οποία ένα στοιχείο περιέχει άλλο στοιχείο.

Άλλα στοιχεία του μοντέλου εκτός από αυτά που περιγράφονται περιλαμβάνουν μηνύματα, ενέργειες και στερεότυπα.

2.3.4 Γενικοί μηχανισμοί

Η UML χρησιμοποιεί ορισμένους γενικούς μηχανισμούς σε όλα τα διαγράμματα, για πρόσθετες πληροφορίες σε ένα διάγραμμα, συνήθως αυτές που δεν μπορούν να αναπαρασταθούν με τις βασικές ικανότητες των στοιχείων του μοντέλου.

Διακοσμήσεις

Γραφικές διακοσμήσεις μπορούν να προσαρτηθούν στα στοιχεία μοντέλου των διαγραμμάτων. Οι διακοσμήσεις προσθέτουν σημασιολογία στο στοιχείο. Ένα στολίδι είναι, για παράδειγμα, μια τεχνική που χρησιμοποιείται για τον διαχωρισμό ενός τύπου και το όνομά του εμφανίζεται με έντονη γραφή. Όταν το ίδιο στοιχείο αντιπροσωπεύει μια περίπτωση του τύπου, το όνομά του είναι υπογραμμισμένο και μπορεί να προσδιορίζει τόσο το όνομα της περίπτωσης όσο και το όνομα του τύπου. Ένα ορθογώνιο κλάσης, με το όνομα με έντονα γράμματα να αντιπροσωπεύει μια κλάση και το όνομα υπογραμμισμένο να αντιπροσωπεύει ένα αντικείμενο, είναι ένα παράδειγμα αυτού. Το ίδιο ισχύει και για τους κόμβους, όπου το σύμβολο του κόμβου μπορεί να είναι είτε ένας τύπος με έντονη γραφή, όπως το Printer, είτε μια περίπτωση ενός τύπου κόμβου, όπως ο εκτυπωτής HP 5MP του John. Άλλα στολίδια είναι οι προδιαγραφές της πολλαπλότητας των σχέσεων, όπου η πολλαπλότητα είναι ένας αριθμός ή ένα εύρος που δείχνει πόσες περιπτώσεις συνδεδεμένων τύπων μπορούν να συμμετέχουν στη σχέση. Τα στολίδια γράφονται κοντά στα στοιχεία στα οποία προσθέτουν πληροφορίες. Όλες οι διακοσμήσεις περιγράφονται σε συνδυασμό με την περιγραφή του στοιχείου που επηρεάζουν (Nguyen et al., 2015).

Σημειώσεις

Δεν μπορούν να οριστούν τα πάντα σε μια γλώσσα μοντελοποίησης, ανεξάρτητα από το πόσο εκτεταμένη είναι η γλώσσα. Για να καταστεί δυνατή η προσθήκη πληροφοριών σε ένα μοντέλο που διαφορετικά δεν μπορούν να αναπαρασταθούν, η UML παρέχει τη δυνατότητα σημειώσεων. Μια σημείωση μπορεί να τοποθετηθεί οπουδήποτε σε οποιοδήποτε διάγραμμα και μπορεί να περιέχει οποιοδήποτε τύπο πληροφορίας. Ο τύπος της πληροφορίας της είναι μια συμβολοσειρά που δεν διακόπτεται από την UML. Η σημείωση συνήθως επισυνάπτεται σε κάποιο στοιχείο του διαγράμματος με μια διακεκομμένη γραμμή που καθορίζει ποιο στοιχείο εξηγείται ή αναλύεται, μαζί με τις πληροφορίες της σημείωσης.

Μια σημείωση συχνά περιέχει σχόλια ή ερωτήσεις από τον μοντελοποιητή ως υπενθύμιση για την επίλυση ενός διλήμματος σε μεταγενέστερο χρόνο. Οι σημειώσεις μπορούν επίσης να έχουν στερεότυπα που περιγράφουν τον τύπο της σημείωσης (Nguyen et al., 2015).

2.4 Χρήση της UML

Η UML χρησιμοποιείται για τη μοντελοποίηση συστημάτων, το εύρος των οποίων είναι πολύ ευρύ- πολλοί διαφορετικοί τύποι συστημάτων μπορούν να περιγραφούν με τη UML. Η UML μπορεί επίσης να χρησιμοποιηθεί στις διάφορες φάσεις της ανάπτυξης ενός συστήματος, από την προδιαγραφή των απαιτήσεων μέχρι τη δοκιμή ενός τελικού συστήματος (Nassar, 2003).

Στόχος της UML είναι η περιγραφή οποιουδήποτε τύπου συστήματος, με τη μορφή αντικειμενοστραφών διαγραμμάτων. Φυσικά, η πιο συνηθισμένη χρήση είναι η δημιουργία μοντέλων συστημάτων λογισμικού, αλλά η UML χρησιμοποιείται επίσης για την περιγραφή μηχανικών συστημάτων χωρίς καθόλου λογισμικό, ή την οργάνωση μιας επιχείρησης.

Οι διάφοροι τύποι συστημάτων που μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση με τη UML περιλαμβάνουν (Nassar, 2003):

- **Πληροφοριακά συστήματα:** Αποθήκευση, ανάκτηση, μετασχηματισμός και παρουσίαση πληροφοριών στους χρήστες. Διαχειρίζονται μεγάλες ποσότητες δεδομένων με πολύπλοκες σχέσεις, τα οποία αποθηκεύονται σε σχεσιακές ή αντικειμενικές βάσεις δεδομένων.
- **Τεχνικά συστήματα:** Χειρίζονται και ελέγχουν τεχνικό εξοπλισμό, όπως τηλεπικοινωνίες, στρατιωτικά συστήματα ή βιομηχανικές διαδικασίες. Πρέπει να χειρίζονται τις ειδικές διεπαφές του εξοπλισμού και διαθέτουν λιγότερο τυποποιημένο λογισμικό από τα πληροφοριακά συστήματα. Τα τεχνικά συστήματα είναι συχνά συστήματα πραγματικού χρόνου.
- **Ενσωματωμένα συστήματα πραγματικού χρόνου:** Εκτελούνται σε απλό υλικό ενσωματωμένο σε κάποιον άλλο εξοπλισμό, όπως κινητό τηλέφωνο, αυτοκίνητο, οικιακή συσκευή κ.λπ. Αυτό επιτυγχάνεται μέσω προγραμματισμού χαμηλού επιπέδου που απαιτεί υποστήριξη πραγματικού χρόνου. Τα συστήματα αυτά συχνά δεν διαθέτουν συσκευές όπως οθόνη, σκληρό δίσκο κ.λπ.
- **Κατανεμημένα συστήματα:** Κατανεμημένα σε έναν αριθμό μηχανών όπου τα δεδομένα μεταφέρονται εύκολα από τη μία μηχανή στην άλλη. Απαιτούν μηχανισμούς συγχρονισμένης επικοινωνίας για να εξασφαλίσουν την ακεραιότητα των δεδομένων και συχνά βασίζονται σε μηχανισμούς αντικειμένων όπως το CORBA, το COM/DCOM ή το Java Beans/RMI.
- **Λογισμικό συστήματος:** Καθορίζει την τεχνική υποδομή που χρησιμοποιεί άλλο λογισμικό. Τα λειτουργικά συστήματα, οι βάσεις δεδομένων και οι διεπαφές χρήστη εκτελούν λειτουργίες χαμηλού επιπέδου στο υλικό, ενώ παράλληλα παρουσιάζουν γενικές διεπαφές για χρήση από άλλο λογισμικό.
- **Επιχειρησιακά συστήματα:** Περιγράφουν τους στόχους, τους πόρους (ανθρώπινους, υπολογιστές κ.λπ.), τους κανόνες (νόμοι, επιχειρηματικές στρατηγικές, πολιτικές κ.λπ.) και την πραγματική εργασία στην επιχείρηση (επιχειρηματικές διαδικασίες).

Είναι σημαντικό να τονιστεί ότι τα περισσότερα συστήματα δεν εντάσσονται με σαφήνεια σε μία από αυτές τις κατηγορίες, αλλά ανήκουν σε περισσότερους από έναν τύπους συστημάτων ή ως συνδυασμός. Για παράδειγμα, πολλά από τα σημερινά πληροφοριακά συστήματα έχουν τόσο κατανεμημένες απαιτήσεις όσο και απαιτήσεις πραγματικού χρόνου. Η UML έχει τη δυνατότητα μοντελοποίησης όλων αυτών των τύπων συστημάτων (Nassar, 2003).

2.5 Φάσεις ανάπτυξης συστήματος

Υπάρχουν πέντε φάσεις ανάπτυξης συστημάτων: ανάλυση απαιτήσεων, ανάλυση, σχεδιασμός, προγραμματισμός και δοκιμή.

1. Ανάλυση απαιτήσεων

Η UML διαθέτει περιπτώσεις χρήσης για την καταγραφή των απαιτήσεων του πελάτη. Μέσω της μοντελοποίησης περιπτώσεων χρήσης, μοντελοποιούνται οι εξωτερικοί παράγοντες που ενδιαφέρονται για το σύστημα μαζί με τη λειτουργικότητα που απαιτούν από το σύστημα (οι περιπτώσεις χρήσης). Οι φορείς και οι περιπτώσεις χρήσης μοντελοποιούνται με σχέσεις και έχουν συνδέσεις επικοινωνίας μεταξύ τους ή αναλύονται σε ιεραρχίες. Οι φορείς και οι περιπτώσεις χρήσης περιγράφονται σε ένα διάγραμμα περιπτώσεων χρήσης της UML. Κάθε περίπτωση χρήσης περιγράφεται με κείμενο, και το οποίο προσδιορίζει τις απαιτήσεις του πελάτη: Τι περιμένει από το σύστημα, χωρίς να εξετάζεται ο τρόπος με τον οποίο θα υλοποιηθεί η λειτουργικότητα. Μια ανάλυση απαιτήσεων μπορεί να γίνει και με επιχειρηματικές διαδικασίες, όχι μόνο για συστήματα λογισμικού (Min et al., 2011).

2. Ανάλυση

Η φάση της ανάλυσης ασχολείται με τις πρωταρχικές αφαιρέσεις (κλάσεις και αντικείμενα) και τους μηχανισμούς που υπάρχουν στον προβληματικό τομέα. Οι κλάσεις που τις μοντελοποιούν προσδιορίζονται, μαζί με τις μεταξύ τους σχέσεις, και περιγράφονται σε ένα διάγραμμα κλάσεων της UML. Περιγράφονται επίσης οι συνεργασίες μεταξύ των κλάσεων για την εκτέλεση των περιπτώσεων χρήσης, μέσω οποιουδήποτε από τα δυναμικά μοντέλα της UML. Στην ανάλυση μοντελοποιούνται μόνο οι κλάσεις που ανήκουν στο πεδίο του προβλήματος (έννοιες του πραγματικού κόσμου) - όχι τεχνικές κλάσεις που καθορίζουν λεπτομέρειες και λύσεις στο σύστημα λογισμικού, όπως κλάσεις για τη διεπαφή χρήστη, τις βάσεις δεδομένων, την επικοινωνία, την ταυτόχρονη χρήση κ.ο.κ. (Dennis et al., 2015).

3. Σχεδιασμός

Στη φάση της σχεδίασης, το αποτέλεσμα της ανάλυσης επεκτείνεται σε μια τεχνική λύση. Προστίθενται νέες κλάσεις για την παροχή της τεχνικής υποδομής: η διεπαφή χρήστη, ο χειρισμός βάσεων δεδομένων για την αποθήκευση αντικειμένων σε μια βάση δεδομένων, η επικοινωνία με άλλα συστήματα, η διασύνδεση με συσκευές του συστήματος και άλλα. Οι κλάσεις του προβλήματος του τομέα από την ανάλυση "ενσωματώνονται" σε αυτή την τεχνική υποδομή, καθιστώντας δυνατή την αλλαγή τόσο του τομέα του προβλήματος όσο και της υποδομής. Ο σχεδιασμός καταλήγει σε λεπτομερείς προδιαγραφές για τη φάση της κατασκευής (Min et al., 2011).

4. Προγραμματισμός

Στη φάση προγραμματισμού ή κατασκευής, οι κλάσεις από τη φάση σχεδιασμού μετατρέπονται σε πραγματικό κώδικα σε μια αντικειμενοστραφή γλώσσα προγραμματισμού (δεν συνιστάται η χρήση διαδικαστικής γλώσσας). Ανάλογα με τις δυνατότητες της χρησιμοποιούμενης γλώσσας, αυτό μπορεί να είναι είτε δύσκολο είτε εύκολο έργο. Κατά τη δημιουργία μοντέλων ανάλυσης και σχεδιασμού σε UML, είναι καλύτερο να αποφεύγεται η προσπάθεια νοητικής μετάφρασης των μοντέλων σε κώδικα. Στις πρώτες φάσεις, τα μοντέλα είναι ένα μέσο για την κατανόηση και τη δόμηση ενός συστήματος- συνεπώς, το να προκύπτουν πρόωρα συμπεράσματα για τον κώδικα μπορεί να είναι αντιπαραγωγικό για τη δημιουργία απλών και σωστών μοντέλων. Ο προγραμματισμός είναι μια ξεχωριστή φάση κατά την οποία τα μοντέλα μετατρέπονται σε κώδικα (Dennis et al., 2015).

5. Δοκιμή

Ένα σύστημα δοκιμάζεται συνήθως με δοκιμές μονάδας, δοκιμές ολοκλήρωσης, δοκιμές συστήματος και δοκιμές αποδοχής. Οι δοκιμές μονάδας αφορούν μεμονωμένες κλάσεις ή μια ομάδα κλάσεων και συνήθως εκτελούνται από τον προγραμματιστή. Η δοκιμή ολοκλήρωσης ενσωματώνει στοιχεία και κλάσεις προκειμένου να επαληθευτεί ότι συνεργάζονται όπως έχει καθοριστεί. Η δοκιμή συστήματος βλέπει το σύστημα ως "μαύρο κουτί" και επικυρώνει ότι το σύστημα διαθέτει την τελική λειτουργικότητα που αναμένει ο τελικός χρήστης. Η δοκιμή αποδοχής που διενεργείται από τον πελάτη για να επαληθεύσει ότι το σύστημα ικανοποιεί τις απαιτήσεις είναι παρόμοια με τη δοκιμή συστήματος. Οι διάφορες ομάδες δοκιμών χρησιμοποιούν διαγράμματα UML ως βάση για την εργασία τους: οι δοκιμές μονάδας χρησιμοποιούν διαγράμματα κλάσεων και προδιαγραφές κλάσεων, οι δοκιμές ολοκλήρωσης χρησιμοποιούν συνήθως διαγράμματα συστατικών και διαγράμματα συνεργασίας, και οι δοκιμές συστήματος εφαρμόζουν διαγράμματα περιπτώσεων χρήστη για να επικυρώσουν ότι το σύστημα συμπεριφέρεται όπως έχει αρχικά οριστεί στα εν λόγω διαγράμματα (Min et al., 2011).

2.6 Διαγράμματα καταστάσεων

Όλα τα συστήματα έχουν στατική δομή και δυναμική συμπεριφορά, και η UML παρέχει διαγράμματα για την καταγραφή και περιγραφή και των δύο αυτών πτυχών. Τα διαγράμματα κλάσεων χρησιμοποιούνται καλύτερα για την τεκμηρίωση και την έκφραση της στατικής δομής ενός συστήματος - των κλάσεων, των αντικειμένων και των σχέσεων τους. Τα διαγράμματα κατάστασης, ακολουθίας, συνεργασίας και δραστηριότητας χρησιμοποιούνται καλύτερα για να εκφράσουν τη συμπεριφορά (τη δυναμική) ενός συστήματος, για να καταδείξουν πώς τα αντικείμενα αλληλοεπιδρούν δυναμικά σε διαφορετικές χρονικές στιγμές κατά τη διάρκεια της εκτέλεσης του συστήματος (Sunitha & Samuel, 2019).

Τα διαγράμματα κατάστασης αποτυπώνουν τους κύκλους ζωής των αντικειμένων, των υποσυστημάτων και των συστημάτων. Λένε τις καταστάσεις που μπορεί να έχει ένα αντικείμενο και πώς τα γεγονότα (λαμβανόμενα μηνύματα, χρόνος που παρήλθε, σφάλματα και συνθήκες που γίνονται αληθινές) επηρεάζουν αυτές τις καταστάσεις με την πάροδο του χρόνου. Ένα διάγραμμα κατάστασης πρέπει να επισυνάπτεται σε όλες τις κλάσεις που έχουν σαφώς αναγνωρίσιμες καταστάσεις και σύνθετη συμπεριφορά- το διάγραμμα καθορίζει τη συμπεριφορά και πώς αυτή διαφέρει ανάλογα με την τρέχουσα κατάσταση. Απεικονίζει επίσης ποια γεγονότα θα αλλάξουν την κατάσταση του αντικειμένου της κλάσης.

Το σύστημα, τα υποσυστήματα και τα συστατικά στοιχεία θεωρούνται συνήθως είτε ως μετασχηματιστικά είτε ως αντιδραστικά. Τα μετασχηματιστικά συστατικά είναι συστατικά χωρίς κατάσταση, δηλαδή δεν έχουν μνήμη που να παραμένει μεταξύ διαδοχικών κλήσεων των συστατικών. Τα αντιδραστικά συστήματα, αντίθετα, εκτελούν έναν συνεχή και συχνά ατελείωτο υπολογισμό, στον οποίο κάθε κλήση χρησιμοποιεί πληροφορίες ενσωματωμένες στη μνήμη του παρελθόντος που απαιτούνται για το μέλλον. Τα μοντέλα που βασίζονται σε καταστάσεις, όπως τα διαγράμματα καταστάσεων της UML, είναι μια από τις πιο δημοφιλείς μεθόδους περιγραφής της συμπεριφοράς των αντιδραστικών στοιχείων (Sunitha & Samuel, 2019).

Οι μηχανές πεπερασμένων καταστάσεων (Finite state machines - FSM) χρησιμοποιούνται για περισσότερο από μισό αιώνα για την περιγραφή αντιδραστικών στοιχείων. Τα διαγράμματα καταστάσεων, που αποτελούν επέκταση των FSM, συνεχίζουν αυτή την παράδοση και είναι η πιο δημοφιλής γλώσσα για τη μοντελοποίηση αντιδραστικών στοιχείων.

Πριν από την εισαγωγή των διαγραμμάτων καταστάσεων, οι FSM θεωρούνταν συχνά μια καλή αναπαράσταση για τα μοντέλα τάξης των αντιδραστικών στοιχείων, αλλά στην πράξη, όταν εφαρμόζονται σε μεγαλύτερα προβλήματα, τα μοντέλα ήταν ακατάστατα και δυσανάγνωστα. Ο κύριος λόγος για αυτό είναι ότι τα FSM είναι επίπεδα και διαδοχικά και επομένως δεν κλιμακώνονται καλά όταν εφαρμόζονται σε μεγάλα συστατικά. Ο David Harel εισήγαγε τα διαγράμματα καταστάσεων τη δεκαετία του 1980 ως γλώσσα για την περιγραφή πολύπλοκων αντιδραστικών συστημάτων που απαντώνται στα σύγχρονα συστήματα αεροναυπηγικής. Τις επόμενες δύο δεκαετίες, τα διαγράμματα καταστάσεων υιοθετήθηκαν από τη μεθοδολογία Object Modelling Technique (OMT) και αργότερα από τον απόγονό της, το πρότυπο Unified Modelling Language (UML).

Με λίγα λόγια, τα διαγράμματα καταστάσεων επεκτείνουν τα FSM με τις εξής δυνατότητες: γεγονότα και συνθήκες, ιεραρχία (φωλιασμός καταστάσεων), συγχρονισμός ή ορθογωνιότητα καταστάσεων και καταστάσεις ιστορικού. Επιπλέον, τα διαγράμματα καταστάσεων έχουν ενσωματωμένες δυνατότητες για την περιγραφή της αλληλεπίδρασής τους με πολλαπλά αντικείμενα στο περιβάλλον τους.

Τα τρία βασικά στοιχεία των διαγραμμάτων καταστάσεων είναι οι καταστάσεις, τα γεγονότα και οι μεταβάσεις (Sunitha & Samuel, 2019):

- **Κατάσταση:** "μια κατάσταση ή κατάσταση κατά τη διάρκεια της ζωής ενός αντικειμένου κατά την οποία ικανοποιεί κάποια συνθήκη, εκτελεί κάποια δραστηριότητα ή περιμένει κάποιο γεγονός",
- **Γεγονός:** "ο προσδιορισμός ενός αξιοσημείωτου συμβάντος που έχει μια θέση στο χρόνο και στο χώρο",
- **Μετάβαση:** η μετακίνηση από μια κατάσταση σε μια άλλη ως απόκριση σε ένα γεγονός.

2.7 Μέθοδοι της UML

Η UML, όπως παρουσιάστηκε σε προηγούμενες ενότητες, δεν είναι μια μεθοδολογία αλλά μια ανοικτή, επεκτάσιμη, τυποποιημένη γλώσσα οπτικής μοντελοποίησης, η οποία έχει εγκριθεί από την Ομάδα Διαχείρισης Αντικειμένων (OMG). Σε αυτή την ενότητα θα περιγράψω μια μεθοδολογία που συμπληρώνει καλύτερα την UML: την ενοποιημένη διαδικασία (Unified Process - UP) (Tiwari et al., 2021).

Μια διαδικασία μηχανικής λογισμικού (Software Engineering Process - SEP), γνωστή και ως διαδικασία ανάπτυξης λογισμικού, ορίζει το "ποιος", το "τι", το "πότε" και το "πώς" της ανάπτυξης λογισμικού. Ενώ η UML είναι το μέρος της οπτικής γλώσσας της UML, η UP και η SEP είναι η διαδικασία κατά την οποία μετατρέπουμε τις απαιτήσεις των χρηστών σε λογισμικό. Η UML είναι τυποποιημένη από την OMG, ενώ η UP δεν είναι.

Για να μοντελοποιήσει το "ποιος" της SEP, η UP εισάγει την έννοια του εργαζόμενου. Αυτή περιγράφει έναν ρόλο που διαδραματίζει ένα άτομο ή μια ομάδα στο πλαίσιο του έργου. Κάθε εργαζόμενος μπορεί να υλοποιηθεί από πολλά άτομα ή ομάδες, και κάθε άτομο ή ομάδα μπορεί να εκτελεί ως πολλοί διαφορετικοί εργαζόμενοι (Tiwari et al., 2021).

Για να μοντελοποιήσει το "ποιος" της SEP, η UP εισάγει την έννοια του εργάτη. Αυτή περιγράφει έναν ρόλο που διαδραματίζει ένα άτομο ή μια ομάδα στο πλαίσιο του έργου. Κάθε εργαζόμενος μπορεί να υλοποιηθεί από πολλά άτομα ή ομάδες, και κάθε άτομο ή ομάδα μπορεί να εκτελεί ως πολλοί διαφορετικοί εργαζόμενοι.

Η UP μοντελοποιεί το "τι" ως δραστηριότητες και τεχνουργήματα. Οι δραστηριότητες είναι εργασίες που θα εκτελούνται από άτομα ή ομάδες στο έργο. Αυτά τα άτομα ή οι ομάδες θα υιοθετούν πάντα συγκεκριμένους ρόλους όταν εκτελούν ορισμένες δραστηριότητες και έτσι, για κάθε δραστηριότητα, η UP μπορεί να μας πει τους εργαζόμενους (ρόλους) που συμμετέχουν σε αυτή τη δραστηριότητα. Οι δραστηριότητες μπορούν να αναλύονται σε πιο λεπτομερή επίπεδα λεπτομέρειας, ανάλογα με τις ανάγκες. Τα τεχνουργήματα είναι πράγματα που αποτελούν εισροές και εκροές στο έργο - μπορεί να είναι πηγαίος κώδικας, εκτελέσιμα προγράμματα, πρότυπα, τεκμηρίωση κ.ο.κ. (Tiwari et al., 2021).

Η UP μοντελοποιεί το "όταν" ως ροές εργασίας. Πρόκειται για ακολουθίες συναφών δραστηριοτήτων που εκτελούνται από εργαζόμενους. Οι ροές εργασίας μπορούν να αναλύονται σε μία ή περισσότερες λεπτομέρειες ροής εργασίας που περιγράφουν τις δραστηριότητες, τους ρόλους και τα αντικείμενα που εμπλέκονται στη ροή εργασίας.

Η UP είναι μια επαναληπτική και σταδιακή διαδικασία

Για να γίνει κατανοητή η UP, πρέπει να γίνουν κατανοητές οι επαναλήψεις. Η ιδέα είναι πολύ απλή, καθώς η εμπειρία λέει, ότι τα μικρά προβλήματα επιλύονται ευκολότερα από τα μεγάλα προβλήματα. Επομένως, τα μεγάλα έργα ανάπτυξης λογισμικού σπάνε σε έναν αριθμό μικρότερων "μίνι έργων", τα οποία είναι πιο εύκολα στη διαχείρισή τους και στην επιτυχή ολοκλήρωσή τους. Καθένα από αυτά τα "μίνι έργα" είναι η επανάληψη. Κάθε επανάληψη περιέχει όλα τα στοιχεία ενός κανονικού έργου ανάπτυξης λογισμικού, όπως παρακάτω (Sarkar, Bhalla & Singal, 2017):

- **Σχεδιασμός**
- **Ανάλυση**
- **Κατασκευή**
- **Ολοκλήρωση και δοκιμή**
- **Εσωτερική ή εξωτερική έκδοση**

Κάθε επανάληψη δημιουργεί ένα βασικό επίπεδο που περιλαμβάνει μια μερικώς ολοκληρωμένη έκδοση του τελικού συστήματος και κάθε σχετική τεκμηρίωση του έργου. Οι βασικές γραμμές αναπτύσσονται η μία πάνω στην άλλη σε διαδοχικές επαναλήψεις μέχρι να επιτευχθεί το τελικό σύστημα. Η διαφορά μεταξύ δύο διαδοχικών γραμμών βάσης είναι γνωστή ως προσαύξηση. Αυτός είναι ο λόγος για τον οποίο ο UP είναι γνωστός ως επαναληπτικός και αυξητικός κύκλος ζωής (Sarkar, Bhalla & Singal, 2017).

3. Δοκιμές λογισμικού

Η δοκιμή λογισμικού είναι μια εμπειρική έρευνα που διεξάγεται για να παρέχει στους ενδιαφερόμενους πληροφορίες σχετικά με την ποιότητα του προϊόντος ή της υπηρεσίας που δοκιμάζεται (Black et al., 2000).

Όσον αφορά έναν γενικό ορισμό της δοκιμής και έναν λεπτομερή ορισμό της δοκιμής λογισμικού ειδικότερα, οι Utting και Legeard (2007) παραθέτουν το IEEE Software Engineering Body of Knowledge ως εξής (πρώτα ο γενικός ορισμός της δοκιμής και στη συνέχεια ο λεπτομερής ορισμός της δοκιμής λογισμικού):

Η δοκιμή είναι μια δραστηριότητα που εκτελείται για την αξιολόγηση της ποιότητας του προϊόντος και για τη βελτίωσή της, με τον εντοπισμό ελαττωμάτων και προβλημάτων.

Η δοκιμή λογισμικού συνίσταται στη δυναμική επαλήθευση της συμπεριφοράς ενός προγράμματος σε ένα πεπερασμένο σύνολο περιπτώσεων δοκιμής, κατάλληλα επιλεγμένων από το συνήθως άπειρο πεδίο εκτελέσεων, σε σχέση με την αναμενόμενη συμπεριφορά.

Σύμφωνα με την τυποποιημένη ορολογία του IEEE για τη μηχανική λογισμικού, η αποτυχία είναι μια ανεπιθύμητη συμπεριφορά. Οι αποτυχίες παρατηρούνται συνήθως κατά τη διάρκεια της εκτέλεσης του συστήματος που δοκιμάζεται. Το σφάλμα είναι η αιτία της αποτυχίας. Είναι ένα σφάλμα στο λογισμικό, που συνήθως προκαλείται από ανθρώπινο λάθος στη διαδικασία προδιαγραφών, σχεδίασης ή κωδικοποίησης. Η εκτέλεση των σφαλμάτων είναι αυτή που προκαλεί αποτυχίες. Μόλις παρατηρηθεί μια αποτυχία, μπορεί να διερευνηθεί για να βρεθεί το σφάλμα που την προκάλεσε και έτσι το σφάλμα αυτό διορθώνεται (Utting & Legeard, 2007).

3.1 Μέθοδοι δοκιμής

Οι μέθοδοι δοκιμής λογισμικού διακρίνονται παραδοσιακά σε δοκιμές "μαύρου κουτιού" (black box testing) και δοκιμές "λευκού κουτιού" (white box testing). Αυτές οι δύο μέθοδοι χρησιμοποιούνται για να περιγράψουν την άποψη που υιοθετεί ένας μηχανικός δοκιμών όταν σχεδιάζει περιπτώσεις δοκιμών (Aniche, 2022).

Δοκιμή μαύρου κουτιού

Η δοκιμή μαύρου κουτιού αντιμετωπίζει το λογισμικό ως μαύρο κουτί χωρίς καμία γνώση της εσωτερικής υλοποίησης. Περιλαμβάνει τη δοκιμή βάσει μοντέλου, τον πίνακα ιχνηλασιμότητας και τη δοκιμή βάσει προδιαγραφών.

Δοκιμές βάσει προδιαγραφών

Η δοκιμή βάσει προδιαγραφών αποσκοπεί στον έλεγχο της λειτουργικότητας σύμφωνα με τις απαιτήσεις. Ως εκ τούτου, ο ελεγκτής εισάγει δεδομένα και βλέπει μόνο την έξοδο από το αντικείμενο δοκιμής. Αυτό το επίπεδο δοκιμών απαιτεί συνήθως την παροχή εμπεριστατωμένων περιπτώσεων δοκιμών στον ελεγκτή, ο οποίος στη συνέχεια μπορεί απλώς να επαληθεύσει ότι για μια δεδομένη είσοδο, η τιμή εξόδου (ή η συμπεριφορά), είναι η ίδια με την αναμενόμενη τιμή που καθορίζεται στην περίπτωση δοκιμής (Aniche, 2022).

Δοκιμές λευκού κουτιού

Η δοκιμή λευκού κουτιού, σε αντίθεση με τη δοκιμή μαύρου κουτιού, είναι όταν ο ελεγκτής έχει πρόσβαση στις εσωτερικές δομές δεδομένων και στους αλγορίθμους (και στον κώδικα που τις υλοποιεί).

Τύποι δοκιμών λευκού κουτιού

Υπάρχουν οι ακόλουθοι τύποι δοκιμών λευκού κουτιού:

- Κάλυψη κώδικα: Δημιουργεί δοκιμές για την ικανοποίηση ορισμένων κριτηρίων κάλυψης κώδικα

- Μέθοδοι δοκιμής μετάλλαξης
- Μέθοδοι έγχυσης σφαλμάτων
- Στατικός έλεγχος: Η δοκιμή λευκού κουτιού περιλαμβάνει όλες τις στατικές δοκιμές

3.2 Δοκιμές βασισμένες σε μοντέλα

Υπάρχει ένα αυξανόμενο ενδιαφέρον για την ανάπτυξη με βάση το μοντέλο για αντικειμενοστραφή συστήματα, με τη χρήση της ενοποιημένης γλώσσας μοντελοποίησης (UML). Εκτός του ότι αποτελούν βασικό πόρο για τη σχεδίαση αντικειμενοστρεφούς λογισμικού, τα μοντέλα είναι πολύ χρήσιμα στη δοκιμή αντικειμενοστρεφούς λογισμικού.

Σύμφωνα με τους Utting και Legeard (2010), υπάρχουν τέσσερις κύριες προσεγγίσεις γνωστές ως δοκιμές βασισμένες σε μοντέλα:

1. Δημιουργία δεδομένων εισόδου δοκιμής από ένα μοντέλο τομέα
2. Δημιουργία περιπτώσεων δοκιμής από ένα μοντέλο περιβάλλοντος
3. Παραγωγή περιπτώσεων δοκιμής με μαντεία από ένα μοντέλο συμπεριφοράς
4. Παραγωγή σεναρίων δοκιμών από αφηρημένη δοκιμή

Στην πρώτη έννοια, όταν η δοκιμή με βάση το μοντέλο χρησιμοποιείται για την παραγωγή δεδομένων εισόδου δοκιμής, το μοντέλο είναι οι πληροφορίες σχετικά με τους τομείς των τιμών εισόδου και η παραγωγή δοκιμής περιλαμβάνει την έξυπνη επιλογή και τον συνδυασμό ενός υποσυνόλου αυτών των τιμών για την παραγωγή τιμών εισόδου δοκιμής.

Η δεύτερη έννοια του model-based testing χρησιμοποιεί ένα διαφορετικό είδος μοντέλου, το οποίο περιγράφει το αναμενόμενο περιβάλλον του υπό δοκιμή συστήματος (System Under Test - SUT). Από αυτά τα μοντέλα περιβάλλοντος είναι δυνατόν να παραχθούν ακολουθίες κλήσεων στο SUT. Ωστόσο, όπως και στην προηγούμενη προσέγγιση, οι παραγόμενες ακολουθίες δεν καθορίζουν τις αναμενόμενες εξόδους του SUT. Δεν είναι δυνατή η πρόβλεψη των τιμών εξόδου επειδή το μοντέλο περιβάλλοντος δεν μοντελοποιεί τη συμπεριφορά του SUT. Έτσι, είναι δύσκολο να προσδιοριστεί με ακρίβεια αν μια δοκιμή έχει περάσει ή αποτύχει - η ετυμηγορία συντριβής/μη συντριβής μπορεί να είναι το μόνο δυνατό (Ammann & Offutt, 2016).

Η τρίτη έννοια της δοκιμής με βάση το μοντέλο είναι η δημιουργία εκτελέσιμων περιπτώσεων δοκιμής που περιλαμβάνουν πληροφορίες μαντείου, όπως οι αναμενόμενες τιμές εξόδου του SUT, ή κάποιος αυτοματοποιημένος έλεγχος των πραγματικών τιμών εξόδου για να διαπιστωθεί αν είναι σωστές. Αυτό είναι προφανώς πιο δύσκολο έργο από την απλή παραγωγή δεδομένων εισόδου δοκιμής ή ακολουθιών δοκιμής που καλούν το SUT αλλά δεν ελέγχουν τα αποτελέσματα. Για τη δημιουργία δοκιμών με μαντεία, η γεννήτρια δοκιμών πρέπει να γνωρίζει αρκετά για την αναμενόμενη συμπεριφορά του SUT ώστε να είναι σε θέση να προβλέψει ή να ελέγξει τις τιμές εξόδου του SUT. Με άλλα λόγια, με αυτόν τον ορισμό των δοκιμών με βάση το μοντέλο, το μοντέλο πρέπει να περιγράφει την αναμενόμενη συμπεριφορά του SUT, όπως η σχέση μεταξύ των εισόδων και των εξόδων του. Το πλεονέκτημα όμως αυτής της προσέγγισης είναι ότι είναι η μόνη από τις τέσσερις που αντιμετωπίζει ολόκληρο το πρόβλημα του σχεδιασμού δοκιμών, από την επιλογή των τιμών εισόδου και τη δημιουργία ακολουθιών κλήσεων λειτουργίας μέχρι τη δημιουργία εκτελέσιμων περιπτώσεων δοκιμής που περιλαμβάνουν πληροφορίες ετυμηγορίας. Αυτό το είδος δοκιμής με βάση το μοντέλο είναι πιο περίπλοκο και πολύπλοκο από τις άλλες έννοιες, αλλά έχει μεγαλύτερη πιθανή απόδοση. Μπορεί να αυτοματοποιήσει την πλήρη διαδικασία σχεδιασμού δοκιμών, δεδομένου ενός κατάλληλου μοντέλου, και παράγει πλήρεις ακολουθίες δοκιμών που μπορούν να μετατραπούν σε εκτελέσιμα σενάρια δοκιμών. Με αυτή την άποψη της δοκιμής βάσει μοντέλου, η δοκιμή βάσει μοντέλου μπορεί να οριστεί ως η αυτοματοποίηση του σχεδιασμού δοκιμών μαύρου κουτιού. Η διαφορά από τις συνήθεις δοκιμές μαύρου κουτιού είναι ότι αντί για χειροκίνητη συγγραφή δοκιμών με βάση την τεκμηρίωση των απαιτήσεων, μπορεί να δημιουργηθεί ένα μοντέλο της αναμενόμενης συμπεριφοράς του SUT, το οποίο αποτυπώνει ορισμένες από τις απαιτήσεις. Στη συνέχεια, τα εργαλεία δοκιμών βασισμένων στο μοντέλο χρησιμοποιούνται για την αυτόματη δημιουργία δοκιμών από το εν λόγω μοντέλο (Ammann & Offutt, 2016).

Η τέταρτη έννοια της δοκιμής βάσει μοντέλου είναι αρκετά διαφορετική: υποθέτει ότι μας δίνεται μια πολύ αφηρημένη περιγραφή μιας περίπτωσης δοκιμής, όπως ένα διάγραμμα ακολουθίας UML ή μια ακολουθία κλήσεων μιας διαδικασίας υψηλού επιπέδου, και επικεντρώνεται στη μετατροπή αυτής της αφηρημένης περίπτωσης δοκιμής σε ένα εκτελέσιμο σενάριο δοκιμής χαμηλού επιπέδου. Με αυτή την προσέγγιση, το μοντέλο είναι οι πληροφορίες σχετικά με τη δομή και το API (Application Programming Interface) του SUT και οι λεπτομέρειες του τρόπου

μετασχηματισμού μιας κλήσης υψηλού επιπέδου σε εκτελέσιμο σενάριο δοκιμής (Ammann & Offutt, 2016).

3.3 Διαδικασία ανάλυσης μετάλλαξης

Η ανάλυση μετάλλαξης εισάγει σφάλματα στο λογισμικό δημιουργώντας πολλές εκδόσεις του λογισμικού, καθεμία από τις οποίες περιέχει ένα σφάλμα. Στη συνέχεια χρησιμοποιούνται περιπτώσεις δοκιμής για την εκτέλεση αυτών των ελαττωματικών προγραμμάτων με στόχο τη διάκριση των ελαττωματικών προγραμμάτων από το αρχικό πρόγραμμα. Αυτές οι ελαττωματικές εκδόσεις των αρχικών προγραμμάτων ονομάζονται μεταλλάξεις του αρχικού, και μια μετάλλαξη εξοντώνεται με τη διάκριση της εξόδου της μετάλλαξης από εκείνη του αρχικού προγράμματος (Black et al., 2000).

Η διαδικασία ελέγχου μετάλλαξης αρχίζει με την επιλογή των τελεστών μετάλλαξης. Οι τελεστές μετάλλαξης σχετίζονται με το σύνολο των κλάσεων σφαλμάτων, οι οποίες αναλύονται από τους Black et al. (2000). Το σύνολο των κλάσεων σφαλμάτων που αναλύονται περιλαμβάνει:

- **Variable Reference Fault (VRF):** αντικατάσταση μιας Boolean μεταβλητής x από μια άλλη μεταβλητή y, $x \neq y$.
- **Variable Negation Fault (VNF):** αντικατάσταση μιας Boolean μεταβλητής x από την x.
- **Expression Negation Fault (ENF):** αντικατάσταση μιας έκφρασης Boolean p από την p.
- **Missing Condition Fault (MCF):** αποτυχία ελέγχου των προαπαιτούμενων συνθηκών.

Κάθε κλάση σφάλματος έχει έναν αντίστοιχο τελεστή μετάλλαξης. Η εφαρμογή ενός τελεστή μετάλλαξης προκαλεί ένα σφάλμα στην εν λόγω κλάση. Ο τελεστής μετάλλαξης που χρησιμοποιείται περισσότερο στις δύο δοκιμές είναι ο Relational Operator Replacement (RRO). Με αυτόν τον τελεστή μετάλλαξης, ένας σχεσιακός τελεστής ($<, \leq, >, \geq, =, \neq$) αντικαθίσταται από οποιονδήποτε άλλο σχεσιακό τελεστή, εκτός από τον αντίθετό του. Ένας άλλος τελεστής μετάλλαξης που επιλέχθηκε ήταν η αντικατάσταση λογικού τελεστή (Logical Operator Replacement - LRO), η οποία αντικαθιστά έναν λογικό τελεστή (&, |) με έναν άλλο λογικό τελεστή (Black et al., 2000).

Ο στόχος της δοκιμής λογισμικού είναι προφανώς η ανίχνευση σφαλμάτων στο SUT. Ως εκ τούτου, ο κύριος λόγος για την επιλογή των τελεστών μετάλλαξης βασίζεται στο κριτήριο επιλογής που επιλέχθηκε για αυτές τις δοκιμές, το οποίο είναι κριτήριο βασισμένο σε σφάλματα (Utting & Legeard, 2007).

Ο χρήστης των δοκιμών μετάλλαξης προσθέτει περιπτώσεις δοκιμών (που παράγονται χειροκίνητα ή αυτόματα) στο σύστημα μετάλλαξης και ελέγχει την έξοδο του προγράμματος σε κάθε περίπτωση δοκιμής για να διαπιστώσει αν είναι σωστή. Εάν η έξοδος είναι λανθασμένη, έχει βρεθεί σφάλμα και το πρόγραμμα πρέπει να τροποποιηθεί και η διαδικασία να επανεκκινήσει. Εάν η έξοδος είναι σωστή, η συγκεκριμένη περίπτωση δοκιμής εκτελείται σε κάθε ζωντανή μετάλλαξη. Εάν η έξοδος ενός μεταλλαγμένου διαφέρει από εκείνη του αρχικού προγράμματος στην ίδια περίπτωση δοκιμής, το μεταλλαγμένο θεωρείται εσφαλμένο και σκοτώνεται (Utting & Legeard, 2007).

Η ανάλυση μετάλλαξης παρέχει ένα κριτήριο δοκιμής και όχι μια διαδικασία δοκιμής. Ένα κριτήριο δοκιμής είναι ένας κανόνας ή μια συλλογή κανόνων που επιβάλλει απαιτήσεις σε ένα σύνολο περιπτώσεων δοκιμής. Οι μηχανικοί δοκιμών μετρούν το βαθμό στον οποίο ικανοποιείται ένα κριτήριο με όρους κάλυψης, ένα σύνολο περιπτώσεων δοκιμής επιτυγχάνει 100% κάλυψη εάν ικανοποιεί πλήρως το κριτήριο. Η κάλυψη μετράται με βάση τις απαιτήσεις που επιβάλλονται. Οι απαιτήσεις δοκιμής είναι συγκεκριμένα πράγματα που πρέπει να ικανοποιούνται ή να καλύπτονται.

Όταν ένα πρόγραμμα υποβάλλεται σε ένα σύστημα μετάλλαξης, το σύστημα δημιουργεί πρώτα πολλές μεταλλαγμένες εκδόσεις του προγράμματος. Ένας τελεστής μετάλλαξης είναι ο κανόνας που εφαρμόζεται σε ένα πρόγραμμα για τη δημιουργία μεταλλάξεων. Οι τυπικοί τελεστές μετάλλαξης, για παράδειγμα, αντικαθιστούν κάθε τελεστή με άλλον συντακτικά νόμιμο τελεστή, ή τροποποιούν εκφράσεις αντικαθιστώντας τελεστές και εισάγοντας νέους τελεστές, ή διαγράφουν ολόκληρες εντολές. Τα συμπαγή πλαίσια αντιπροσωπεύουν βήματα που είναι αυτοματοποιημένα και τα διακεκομμένα πλαίσια αντιπροσωπεύουν βήματα που γίνονται χειροκίνητα (Black et al., 2000).

Στη συνέχεια, παρέχονται στο σύστημα περιπτώσεις δοκιμών που χρησιμεύουν ως είσοδοι στο πρόγραμμα. Κάθε περίπτωση δοκιμής εκτελείται στο αρχικό πρόγραμμα και ο ελεγκτής

επαληθεύει ότι η έξοδος είναι σωστή. Εάν είναι σωστή, οι περιπτώσεις δοκιμής εκτελούνται σε κάθε μεταλλαγμένο πρόγραμμα. Εάν η έξοδος ενός μεταλλαγμένου προγράμματος διαφέρει από την αρχική (σωστή) έξοδο, το μεταλλαγμένο πρόγραμμα χαρακτηρίζεται ως νεκρό. Τα νεκρά μεταλλαγμένα δεν εκτελούνται έναντι των επόμενων περιπτώσεων δοκιμής.

Αφού εκτελεστούν όλες οι περιπτώσεις δοκιμής, υπολογίζεται ένα σκορ μετάλλαξης. Το σκορ μετάλλαξης είναι ο λόγος των νεκρών μεταλλαγμένων προς το συνολικό αριθμό των μη ισοδύναμων μεταλλαγμένων. Έτσι, στόχος του ελεγκτή είναι να αυξήσει το σκορ μετάλλαξης στο 1,00, υποδεικνύοντας ότι έχουν εντοπιστεί όλες οι μεταλλάξεις. Ένα σύνολο δοκιμών που σκοτώνει όλα τα μεταλλαγμένα λέγεται ότι είναι επαρκές σε σχέση με τα μεταλλαγμένα. Ένα σύνολο δεδομένων δοκιμής καλείται κατάλληλο για μετάλλαξη εάν το σκορ μετάλλαξης είναι 100% (Black et al., 2000).

4. Το πλαίσιο της κινητής τηλεφωνίας

4.1 Μοντελοποίηση εφαρμογών για κινητά τηλέφωνα: κατάσταση προόδου

Κατά τη διάρκεια της μελέτης μας, βρήκαμε αρκετές ερευνητικές εργασίες σχετικά με την ανάπτυξη και μοντελοποίηση εφαρμογών για κινητά. Το κύριο μέρος των ερευνών τα προηγούμενα χρόνια αφορούσε ειδική μοντελοποίηση για την υποστήριξη της αυτόματης παραγωγής κώδικα για εφαρμογές. Ταυτόχρονα, πραγματοποιήθηκαν διαφορετικές προσεγγίσεις προκειμένου να προταθούν ειδικά διαγράμματα UML για Android και εφαρμογές πολλαπλών πλατφορμών. Σε αυτή την ενότητα παρουσιάζουμε σχετικές εργασίες σχετικά με προσεγγίσεις φορητής μοντελοποίησης εστιάζοντας την προσοχή μας σε τρεις κύριους τομείς (Zander et al., 2017):

1. Σχεδιασμός βάσει μοντέλου για υποστήριξη εφαρμογών ανάπτυξης.
2. Γενικές προσεγγίσεις επέκτασης UML.
3. Επεκτάσεις UML για εφαρμογές για φορητές συσκευές.
4. Εμπειρικά πειράματα για την αξιολόγηση της χρησιμότητας των συμβολισμών.

4.1.1 Σχεδιασμός βάσει μοντέλου για την υποστήριξη της ανάπτυξης εφαρμογών

Διαφορετικές προσεγγίσεις αφορούν τον σχεδιασμό με γνώμονα το μοντέλο για την υποστήριξη της ανάπτυξης εφαρμογών. Επικεντρώνονται στη δημιουργία κώδικα για την υποστήριξη της ανάπτυξης πολλαπλών πλατφορμών. Οι Parada, Marques και de Brisolara (2015) πρότειναν μια προσέγγιση βάσει μοντέλου για την ανάπτυξη εφαρμογών Android. Ο κύριος στόχος αυτής της προσέγγισης ήταν να μειωθεί το χάσμα μεταξύ του τομέα του προβλήματος και της υλοποίησης λογισμικού μέσω της χρήσης τεχνολογιών που υποστηρίζουν συστηματικούς μετασχηματισμούς μοντέλων. Η προσέγγιση περιελάμβανε μοντελοποίηση βάσει UML και αυτόματη δημιουργία κώδικα για τη διευκόλυνση και την επιτάχυνση της ανάπτυξης εφαρμογών Android. Η μοντελοποίησή τους βασίστηκε σε UML, χρησιμοποιώντας διάγραμμα κλάσης για να περιγράψει τη δομική προβολή της εφαρμογής και διαγράμματα ακολουθίας για να αναπαραστήσει την άποψη συμπεριφοράς. Αυτή η προσέγγιση ήταν χρήσιμη για την καλύτερη κατανόηση του είδους των στοιχείων που λαμβάνονται υπόψη στις προσεγγίσεις μας, αλλά δεν υπάρχει μια ποιοτική αξιολόγηση της προτεινόμενης επέκτασης και του κώδικα που δημιουργείται.

Μια άλλη ενδιαφέρουσα προσέγγιση προτάθηκε από τους Sabraoiu et al.(2012). Πρότειναν μια προσέγγιση βασισμένη στο MDA, για τη δημιουργία GUI για κινητές εφαρμογές σε smartphone. Η υιοθετηθείσα προσέγγιση αποτελείται από τρία βασικά βήματα:

i) ανάλυση και μοντελοποίηση του γραφικού περιβάλλοντος στο UML
ii) μετατροπή των ληφθέντων διαγραμμάτων σε ένα απλοποιημένο σχήμα XMI χρησιμοποιώντας JDOM API
iii) δημιουργία του GUI με βάση το MDA.

Η μέθοδός τους έχει τα πλεονεκτήματα να δημιουργεί αυτόματα GUI για πολλές πλατφόρμες και δίνει έναν γραφικό τρόπο σχεδίασης σε UML. Αυτή η προσέγγιση δεν προτείνει μια πραγματική επέκταση UML για εφαρμογές για κινητές συσκευές, αλλά εξετάζει το πλαίσιο για φορητές συσκευές καθορίζοντας ένα μετα-μοντέλο για αυτόματη δημιουργία κώδικα.

Οι Usman et al. (2014) πρότειναν μια προσέγγιση βάσει μοντέλου για τη δημιουργία κώδικα εφαρμογών για κινητά για πολλαπλές πλατφόρμες. Πρότειναν μια μεθοδολογία μοντελοποίησης χρησιμοποιώντας πραγματική περίπτωση χρήσης για τη συλλογή απαιτήσεων, διαγράμματα κλάσεων για δομική μοντελοποίηση και μηχανή κατάστασης για μοντελοποίηση συμπεριφοράς. Για να δημιουργήσουν αυτόματα εφαρμογές για κινητά, αναπτύσσουν ένα εργαλείο που ονομάζεται Mobile Application Generator (MAG) που λαμβάνει τα αναπτυγμένα μοντέλα UML ως είσοδο και δημιουργεί εφαρμογή για τις καθορισμένες πλατφόρμες φορητών στόχων. Πρότειναν επίσης ένα πρόγραμμα UML για τη μοντελοποίηση ειδικών εννοιών τομέα. Η προσέγγιση που παρουσιάστηκε αφορά εφαρμογές πολλαπλών πλατφορμών και επικεντρώθηκε στη φάση δημιουργίας κώδικα. Παρουσιάζει επίσης μια αξιολόγηση κώδικα,

αλλά χάνει μια αξιολόγηση σχετικά με την αποτελεσματική χρησιμότητα σε εφαρμογές προγραμματισμού.

Οι Botturi et al. (2013), επίσης, πρότειναν μια προσέγγιση βάσει μοντέλου που βασίζεται στη δημιουργία κώδικα, έτσι ώστε να μην απαιτείται επιπλέον βιβλιοθήκη ή διαδικασία στο smartphone για την υποστήριξη διαφορετικών πλατφορμών. Χρησιμοποίησαν το προφίλ UML2 για να αναπαραστήσουν τα στοιχεία της εφαρμογής ανεξάρτητα από την πλατφόρμα-στόχο. Αυτή η προσέγγιση, όπως και οι άλλες, προσανατολίζεται στη δημιουργία κώδικα και λαμβάνει υπόψη στοιχεία διεπαφής χρήστη. Του λείπουν τα δομικά στοιχεία του λειτουργικού συστήματος.

Μια άλλη ενδιαφέρουσα πρόταση προκύπτει από τους Freitas και Maia (2016). Ισχυρίστηκαν ότι το Model-driven Engineering (MDE) έχει προκύψει ως μια συγκεκριμένη εναλλακτική για την αυτόματη δημιουργία εφαρμογών Android και πρότειναν το JustModeling, μια προσέγγιση MDE που δημιουργήθηκε από το JBModel, ένα εργαλείο γραφικής μοντελοποίησης με το οποίο ο χρήστης μοντελοποιεί τις επιχειρηματικές τάξεις εφαρμογής χρησιμοποιώντας το διάγραμμα κλάσης UML και αυτό παρέχει ένα σύνολο μετασχηματισμών μοντέλων για τη δημιουργία κώδικα για το πλαίσιο JustBusiness, το οποίο δημιουργεί αυτόματα όλους τους απαραίτητους πόρους της εφαρμογής για κινητά. Αυτό επιτρέπει στους προγραμματιστές να εργαστούν σε υψηλότερο επίπεδο αφαίρεσης, εστιάζοντας στον σχεδιασμό της εφαρμογής και όχι σε ζητήματα υλοποίησης. Η προσέγγιση προσανατολίζεται κυρίως σε εφαρμογές Android Business και χάνει δομικό στοιχείο.

4.1.2 Προσεγγίσεις επεκτάσεων της UML

Όλες αυτές οι προσεγγίσεις αφορούν τη μοντελοποίηση και την αυτοματοποιημένη δημιουργία κώδικα για εφαρμογές Android ή πολλαπλών πλατφορμών. Δεδομένου ότι το ενδιαφέρον μας εστιάζεται κυρίως στη μοντελοποίηση εφαρμογών για τη βελτίωση της κατανόησης κώδικα κατά τις φάσεις ανάπτυξης και συντήρησης, αναλύσαμε προσεγγίσεις που πρότειναν επεκτάσεις UML.

Η πρώτη σημαντική προσέγγιση στην επέκταση UML παρουσιάστηκε από τον Conallen (2003). Πρότεινε μια σημαντική επέκταση της γλώσσας UML για να αντιμετωπίσει τη μοντελοποίηση εφαρμογών Ιστού. Το έργο του στοχεύει στην πρόταση μιας εφαρμόσιμης λύσης για την κυκλοφορία διαδικτυακών εφαρμογών. Η πρόταση παρέχει προνόμια στις αλληλεπιδράσεις πελάτη-διακομιστή και υποτιμά τη λογική και τη φυσική σχεδίαση τόσο των δομών πληροφοριών όσο και των δομών πλοήγησης. Καθορίζει τα στερεότυπα, τις τιμές με ετικέτα και τους περιορισμούς OCL για να μοντελοποιήσει ιστοσελίδες και υπερσυνδέσμους, φόρμες, πλαίσια και στοιχεία πελάτη-διακομιστή σε συγκεκριμένο επίπεδο. Ο Conallen (2003) προσαρμόζει επίσης όλες τις κλασσικές φάσεις της ανάπτυξης λογισμικού σε αρχιτεκτονικές ιστού και προσαρμόζει σχεδόν όλα τα διαγράμματα UML για να αποδώσει έννοιες που σχετίζονται με τον ιστό.

Από την Conallen βγήκε διαφορετική πρόταση. Οι Baumeister et al. (1999) πρότειναν μια UML επεκτάσεις για τη μοντελοποίηση εφαρμογών υπερμέσων που εκτελούνται στο Διαδίκτυο. Στην εργασία τους πρότειναν μια τέτοια επέκταση για τη μοντελοποίηση της πλοήγησης και των διεπαφών χρήστη των συστημάτων υπερμέσων. Παρόμοια με άλλες μεθόδους σχεδιασμού για συστήματα υπερμέσων, θεώρησαν τον σχεδιασμό των συστημάτων υπερμέσων ως αποτελούμενο από τρία μοντέλα: το εννοιολογικό, το μοντέλο πλοήγησης και το μοντέλο παρουσίασης.

Οι Koch et al. (2000) παρουσίασαν μια άλλη προσέγγιση του Προφίλ UML για εφαρμογές Ιστού. Ήταν μια επέκταση UML βασισμένη στον γενικό μηχανισμό επέκτασης που παρέχεται από το UML που καθορίζει συγκεκριμένα στερεότυπα για να μοντελοποιήσει πτυχές που σχετίζονται με την πλοήγηση και την παρουσίαση εφαρμογών Ιστού. Αυτό το έργο αποτελεί μέρος μιας μεθοδολογίας για την ανάλυση και το σχεδιασμό εφαρμογών Ιστού. Αυτή η μεθοδολογία εκτελεί ξεχωριστά βήματα για εννοιολογική, πλοήγηση και μοντελοποίηση παρουσίασης με παρόμοιο τρόπο όπως προτείνεται από άλλες μεθόδους για υπερμέσα ή σχεδιασμό Ιστού. Η καινοτομία αυτής της προσέγγισης συνίσταται στις τεχνικές μοντελοποίησης και στη σημειογραφία που χρησιμοποιούνται, οι οποίες βασίζονται εξ ολοκλήρου στην Ενοποιημένη Γλώσσα Μοντελοποίησης.

Μια ενδιαφέρουσα προσέγγιση προέρχεται από τον Nassar (2003) που πρότεινε μια επέκταση της UML που ονομάζεται VUML (View based Uni ed Modeling Language). Το VUML βασίστηκε στην έννοια του στοιχείου πολλαπλών προβολών του οποίου ο στόχος είναι να αποθηκεύει και να παρέχει πληροφορίες σύμφωνα με τις απόψεις των χρηστών. Αυτή η προσέγγιση επιτρέπει τη δυναμική αλλαγή των απόψεων και προσφέρει μηχανισμούς για την περιγραφή των εξαρτήσεων απόψεων.

Άλλες έρευνες αφορούν διαφορετικούς τομείς εφαρμογής. Μια φορά από αυτά υλοποιήθηκε στην Ανάπτυξη Ασφαλών Συστημάτων. Ο Jürjens (2002) πρότεινε μια επέκταση UML για υποστήριξη χρήσης UML για ασφαλή ανάπτυξη συστήματος (UMLsec). Με την πρότασή τους, περιέκοψαν τη γνώση σχετικά με τη συνετή μηχανική ασφάλειας και έτσι την έθεσαν διαθέσιμη σε προγραμματιστές που μπορεί να μην είναι εξειδικευμένοι στην ασφάλεια. Ο Jürjens (2002) δημιούργησε νέα στερεότυπα και ετικέτες και επέτρεψε σε προγραμματιστές με υπόβαθρο στην ασφάλεια να κάνουν χρήση των γνώσεων της μηχανικής ασφάλειας που περικλείονται σε ευρέως χρησιμοποιούμενες σημειώσεις σχεδίασης.

4.1.3 Επεκτάσεις της UML για κινητές εφαρμογές

Βρήκαμε στη βιβλιογραφία διαφορετικές προσεγγίσεις που αφορούν επεκτάσεις UML για περιβάλλον εφαρμογών για κινητές συσκευές.

Η πρόταση των Ko et al. (2012) φαινόταν να είναι η πιο ενδιαφέρουσα προσέγγιση για την επέκταση της UML για εφαρμογές Android. Χρησιμοποίησαν μηχανισμό επέκτασης UML και πρότειναν ένα μετα-μοντέλο για την ανάπτυξη μιας εφαρμογής στην πλατφόρμα Android χρησιμοποιώντας αυτόν τον μηχανισμό. Προσδιόρισαν τα κύρια χαρακτηριστικά εφαρμογών Android και παρείχαν ένα πιο συγκεκριμένο διάγραμμα κλάσης UML που περιέχει συγκεκριμένους ορισμούς τομέα. Ούτως ή άλλως η έρευνα χάνει τα πειράματα για την κατανόηση της αποτελεσματικής χρησιμότητας του προτεινόμενου μετα-μοντέλου.

Οι Min et al. (2011) πρότειναν ένα μετα-μοντέλο UML για εφαρμογή με βάση το Windows Phone. Πρότειναν ένα εκτεταμένο μετα-μοντέλο για τη μοντελοποίηση εφαρμογών με βάση το Windows Phone 7 χρησιμοποιώντας τον μηχανισμό επέκτασης UML. Για να γίνει αυτό, ανέλυσαν τα χαρακτηριστικά του Windows Phone 7s και τα ταξινόμησαν σε σχέση με στοιχεία λογισμικού και πόρους υλικού. Επέκτειναν το υλικό λογισμικού και τις θεμελιώδεις λειτουργίες χρησιμοποιώντας στερεότυπο μηχανισμό που παρέχεται από την UML.

Μια άλλη ενδιαφέρουσα πρόταση έρχεται από τους Perego και Pezzetti (2013). Ανέλυσαν το πλαίσιο ανάπτυξης εφαρμογών για κινητά και πρότειναν ένα αφηρημένο μετα-μοντέλο UML του οποίου οι περιπτώσεις αντιπροσωπεύουν μοντέλα υψηλού επιπέδου για εφαρμογές για κινητές συσκευές. Στη συνέχεια δημιούργησαν μια πιο συγκεκριμένη έκδοση του μετα-μοντέλου με στόχο να καθορίσουν ένα πιο λεπτομερές μοντέλο των εφαρμογών. Παρουσίασαν επίσης ένα εργαλείο για τη δημιουργία πηγαίου κώδικα Android και iOS ξεκινώντας από το μετα-μοντέλο που ορίζεται. Οι αξιολογήσεις που έγιναν αφορούσαν μόνο την ποιότητα του παραγόμενου κώδικα και όχι την αποτελεσματική χρησιμότητα της πρότασης.

4.1.4 Εμπειρικά πειράματα για την αξιολόγηση της χρησιμότητας των συμβολισμών

Κατά τη διάρκεια της έρευνας που έγινε, αναλύθηκαν διαφορετικές εμπειρικές μελέτες για μια σωστή αξιολόγηση της χρησιμότητας της προτεινόμενης επέκτασης Droid-UML.

Το πρώτο ελεγχόμενο πείραμα διεξήχθη από τους Dzidek et al. (2008). Διερεύνησαν το κόστος συντήρησης και τα οφέλη της χρήσης της τεκμηρίωσης UML κατά τη συντήρηση και την εξέλιξη ενός πραγματικού, μη τετριμμένου συστήματος, χρησιμοποιώντας επαγγελματίες προγραμματιστές ως υποκείμενα, δουλεύοντας με ένα υπερσύγχρονο εργαλείο UML για μεγάλο χρονικό διάστημα. .

Οι Kuzniarz et al. (2004) ανέλυσαν τη χρήση των στερεοτύπων για τη βελτίωση της κατανόησης των μοντέλων UML. Η εργασία επεξεργάζεται αυτόν τον ρόλο των στερεοτύπων από την οπτική γωνία της UML, διευκρινίζει τον ρόλο και περιγράφει ένα ελεγχόμενο πείραμα που στοχεύει στην αξιολόγηση του ρόλου - στο πλαίσιο της κατανόησης του μοντέλου. Τα αποτελέσματα του πειράματος υποστηρίζουν τον ισχυρισμό ότι τα στερεότυπα με γραφικά εικονίδια για την

αναπαράστασή τους παίζουν σημαντικό ρόλο στην κατανόηση των μοντέλων και δείχνουν το μέγεθος της βελτίωσης.

Οι Briand et al. (2005) πρότειναν ένα ελεγχόμενο πείραμα που διερευνά τον αντίκτυπο της χρήσης OCL σε τρεις δραστηριότητες μηχανικής λογισμικού χρησιμοποιώντας μοντέλα ανάλυσης UML: ανίχνευση ελαττωμάτων μοντέλου μέσω επιθεωρήσεων, κατανόηση της λογικής και της λειτουργικότητας του συστήματος και ανάλυση επιπτώσεων των αλλαγών. Προκειμένου να διερευνηθεί ο αντίκτυπος του OCL στην ανάπτυξη UML, σχεδίασαν, πραγματοποίησαν και αναπαρήγαγαν ένα ελεγχόμενο πείραμα. Αφορούσε φοιτητές τετάρτου έτους λογισμικού/μηχανικών υπολογιστών που έλαβαν ουσιαστική εκπαίδευση σε UML και OCL. Ερεύνησαν τον αντίκτυπο της χρήσης OCL σε τρεις σημαντικές δραστηριότητες μηχανικής λογισμικού:

1. Κατανόηση της λειτουργικότητας και της εσωτερικής λογικής των μοντελοποιημένων συστημάτων
2. Εκτέλεση ανάλυσης επιπτώσεων αλλαγής με βάση μοντέλα UML
3. Εντοπισμός ελαττωμάτων μέσω επιθεωρήσεων μοντέλων.

Οι Safdar et al. (2015) πραγματοποίησαν ένα ελεγχόμενο πείραμα για σύγκριση εργαλείων μοντελοποίησης MDSE. Μέτρησαν την παραγωγικότητα ως προς την προσπάθεια μοντελοποίησης που απαιτείται για τη σωστή ολοκλήρωση μιας εργασίας, τη δυνατότητα εκμάθησης, τον χρόνο και τον αριθμό των απαιτούμενων κλικ και το φορτίο μνήμης που απαιτείται για να ολοκληρώσει μια εργασία ο μηχανικός λογισμικού.

Μια άλλη προσέγγιση που προτείνεται από τους Bavota et al. (2015). Επισημοποίησαν μια εμπειρική μελέτη με στόχο τη σύγκριση της υποστήριξης που παρέχεται από τα διαγράμματα κλάσεων ER και UML κατά τη συντήρηση των μοντέλων δεδομένων. Το πείραμα που έγινε είχε στόχο να κατανοήσει την αποτελεσματικότητα των διαγραμμάτων κλάσης UML και διαγραμμάτων ER με σκοπό την κατανόηση που παρέχει καλύτερη υποστήριξη σε σχέση με την κατανόηση και την τροποποίηση μοντέλων δεδομένων.

5. Κινητές τεχνολογίες και ανάπτυξη εφαρμογών

Στο παρελθόν, οι εφαρμογές της τεχνητής νοημοσύνης έχουν χρησιμοποιηθεί σε μια σειρά από πεδία. Ο τρόπος χρησιμοποίησής της είναι η καινοτομία της τεχνολογίας σε διάφορους κλάδους. Η Virvou, (2022) αναλύει την αλληλεπίδραση ανθρώπου τεχνητής νοημοσύνης και τα θετικά της ως προς τη βοήθεια προς τον άνθρωπο. Η μελέτη αναλύει τα πλεονεκτήματα και τα μειονεκτήματα της σχέσης αυτής με κατεύθυνση προς το μέλλον όπου μπορεί να γίνει πιο αξιόπιστη και επεξεργάσιμη. Σε μια ακόμη μελέτη η Virvou (2023) παραθέτει τις βέλτιστες πρακτικές της βιβλιογραφία που αναφέρονται στην επίτευξη μιας επιτυχημένης αλληλεπίδρασης της τεχνητής νοημοσύνης στο UX, οι οποίες προσδιορίζονται και παρατίθενται βάσει καθιερωμένων μεθόδων ή τεχνικών που έχουν αποδειχθεί αποτελεσματικές και συζητάει πλεονεκτήματα και μειονεκτήματα.

Ένα από αυτά είναι τα παιχνίδια εκμάθησης για μαθητές, όπου διάφορες εφαρμογές μέσω αλγορίθμου χρησιμοποιούνται για τη μαθησιακή διαδικασία. Οι Chrysafiadi, et al. (2023) παρουσιάζουν ένα λογισμικό (k-NN) που διδάσκει μια γλώσσα προγραμματισμού μέσω εξατομικευμένης μάθησης σε κάθε μαθητή ξεχωριστά ανάλογα με τις αλληλεπιδράσεις του. Το λογισμικό καταγράφει το επίπεδο γνώσεων του μαθητή και τις παρανοήσεις του. Αυτή η μέθοδος γίνεται ολοένα και πιο δημοφιλής στις μέρες μας καθώς τέτοια παιχνίδια προσφέρουν ένα συναρπαστικό περιβάλλον που τα καθιστούν ένα ισχυρό εργαλείο για την επίτευξη μαθησιακών αποτελεσμάτων υψηλής ποιότητας. Τα αποτελέσματα δείχνουν ότι ο συνδυασμός της μηχανικής μάθησης με τη ασαφή λογική παρέχει πιο εξατομικευμένη μαθησιακή εμπειρία, προωθεί την ενεργό συμμετοχή των μαθητών στη μαθησιακή διαδικασία και έχει ως αποτέλεσμα τη μείωση της παραίτησής τους από αυτή.

Οι Chrysafiadi, Kamitsios & Virvou (2023) αναφέρουν ότι τα εκπαιδευτικά παιχνίδια μέσω αλγορίθμων δεν εξυπηρετούν ενιαία τους μαθητές που έχουν διαφορετικά επίπεδα μαθησιακών αναγκών και δεξιοτήτων και άρα κάποιοι από αυτούς μπορεί να αντιμετωπίσουν δυσκολίες ή μεγάλη ευκολία σε αυτό. Χρησιμοποιούν γλώσσα προγραμματισμού HTML που προσαρμόζεται στις ανάγκες του κάθε μαθητή – παίχτη, ξεχωριστά. Ο βαθμός δυσκολίας αναπροσαρμόζεται σε κάθε επίπεδο (Chrysafiadi, Papadimitriou, & Virvou, 2022; Chrysafiadi, Kamitsios & Virvou, 2023). Οι Chrysafiadi et al. (2022) αναφέρουν ότι τα κριτήρια που χρησιμοποιούνται για την εξατομίκευση των τεστ και την αναπροσαρμογή τους όσο προχωράει η διαδικασία, είναι το επίπεδο γνώσεων του μαθητή, οι προηγούμενες γνώσεις του στον προγραμματισμό, τα λάθη που κάνει και το επίπεδο δυσκολίας του τεστ.

Αυτές οι διαδικασίες έχουν στόχο όπως αναφέρουν οι Virvou et al. (2020) να μετρηθεί η δέσμευση των μαθητών, η ποσοτικοποίηση της μαθησιακής εμπειρίας και η διευκόλυνση της αυτορρύθμισης, να γίνει η πρόβλεψη της απόδοσης των μαθητών, να ενσωματωθούν σε Εργαλεία για Δόμηση Μαθησιακού Υλικού και Εκπαιδευτικά Μαθήματα και να χρησιμοποιηθούν ως Εργαλεία για την υποστήριξη Μαθητών και Εκπαιδευτών στη Σύγχρονη και Ασύγχρονη ηλεκτρονική μάθηση.

Τα αποτελέσματα αυτών των αξιολογήσεων έδειξαν ότι η προσαρμογή αυξάνει τα κίνητρα του χρήστη και καταφέρνει να διατηρεί αμείωτο το ενδιαφέρον των παικτών για το παιχνίδι. Αντίστοιχα και τα αποτελέσματα των Chrysafiadi, Papadimitriou, & Virvou, (2022) υποδεικνύουν υψηλό ποσοστό αποδοχής του παιχνιδιού από μαθητές και δασκάλους και υπογραμμίζουν την αποτελεσματικότητά του στα εκπαιδευτικά αποτελέσματα.

Εκτός από τη μαθησιακή διαδικασία, αυτές οι εφαρμογές χρησιμοποιούνται σε κλάδους όπως η ιατρική. Οι Panagoulias et al. (2022) ανέπτυξαν μέσω AI μιας ιατρικής εφαρμογής για τη βελτίωση των αναγκών και των απαιτήσεων των ασθενών με απώτερο σκοπό το μεγαλύτερο όφελος για την ποιότητα της παροχής φροντίδας. Ακόμη ένας κλάδος που εφαρμόζονται είναι η επιθεώρηση εργασίας όπου οι Alogogianni, & Virvou, (2022) μελετούν την πρόβλεψη παραβάσεων για την αδήλωτη εργασία μέσω μηχανικής μάθησης αξιοποιώντας το ιστορικό παραβάσεων νομοθεσίας. Το λογισμικό που χρησιμοποιούν επιδέχεται ως είσοδο τα στοιχεία όχι μονάχα για τις παραβάσεις για αδήλωτη εργασία και δείχνει σημαντικά αποτελέσματα. Αντίστοιχα οι Panagoulias, et al. (2022) εξετάζουν τον τρόπο που η μηχανική μάθηση μπορεί να βοηθήσει την ιατρική. Αυτό γίνεται μέσω μιας πρόβλεψης με βάση τους βιοδείκτες των ανθρώπων για τον προσδιορισμό της κατάστασης της υγείας τους.

Η χρήση της τεχνητής νοημοσυνης και της μηχανικής μάθησης αντιμετωπίζει ήδη μεγάλες προκλήσεις, μια από αυτές είναι η εύρεση εξειδικευμένου δυναμικού που θα εργαστεί στην τεχνητή νοημοσύνη. Οι Virvou, et al. (2023) αναφέρουν ως λύση σε αυτό το πρόβλημα, την ενθάρρυνση μέσω ειδικών σχεδιασμένων προγραμμάτων, περισσότερων γυναικών να εργαστούν στον τομέα, που υπό-εκπροσωπούνται σε σχέση με τους άνδρες συναδέλφους τους . Μια τέτοια πρωτοβουλία θα αναδείξει και τις γυναίκες ως επιστήμονες – πρότυπο θα εμπνεύσει ακόμη περισσότερες να εργαστούν στον κλάδο.

5.1 Τα κινητά λειτουργικά συστήματα

Ένα λειτουργικό σύστημα κινητής τηλεφωνίας (ή λειτουργικό σύστημα κινητής τηλεφωνίας) είναι ένα λειτουργικό σύστημα για τηλέφωνα, tablet, smartwatches ή άλλες κινητές συσκευές (Bhattacherjee & Lin, 2015).

Τα λειτουργικά συστήματα κινητής τηλεφωνίας συνδυάζουν χαρακτηριστικά ενός λειτουργικού συστήματος προσωπικού υπολογιστή με άλλα χαρακτηριστικά χρήσιμα για φορητή ή φορητή χρήση. Ορισμένες από αυτές τις λειτουργίες θεωρούνται απαραίτητες στα σύγχρονα κινητά συστήματα: ασύρματο ενσωματωμένο μόντεμ και δίσκος SIM για σύνδεση τηλεφωνίας και δεδομένων, οθόνη αφής, κινητής τηλεφωνίας, Bluetooth, Wi-Fi Protected Access, Wi-Fi, Παγκόσμιο Σύστημα Εντοπισμού (Global Positioning System - GPS) κινητή πλοήγηση, κάμερες εικόνας βίντεο και ενός καρέ, αναγνώριση ομιλίας, συσκευή εγγραφής φωνής, συσκευή αναπαραγωγής μουσικής, επικοινωνία κοντά στην οθόνη και υπέρυθρη εκπομπή (Bhattacherjee & Lin, 2015). Παρά την έντονη διάδοσή τους, τα συστήματα αυτά εγείρουν και ζητήματα προστασίας δεδομένων καθώς τα δεδομένα που προέρχονται από αισθητήρες κινητής τηλεφωνίας μπορούν να αποκαλύψουν πληροφορίες που οι περισσότεροι χρήστες δεν θα είχαν εκθέσει πρόθυμα αν τους είχαν ζητήσει, όπως συνομιλίες κλήσεων και SMS, φωτογραφίες και δεδομένα τοποθεσίας (Politou, et al., 2022).

Τα κύρια λειτουργικά συστήματα για κινητά είναι (Soldani & Manzalini, 2015):

Android OS (Google Inc.): Το λειτουργικό σύστημα Android για κινητά είναι η ανοιχτή και δωρεάν στοίβα λογισμικού της Google που περιλαμβάνει ένα λειτουργικό σύστημα, ενδιάμεσο λογισμικό και επίσης βασικές εφαρμογές για χρήση σε κινητές συσκευές, συμπεριλαμβανομένων των smartphone.

Bada (Samsung Electronics): Το Bada είναι ένα ιδιόκτητο λειτουργικό σύστημα κινητής τηλεφωνίας της Samsung που κυκλοφόρησε για πρώτη φορά το 2010. Το Samsung Wave ήταν το πρώτο smartphone που χρησιμοποίησε αυτό το λειτουργικό σύστημα για φορητές συσκευές. Το Bada παρέχει λειτουργίες για κινητά, όπως multipoint-touch, τρισδιάστατα γραφικά και φυσικά, λήψεις και εγκατάσταση εφαρμογών.

BlackBerry OS (Research In Motion): Το BlackBerry OS είναι ένα ιδιόκτητο λειτουργικό σύστημα για κινητά που αναπτύχθηκε από την Research In Motion για χρήση στις δημοφιλείς φορητές συσκευές BlackBerry της εταιρείας. Η πλατφόρμα BlackBerry είναι δημοφιλής στους εταιρικούς χρήστες καθώς προσφέρει συγχρονισμό με Microsoft Exchange, Lotus Domino, Novell Group-Wise email και άλλο επιχειρηματικό λογισμικό, όταν χρησιμοποιείται με τον Black-Berry Enterprise Server.

iPhone OS / iOS (Apple): Το iPhone OS της Apple αναπτύχθηκε αρχικά για χρήση στις συσκευές iPhone της. Τώρα, το λειτουργικό σύστημα για κινητά αναφέρεται ως iOS και υποστηρίζεται σε πολλές συσκευές Apple, συμπεριλαμβανομένων των iPhone, iPad, iPad 2 και iPod Touch. Το λειτουργικό σύστημα για κινητά iOS είναι διαθέσιμο μόνο στις συσκευές που κατασκευάζει η Apple, καθώς η εταιρεία δεν αδειοδοτεί το λειτουργικό σύστημα για υλικό τρίτων κατασκευαστών. Το Apple iOS προέρχεται από το λειτουργικό σύστημα Mac OS X της Apple.

MeeGo OS (Nokia και Intel): Ένα κοινό λειτουργικό σύστημα ανοιχτού κώδικα για κινητά που είναι το αποτέλεσμα της συγχώνευσης δύο προϊόντων που βασίζονται σε τεχνολογίες ανοιχτού κώδικα: Maemo (Nokia) και Moblin (Intel). Το MeeGo είναι ένα φορητό λειτουργικό σύστημα που έχει σχεδιαστεί για να λειτουργεί σε πολλές συσκευές, όπως smartphone, netbook, tablet, συστήματα πληροφοριών εντός του οχήματος και διάφορες συσκευές που χρησιμοποιούν αρχιτεκτονικές Intel Atom και ARMv7.

Palm OS (Garnet OS): Το Palm OS είναι ένα ιδιόκτητο λειτουργικό σύστημα για κινητά (λειτουργικό σύστημα PDA) που κυκλοφόρησε αρχικά το 1996 στο Pilot 1000 χειρός. Οι νεότερες εκδόσεις του Palm OS έχουν προσθέσει υποστήριξη για θύρες επέκτασης, νέους επεξεργαστές, εξωτερικές κάρτες μνήμης, βελτιωμένη ασφάλεια και υποστήριξη για επεξεργαστές ARM και smartphone. Το Palm OS 5 επεκτάθηκε για να παρέχει υποστήριξη για ένα ευρύ φάσμα αναλύσεων οθόνης, ασύρματες συνδέσεις και βελτιωμένες δυνατότητες πολυμέσων και ονομάζεται Garnet OS.

Symbian OS (Nokia): Το Symbian είναι ένα λειτουργικό σύστημα κινητής τηλεφωνίας (OS) που στοχεύει σε κινητά τηλέφωνα που προσφέρει υψηλού επιπέδου ενοποίηση με λειτουργίες επικοινωνίας και διαχείρισης προσωπικών πληροφοριών (PIM). Το Symbian OS συνδυάζει ενδιάμεσο λογισμικό με ασύρματες επικοινωνίες μέσω ενός ενσωματωμένου γραμματοκιβωτίου και την ενσωμάτωση λειτουργιών Java και PIM (ατζέντα και επαφές). Η Nokia έχει διαθέσει την πλατφόρμα Symbian με ένα εναλλακτικό, ανοιχτό και άμεσο μοντέλο, για να συνεργαστεί με ορισμένους κατασκευαστές OEM και τη μικρή κοινότητα των συνεργατών ανάπτυξης πλατφόρμας. Η Nokia δεν διατηρεί το Symbian ως έργο ανάπτυξης ανοιχτού κώδικα.

webOS (Palm/HP): Το WebOS είναι ένα λειτουργικό σύστημα για φορητές συσκευές που εκτελείται στον πυρήνα του Linux. Το WebOS αναπτύχθηκε αρχικά από την Palm ως διάδοχος του λειτουργικού της συστήματος για φορητές συσκευές Palm OS. Είναι ένα ιδιόκτητο λειτουργικό σύστημα για φορητές συσκευές, το οποίο τελικά αποκτήθηκε από την HP και τώρα αναφέρεται ως webOS (πεζά w) στη βιβλιογραφία της HP. Η HP χρησιμοποιεί webOS σε πολλές συσκευές, συμπεριλαμβανομένων πολλών smartphone και HP Touch-Pads. Η HP ώθησε το webOS της στην αγορά εταιρικών κινητών τηλεφώνων εστιάζοντας στη βελτίωση των χαρακτηριστικών ασφαλείας και της διαχείρισης με την κυκλοφορία του webOS 3.x. Η HP ανακοίνωσε επίσης σχέδια για μια έκδοση του webOS που θα τρέχει στο λειτουργικό σύστημα Microsoft Windows και θα εγκατασταθεί σε όλους τους επιτραπέζιους και φορητούς υπολογιστές HP το 2012.

Windows Mobile (Windows Phone): Το Windows Mobile είναι το λειτουργικό σύστημα για φορητές συσκευές της Microsoft που χρησιμοποιείται σε smartphone και φορητές συσκευές με ή χωρίς οθόνες αφής. Το Mobile OS βασίζεται στον πυρήνα των Windows CE 5.2. Το 2010 η Microsoft ανακοίνωσε μια νέα πλατφόρμα smartphone που ονομάζεται Windows Phone.

Με εξαίρεση το Android (που αναπτύχθηκε από την Google), τα λειτουργικά συστήματα για κινητά αναπτύσσονται από διαφορετικούς κατασκευαστές κινητών τηλεφώνων, συμπεριλαμβανομένης της Nokia (Symbian, MeeGo, Maemo). Apple (Apple iOS); Research In Motion (RIM) (BlackBerry OS); Microsoft (Windows Mobile, Windows Phone) και Samsung (Palm WebOS και bada). Τα Android, LiMo, Maemo, Openmoko και Qt Extende (Qtopia) βασίζονται στο λειτουργικό σύστημα ανοιχτού κώδικα Linux (Soldani & Manzalini, 2015).

5.1.1 Λειτουργικό σύστημα Android

Το Android είναι ένα λειτουργικό σύστημα για κινητά που αναπτύχθηκε από την Google. Βασίζεται σε μια τροποποιημένη έκδοση του πυρήνα Linux και άλλου λογισμικού ανοιχτού κώδικα και έχει σχεδιαστεί κυρίως για κινητές συσκευές με οθόνη αφής, όπως smartphone και tablet. Αρχικά αναπτύχθηκε από την Android Inc., την οποία αγόρασε η Google το 2005, το Android παρουσιάστηκε το 2007, με την πρώτη εμπορική συσκευή Android που κυκλοφόρησε τον Σεπτέμβριο του 2008. Το λειτουργικό σύστημα έχει περάσει από πολλές σημαντικές εκδόσεις, με την τρέχουσα έκδοση να είναι το 9 "Pie" , κυκλοφόρησε τον Αύγουστο του 2018 (Sarkar et al., 2019).

Εξέλιξη του λειτουργικού συστήματος Android

Το Android έκανε το επίσημο δημόσιο ντεμπούτο του το 2008 με το Android 1.0 μια κυκλοφορία τόσο αρχαία που δεν είχε καν ένα χαριτωμένο κωδικό όνομα.

Το Android του 2009 κυκλοφόρησε την έκδοση 1.5: Cupcake. Η παράδοση των ονομάτων εκδόσεων Android γεννήθηκε. Το Cupcake εισήγαγε πολυάριθμες βελτιώσεις στη διεπαφή Android, συμπεριλαμβανομένου του πρώτου πληκτρολογίου οθόνης, κάτι απαραίτητο καθώς τα

τηλέφωνα απομακρύνθηκαν από το κάποτε πανταχού παρόν μοντέλο φυσικού πληκτρολογίου. Αυτή η έκδοση έφερε επίσης το πλαίσιο για γραφικά στοιχεία εφαρμογών τρίτων, τα οποία θα μετατραπούν γρήγορα σε ένα από τα πιο ξεχωριστά στοιχεία του Android και παρείχε την πρώτη επιλογή της πλατφόρμας για εγγραφή βίντεο (Gilski & Stefanski, 2015).

Το Android 1.6, Donut, κυκλοφόρησε στον κόσμο το φθινόπωρο του 2009. Το Donut παρουσίασε ορισμένες σημαντικές τρύπες στο κέντρο του Android, συμπεριλαμβανομένης της δυνατότητας του λειτουργικού συστήματος να λειτουργεί σε μια ποικιλία διαφορετικών μεγεθών και αναλύσεων οθόνης, ένας παράγοντας που θα πρέπει να είναι κρίσιμος για τα επόμενα χρόνια.

Το Android 2.0 Eclair, εμφανίστηκε μόλις έξι εβδομάδες μετά το Donut και η ενημερωμένη έκδοση του "pointone", που ονομάζεται επίσης Eclair, κυκλοφόρησε λίγους μήνες αργότερα. Το Eclair ήταν η πρώτη έκδοση Android που μπήκε στην επικρατούσα συνείδηση χάρη στο αρχικό τηλέφωνο Motorola Droid και την τεράστια εκστρατεία μάρκετινγκ υπό την ηγεσία της Verizon που το περιβάλλει (Gilski & Stefanski, 2015).

Μόλις τέσσερις μήνες μετά την κυκλοφορία του Android 2.1, η Google εξυπηρέτησε το Android 2.2, Froyo, το οποίο περιστράφηκε σε μεγάλο βαθμό γύρω από βελτιώσεις απόδοσης κάτω από το καπό. Ωστόσο, το Froyo παρείχε ορισμένες σημαντικές λειτουργίες στο μπροστινό μέρος, συμπεριλαμβανομένης της προσθήκης του πλέον τυπικού dock στο κάτω μέρος της αρχικής οθόνης, καθώς και της πρώτης ενσάρκωσης των Φωνητικών ενεργειών, που επιτρέπει την εκτέλεση βασικών λειτουργιών, όπως λήψη οδηγιών και λήψη σημειώσεις πατώντας ένα εικονίδιο και μετά εκφωνώντας μια εντολή.

Η πρώτη αληθινή οπτική ταυτότητα του Android άρχισε να έρχεται στο επίκεντρο με την κυκλοφορία του Gingerbread του 2010. Το έντονο πράσινο ήταν από καιρό το χρώμα της μασκότ ρομπότ του Android και με το Gingerbread έγινε αναπόσπαστο μέρος της εμφάνισης του λειτουργικού συστήματος. Το μαύρο και το πράσινο έμπαιναν σε όλη τη διεπαφή χρήστη καθώς το Android ξεκίνησε την αργή του πορεία προς την ξεχωριστή σχεδίαση (Gilski & Stefanski, 2015).

Το Android 3.0 (Honeycomb, 2011) ήρθε στον κόσμο ως έκδοση μόνο για tablet για να συνοδεύσει την κυκλοφορία του Motorola Xoom και μέσω των επακόλουθων ενημερώσεων 3.1 και 3.2, παρέμεινε μια οντότητα αποκλειστική (και κλειστού κώδικα) για tablet. Υπό την καθοδήγηση του νεοαφιχθέντος επικεφαλής σχεδιασμού Matias Duarte, η Honeycomb παρουσίασε ένα δραματικά επανασχεδιασμένο UI για Android. Είχε μια "ολογραφική" σχεδίαση που έμοιαζε με χώρο που αντικατέστησε το σήμα κατατεθέν της πλατφόρμας με το πράσινο με το μπλε και έδινε έμφαση στην αξιοποίηση του χώρου της οθόνης ενός tablet.

Το Ice Cream Sandwich, που κυκλοφόρησε επίσης το 2011, αποτέλεσε την επίσημη είσοδο της πλατφόρμας στην εποχή του μοντέρνου σχεδιασμού. Η κυκλοφορία προσδιόρισε τις οπτικές ιδέες που εισήχθησαν με το Honeycomb και επανενώθηκαν τα tablet και τα τηλέφωνα με ένα ενιαίο, ενιαίο όραμα διεπαφής χρήστη (Irwansyah et al., 2018).

Το 2012 και το 2013 κυκλοφόρησαν τρεις εκδόσεις του Android Jelly Bean. Πήραν τα φρέσκα θεμέλια της ICS και έκαναν σημαντικά βήματα για τη βελτίωση και την οικοδόμηση πάνω σε αυτό. Οι εκδόσεις πρόσθεσαν αρκετή ισορροπία και λάμψη στο λειτουργικό σύστημα και συνέβαλαν πολύ στο να κάνουν το Android πιο ελκυστικό για τον μέσο χρήστη. Πέρα από τα οπτικά στοιχεία, το Jelly Bean έφερε την πρώτη μας γεύση από το Google Now, το θεαματικό βοηθητικό πρόγραμμα πρόβλεψης-νοημοσύνης που δυστυχώς από τότε μετατράπηκε σε μια ένδοξη ροή ειδήσεων. Μας έδωσε επεκτάσιμες και διαδραστικές ειδοποιήσεις, ένα διευρυμένο σύστημα φωνητικής αναζήτησης και ένα πιο προηγμένο σύστημα για την εμφάνιση των αποτελεσμάτων αναζήτησης γενικά, με έμφαση σε αποτελέσματα που βασίζονται σε κάρτες που προσπαθούσαν να απαντήσουν άμεσα σε ερωτήσεις. Η υποστήριξη πολλών χρηστών τέθηκε επίσης στο παιχνίδι, αν και μόνο tablet σε αυτό το σημείο και μια πρώιμη έκδοση του πίνακα Γρήγορων ρυθμίσεων του Android έκανε την πρώτη της εμφάνιση (Irwansyah et al., 2018).

Το 2013, η κυκλοφορία του KitKat σηματοδότησε το τέλος της σκοτεινής εποχής του Android, καθώς οι μαύροι του Gingerbread και τα μπλουζ του Honeycomb έβγαλαν τελικά το δρόμο τους από το λειτουργικό σύστημα. Πιο ανοιχτόχρωμα φόντο και πιο ουδέτερες επισημάνσεις πήραν τη θέση τους, με μια διαφανή γραμμή κατάστασης και λευκά εικονίδια που δίνουν στο λειτουργικό σύστημα μια πιο σύγχρονη εμφάνιση. Το Android 4.4 είδε επίσης την πρώτη έκδοση

της υποστήριξης "OK, Google", αλλά στο KitKat, η προτροπή ενεργοποίησης hands-free λειτουργούσε μόνο όταν η οθόνη ήταν ήδη ενεργοποιημένη και ήταν είτε στην αρχική οθόνη είτε μέσα στην εφαρμογή Google.

Η Google ουσιαστικά επανεφηύρε το Android με την κυκλοφορία του Android 5.0 Lollipop το φθινόπωρο του 2014. Το Lollipop κυκλοφόρησε το πρότυπο Material Design, που εξακολουθεί να υπάρχει σήμερα, το οποίο έφερε μια εντελώς νέα εμφάνιση που επεκτάθηκε σε όλο το Android, τις εφαρμογές του και ακόμη και σε άλλα προϊόντα της Google. Η ιδέα που βασίζεται στην κάρτα που είχε διασκορπιστεί σε όλο το Android έγινε ένα βασικό μοτίβο διεπαφής χρήστη, που θα καθοδηγούσε την εμφάνιση όλων, από τις ειδοποιήσεις, οι οποίες εμφανίζονταν τώρα στην οθόνη κλειδώματος για πρόσβαση με μια ματιά, έως τη λίστα Πρόσφατων εφαρμογών , που πήρε μια ασύστολη εμφάνιση βασισμένη σε κάρτες. Το Lollipop εισήγαγε μια σειρά από νέες δυνατότητες στο Android, συμπεριλαμβανομένου του φωνητικού ελέγχου πραγματικά hands-free μέσω της εντολής "OK, Google", υποστήριξη για πολλούς χρήστες σε τηλέφωνα και μια λειτουργία προτεραιότητας για καλύτερη διαχείριση ειδοποιήσεων. Άλλαξε τόσο πολύ, δυστυχώς, που εισήγαγε επίσης ένα σωρό προβληματικά σφάλματα, πολλά από τα οποία δεν θα είχαν εξαλειφθεί πλήρως μέχρι την κυκλοφορία 5.1 του επόμενου έτους.

Στο μεγάλο σχέδιο των πραγμάτων, το Marshmallow του 2015 (Android 6.0) ήταν μια αρκετά μικρή έκδοση Android, μια έκδοση που έμοιαζε περισσότερο με μια ενημέρωση επιπέδου 0,1 παρά με οτιδήποτε άξιζε μια πλήρη αύξηση αριθμού. Αλλά ξεκίνησε την τάση της Google να κυκλοφορεί μια σημαντική έκδοση Android ετησίως και αυτή η έκδοση να λαμβάνει πάντα τον δικό της ακέραιο αριθμό. Το πιο σημαντικό στοιχείο αυτής της έκδοσης ήταν μια δυνατότητα αναζήτησης οθόνης που ονομάζεται Now On Tap, κάτι που είχε πολλές δυνατότητες που δεν αξιοποιήθηκε πλήρως. Η Google ποτέ δεν τελειοποίησε αρκετά το σύστημα και κατέληξε αθόρυβα να αποσύρει την επωνυμία της και να την απομακρύνει από το προσκήνιο το επόμενο έτος. Ωστόσο, το Android 6.0 εισήγαγε ορισμένα πράγματα με διαρκή αντίκτυπο, συμπεριλαμβανομένων πιο λεπτομερών αδειών εφαρμογών, υποστήριξης για προγράμματα ανάγνωσης ψηφιακών αποτυπωμάτων και υποστήριξης για USB-C (Irwansyah et al., 2018).

Οι εκδόσεις Android Nougat του 2016 της Google (7.0 και 7.1) παρείχαν στο Android μια εγγενή λειτουργία διαχωρισμού οθόνης, ένα σύστημα οργάνωσης ειδοποιήσεων και μια λειτουργία Εξοικονόμησης δεδομένων. Το Nougat πρόσθεσε μερικές μικρότερες αλλά ακόμα σημαντικές λειτουργίες, όπως μια συντόμευση τύπου Alt-Tab για εναλλαγή μεταξύ εφαρμογών. Ίσως το πιο σημαντικό μεταξύ των βελτιώσεων του Nougat, ωστόσο, ήταν η κυκλοφορία του Google Assistant, το οποίο ήρθε παράλληλα με την ανακοίνωση του πρώτου πλήρως αυτοδημιούργητου τηλεφώνου της Google, του Pixel, περίπου δύο μήνες μετά το ντεμπούτο του Nougat. Ο Βοηθός θα συνεχίσει να γίνεται ένα κρίσιμο στοιχείο του Android και των περισσότερων άλλων προϊόντων της Google και είναι αναμφισβήτητα η κορυφαία προσπάθεια της εταιρείας σήμερα.

Το Android 8.0 (Oreo) πρόσθεσε μια ποικιλία ωραιοτήτων στην πλατφόρμα, συμπεριλαμβανομένης μιας εγγενούς λειτουργίας εικόνας σε εικόνα, μιας επιλογής αναβολής ειδοποιήσεων και καναλιών ειδοποιήσεων που προσφέρουν τον έλεγχο του τρόπου με τον οποίο οι εφαρμογές μπορούν να ειδοποιήσουν το χρήστη. Η κυκλοφορία του 2017 περιλάμβανε επίσης ορισμένα αξιοσημείωτα στοιχεία που προώθησαν τον στόχο της Google να ευθυγραμμίσει το Android και το Chrome OS και να βελτιώσει την εμπειρία χρήσης εφαρμογών Android σε Chromebook, και ήταν η πρώτη έκδοση Android που περιείχε το Project Treble, μια φιλόδοξη προσπάθεια δημιουργίας μιας αρθρωτής βάσης για Κώδικας Android με την ελπίδα να διευκολύνει τους κατασκευαστές συσκευών να παρέχουν έγκαιρες ενημερώσεις λογισμικού (Sarkar et al., 2018),

Η πιο πρόσφατη προσθήκη στη λίστα εκδόσεων Android είναι το Android Pie. Το Android 9 μπήκε στον κόσμο στις αρχές Αυγούστου 2018 μετά από αρκετούς μήνες εξέλιξης στις δημόσιες προεπισκοπήσεις beta. Η πιο μεταμορφωτική αλλαγή του Pie είναι το νέο του σύστημα πλοήγησης με χειρονομίες, το οποίο ανταλλάσσει τα παραδοσιακά πλήκτρα Android Back, Home και Overview με ένα μόνο πολυλειτουργικό κουμπί Home και μια σειρά από εντολές που βασίζονται σε χειρονομίες. επίμηκες κουμπί Home και ένα μικρό κουμπί Πίσω που εμφανίζεται όπως απαιτείται. Το Android 9 διαθέτει πολλές άλλες αξιοσημείωτες δυνατότητες παραγωγικότητας, όπως ένα καθολικό σύστημα προτεινόμενης απάντησης για ειδοποιήσεις μηνυμάτων, μια πιο αποτελεσματική μέθοδο διαχείρισης στιγμιότυπων οθόνης και πιο έξυπνα συστήματα για διαχείριση ενέργειας και έλεγχο φωτεινότητας οθόνης. Και, φυσικά, δεν υπάρχει

έλλειψη μικρότερων αλλά ακόμα σημαντικών εξελίξεων που κρύβονται σε όλη τη διάρκεια του Pie, όπως ένας πιο έξυπνος τρόπος χειρισμού των hotspot Wi-Fi, μια ευπρόσδεκτη αλλαγή στη λειτουργία εξοικονόμησης μπαταρίας του Android και μια χρήσιμη νέα πινελιά για τους αισθητήρες δακτυλικών αποτυπωμάτων (Sarkar et al., 2018),

Στοιχεία της πλατφόρμας Android

Τα στοιχεία αρχιτεκτονικής Android είναι μια συλλογή από βιβλιοθήκες που βοηθούν στον σχεδιασμό ισχυρών, ελεγχόμενων και συντηρήσιμων εφαρμογών. Τα στοιχεία εφαρμογής είναι τα βασικά δομικά στοιχεία μιας εφαρμογής Android. Κάθε στοιχείο είναι ένα σημείο εισόδου μέσω του οποίου το σύστημα ή ένας χρήστης μπορεί να εισέλθει στην εφαρμογή. Ορισμένα στοιχεία εξαρτώνται από άλλα.

Υπάρχουν τέσσερις διαφορετικοί τύποι στοιχείων εφαρμογής: Δραστηριότητες, Υπηρεσίες, Δέκτες εκπομπής και πάροχοι περιεχομένου. Κάθε τύπος εξυπηρετεί έναν ξεχωριστό σκοπό και έχει έναν ξεχωριστό κύκλο ζωής που καθορίζει τον τρόπο δημιουργίας και καταστροφής του στοιχείου (Bala, Sharma & Kaur, 2015).

Δραστηριότητα: Μια δραστηριότητα είναι το σημείο εισόδου για την αλληλεπίδραση με τον χρήστη. Αντιπροσωπεύει μια ενιαία οθόνη με διεπαφή χρήστη. Μια δραστηριότητα διευκολύνει τις ακόλουθες βασικές αλληλεπιδράσεις μεταξύ συστήματος και εφαρμογής (Bala, Sharma & Kaur, 2015):

- Παρακολούθηση του τι ενδιαφέρεται επί του παρόντος ο χρήστης (τι εμφανίζεται στην οθόνη) για να διασφαλιστεί ότι το σύστημα συνεχίζει να εκτελεί τη διαδικασία που φιλοξενεί τη δραστηριότητα. γνωρίζοντας ότι οι διαδικασίες που χρησιμοποιήθηκαν προηγουμένως περιέχουν πράγματα στα οποία μπορεί να επιστρέψει ο χρήστης (δραστηριότητες που έχουν διακοπεί) και επομένως δίνουν μεγαλύτερη προτεραιότητα στη διατήρηση αυτών των διεργασιών.
- Βοηθώντας την εφαρμογή να χειριστεί τη διακοπή της διεργασίας της, ώστε ο χρήστης να μπορεί να επιστρέψει στις δραστηριότητές του με επαναφορά της προηγούμενης κατάστασής του.
- Παρέχοντας έναν τρόπο για τις εφαρμογές να εφαρμόζουν μεταξύ τους τα OW των χρηστών και για το σύστημα να συντονίζει αυτά τα OW.

Μια δραστηριότητα υλοποιείται ως υποκλάση της κλάσης Δραστηριότητα.

Υπηρεσία: Μια υπηρεσία είναι ένα σημείο εισόδου γενικής χρήσης για τη διατήρηση μιας εφαρμογής σε λειτουργία στο παρασκήνιο για κάθε είδους λόγους. Είναι ένα στοιχείο που εκτελείται στο παρασκήνιο για την εκτέλεση μακροχρόνιων λειτουργιών ή για την εκτέλεση εργασιών για απομακρυσμένες διεργασίες. Μια υπηρεσία δεν παρέχει διεπαφή χρήστη. Ένα άλλο στοιχείο, όπως μια δραστηριότητα, μπορεί να ξεκινήσει την υπηρεσία και να την αφήσει να εκτελεστεί ή να συνδεθεί με αυτήν για να αλληλεπιδράσει μαζί της. Στην πραγματικότητα, υπάρχουν δύο πολύ διακριτές υπηρεσίες σημασιολογίας που λένε στο σύστημα πώς να διαχειρίζεται μια εφαρμογή: Οι εκκινημένες υπηρεσίες λένε στο σύστημα να τις κρατήσει σε λειτουργία μέχρι να ολοκληρωθεί η εργασία τους. Αυτό θα μπορούσε να είναι ο συγχρονισμός ορισμένων δεδομένων στο παρασκήνιο ή η αναπαραγωγή μουσικής ακόμα και μετά την αποχώρηση του χρήστη από την εφαρμογή. Οι δεσμευμένες υπηρεσίες εκτελούνται επειδή κάποια άλλη εφαρμογή (ή το σύστημα) έχει πει ότι θέλει να κάνει χρήση της υπηρεσίας. Αυτή είναι βασικά η υπηρεσία που παρέχει ένα API σε μια άλλη διαδικασία. Το σύστημα έτσι γνωρίζει ότι υπάρχει μια εξάρτηση μεταξύ αυτών των διαδικασιών. Μια υπηρεσία υλοποιείται ως υποκατηγορία Υπηρεσίας (Bala, Sharma & Kaur, 2015).

Δέκτης εκπομπής: Ο δέκτης εκπομπής είναι ένα στοιχείο που επιτρέπει στο σύστημα να παραδίδει συμβάντα στην εφαρμογή εκτός ενός κανονικού χρήστη, επιτρέποντας στην εφαρμογή να ανταποκρίνεται σε ανακοινώσεις εκπομπής σε όλο το σύστημα. Επειδή οι δέκτες εκπομπής είναι μια άλλη καλά διαμορφωμένη είσοδος στην εφαρμογή, το σύστημα μπορεί να παρέχει εκπομπές ακόμα και σε εφαρμογές που δεν εκτελούνται αυτήν τη στιγμή. Οι δέκτες εκπομπής δεν εμφανίζουν διεπαφή χρήστη, μπορούν να δημιουργήσουν μια ειδοποίηση

γραμμής κατάστασης για να ειδοποιούν τον χρήστη όταν συμβαίνει ένα συμβάν μετάδοσης. Συνηθέστερα, ωστόσο, ένας δέκτης εκπομπής είναι απλώς μια πύλη σε άλλα εξαρτήματα και προορίζεται να κάνει μια πολύ ελάχιστη ποσότητα εργασίας. Ένας δέκτης εκπομπής υλοποιείται ως υποκατηγορία του BroadcastReceiver και κάθε εκπομπή παραδίδεται ως αντικείμενο Intent (Uttarwar et al., 2021).

Πάροχος περιεχομένου: Ένας πάροχος περιεχομένου διαχειρίζεται ένα κοινόχρηστο σύνολο δεδομένων της εφαρμογής, τα οποία μπορούν να αποθηκευτούν σε διάφορες μόνιμες θέσεις αποθήκευσης, όπως σε ένα σύστημα αρχείων, μια βάση δεδομένων SQLite, στον ιστό, ή οποιαδήποτε άλλη παρόμοια τοποθεσία, στην οποία η εφαρμογή έχει πρόσβαση. Μέσω του παρόχου περιεχομένου, άλλες εφαρμογές μπορούν να υποβάλουν ερωτήματα ή να τροποποιήσουν τα δεδομένα εάν το επιτρέπει ο πάροχος περιεχομένου. Για το σύστημα, ένας πάροχος περιεχομένου είναι ένα σημείο εισόδου σε μια εφαρμογή για τη δημοσίευση επώνυμων στοιχείων δεδομένων, που προσδιορίζονται από ένα σχήμα URI. Έτσι, μια εφαρμογή μπορεί να αποφασίσει πώς θέλει να αντιστοιχίσει τα δεδομένα που περιέχει σε έναν χώρο ονομάτων URI, διανέμοντας αυτά τα URI σε άλλες οντότητες που μπορούν με τη σειρά τους να τα χρησιμοποιήσουν για πρόσβαση στα δεδομένα. Οι πάροχοι περιεχομένου είναι επίσης χρήσιμοι για την ανάγνωση και τη σύνταξη δεδομένων που είναι ιδιωτικά στην εφαρμογή και δεν είναι κοινά. Ένας πάροχος περιεχομένου υλοποιείται ως υποκατηγορία του Παρόχου Περιεχομένου και πρέπει να εφαρμόσει ένα τυπικό σύνολο API που επιτρέπει σε άλλες εφαρμογές να εκτελούν συναλλαγές (Uttarwar et al., 2021).

Πρόθεση: Τρεις από τους τέσσερις τύπους στοιχείων, οι δραστηριότητες, οι υπηρεσίες και οι δέκτες εκπομπής ενεργοποιούνται από ένα ασύγχρονο μήνυμα που ονομάζεται πρόθεση. Οι προθέσεις συνδέουν μεμονωμένα στοιχεία μεταξύ τους κατά το χρόνο εκτέλεσης. Μπορούν να θεωρηθούν ως οι αγγελιοφόροι που ζητούν μια ενέργεια από άλλα στοιχεία, είτε το στοιχείο ανήκει στην εφαρμογή είτε σε άλλη. Μια πρόθεση δημιουργείται με ένα αντικείμενο Intent, το οποίο ορίζει ένα μήνυμα για την ενεργοποίηση είτε ενός συγκεκριμένου στοιχείου (σαφής πρόθεση) είτε ενός συγκεκριμένου τύπου στοιχείου (σιωπηρής πρόθεσης) (Uttarwar et al., 2021).

Για να μπορέσει το σύστημα Android να ξεκινήσει ένα στοιχείο εφαρμογής, το σύστημα πρέπει να γνωρίζει ότι το στοιχείο υπάρχει διαβάζοντας το μανιφέστο της εφαρμογής, το AndroidManifest. xml. Η εφαρμογή πρέπει να δηλώσει όλα τα στοιχεία της σε αυτό το αρχείο, το οποίο πρέπει να βρίσκεται στη ρίζα του καταλόγου του έργου της εφαρμογής.

Το μανιφέστο κάνει πολλά πράγματα εκτός από τη δήλωση των στοιχείων της εφαρμογής, όπως τα εξής (Sarkar et al., 2019):

- Προσδιορίζει τυχόν δικαιώματα χρήστη που απαιτεί η εφαρμογή, όπως πρόσβαση στο Διαδίκτυο ή πρόσβαση ανάγνωσης στις επαφές του χρήστη.
- Δηλώνει το ελάχιστο επίπεδο API που απαιτείται από την εφαρμογή, με βάση τα API που χρησιμοποιεί η εφαρμογή.
- Δηλώνει χαρακτηριστικά υλικού και λογισμικού που χρησιμοποιούνται ή απαιτούνται από την εφαρμογή, όπως κάμερα, υπηρεσίες bluetooth ή οθόνη πολλαπλής αφής.
- Δηλώνει βιβλιοθήκες API με τις οποίες πρέπει να συνδεθεί η εφαρμογή (εκτός από τα API πλαισίου Android), όπως η βιβλιοθήκη των Χαρτών Google.

Το Android παρέχει μια ποικιλία από προκατασκευασμένα στοιχεία διεπαφής χρήστη, όπως αντικείμενα δομημένης διάταξης και στοιχεία ελέγχου διεπαφής χρήστη που επιτρέπουν τη δημιουργία της γραφικής διεπαφής χρήστη για την εφαρμογή. Το Android παρέχει επίσης άλλες μονάδες διεπαφής χρήστη για ειδικές διεπαφές, όπως διαλόγους, ειδοποιήσεις και μενού.

5.1.2 Λειτουργικό σύστημα iPhone

Το iOS είναι ένα λειτουργικό σύστημα για κινητά που δημιουργήθηκε και αναπτύχθηκε από την Apple Inc. αποκλειστικά για το υλικό της. Είναι το λειτουργικό σύστημα που τροφοδοτεί επί του παρόντος πολλές από τις κινητές συσκευές της εταιρείας, συμπεριλαμβανομένων των iPhone, iPad και iPod Touch. Είναι το δεύτερο πιο δημοφιλές λειτουργικό σύστημα για κινητά

παγκοσμίως μετά το Android. Το iOS έχει επεκταθεί για να υποστηρίζει άλλες συσκευές Apple όπως το iPod Touch (Σεπτέμβριος 2007) και το iPad (Ιανουάριος 2010).

Το iOS χρησιμοποιεί μια διεπαφή πολλαπλής αφής στην οποία χρησιμοποιούνται απλές χειρονομίες για τον χειρισμό της συσκευής, όπως το σάρωση του δαχτύλου στην οθόνη για να μετακινηθεί κανείς στην επόμενη σελίδα ή το «τσίμπημα» των δακτύλων για σμίκρυνση (Masruroh & Saputra, 2016).

Εξέλιξη του λειτουργικού συστήματος iPhone

Η Apple ανακοίνωσε το iPhone OS 1 στην κεντρική παρουσίαση του iPhone στις 9 Ιανουαρίου 2007 και κυκλοφόρησε στο κοινό μαζί με το αρχικό iPhone στις 29 Ιουνίου 2007. Δεν δόθηκε επίσημο όνομα κατά την αρχική του κυκλοφορία.

Τον Μάρτιο του 2008 η Apple ανακοίνωσε το iPhone OS 2b αλλά κυκλοφόρησε στο κοινό στις 11 Ιουλίου 2008 μαζί με το iPhone 3G. Η Apple δεν εγκατέλειψε την υποστήριξη για καμία συσκευή με αυτή την έκδοση. Το iPhone OS 2 ήταν συμβατό με όλες τις συσκευές που είχαν κυκλοφορήσει μέχρι τότε. Η κυκλοφορία του iPhone OS 2.1.1 έφερε υποστήριξη για το iPod Touch (2ης γενιάς). Η πιο βαθιά αλλαγή που εισήχθη σε αυτή την έκδοση ήταν το App Store και η υποστήριξή του για εγγενείς εφαρμογές τρίτων κατασκευαστών. Περίπου 500 εφαρμογές ήταν διαθέσιμες στο App Store κατά την κυκλοφορία. Προστέθηκαν επίσης εκατοντάδες άλλες κρίσιμες βελτιώσεις. Άλλες σημαντικές αλλαγές που εισήχθησαν στις 5 ενημερώσεις του iPhone OS 2.0 περιλάμβαναν την υποστήριξη podcast και τις οδηγίες για τα μέσα μαζικής μεταφοράς και το περπάτημα στους Χάρτες (και οι δύο στην έκδοση 2.2) (Masruroh & Saputra, 2016).

Το iPhone OS 3 κυκλοφόρησε στο κοινό στις 17 Ιουνίου 2009 μαζί με το iPhone 3GS. Η κυκλοφορία αυτής της έκδοσης του iOS συνόδευσε το ντεμπούτο του iPhone 3GS. Πρόσθεσε χαρακτηριστικά όπως αντιγραφή και επικόλληση, αναζήτηση Spotlight, υποστήριξη MMS στην εφαρμογή Μηνύματα και δυνατότητα εγγραφής βίντεο με την εφαρμογή Κάμερα. Επίσης, αξιοσημείωτο σε αυτή την έκδοση του iOS είναι ότι ήταν η πρώτη που υποστήριζε το iPad. Το iPad 1ης γενιάς κυκλοφόρησε το 2010 και η έκδοση 3.2 του λογισμικού το συνόδευε.

Στις 21 Ιουνίου 2010 η Apple κυκλοφόρησε το OS 4 μαζί με το iPhone 4. Με αυτή την έκδοση, η Apple διέκοψε την υποστήριξη για το αρχικό iPhone και το iPod Touch (1ης γενιάς), η οποία είναι η πρώτη φορά που η Apple διέκοψε την υποστήριξη για οποιαδήποτε συσκευή σε μια έκδοση iOS. Το iPhone 3G και το iPod Touch (2ης γενιάς) ήταν ικανά να τρέξουν το iOS 4, αλλά είχαν περιορισμένες δυνατότητες. Για παράδειγμα, και οι δύο συσκευές δεν διέθεταν δυνατότητες multitasking και τη δυνατότητα ορισμού ταπετσαρίας αρχικής οθόνης. Ωστόσο, το iOS 4 ήταν η πρώτη μεγάλη έκδοση για την οποία οι χρήστες του iPod Touch δεν χρειάστηκε να πληρώσουν χρήματα. Η κυκλοφορία του iOS 4.2.1 έφερε συμβατότητα με το αρχικό iPad και ήταν η τελευταία έκδοση που υποστηριζόταν στο iPhone 3G και το iPod Touch (2ης γενιάς) λόγω σημαντικών προβλημάτων απόδοσης. Η κυκλοφορία του iOS 4.3 έφερε συμβατότητα με το iPad 2. Δεν υποστηρίζεται πλέον από τις 18 Δεκεμβρίου 2013 (Masruroh & Saputra, 2016).

Το iOS 5 ανακοινώθηκε στις 6 Ιουνίου 2011 στην ετήσια εκδήλωση της Apple Worldwide Developers Conference (WWDC) και κυκλοφόρησε στο κοινό στις 12 Οκτωβρίου 2011 μαζί με το iPhone 4S. Η Apple δεν εγκατέλειψε την υποστήριξη για καμία συσκευή με αυτή την έκδοση- η υποστήριξη για το iPhone 3G και το iPod Touch 2ης γενιάς είχε ήδη εγκαταλειφθεί με την κυκλοφορία του iOS 4.3 επτά μήνες νωρίτερα. Ως εκ τούτου, το iOS 5 κυκλοφόρησε για το iPhone 3GS και μετά, το iPod Touch (3ης γενιάς) και μετά και όλα τα μοντέλα iPad. Με αυτή την έκδοση η Apple ανταποκρίθηκε στην αυξανόμενη τάση του wirelessness, και του cloud computing, στο iOS 5, εισάγοντας βασικές νέες λειτουργίες και πλατφόρμες. Μεταξύ αυτών ήταν το iCloud, η δυνατότητα ενεργοποίησης ενός iPhone ασύρματα (προηγουμένως χρειαζόταν σύνδεση με υπολογιστή) και ο συγχρονισμός με το iTunes μέσω Wi-Fi.

Το iOS 6 κυκλοφόρησε στο κοινό στις 19 Σεπτεμβρίου 2012 μαζί με το iPhone 5, το iPod Touch (5ης γενιάς) και το iPad 4. Με αυτή την έκδοση, η Apple εγκατέλειψε την υποστήριξη για το iPod Touch (3ης γενιάς) και το iPad (1ης γενιάς) λόγω περιορισμών υλικού και προσέφερε περιορισμένη μόνο υποστήριξη στο iPhone 3GS, το iPad 2 και το iPod Touch (4ης γενιάς). Το iOS 6.1.6 ήταν η τελευταία έκδοση που υποστηριζόταν για το iPhone 3GS και το iPod Touch (4ης γενιάς). Αυτή η έκδοση εισήγαγε στον κόσμο το Siri, το οποίο, παρά το γεγονός ότι αργότερα ξεπεράστηκε από τους ανταγωνιστές, ήταν μια πραγματικά επαναστατική τεχνολογία.

Η Apple εισήγαγε επίσης τη δική της εφαρμογή Χάρτες, η οποία έτυχε κακής υποδοχής λόγω σφαλμάτων, κακών οδηγιών και προβλημάτων με ορισμένες λειτουργίες (Shaw et al., 2016).

Η Apple ανακοίνωσε το iOS 7 στις 10 Ιουνίου 2013 στην ετήσια εκδήλωση της Apple Worldwide Developers Conference (WWDC) και κυκλοφόρησε στο κοινό στις 18 Σεπτεμβρίου 2013 μαζί με τα iPhone 5C και iPhone 5S. Με αυτή την έκδοση, η Apple έπαψε να υποστηρίζει το iPhone 3GS (λόγω περιορισμών υλικού) και το iPod Touch (4ης γενιάς) (λόγω προβλημάτων απόδοσης). Σε αυτή την έκδοση του iOS, η Apple εγκαινίασε μια σημαντική αναμόρφωση της διεπαφής χρήστη, με σκοπό να την κάνει πιο σύγχρονη (Shaw et al., 2016).

Η Apple κυκλοφόρησε το iOS 8 στο κοινό στις 17 Σεπτεμβρίου 2014 μαζί με το iPhone 6 και το iPhone 6 Plus. Η κυκλοφορία του iOS 8.1 έφερε υποστήριξη για τα iPad Air 2 και iPad Mini 3, ενώ η κυκλοφορία του iOS 8.4 έφερε υποστήριξη για το iPod Touch (6ης γενιάς). Το iOS 8.3 ήταν η πρώτη έκδοση του iOS που είχε διαθέσιμη δημόσια δοκιμή beta, όπου οι χρήστες μπορούσαν να δοκιμάσουν τη beta για τις επερχόμενες εκδόσεις του iOS και να στείλουν σχόλια στην Apple σχετικά με σφάλματα ή δυσλειτουργίες. Η τελική έκδοση του iOS 8 ήταν το iOS 8.4.1. Με τις ριζικές αλλαγές των δύο τελευταίων εκδόσεων να ανήκουν πλέον στο παρελθόν, η Apple επικεντρώθηκε και πάλι στην παροχή σημαντικών νέων χαρακτηριστικών. Μεταξύ αυτών των χαρακτηριστικών ήταν το ασφαλές, ανέπαφο σύστημα πληρωμών Apple Pay και, με την ενημέρωση iOS 8.4, η συνδρομητική υπηρεσία Apple Music. Συνεχίστηκαν επίσης οι βελτιώσεις στην πλατφόρμα iCloud, με την προσθήκη του iCloud Drive που μοιάζει με το Dropbox, της iCloud Photo Library και της iCloud Music Library (Shaw et al., 2016).

Η Apple ανακοίνωσε το iOS 9 στις 8 Ιουνίου 2015 και το κυκλοφόρησε στο κοινό στις 16 Σεπτεμβρίου 2015 μαζί με τα iPhone 6S, iPhone 6S Plus και iPad Mini 4. Με αυτή την έκδοση, η Apple δεν εγκατέλειψε την υποστήριξη για καμία συσκευή iOS. Ως εκ τούτου, το iOS 9 υποστηριζόταν από το iPhone 4S και μετά, από το iPod Touch (5ης γενιάς) και μετά, από το iPad 2 και μετά και από το iPad Mini (1ης γενιάς) και μετά. Αυτή η έκδοση έκανε το iPad 2 την πρώτη συσκευή που υποστήριζε έξι μεγάλες εκδόσεις του iOS, υποστηρίζοντας το iOS 4 έως 9. Παρά την υπόσχεση της Apple για καλύτερες επιδόσεις σε αυτές τις συσκευές, υπήρχαν ακόμα εκτεταμένα παράπονα ότι το πρόβλημα δεν είχε επιλυθεί. Το iOS 9.3.5 είναι η τελική έκδοση για τα iPhone 4S, iPad 2 και 3, iPod Touch (5ης γενιάς) και iPad Mini (1ης γενιάς). Αυτή η έκδοση αποσκοπούσε γενικά στην εδραίωση των θεμελίων του λειτουργικού συστήματος για το μέλλον. Παρουσιάστηκαν σημαντικές βελτιώσεις στην ταχύτητα και την απόκριση, τη σταθερότητα και την απόδοση σε παλαιότερες συσκευές.

Στις 13 Σεπτεμβρίου 2016 η Apple κυκλοφόρησε το iOS 10 μαζί με το iPhone 7 και το iPhone 7 Plus.

Αυτή η έκδοση έχει περιορισμένη υποστήριξη στα iPhone 5, iPhone 5C και iPad 4 επειδή αυτές οι συσκευές διαθέτουν επεξεργαστές 32bit (Shaw et al., 2016).

Ωστόσο, το iPhone 5S και μετά, το iPod Touch (6ης γενιάς) και μετά και το iPad Mini 2 και μετά υποστηρίζονται πλήρως. Τα κύρια θέματα του iOS 10 ήταν η διαλειτουργικότητα και η προσαρμογή. Οι εφαρμογές μπορούσαν πλέον να επικοινωνούν απευθείας μεταξύ τους σε μια συσκευή, επιτρέποντας σε μια εφαρμογή να χρησιμοποιεί κάποια χαρακτηριστικά από μια άλλη χωρίς να ανοίγει τη δεύτερη εφαρμογή Η Siri έγινε διαθέσιμη σε εφαρμογές τρίτων με νέους τρόπους.

Το iOS 11 ανακοινώθηκε στις 5 Ιουνίου 2017 και κυκλοφόρησε στο κοινό στις 19 Σεπτεμβρίου 2017 μαζί με το iPhone 8 και το iPhone 8 Plus. Με αυτή την έκδοση, η Apple εγκατέλειψε την υποστήριξη για τα 32bit iPhone 5 και iPhone 5C και το iPad 4, καθιστώντας το iOS ένα λειτουργικό σύστημα μόνο 64bit που εκτελεί μόνο εφαρμογές 64bit. Όλες οι άλλες συσκευές από το iPhone 6S 6S Plus και μετά, το iPad Pro και μετά και το iPad (2017) και μετά υποστηρίζονται πλήρως. Το iOS 11 περιέχει πολλές βελτιώσεις για το iPhone, αλλά η κύρια εστίασή του είναι να μετατρέψει τα μοντέλα της σειράς iPad Pro σε νόμιμους αντικαταστάτες φορητών υπολογιστών για ορισμένους χρήστες (Harkin & Molnar, 2021).

Η Apple ανακοίνωσε το iOS 12 στις 4 Ιουνίου 2018 και το κυκλοφόρησε στο κοινό στις 17 Σεπτεμβρίου 2018 μαζί με τα iPhone XS και iPhone XS Max. Με αυτή την έκδοση, η Apple δεν εγκατέλειψε την υποστήριξη για καμία συσκευή iOS. Ως εκ τούτου, το iOS 12 υποστηριζόταν από το iPhone 5S και μετά, από το iPod Touch (6ης γενιάς) και μετά, από το iPad Air και μετά και από το iPad Mini 2 και μετά. Τα νέα χαρακτηριστικά και οι βελτιώσεις που προστέθηκαν στο iOS 12 δεν είναι τόσο εκτεταμένα ή επαναστατικά όσο σε ορισμένες προηγούμενες ενημερώσεις

του λειτουργικού συστήματος. Αντ' αυτού, το iOS 12 επικεντρώθηκε περισσότερο στη βελτίωση συχνά χρησιμοποιούμενων λειτουργιών και στην προσθήκη "ρυτίδων" που βελτιώνουν τον τρόπο με τον οποίο οι χρήστες χρησιμοποιούν τις συσκευές τους (Harkin & Molnar, 2021).

Δομή εφαρμογών iOS

Οι εφαρμογές iOS διαφέρουν από αυτές του Android. Όταν δημιουργείται μια εφαρμογή, πρέπει να έχει διαφορετικούς πόρους και μεταδεδομένα, ώστε να μπορεί να εμφανίζεται σωστά σε συσκευές iOS (Hu et al., 2023):

- Ενημερωτικό αρχείο καταλόγου ιδιοκτησίας. Το αρχείο Info.plist περιέχει μεταδεδομένα σχετικά με την εφαρμογή, τα οποία χρησιμοποιεί το σύστημα για να αλληλεπιδράσει με την εφαρμογή.
- Μια δήλωση των απαιτούμενων δυνατοτήτων των εφαρμογών. Κάθε εφαρμογή πρέπει να δηλώνει τις δυνατότητες ή τις δυνατότητες υλικού που απαιτεί για να εκτελεστεί.
- Ένα ή περισσότερα εικονίδια. Το σύστημα εμφανίζει το εικονίδιο της εφαρμογής στην αρχική οθόνη μιας συσκευής χρήστη. Το σύστημα μπορεί επίσης να χρησιμοποιήσει άλλες εκδόσεις του εικονιδίου στην εφαρμογή Ρυθμίσεις ή κατά την εμφάνιση των αποτελεσμάτων μιας αναζήτησης.
- Μία ή περισσότερες εικόνες εκκίνησης. Όταν εκκινείται μια εφαρμογή, το σύστημα εμφανίζει μια προσωρινή εικόνα μέχρι να μπορέσει η εφαρμογή να παρουσιάσει τη διεπαφή χρήστη της.

Κατά την εκκίνηση, η συνάρτηση UIApplicationMain ρυθμίζει πολλά βασικά αντικείμενα και ξεκινά την εκτέλεση της εφαρμογής. Σημειώστε ότι οι εφαρμογές iOS χρησιμοποιούν αρχιτεκτονική μοντέλου-προβολή-ελεγκτή. Αυτό το μοτίβο διαχωρίζει τα δεδομένα των εφαρμογών και την επιχειρηματική λογική από την οπτική παρουσίαση αυτών των δεδομένων. Αυτή η αρχιτεκτονική είναι ζωτικής σημασίας για τη δημιουργία εφαρμογών που μπορούν να εκτελούνται σε διαφορετικές συσκευές με διαφορετικά μεγέθη οθόνης (Hu et al., 2023).

Το αντικείμενο UIApplication διαχειρίζεται τον βρόχο συμβάντος και άλλες συμπεριφορές εφαρμογής υψηλού επιπέδου. Αναφέρει επίσης βασικές μεταβάσεις εφαρμογών και ορισμένα ειδικά συμβάντα (όπως εισερχόμενες ειδοποιήσεις push) στον εκπρόσωπο του, το οποίο είναι ένα προσαρμοσμένο αντικείμενο.

Ο εκπρόσωπος της εφαρμογής είναι η καρδιά του προσαρμοσμένου κωδικού. Αυτό το αντικείμενο λειτουργεί παράλληλα με το αντικείμενο UIApplication για να χειριστεί την προετοιμασία της εφαρμογής, τις μεταβάσεις καταστάσεων και πολλά συμβάντα εφαρμογής υψηλού επιπέδου. Αυτό το αντικείμενο είναι επίσης το μόνο που εγγυάται ότι υπάρχει σε κάθε εφαρμογή, επομένως χρησιμοποιείται συχνά για τη ρύθμιση των αρχικών δομών δεδομένων της εφαρμογής.

Τα αντικείμενα μοντέλου δεδομένων αποθηκεύουν το περιεχόμενο των εφαρμογών και είναι συγκεκριμένα για την εφαρμογή. Για παράδειγμα, μια τραπεζική εφαρμογή μπορεί να αποθηκεύσει μια βάση δεδομένων που περιέχει οικονομικές συναλλαγές, ενώ μια εφαρμογή ζωγραφικής μπορεί να αποθηκεύσει ένα αντικείμενο εικόνας ή ακόμα και την ακολουθία εντολών σχεδίασης που οδήγησαν στη δημιουργία αυτής της εικόνας. (Στην τελευταία περίπτωση, ένα αντικείμενο εικόνας εξακολουθεί να είναι αντικείμενο δεδομένων επειδή είναι απλώς ένα κοντέινερ για τα δεδομένα εικόνας.)

Οι εφαρμογές μπορούν επίσης να χρησιμοποιούν αντικείμενα εγγράφων (προσαρμοσμένες υποκατηγορίες του UIDocument) για τη διαχείριση ορισμένων ή όλων των αντικειμένων του μοντέλου δεδομένων τους. Τα αντικείμενα εγγράφου δεν απαιτούνται, αλλά προσφέρουν έναν βολικό τρόπο ομαδοποίησης δεδομένων που ανήκουν σε ένα μόνο αρχείο ή πακέτο αρχείων (Hu et al., 2023).

Η προβολή αντικειμένων ελεγκτή διαχειρίζεται την παρουσίαση του περιεχομένου των εφαρμογών στην οθόνη. Ένας ελεγκτής προβολής διαχειρίζεται μια μεμονωμένη προβολή και τη συλλογή υποπροβολών του. Όταν παρουσιάζεται, ο ελεγκτής προβολής κάνει τις προβολές του ορατές εγκαθιστώντας τες στο παράθυρο εφαρμογών.

Η κλάση UIViewController είναι η βασική κλάση για όλα τα αντικείμενα του ελεγκτή προβολής. Παρέχει προεπιλεγμένη λειτουργικότητα για τη φόρτωση προβολών, την παρουσίασή τους, την

περιστροφή τους ως απόκριση στις περιστροφές της συσκευής και πολλές άλλες τυπικές συμπεριφορές συστήματος. Το UIKit και άλλα πλαίσια παρέχουν πρόσθετες κλάσεις ελεγκτή προβολής για την υλοποίηση τυπικών διεπαφών συστήματος, όπως το εργαλείο επιλογής εικόνας, τη διεπαφή της γραμμής καρτελών και τη διεπαφή πλοήγησης.

Ένα αντικείμενο UIWindow συντονίζει την παρουσίαση μιας ή περισσότερων προβολών σε μια οθόνη. Οι περισσότερες εφαρμογές έχουν μόνο ένα παράθυρο, το οποίο παρουσιάζει περιεχόμενο στην κύρια οθόνη, αλλά οι εφαρμογές μπορεί να έχουν ένα επιπλέον παράθυρο για περιεχόμενο που εμφανίζεται σε εξωτερική οθόνη. Για να αλλάξει το περιεχόμενο της εφαρμογής, γίνεται χρήση ενός ελεγκτή προβολής για να αλλάξει τις προβολές που εμφανίζονται στο αντίστοιχο παράθυρο. Ποτέ δεν γίνεται αντικατάσταση του ίδιου του παραθύρου. Εκτός από τη φιλοξενία προβολών, τα παράθυρα συνεργάζονται με το αντικείμενο UIApplication για την παράδοση συμβάντων στις προβολές και τους ελεγκτές προβολής (Hu et al., 2023).

Οι προβολές και τα στοιχεία ελέγχου παρέχουν την οπτική αναπαράσταση του περιεχομένου των εφαρμογών. Μια προβολή είναι ένα αντικείμενο που σχεδιάζει περιεχόμενο σε μια καθορισμένη ορθογώνια περιοχή και ανταποκρίνεται σε γεγονότα εντός αυτής της περιοχής. Τα στοιχεία ελέγχου είναι ένας εξειδικευμένος τύπος προβολής που είναι υπεύθυνος για την υλοποίηση οικείων αντικειμένων διεπαφής, όπως κουμπιά, τμήματα κειμένου και διακόπτες εναλλαγής. Το πλαίσιο UIKit παρέχει τυπικές προβολές για την παρουσίαση πολλών διαφορετικών τύπων περιεχομένου. Υπάρχει η δυνατότητα επίσης να οριστούν προσαρμοσμένες προβολές υποκατηγορώντας απευθείας το UIView (ή τους απογόνους του). Εκτός από την ενσωμάτωση προβολών και στοιχείων ελέγχου, οι εφαρμογές μπορούν επίσης να ενσωματώσουν επίπεδα Κινούμενου πυρήνα στις ιεραρχίες προβολής και ελέγχου τους. Τα αντικείμενα επιπέδου είναι στην πραγματικότητα αντικείμενα δεδομένων που αντιπροσωπεύουν οπτικό περιεχόμενο. Οι προβολές χρησιμοποιούν αντικείμενα επιπέδων εντατικά στο παρασκήνιο για να αποδώσουν το περιεχόμενό τους. Υπάρχει επίσης η δυνατότητα να προστεθούν αντικείμενα προσαρμοσμένου επιπέδου στη διεπαφή για να εφαρμοστούν πολύπλοκα κινούμενα σχέδια και άλλους τύπους εξελιγμένων οπτικών εφέ (Hu et al., 2023).

5.2 Ανάπτυξη εφαρμογών Android

Οι εφαρμογές Android βασίζονται κυρίως στη γλώσσα java. Γενικά, η δημιουργία μιας εφαρμογής Android απαιτεί το SDK (Κιτ ανάπτυξης λογισμικού), ένα IDE (ενσωματωμένο περιβάλλον ανάπτυξης) όπως το Android Studio ή το Eclipse, το Java Software Development Kit (JDK) και μια εικονική συσκευή για δοκιμή.

Οι εφαρμογές Android μπορούν να αναπτυχθούν με δύο διαφορετικές γλώσσες προγραμματισμού Java και Kotlin (Ardito et al., 2020).

5.2.1 Kotlin

Η Kotlin είναι μια γλώσσα προγραμματισμού με στατική πληκτρολόγηση που εκτελείται στην εικονική μηχανή Java και μπορεί επίσης να μεταγλωττιστεί σε πηγαίο κώδικα JavaScript ή να χρησιμοποιήσει τον μεταγλωττιστή LLVM. Αναπτύσσεται και υποστηρίζεται από την εταιρεία JetBrains. Αν και η σύνταξή της διαφέρει από αυτήν της Java, η υλοποίηση της Kotlin στην εικονική μηχανή Java (JVM) είναι σχεδιασμένη για να διαλειτουργεί με τον κώδικα Java και βασίζεται σε κώδικα Java από την υπάρχουσα Βιβλιοθήκη Java Class, όπως το πλαίσιο συλλογών. Τα κύρια χαρακτηριστικά της Kotlin περιλαμβάνουν (Ardito et al., 2020):

1. Συνοπτικότητα: Η Kotlin μειώνει τον αναγκαίο κώδικα boilerplate, κάνοντας τον κώδικα πιο συνοπτικό.
2. Ασφάλεια: Προσφέρει ασφάλεια στον προγραμματισμό, αποφεύγοντας κλάσεις σφαλμάτων όπως οι εξαιρέσεις μηδενικού δείκτη και άλλα.

3. Διαλειτουργικότητα: Η Kotlin είναι σχεδιασμένη για να συνεργάζεται με υπάρχουσες βιβλιοθήκες και πλατφόρμες, όπως το JVM, το Android και τον κώδικα προγράμματος περιήγησης.
4. Φιλική προς τα εργαλεία: Οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές Kotlin από οποιοδήποτε περιβάλλον ανάπτυξης Java ή από τη γραμμή εντολών.

5.2.2 Java

Η Java είναι μια πολύπλευρη γλώσσα προγραμματισμού που υποστηρίζει παράλληλη εκτέλεση, βασίζεται σε κλάσεις, είναι προσανατολισμένη στα αντικείμενα και έχει σχεδιαστεί με σκοπό να ελαχιστοποιεί τις εξαρτήσεις από την υλοποίηση. Οι εφαρμογές που γράφονται σε Java συνήθως μεταγλωττίζονται σε bytecode, που μπορεί να εκτελεστεί σε οποιαδήποτε εικονική μηχανή Java (Java virtual machine - JVM), ανεξάρτητα από την αρχιτεκτονική του υπολογιστή (Ardito et al., 2020).

Κατά τον σχεδιασμό της, η Java επιδίωξε να προσφέρει τέσσερις κύριες αρχές (Ardito et al., 2020):

Ευκολία Χρήσης: Η Java απορροφά τα βασικά στοιχεία της γλώσσας προγραμματισμού C++, καθώς ήταν πιο δύσκολη στη χρήση. Η Java βελτίωσε τη σύνταξη της γλώσσας και την έκανε πιο προσβάσιμη για τους προγραμματιστές.

Αξιοπιστία: Η Java εισήγαγε τον αντικειμενοστραφή προγραμματισμό για να μειώσει τα σφάλματα του προγραμματιστή και να ενισχύσει την αξιοπιστία των προγραμμάτων.

Ασφάλεια: Λόγω του πρωταρχικού της στόχου να υποστηρίξει κινητές συσκευές που επικοινωνούν μέσω δικτύων, η Java εστίασε στην προσθήκη υψηλού επιπέδου ασφάλειας στη γλώσσα της.

Ανεξαρτησία Πλατφόρμας: Η Java είναι πολυφορετική και δεν εξαρτάται από το λειτουργικό σύστημα ή τον τύπο του υπολογιστή στον οποίο εκτελείται, κάτι που την καθιστά ιδανική για πολλαπλές πλατφόρμες.

Συνοψίζοντας, η Java είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε με γνώμονα την ευκολία χρήσης, την αξιοπιστία, την ασφάλεια και την ανεξαρτησία πλατφόρμας, και έχει εξελιχθεί σε μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο.

5.2.3 Android Studio

Το πιο δημοφιλές εργαλείο ανάπτυξης εφαρμογών Android είναι το Android Studio. Το Android Studio είναι το επίσημο περιβάλλον ανάπτυξης (IDE) για το λειτουργικό σύστημα Android της Google, χτισμένο πάνω στο λογισμικό IntelliJ IDEA της JetBrains και σχεδιασμένο ειδικά για την ανάπτυξη εφαρμογών Android. Είναι διαθέσιμο για λήψη σε υπολογιστές με λειτουργικά συστήματα Windows, macOS και Linux. Αντικαθιστά το Eclipse Android Development Tools (ADT) ως το κύριο IDE για την ανάπτυξη εφαρμογών Android. Η εγκατάσταση του Android Studio παρέχει εργαλεία SDK και τον διαχειριστή ADB για τον συναρμολόγηση και τον έλεγχο εφαρμογών μέσω εξομοιωτών (Ribeiro, Ferreira & Mendes, 2021).

5.2.4 Άλλα εργαλεία

ADB (Android Debug Bridge): Το Android Studio περιλαμβάνει το Android Debug Bridge, που είναι ένα εργαλείο γραμμής εντολών ή μια γέφυρα επικοινωνίας μεταξύ συσκευών Android και άλλων υπολογιστών. Αυτό το εργαλείο χρησιμοποιείται κατά τη διάρκεια της ανάπτυξης και της διαδικασίας εντοπισμού σφαλμάτων και της ποιοτικής ασφάλειας. Μέσω της σύνδεσης μιας συσκευής Android στον υπολογιστή ανάπτυξης και της χρήσης εντολών τερματικού, ένας προγραμματιστής μπορεί να κάνει τις απαραίτητες τροποποιήσεις σε και τις δύο συσκευές (Sarkar et al., 2019).

Διαχειριστής AVD: Μια άλλη χρήσιμη δυνατότητα του Android Studio είναι ο Διαχειριστής AVD, ο οποίος αντιστοιχεί στην εικονική συσκευή Android. Ο Διαχειριστής AVD είναι ένας εξομοιωτής που χρησιμοποιείται για την εκτέλεση εφαρμογών Android σε έναν υπολογιστή. Αυτό επιτρέπει στους προγραμματιστές να δοκιμάσουν την απόδοση και την απόκριση σε διάφορες εκδόσεις, μεγέθη οθόνης και αναλύσεις για όλους τους τύπους συσκευών Android.

Eclipse: Όπως προαναφέρθηκε, το Eclipse χρησιμοποιούνταν ως το επίσημο IDE για την ανάπτυξη εφαρμογών Android πριν το Android Studio. Παρόλο που η Google δεν υποστηρίζει πλέον το Eclipse, πολλοί προγραμματιστές εξακολουθούν να το χρησιμοποιούν για τη δημιουργία εφαρμογών Android και πολλαπλών πλατφορμών, καθώς λειτουργεί καλά με πολλές διαφορετικές γλώσσες προγραμματισμού.

Fabric: Το Fabric είναι η πλατφόρμα πίσω από την εφαρμογή Twitter για κινητά. Παρέχει στους προγραμματιστές ένα σύνολο εργαλείων για τη βελτίωση των εφαρμογών τους για κινητά, συμπεριλαμβανομένων διαφόρων κιτ που μπορούν να επιλέξουν και να χρησιμοποιήσουν. Αυτά τα κιτ περιλαμβάνουν τα πάντα, από δοκιμές beta μέχρι εργαλεία μάρκετινγκ και διαφήμισης. Η Google αγόρασε το Fabric από το Twitter τον Ιανουάριο του 2017, και πολλές επώνυμες εταιρείες, όπως οι Uber, Spotify, Square, Groupon, Yelp και άλλες, το χρησιμοποιούν για την ανάπτυξη των κινητών εφαρμογών τους.

FlowUp: Το FlowUp επιτρέπει την παρακολούθηση της απόδοσης όλων των παραγωγικών εφαρμογών. Τα εύχρηστα πίνακες ελέγχου δίνουν τη δυνατότητα παρακολούθησης στατιστικών και μετρήσεων, όπως τη χρήση CPU και δίσκου, τη χρήση μνήμης, τα καρέ ανά δευτερόλεπτο, το εύρος ζώνης και πολλά άλλα.

Gradle: Το 2013, η Google ενέκρινε το Gradle ως σύστημα κατασκευής για εφαρμογές Android. Βασίζεται στο Apache Maven και το Apache Ant και είναι ένα από τα πιο δημοφιλή εργαλεία ανάπτυξης για τη δημιουργία μεγάλης κλίμακας εφαρμογών που χρησιμοποιούν την Java. Οι προγραμματιστές αγαπούν τη χρήση του Gradle σε συνδυασμό με το Android Studio, καθώς είναι εύκολο να προσθέσουν εξωτερικές βιβλιοθήκες με μία απλή γραμμή κώδικα.

IntelliJ IDEA: Από την ομάδα ανάπτυξης της JetBrains, το IntelliJ IDEA έχει σχεδιαστεί για να προσφέρει τη μέγιστη παραγωγικότητα στους προγραμματιστές. Είναι πολύ γρήγορο και παρέχει μια πλήρη σουίτα εργαλείων ανάπτυξης αμέσως (Sarkar et al., 2019).

RAD Studio: Το RAD Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης που επιτρέπει τη συγγραφή, μεταγλώττιση, πακετάρισμα και ανάπτυξη πολλαπλών πλατφορμών εφαρμογών. Παρέχει υποστήριξη για τον πλήρη κύκλο ζωής ανάπτυξης, επιτρέποντας τη χρήση ενός κώδικα πηγής που μπορεί να μεταγλωττιστεί εκ νέου και να διανεμηθεί ξανά.

Unity 3D: Το Unity 3D είναι ένα περιβάλλον ανάπτυξης παιχνιδιών πολλαπλών πλατφορμών που χρησιμοποιείται για τη δημιουργία προηγμένων παιχνιδιών με υψηλή γραφική ποιότητα, συμπεριλαμβανομένης της εικονικής και της επαυξημένης πραγματικότητας. Μπορεί επίσης να χρησιμοποιηθεί για τη δημιουργία απλούστερων παιχνιδιών βασισμένων σε 2D, αλλά συνήθως χρησιμοποιείται για προηγμένη ανάπτυξη παιχνιδιών.

Unreal Engine: Μια άλλη προηγμένη πλατφόρμα ανάπτυξης παιχνιδιών, η Unreal Engine είναι μια δωρεάν, ανοιχτού κώδικα λύση πολλαπλών πλατφορμών για τη δημιουργία διαδραστικών παιχνιδιών υψηλού επιπέδου.

Visual Studio με Xamarin: Το Visual Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης της Microsoft και είναι ένα δωρεάν εργαλείο για τους προγραμματιστές. Μπορεί να χρησιμοποιηθεί για τη δημιουργία εφαρμογών Windows, Android και iOS όταν συνδυάζεται με το Xamarin (Sarkar et al., 2019).

Υπάρχουν εκατοντάδες άλλα χρήσιμα εργαλεία για την ανάπτυξη Android. Κάθε προγραμματιστής έχει τις δικές του προτιμήσεις όσον αφορά τα εργαλεία και τα περιβάλλοντα ανάπτυξης που χρησιμοποιεί, ανάλογα με τη συγκεκριμένη εφαρμογή που αναπτύσσει. Καθώς η ζήτηση για εφαρμογές Android συνεχίζει να αυξάνεται, το φάσμα των πλατφορμών και των λύσεων που βοηθούν τους προγραμματιστές να εξοικονομούν χρόνο και να βελτιώνουν την ποιότητα των εφαρμογών τους θα συνεχίσει να επεκτείνεται (Sarkar et al., 2019).

5.3 Ανάπτυξη εφαρμογών iOS

Για την ανάπτυξη εφαρμογών iOS, χρειάζεστε έναν υπολογιστή Mac με την τελευταία έκδοση του Xcode. Το Xcode περιλαμβάνει όλες τις λειτουργίες που χρειάζεστε για το σχεδιασμό, την ανάπτυξη και την αποσφαλμάτωση μιας εφαρμογής. Το Xcode περιέχει επίσης το iOS SDK, το οποίο επεκτείνει το Xcode ώστε να περιλαμβάνει τα εργαλεία, τους μεταγλωττιστές και τα πλαίσια που χρειάζεστε ειδικά για την ανάπτυξη iOS. Οι γλώσσες που χρησιμοποιούνται για την ανάπτυξη εφαρμογών είναι η Swift και η Objective C (Yamacli, 2018).

5.3.1 Swift

Η Swift είναι μια γενικής χρήσης γλώσσα προγραμματισμού που δημιουργήθηκε με μια προσέγγιση που έχει έμφαση στην ασφάλεια, την απόδοση και τα πρότυπα σχεδίασης λογισμικού. Ο στόχος της Swift είναι να αποτελέσει την καλύτερη διαθέσιμη γλώσσα προγραμματισμού για μια ευρεία γκάμα εφαρμογών, από συστήματα προγραμματισμού έως κινητές εφαρμογές και υπηρεσίες στο cloud. Ο σημαντικότερος στόχος είναι να καθιστά την ανάπτυξη και τη συντήρηση λογισμικού πιο εύκολη για τους προγραμματιστές. Για να επιτευχθεί αυτό, η Swift προσανατολίζεται προς τρία κύρια χαρακτηριστικά (Yamacli, 2018):

1. Ασφάλεια: Η γλώσσα Swift έχει σχεδιαστεί με έμφαση στην ασφάλεια του κώδικα. Αυτό σημαίνει ότι η γλώσσα επιδιώκει να αποτρέπει σφάλματα και αδικαιολόγητη συμπεριφορά από τον προγραμματιστή. Αυτό συμβάλλει στη δημιουργία αξιόπιστων και ασφαλών εφαρμογών.
2. Απόδοση: Παράλληλα με την ασφάλεια, η Swift στοχεύει σε αποδοτική εκτέλεση. Οι επιδόσεις της γλώσσας πρέπει να είναι συγκρίσιμες με άλλες γλώσσες όπως η C, C++ και Objective-C, ιδίως για προγραμματισμό συστημάτων.
3. Εκφραστικότητα: Η Swift δίνει έμφαση στην ευκολία χρήσης και στην κατανόηση του κώδικα. Η συντακτική της φιλοσοφία επιδιώκει να κάνει τον κώδικα πιο ευανάγνωστο και εκφραστικό για τους προγραμματιστές.

Σε γενικές γραμμές, η Swift αποτελεί ένα προηγμένο και φιλικό προς τον προγραμματιστή εργαλείο που ενσωματώνει τα παραπάνω χαρακτηριστικά για την ανάπτυξη λογισμικού.

5.3.2 Objective C

Η Objective C είναι μια αντικειμενοστραφής γλώσσα και ως εκ τούτου, θα ήταν εύκολο για όσους έχουν κάποιο υπόβαθρο σε αντικειμενοστραφείς γλώσσες προγραμματισμού. Η Objective-C είναι η κύρια γλώσσα προγραμματισμού που χρησιμοποιείται όταν γράφει κανείς το λογισμικό για OS X και iOS. Αποτελεί υπερσύνολο της γλώσσας προγραμματισμού C και παρέχει αντικειμενοστραφείς δυνατότητες και δυναμικό χρόνο εκτέλεσης. Η Objective-C κληρονομεί τη σύνταξη, τους πρωτόγονους τύπους και τις εντολές ελέγχου OW της C και προσθέτει σύνταξη για την ονομασία κλάσεων και μεθόδων. Προσθέτει επίσης υποστήριξη σε επίπεδο γλώσσας για τη διαχείριση του γράφου αντικειμένων και τα γραμματικά αντικείμενα, ενώ παρέχει δυναμική τυποποίηση και δέσμευση, αναβάλλοντας πολλές ευθύνες μέχρι το χρόνο εκτέλεσης (Kochan, 2011).

5.3.3 Xcode

Το εργαλείο που χρησιμοποιείται για τη δημιουργία εφαρμογών iOS είναι το Xcode. Το Xcode είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) για macOS που περιέχει μια σουίτα εργαλείων ανάπτυξης λογισμικού που αναπτύχθηκε από την Apple για την ανάπτυξη λογισμικού για macOS, iOS, watchOS και tvOS. Κυκλοφόρησε για πρώτη φορά το 2003, η τελευταία σταθερή έκδοση είναι η έκδοση 10.1 και διατίθεται δωρεάν μέσω του Mac App Store για τους χρήστες των macOS High Sierra και macOS Mojave. Οι εγγεγραμμένοι προγραμματιστές μπορούν να κατεβάσουν εκδόσεις προεπισκόπησης και προηγούμενες εκδόσεις της σουίτας μέσω του ιστότοπου Apple Developer (Yamacli, 2018).

Το Xcode υποστηρίζει πηγαίο κώδικα για τις γλώσσες προγραμματισμού C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez) και Swift, με μια ποικιλία μοντέλων προγραμματισμού, συμπεριλαμβανομένων, μεταξύ άλλων, των Cocoa, Carbon και Java. Τρίτα μέρη έχουν προσθέσει υποστήριξη για GNU Pascal Free Pascal, Ada, C#,Perl, D και Fortran.

Το Xcode μπορεί να δημιουργήσει λιπαρά δυαδικά αρχεία που περιέχουν κώδικα για πολλαπλές αρχιτεκτονικές με την εκτελέσιμη μορφή Mach-O. Αυτά ονομάζονται καθολικά δυαδικά αρχεία, τα οποία επιτρέπουν στο λογισμικό να εκτελείται τόσο σε πλατφόρμες PowerPC όσο και σε πλατφόρμες Intel (x86) και μπορεί να περιλαμβάνει κώδικα 32 bit και 64 bit και για τις δύο αρχιτεκτονικές. Χρησιμοποιώντας το iOS SDK, το Xcode μπορεί επίσης να

χρησιμοποιηθεί για τη μεταγλώττιση και την αποσφαλμάτωση εφαρμογών για iOS που εκτελούνται σε επεξεργαστές αρχιτεκτονικής ARM.

Το Xcode περιλαμβάνει το εργαλείο GUI Instruments, το οποίο εκτελείται πάνω σε ένα πλαίσιο δυναμικής ανίχνευσης, το DTrace, που δημιουργήθηκε από τη Sun Microsystems και κυκλοφόρησε ως μέρος του OpenSolaris (Yamacli, 2018).

5.4 Ανάπτυξη εφαρμογών για πολλαπλές πλατφόρμες

Η ανάπτυξη κινητών συσκευών σε πολλαπλές πλατφόρμες είναι η δημιουργία εφαρμογών λογισμικού που είναι συμβατές με πολλαπλά λειτουργικά συστήματα κινητών συσκευών. Αρχικά, η πολυπλοκότητα της ανάπτυξης εφαρμογών για κινητά επιδεινωνόταν από τη δυσκολία δημιουργίας ενός backend που να λειτουργεί σε πολλαπλές πλατφόρμες. Αν και ήταν χρονοβόρο και ακριβό, ήταν συχνά ευκολότερο να δημιουργηθούν εγγενείς εφαρμογές για κάθε λειτουργικό σύστημα (OS) κινητής τηλεφωνίας. Το πρόβλημα ήταν ότι ο κώδικας που κατασκευάστηκε για ένα λειτουργικό σύστημα δεν μπορούσε να επαναχρησιμοποιηθεί για άλλο λειτουργικό σύστημα (Martinez & Lecomte, 2017).

Αναπτύχθηκαν εργαλεία ανάπτυξης κινητών συσκευών πολλαπλών πλατφορμών με σκοπό να τους δοθεί η δυνατότητα να γράφουν τον πηγαίο κώδικα της εφαρμογής μία φορά και να τον εκτελούν σε διαφορετικά λειτουργικά συστήματα (Martinez & Lecomte, 2017).

Τα σημαντικότερα πλεονεκτήματα που έχουν επιφέρει αυτά τα εργαλεία είναι τα εξής:

- Μείωση των απαιτούμενων δεξιοτήτων των προγραμματιστών για την ανάπτυξη εφαρμογών λόγω της χρήσης κοινών γλωσσών προγραμματισμού.
- Μείωση της κωδικοποίησης, επειδή ο πηγαίος κώδικας γράφεται μία φορά και μεταγλωττίζεται για κάθε υποστηριζόμενο λειτουργικό σύστημα.
- Μείωση του χρόνου ανάπτυξης και του μακροπρόθεσμου κόστους συντήρησης,
- Μείωση των γνώσεων API, επειδή με αυτά τα εργαλεία.
- Μεγαλύτερη ευκολία ανάπτυξης σε σύγκριση με τη δημιουργία εγγενών εφαρμογών για κάθε λειτουργικό σύστημα.

Οι προσεγγίσεις για την ανάπτυξη διαπλατφορμών περιλαμβάνουν (Martinez & Lecomte, 2017):

1. Υβριδική ανάπτυξη εφαρμογών για κινητά: οι προγραμματιστές γράφουν τον πυρήνα της εφαρμογής ως εφαρμογή HTML5 ή JavaScript για κινητά και στη συνέχεια τοποθετούν ένα περιτύλιγμα για εγγενείς συσκευές γύρω από αυτήν.
2. Ταχεία ανάπτυξη εφαρμογών για κινητά (Rapid mobile app development - RMAD): οι προγραμματιστές χρησιμοποιούν εργαλεία προγραμματισμού χωρίς κώδικα. Το RMAD προσφέρει στους επιχειρηματικούς χρήστες τη δυνατότητα να δημιουργούν και να διαχειρίζονται γρήγορα επαρκώς καλές εσωτερικές εφαρμογές για την αντιμετώπιση συγκεκριμένων επιχειρηματικών ζητημάτων.
3. Καθολικές εφαρμογές Windows: μία βάση κώδικα για όλες τις συσκευές Windows. Ο στόχος είναι να καταστεί δυνατή η εκτέλεση της ίδιας εφαρμογής σε υπολογιστή με Windows, tablet, smartphone, smartwatch ή XBox.
4. Προοδευτικές εφαρμογές ιστού (Progressive web apps - PWA): ιστότοποι που μοιάζουν και συμπεριφέρονται σαν να είναι εφαρμογές για κινητά. Οι PWAs κατασκευάζονται για να εκμεταλλεύονται τα εγγενή χαρακτηριστικά των κινητών συσκευών, χωρίς να απαιτείται από τον τελικό χρήστη να επισκεφθεί ένα κατάστημα εφαρμογών, να προβεί σε αγορά και να κατεβάσει λογισμικό τοπικά.

Τα πιο δημοφιλή εργαλεία ανάπτυξης διασταυρούμενων πλατφορμών είναι: Xamarin, PhoneGap, Sencha, Appcelerator, iFactr, Kony, AlphaAnywhere, Redhat. Εν πάση περιπτώσει, η ανάπτυξη διασταυρούμενων πλατφορμών παρουσιάζει ορισμένους περιορισμούς. Αν και οι εφαρμογές για κινητά με διασταυρούμενη πλατφόρμα είναι ταχύτερες και φιλικότερες, αλλά έχουν μικρότερη ποιότητα απόδοσης σε σύγκριση με τις εγγενείς εφαρμογές. Δεδομένου ότι κάθε εφαρμογή σχεδιάζεται για μία ή περισσότερες πλατφόρμες λειτουργικού συστήματος, είναι δύσκολο να υποστηρίζεται από κάθε χαρακτηριστικό του λειτουργικού συστήματος. Κάθε φορά που το λειτουργικό σύστημα ενημερώνεται με νέα χαρακτηριστικά, η εφαρμογή χρειάζεται επίσης ενημέρωση. Για την εκκίνηση μιας εφαρμογής cross platform πρέπει να δημιουργηθεί μια σύνδεση, κάτι που δεν απαιτείται με τις εγγενείς εφαρμογές. Οι κινητές εφαρμογές cross

platform έχουν περιορισμούς σε ορισμένα από τα χαρακτηριστικά του υλικού, καθώς αυτές οι εφαρμογές έρχονται με πολλαπλές πλατφόρμες κινητών λειτουργικών συστημάτων. Πολλά πλαίσια κατασκευάζονται με τη χρήση Javascript και, ως εκ τούτου, αν πρέπει να γίνει μετακίνηση από ένα λειτουργικό σύστημα σε άλλο, ο κώδικας που έχει ήδη χρησιμοποιηθεί δεν πρόκειται να λειτουργήσει. Μπορεί να γίνει επαναχρησιμοποιήσιμος βάζοντας πολλή δουλειά στους κώδικες. Οι υβριδικές εφαρμογές έρχονται με χαρακτηριστικά και περιορισμούς που θα μπορούσαν είτε να ενθουσιάσουν είτε να απογοητεύσουν κάθε προγραμματιστή (Martinez & Lecomte, 2017).

5.5 Javascript

Η JavaScript είναι μια υψηλού επιπέδου, ερμηνευμένη γλώσσα προγραμματισμού που είναι κυρίως γνωστή για τη χρήση της στην ανάπτυξη Ιστού. Είναι μια από τις βασικές τεχνολογίες για τη δημιουργία διαδραστικών και δυναμικών ιστοσελίδων. Ακολουθούν ορισμένες βασικές πτυχές της JavaScript (Ardito et al., 2020):

Σενάρια από την πλευρά του πελάτη: Η JavaScript χρησιμοποιείται πιο συχνά ως γλώσσα προγραμματισμού από την πλευρά του πελάτη, που εκτελείται στο πρόγραμμα περιήγησης ιστού ενός χρήστη. Επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικό περιεχόμενο, να χειρίζονται το μοντέλο αντικειμένου εγγράφου (DOM), να χειρίζονται συμβάντα και να αλληλοεπιδρούν με τους χρήστες.

Πρότυπο ECMAScript: Η JavaScript βασίζεται στο πρότυπο ECMAScript, το οποίο καθορίζει τις προδιαγραφές της γλώσσας δέσμης ενεργειών. Το ECMAScript διατηρείται από τον οργανισμό προτύπων Ecma International.

Δυναμική πληκτρολόγηση: Η JavaScript πληκτρολογείται δυναμικά, πράγμα που σημαίνει ότι οι τύποι μεταβλητών καθορίζονται κατά το χρόνο εκτέλεσης. Αυτό παρέχει ευελιξία, αλλά απαιτεί επίσης ιδιαίτερη προσοχή στους μεταβλητούς τύπους κατά την ανάπτυξη.

Αντικειμενοστραφής προγραμματισμός που βασίζεται σε πρωτότυπα: Η JavaScript είναι μια αντικειμενοστραφή γλώσσα, αλλά χρησιμοποιεί ένα μοντέλο που βασίζεται σε πρωτότυπα για κληρονομικότητα. Τα αντικείμενα μπορούν να χρησιμεύσουν ως πρωτότυπα για άλλα αντικείμενα, επιτρέποντας μια ευέλικτη και δυναμική προσέγγιση στη δημιουργία αντικειμένων.

Ασύγχρονος προγραμματισμός: Η JavaScript υποστηρίζει ασύγχρονο προγραμματισμό μέσω λειτουργιών όπως επανακλήσεις, υποσχέσεις και ασυγχρονισμός/αναμονή. Αυτό είναι ζωτικής σημασίας για το χειρισμό εργασιών όπως η ανάκτηση δεδομένων από διακομιστές ή ο χειρισμός των αλληλεπιδράσεων των χρηστών χωρίς να εμποδίζεται η εκτέλεση άλλου κώδικα.

Προγραμματισμός βάσει συμβάντων: Πολλές ενέργειες στο JavaScript ξεκινούν από συμβάντα όπως κλικ κουμπιών, υποβολές φορμών ή χρονόμετρα. Η JavaScript επιτρέπει στους προγραμματιστές να ανταποκρίνονται σε αυτά τα συμβάντα εκτελώντας συγκεκριμένα κομμάτια κώδικα.

Συμβατότητα μεταξύ προγραμμάτων περιήγησης: Η JavaScript έχει σχεδιαστεί για να λειτουργεί σε διαφορετικά προγράμματα περιήγησης ιστού, παρέχοντας μια συνεπή εμπειρία στους χρήστες ανεξάρτητα από το πρόγραμμα περιήγησης που χρησιμοποιούν.

Βιβλιοθήκες και πλαίσια: Υπάρχουν πολλές βιβλιοθήκες και πλαίσια χτισμένα πάνω από JavaScript που απλοποιούν και βελτιώνουν τη διαδικασία ανάπτυξης. Στα παραδείγματα περιλαμβάνονται τα jQuery, React, Angular και Vue.js.

Ανάπτυξη από την πλευρά του διακομιστή: Με την εισαγωγή περιβαλλόντων όπως το Node.js, η JavaScript μπορεί πλέον να χρησιμοποιηθεί και για ανάπτυξη από την πλευρά του διακομιστή. Αυτό επιτρέπει στους προγραμματιστές να χρησιμοποιούν την ίδια γλώσσα τόσο στην πλευρά του πελάτη όσο και του διακομιστή μιας εφαρμογής Ιστού.

JSON (JavaScript Object Notation): Η JavaScript συνδέεται στενά με το JSON, μια ελαφριά μορφή ανταλλαγής δεδομένων. Το JSON χρησιμοποιείται συχνά για τη μετάδοση δεδομένων μεταξύ ενός διακομιστή και μιας εφαρμογής Ιστού.

Η JavaScript είναι μια ευέλικτη γλώσσα που έχει εξελιχθεί με τα χρόνια, επεκτείνοντας τις δυνατότητές της πέρα από το πρόγραμμα περιήγησης ιστού. Πλέον χρησιμοποιείται σε

διάφορους τομείς εφαρμογών, συμπεριλαμβανομένης της ανάπτυξης από την πλευρά του διακομιστή, της ανάπτυξης εφαρμογών για κινητά, ακόμη και στη δημιουργία εφαρμογών για επιτραπέζιους υπολογιστές (Ardito et al., 2020).

5.6 MongoDB

Το MongoDB είναι ένα δημοφιλές, ανοιχτού κώδικα NoSQL (μη σχεσιακό) σύστημα διαχείρισης βάσεων δεδομένων που παρέχει υψηλή απόδοση, υψηλή διαθεσιμότητα και εύκολη επεκτασιμότητα. Αποθηκεύει δεδομένα σε μια ευέλικτη μορφή JSON που ονομάζεται BSON (Binary JSON). Το MongoDB έχει σχεδιαστεί για να χειρίζεται μεγάλους όγκους δεδομένων και χρησιμοποιείται συχνά σε εφαρμογές όπου τα δεδομένα δεν είναι δομημένα ή ημιδομημένα (Banker et al., 2016).

Τα βασικά χαρακτηριστικά του MongoDB περιλαμβάνουν:

Προσανατολισμός εγγράφων: Το MongoDB αποθηκεύει δεδομένα σε ευέλικτα έγγραφα τύπου JSON που ονομάζονται BSON. Κάθε έγγραφο μπορεί να έχει διαφορετική δομή, καθιστώντας εύκολη την αναπαράσταση πολύπλοκων ιεραρχικών σχέσεων.

Χωρίς σχήμα: Σε αντίθεση με τις παραδοσιακές σχεσιακές βάσεις δεδομένων, το MongoDB είναι χωρίς σχήμα, που σημαίνει ότι μπορείτε να προσθέσετε ή να αφαιρέσετε πεδία από έγγραφα χωρίς να επηρεαστεί ολόκληρη η βάση δεδομένων. Αυτή η ευελιξία είναι ιδιαίτερα χρήσιμη σε καταστάσεις όπου το σχήμα δεδομένων εξελίσσεται με την πάροδο του χρόνου.

Ευρετηρίαση: Το MongoDB υποστηρίζει την ευρετηρίαση, η οποία επιτρέπει την αποτελεσματική αναζήτηση δεδομένων. Τα ευρετήρια μπορούν να δημιουργηθούν σε οποιοδήποτε πεδίο ενός εγγράφου, καθιστώντας δυνατή τη γρήγορη ανάκτηση και ανάλυση δεδομένων.

Γλώσσα ερωτήματος: Η MongoDB χρησιμοποιεί μια πλούσια γλώσσα ερωτημάτων που υποστηρίζει ένα ευρύ φάσμα ερωτημάτων, συμπεριλαμβανομένων των ερωτημάτων πεδίου, των ερωτημάτων εύρους και των αναζητήσεων κανονικών εκφράσεων. Υποστηρίζει επίσης πλαίσιο συγκέντρωσης για πολύπλοκες εργασίες επεξεργασίας δεδομένων.

Επεκτασιμότητα: Το MongoDB έχει σχεδιαστεί για οριζόντια κλίμακα, που σημαίνει ότι μπορείτε να προσθέσετε περισσότερους διακομιστές για τη διανομή του φορτίου και τη διαχείριση αυξανόμενων ποσοτήτων δεδομένων. Αυτό επιτυγχάνεται μέσω λειτουργιών όπως η κοινή χρήση, η οποία επιτρέπει τη διανομή δεδομένων σε πολλούς διακομιστές.

Υψηλή διαθεσιμότητα: Το MongoDB παρέχει δυνατότητες για αναπαραγωγή και αυτόματη ανακατεύθυνση, διασφαλίζοντας ότι τα δεδομένα παραμένουν διαθέσιμα ακόμη και σε περίπτωση αστοχίας διακομιστή.

Ερωτήματα Ad Hoc: Το MongoDB υποστηρίζει ad-hoc ερωτήματα, επιτρέποντας την ανάλυση δεδομένων σε πραγματικό χρόνο. Αυτό είναι ιδιαίτερα χρήσιμο σε περιπτώσεις όπου το μοντέλο δεδομένων δεν είναι καλά καθορισμένο εκ των προτέρων.

Πλαίσιο συγκέντρωσης: Το πλαίσιο συγκέντρωσης του MongoDB επιτρέπει την επεξεργασία και μετασχηματισμό δεδομένων εντός της βάσης δεδομένων. Υποστηρίζει λειτουργίες όπως φιλτράρισμα, ομαδοποίηση, ταξινόμηση και προβολή, καθιστώντας το ισχυρό για αναλυτικά στοιχεία.

Το MongoDB χρησιμοποιείται συνήθως σε διάφορους τύπους εφαρμογών, συμπεριλαμβανομένων εφαρμογών ιστού και κινητών, συστημάτων διαχείρισης περιεχομένου και αναλυτικών στοιχείων σε πραγματικό χρόνο. Είναι κατάλληλο για σενάρια όπου οι δομές δεδομένων είναι δυναμικές και εξελισσόμενες και υπάρχει ανάγκη για οριζόντια επεκτασιμότητα και υψηλή απόδοση (Banker et al., 2016).

6. Δημιουργία εξατομικευμένης εφαρμογής

Στο υλοποιητικό σκέλος της εργασίας παρακάτω, προχωρούμε στην δημιουργία μιας εξατομικευμένης διαδικτυακής (web) εφαρμογής (app) με σκοπό την ανάδειξη των UML διαγραμμάτων.

Προαπαιτούμενα για να τρέξει το app:

1. Εγκατάσταση Node.js
2. Εγκατάσταση το VSCode IDE

Χρησιμοποιήθηκε το stack MERN , το οποίο αποτελείται από MongoDB για τη βάση δεδομένων, Node.js και Express.js για την ανάπτυξη του back end και React.js για την ανάπτυξη του front end.Για το styling , χρησιμοποιήθηκε η βιβλιοθήκη TailwindCSS. Η ανάπτυξη έγινε με χρήση VSCode IDE.

Η δομή των αρχείων ακολουθεί το μοτίβο MVC όπως φαίνεται στην παρακάτω αναπαράσταση (περιέχει μόνο μερικά από τα αρχεία του project προκειμένου να παρουσιαστεί η δομή). Ένα έργο MVC αποτελείται από τρία κύρια μέρη, τα Models, τις Views και τους Controllers. Το V αντιπροσωπεύει τα Views, τα οποία αποτελούν τα front end components της εφαρμογής, τις σελίδες που βλέπει ο χρήστης στο πρόγραμμα περιήγησής του (React.js στην προκειμένη περίπτωση). Το M αντιπροσωπεύει τα Models, κλάσεις που χρησιμοποιούνται για την περιγραφή των κύριων οντοτήτων του έργου, οι οποίες χρησιμοποιούνται για την αποθήκευση δεδομένων στη βάση δεδομένων. Τέλος το C, αντιπροσωπεύει τους Controllers, το back end του έργου, αυτά τα αρχεία περιέχουν την επιχειρησιακή λογική, χειρίζονται τα αιτήματα που έρχονται από το front end, φέρνουν δεδομένα από τη βάση δεδομένων για να απαντήσουν στο χρήστη. Η δομή των αρχείων του έργου είναι η εξής:

root

```
├── client/  # Front end directory
│   ├── public/
│   │   └── index.html
│   ├── src/
│   │   ├── partials/  # Reusable UI components
│   │   │       ├── NavBar.js
│   │   ├── views/  # Pages components
│   │   │       ├── Login.js
│   │   │       └── Home.js
│   │   └── App.js # Main React file
│   └── tailwind.config.js # Configuration file for front end styling library
├── models/ # Database models
│   └── userModels.js
├── controllers/ # Business logic handlers
│   └── userControllers.js
├── routes/ # Express route definitions
│   └── userRoutes.js
├── middlewares/ # Express middlewares ( authentications , etc )
│   └── authMiddleware.js
├── server.js # Server setup, where Express app starts
├── package.json
└── .env # For configuration files like database configuration, jwt, etc.
```

Αφού εγκατασταθούν όλες οι απαραίτητες βιβλιοθήκες, για να συνεχιστεί η ανάπτυξη πρέπει να πραγματοποιηθεί:

Server Setup: Δημιουργία server.js αρχείου στο root directory του project. Αυτό το αρχείο περιέχει τη βασική ρύθμιση του διακομιστή express, ξεκινάει τον διακομιστή σε μια συγκεκριμένη θύρα και συνδέει τον διακομιστή με τη βάση δεδομένων.

```
const express = require('express');
const mongoose = require('mongoose');
const dotenv = require('dotenv');
const cors = require('cors');

// Import routes
const userRoutes = require('./routes/userRoutes');
const postRoutes = require('./routes/postRoutes');
const tripRoutes = require('./routes/tripRoutes');
const destinationRoutes = require('./routes/destinationRoutes');

// Initialize dotenv to use environment variables
dotenv.config();

// Create the Express application
const app = express();
// CORS middleware This should come before your route definitions
app.use(cors());
//  JSON middleware
app.use(express.json());


// MongoDB connection
mongoose.connect(process.env.MONGODB_URI)
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('Could not connect to MongoDB:', err));

// Use the user routes
app.use(userRoutes);
app.use(postRoutes);
app.use(destinationRoutes);
app.use(tripRoutes);

// Start the server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Το dotenv που αναφέρεται στο παραπάνω αρχείο server.js , είναι ένα αρχείο που χρησιμοποιείται για την αποθήκευση ευαίσθητων μεταβλητών.

- Define Models: Τα μοντέλα δημιουργούνται στον φάκελο /models. Όπως έχει ήδη εξηγηθεί, τα μοντέλα είναι ουσιαστικά κλάσεις, αναπαριστούν οντότητες μέσα στην εφαρμογή και χρησιμοποιούνται για την αποθήκευση δεδομένων. Παρακάτω είναι ο κώδικας του μοντέλου user που είναι το βασικό μοντέλο του project. Με τον ίδιο τρόπο δημιουργήσαμε τα μοντέλα για το Trip, Post και Destination.

```
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  destinations: [String],
  accommodationPreferences: [String],
  prefferedMonths: [String],
  prefferedWeather: [String],
  activities: [String],
  followers: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
  following: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }],
  posts:[String]
});

userSchema.pre('save', async function (next) {
  if (this.isModified('password')) {
     this.password = await bcrypt.hash(this.password, 8);
  }
  next();
});


module.exports = mongoose.model('User', userSchema);
```

Η βιβλιοθήκη bcrypt χρησιμοποιείται για την κρυπτογράφηση πληροφοριών όπως οι κωδικοί πρόσβασης των χρηστών.

- Define Routes: Οι Routes δημιουργούνται στον φάκελο /routes. Οι Routes αποτελούν μέρος του back end τμήματος του έργου και χρησιμοποιούνται σε συνδυασμό με τους Controllers. Χρησιμοποιούνται για την προώθηση των διαφόρων request που προέρχονται από τον client στον αντίστοιχο Controller που μπορεί να δώσει απάντηση. Οι Routes χρησιμοποιούνται επίσης για τον έλεγχο ταυτότητας, την εξουσιοδότηση κ.λπ. πριν το αίτημα του client αρχίσει να επεξεργάζεται η επιχειρησιακή λογική του έργου.

Παρακάτω είναι ο κώδικας του Route χρήστη (userRoutes.js):

```
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userControllers');
const authMiddleware = require('../middleware/authMiddleware');

// POST /register - Register a new user
router.post('/register', userController.registerUser);
```

```
// POST /login - Log in a user and return a token
router.post('/login', userController.loginUser);

// GET /userProfile - Get the user profile for the logged-in user
router.get('/userProfile', authMiddleware, userController.getUserProfile);

router.put('/userProfile', authMiddleware, userController.updateUserProfile);

router.get('/searchUsers', authMiddleware, userController.getUsersByName);

router.get('/viewProfile/:userId', authMiddleware, userController.getSingleUserById);

router.post('/followUnfollow/:userId', authMiddleware, userController.followUnfollowUser);

module.exports = router;
```

Όπως μπορεί να δει κανείς, η διαδρομή προωθεί τις αιτήσεις HTTP get,post,put στην αντίστοιχη function του αντίστοιχου αρχείου του controller. Χρησιμοποιεί επίσης το αρχείο authMiddleware για τον έλεγχο ταυτότητας του χρήστη.

- Define Controllers: Οι Controllers είναι υπεύθυνοι για τον χειρισμό των εισερχόμενων αιτήσεων, την αλληλεπίδραση με τα μοντέλα και την επιστροφή των απαντήσεων στον πελάτη. Αυτοί χειρίζονται την επιχειρησιακή λογική για διάφορες διαδρομές (π.χ. δημιουργία προφίλ χρήστη, άντληση προτάσεων προορισμού). Όπως και προηγουμένως εδώ είναι το αρχείο userController για να το παρουσιαστεί ως παράδειγμα:

```
const jwt = require('jsonwebtoken');
const bcrypt = require('bcrypt');
const User = require('../models/userModels');

const registerUser = async (req, res) => {...};

const loginUser = async (req, res) => {...};

const deleteUser = async (req, res) => {...};

const getUserProfile = async (req, res) => {...};

const updateUserProfile = async (req, res) => {...};

const getUsersByName = async (req, res) => {...};

const getSingleUserByName = async (req, res) => {...};

const getSingleUserById = async (req, res) => {...};
```

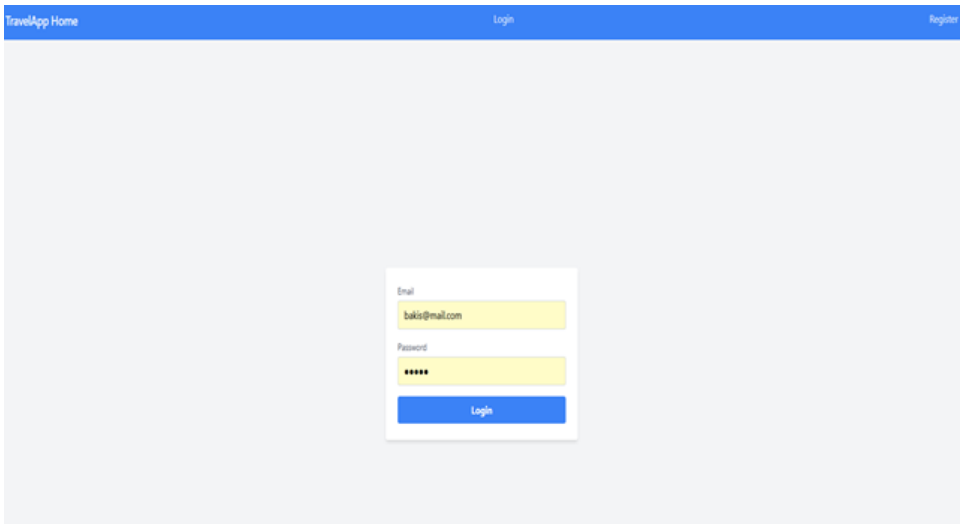
```
const followUnfollowUser = async (req, res) => {...};

module.exports = {
    registerUser,
    loginUser,
    getUserProfile,
    updateUserProfile,
    getUsersByName,
    getSingleUserByName,
    followUnfollowUser,
    getSingleUserById
  };
```

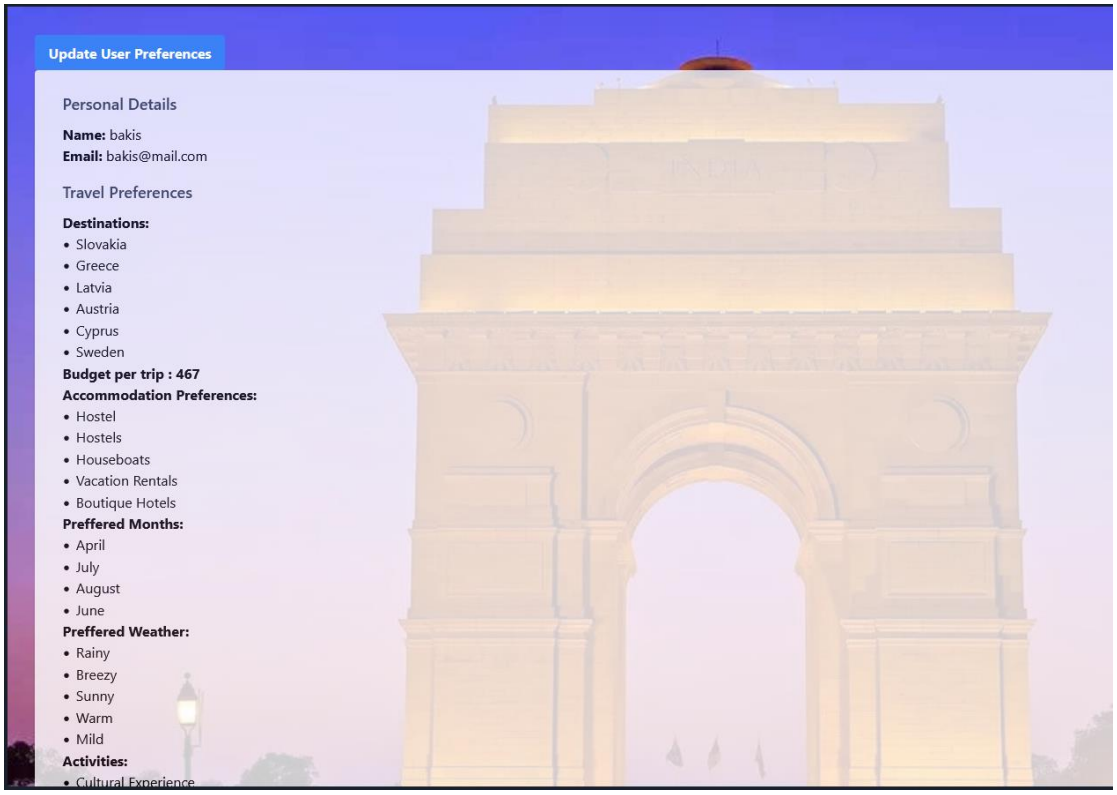
Όπως μπορεί να δει κανείς, το αρχείο userController περιέχει διάφορες functions που εκτελούν διάφορες ενέργειες προκειμένου να χειριστούν τα διάφορα αιτήματα που αποστέλλονται από τον client και προωθούνται από τις διαδρομές. Ο χρήστης πρέπει να ανοίξει το πηγαίο αρχείο σε έναν επεξεργαστή κειμένου ή IDE για να εξετάσει τη λογική μέσα σε κάθε συνάρτηση.

- Create React Components: Μέσα στο φάκελο src πραγματοποιείται η δημιουργία των React components του έργου. Εντός του src υπάρχουν δύο φάκελοι, ο partials και ο views. Το Partials χρησιμοποιείται για επαναχρησιμοποιήσιμα στοιχεία του front end όπως κουμπιά, μπάρες αναζήτησης, κλπ.

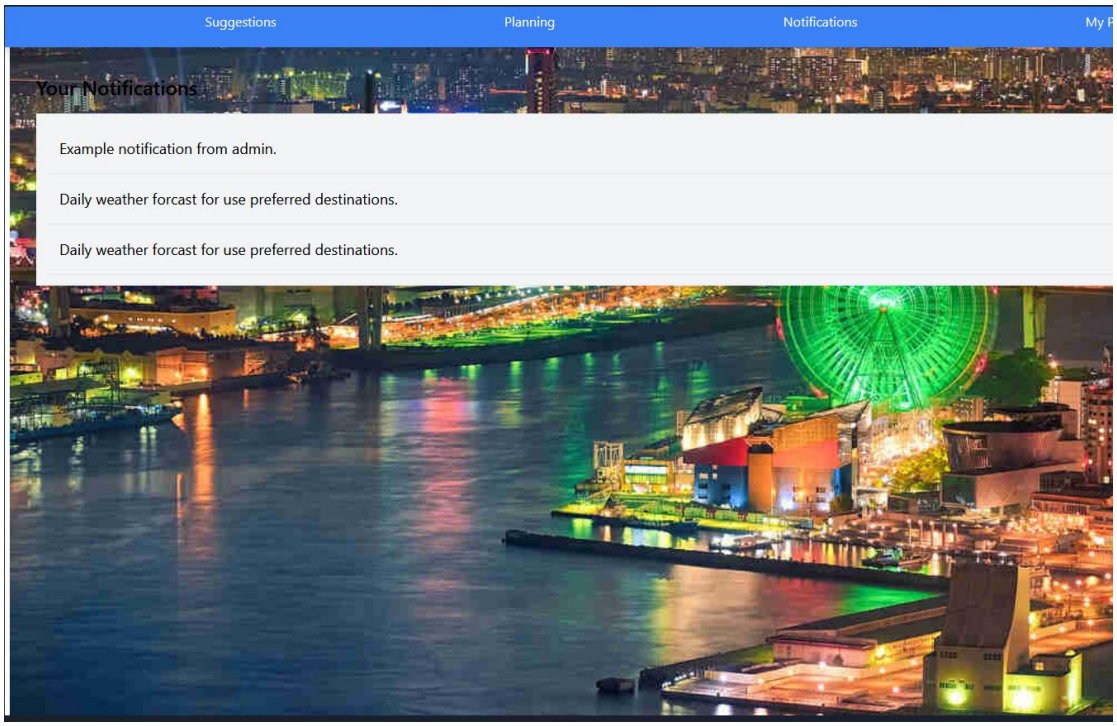
Η προβολή της οθόνης σύνδεσης, οι υπόλοιπες σελίδες της εφαρμογής και ο κώδικας του front-end παρουσιάζονται παρακάτω:



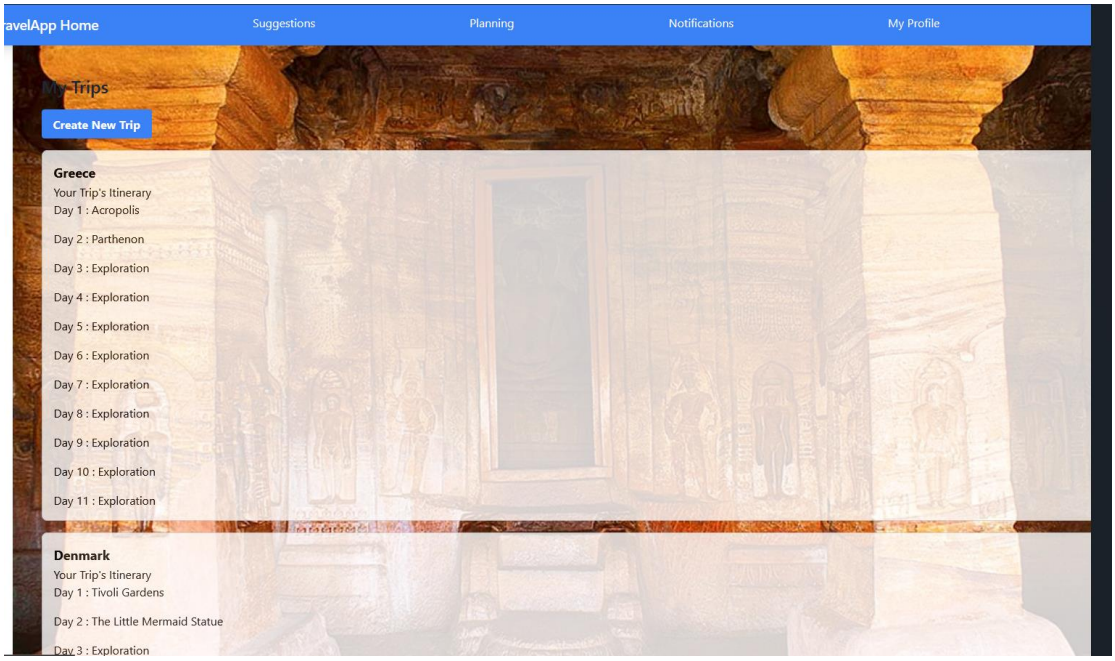
Εικόνα 1: Η προβολή της οθόνης σύνδεσης και ο κώδικας του front end.



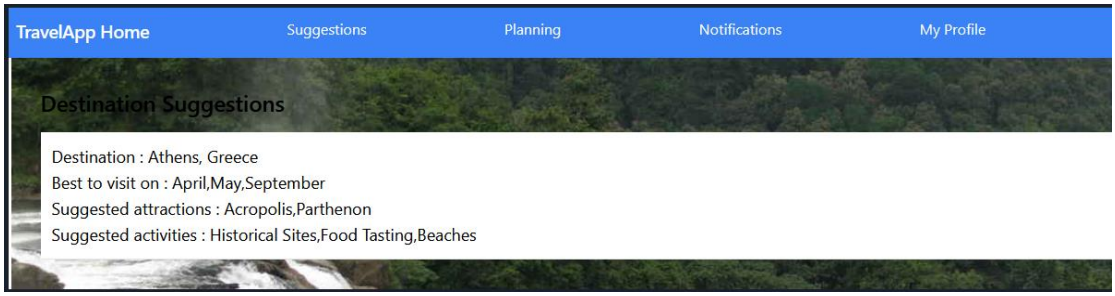
Εικόνα 2: Σελίδα της εφαρμογής.



Εικόνα 3: Σελίδα της εφαρμογής.



Εικόνα 4: Σελίδα της εφαρμογής.



Εικόνα 5: Σελίδα της εφαρμογής.

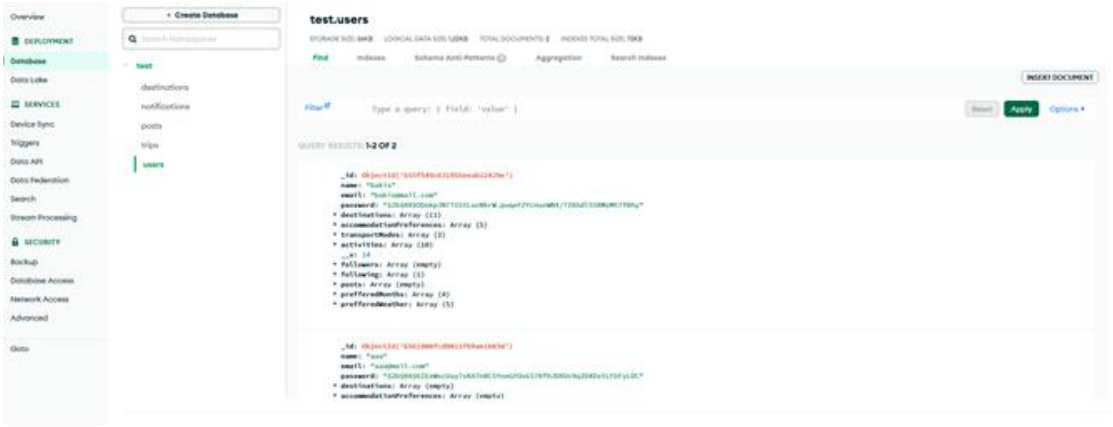
```
import axios from 'axios';
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

const Login = () => {
  const [userData, setUserData] = useState({ email: '', password: '' });
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://localhost:5000/login', userData);
      localStorage.setItem('token', response.data.token);
      localStorage.setItem('isLoggedIn', 'true'); // Set the login flag
      navigate('/home');
```

```
    } catch (error) {
      console.error('Error:', error);
    }
  };
  return (
    <div className="flex justify-center items-center h-screen bg-gray-100">
      <div className="p-6 max-w-sm w-full bg-white rounded shadow-md">
        <form onSubmit={handleSubmit} className="space-y-4">
          <div>
            <label className="block text-sm font-medium text-gray-700">
                                Email</label>
            <input type="email" placeholder="Email" value={userData.email} onChange={e =>
setUserData({ ...userData, email: e.target.value })}
              className="mt-1 px-3 py-2 border border-gray-300 rounded w-full" />
          </div>
          <div>
            <label className="block text-sm font-medium text-gray-700">Password</label>
            <input type="password" placeholder="Password" value={userData.password}
onChange={e => setUserData({ ...userData, password: e.target.value })}
              className="mt-1 px-3 py-2 border border-gray-300 rounded w-full" />
          </div>
          <button type="submit" className="w-full bg-blue-500 hover:bg-blue-700 text-white font-
bold py-2 px-4 rounded">
            Login
          </button>
        </form>
      </div>
    </div>
  );
};
export default Login;
```

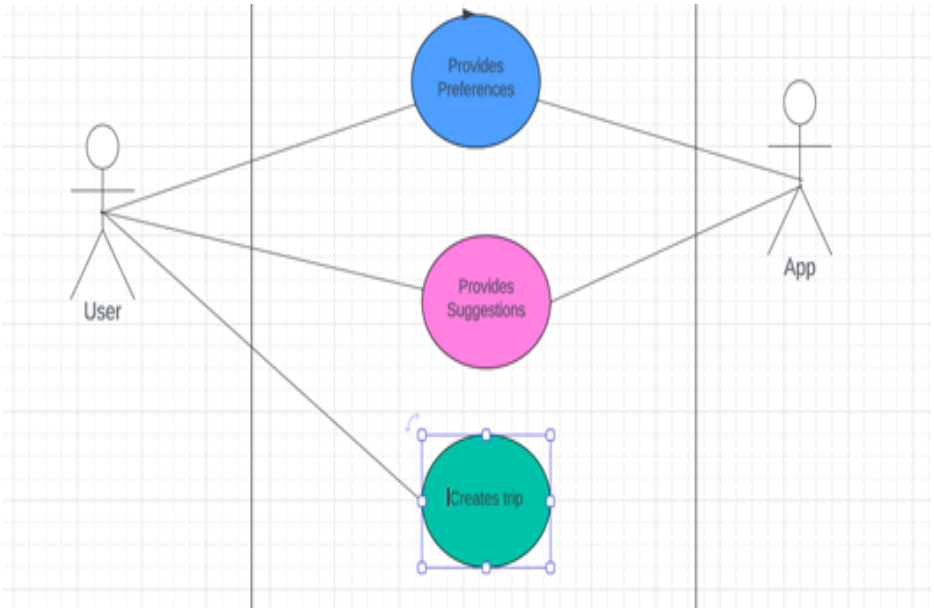
- Create Database: Χρησιμοποιήθηκε το MongoDB Atlas. Με τη χρήση δεν χρειάζεται να εγκαταστήσουμε το mongoDb στο σύστημα που φιλοξενεί το app. Αντ' αυτού υπάρχει η δυνατότητα δημιουργίας και να υπάρχει πρόσβαση σε μια βάση δεδομένων MongoDB στο cloud χρησιμοποιώντας το Atlas, απλά πρέπει να διατηρηθεί το MONGODB_URI στις μεταβλητές enviroment μέσα στο αρχείο .env.


Overview
DEPLOYMENT
Database
Data Lake
SERVICES
Device Sync
Triggers
Data API
Data Federation
Search
Stream Processing
SECURITY
Backup
Database Access
Network Access
Advanced
Goto
+ Create Database
test
destinations
notifications
posts
trips
users
test.users
Find
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
1-2 OF 2

6.1 Διαγράμματα UML (usecase, activity, class, sequence)

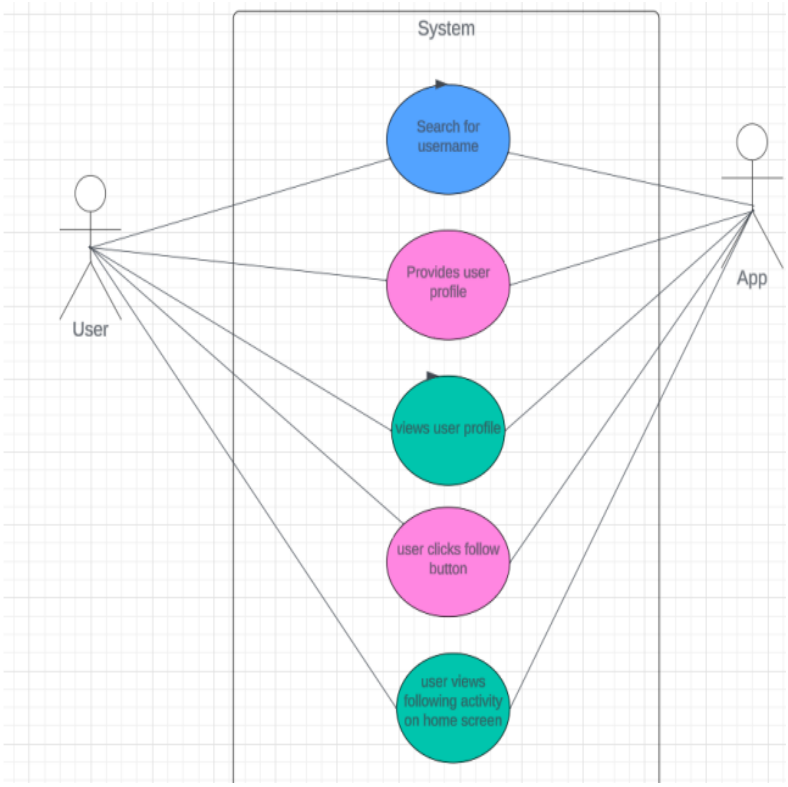
6.1.1 Use Case Diagram

Αυτό το διάγραμμα θα προσδιορίζει τους διάφορους φορείς (χρήστες, διαχειριστές, κοινωνικά δίκτυα) και τις αντίστοιχες περιπτώσεις χρήσης εντός της εφαρμογής. Παραδείγματα περιπτώσεων χρήσης μπορεί να περιλαμβάνουν τη δημιουργία προφίλ, τον προγραμματισμό ενός ταξιδιού, την αναζήτηση καταλυμάτων και την ανταλλαγή ταξιδιωτικών σχεδίων.



Εικόνα 6: Use Case Diagram – Παράδειγμα Α.

Παράδειγμα Α , ο χρήστης παρέχει τις ταξιδιωτικές προτιμήσεις , η εφαρμογή παρέχει προτάσεις και στη συνέχεια ο χρήστης δημιουργεί το ταξίδι.

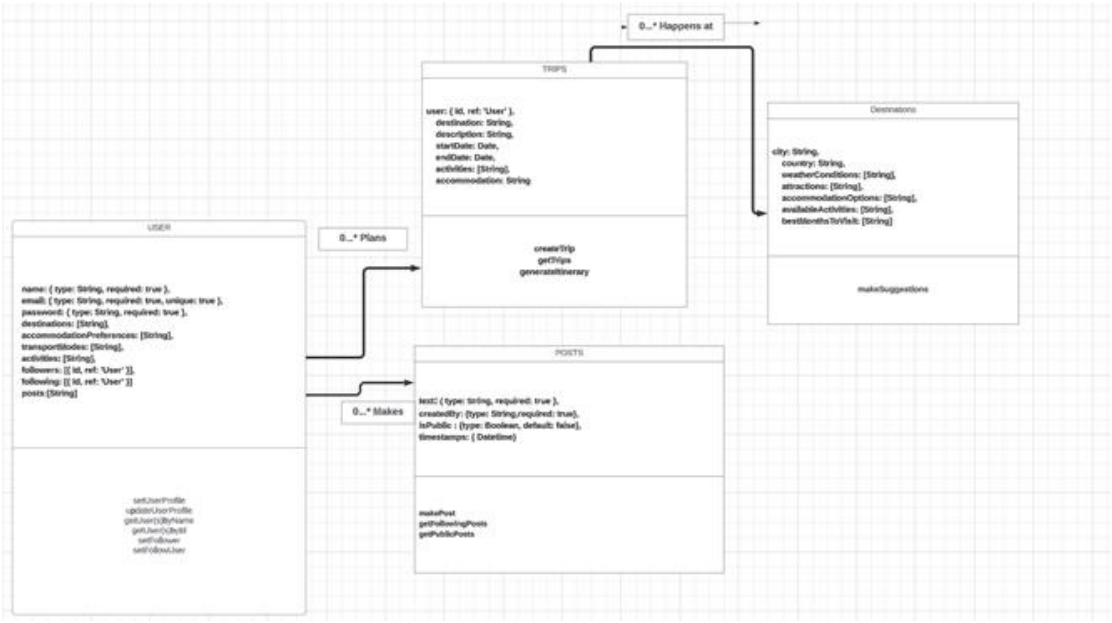


Εικόνα 7: Use Case Diagram – Παράδειγμα Β.

Παράδειγμα Β , ο χρήστης αναζητά άλλον χρήστη βάσει ονόματος , αποκτά πρόσβαση στο προφίλ του αποτελέσματος της αναζήτησης εφόσον αυτο υπάρχει και στη συνέχεια έχει τη δυνατότητα να κάνει Follow / Unfollow .

6.1.2 Class Diagram

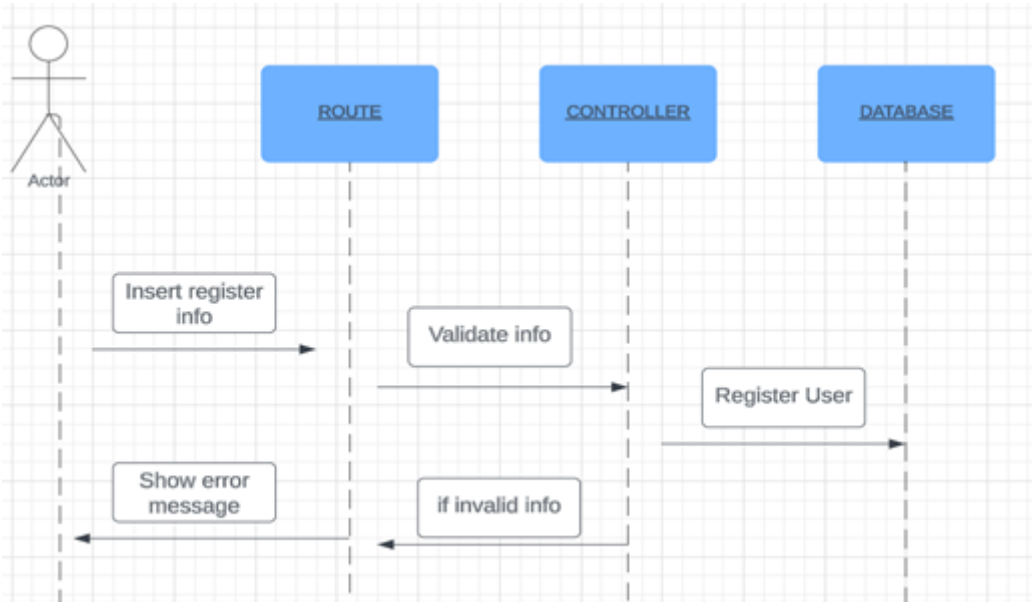
Αυτό το διάγραμμα θα απεικονίσει τις διάφορες κλάσεις και τις σχέσεις τους εντός της εφαρμογής. Οι κλάσεις μπορεί να περιλαμβάνουν Χρήστη, Ταξίδι, Προορισμό, Διαμονή, Δραστηριότητα και άλλα. Οι σχέσεις μεταξύ των κλάσεων μπορούν να αντιπροσωπεύουν συσχετίσεις, συσσωρεύσεις ή κληρονομικότητα.



Εικόνα 8: Class Diagram.

6.1.3 Sequence Diagram

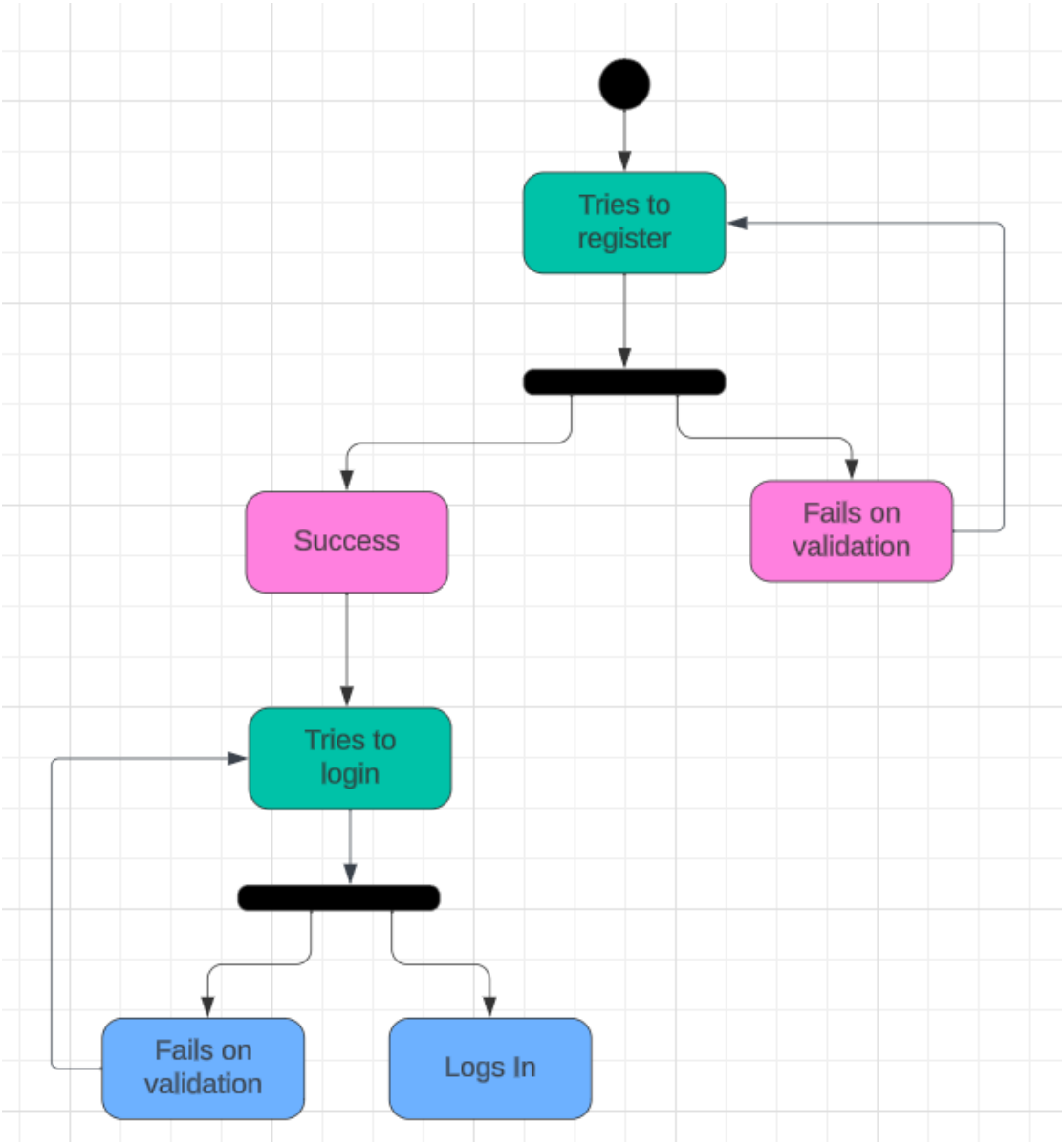
Αυτό το διάγραμμα μπορεί να απεικονίσει την αλληλουχία των αλληλεπιδράσεων μεταξύ διαφορετικών στοιχείων ή παραγόντων στην εφαρμογή. Μπορεί να δείξει πώς εκτελούνται διαφορετικές λειτουργίες, όπως η διαδικασία δημιουργίας μιας προσαρμοσμένης διαδρομής ή η πραγματοποίηση μιας κράτησης.



Εικόνα 9: Sequence Diagram: Παράδειγμα διαγράμματος ακολουθίας για την εγγραφή χρήστη.

6.1.4 Activity Diagram

Αυτό το διάγραμμα απεικονίζει την εφαρμογή ροής εργασιών/διαδικασιών για διάφορες ενέργειες. Εμφανίζει τα βήματα που εμπλέκονται στον προγραμματισμό ταξιδιών (επιλογή προορισμού, ρύθμιση προτιμήσεων, δημιουργία δρομολογίων, πραγματοποίηση κρατήσεων.



Εικόνα 10: Activity Diagram.

7. Συμπεράσματα

7.1 Use Case Diagrams

Τα Use Case Diagrams χρησιμοποιούνται κυρίως για την αποτύπωση των λειτουργικών απαιτήσεων ενός συστήματος. Δείχνουν τι πρέπει να κάνει το σύστημα από την οπτική γωνία του χρήστη. Απεικονίζουν πώς αλληλοεπιδρούν εξωτερικοί χρήστες (παράγοντες) με το σύστημα, απεικονίζοντας τις σχέσεις μεταξύ των χρηστών και των διαφορετικών περιπτώσεων χρήσης.

Χρησιμοποιούνται συχνά στη φάση της ανάλυσης απαιτήσεων της ανάπτυξης λογισμικού για την κατανόηση των λειτουργικών απαιτήσεων ενός συστήματος και βοηθά τους προγραμματιστές και τους σχεδιαστές να κατανοήσουν τις λειτουργίες που χρειάζεται να παρέχει το σύστημα έτσι ώστε να δράσουν ανάλογα για το σχεδιασμό και την ανάπτυξη της εφαρμογής.

Πλεονεκτήματα:

Απλότητα: Τα Use Case Diagrams είναι απλά και εύκολα κατανοητά, καθιστώντας τα ένα αποτελεσματικό εργαλείο επικοινωνίας μεταξύ τεχνικών και μη τεχνικών ενδιαφερομένων.

Εστίαση στην αλληλεπίδραση χρήστη: Εστιάζουν στην οπτική γωνία του χρήστη, διασφαλίζοντας ότι το σύστημα πληροί τις απαιτήσεις των χρηστών.

Προσδιορισμός εύρους συστήματος: Βοηθά στον καθορισμό των ορίων του συστήματος και του τι πρέπει να κάνει.

Μειονεκτήματα:

Έλλειψη λεπτομέρειας: Τα Use Case Diagrams παρέχουν μια επιφανειακή προβολή και συχνά λείπουν λεπτομερείς πληροφορίες σχετικά με τη λειτουργικότητα του συστήματος.

Δεν είναι κατάλληλο για σύνθετες λειτουργίες: Μπορεί να μην είναι το καλύτερο εργαλείο για την αναπαράσταση πολύπλοκων ή λεπτομερών λειτουργικών απαιτήσεων.

Συνοπτικά, χρησιμοποιούνται για την περιγραφή της αλληλεπίδρασης μεταξύ των χρηστών και ενός συστήματος. Διαδραματίζουν κρίσιμο ρόλο στην αποτύπωση λειτουργικών απαιτήσεων και στην επικοινωνία του εύρους και των δυνατοτήτων ενός συστήματος με έναν χρηστοκεντρικό τρόπο.

7.2 Class Diagrams

Τα Class Diagrams παρέχουν μια στατική άποψη του συστήματος, απεικονίζοντας τις κλάσεις του συστήματος, τις ιδιότητες, τις μεθόδους και τις σχέσεις μεταξύ αυτών των κλάσεων. Χρησιμεύουν ως προσχέδιο για την ανάπτυξη του συστήματος, δείχνοντας όχι μόνο τις κλάσεις αλλά και τις αλληλεπιδράσεις τους, κάτι που βοηθά στην κατανόηση και την υλοποίηση του σχεδιασμού του συστήματος.

Χρησιμοποιούνται ευρέως στη φάση σχεδιασμού της ανάπτυξης λογισμικού για τη λεπτομέρεια της δομής του συστήματος. Είναι ζωτικής σημασίας στις μεθοδολογίες OOAD για τη μοντελοποίηση και το σχεδιασμό αντικειμενοστρεφών συστημάτων. Ενώ μπορούν να χρησιμοποιηθούν για το σχεδιασμό σχημάτων βάσεων δεδομένων.

Πλεονεκτήματα:

Clear Structure Representation: Προσφέρει μια σαφή εικόνα της δομής του συστήματος και των σχέσεων μεταξύ των κλάσεων.

Δημιουργία κώδικα: Πολλά σύγχρονα εργαλεία ανάπτυξης μπορούν να δημιουργήσουν δομές κώδικα από διαγράμματα κλάσεων.

Τεκμηρίωση: Χρήσιμο για την τεκμηρίωση ενός συστήματος, παρέχοντας έναν σαφή και συνοπτικό τρόπο μετάδοσης της δομής του συστήματος.

Μειονεκτήματα:

Πολυπλοκότητα: Για μεγάλα συστήματα, τα διαγράμματα κλάσεων μπορεί να γίνουν πολύ περίπλοκα και δύσκολο να ερμηνευτούν.

Στατική φύση: Αντιπροσωπεύουν μια στατική προβολή και δεν αποτυπώνουν τις δυναμικές πτυχές ενός συστήματος, όπως η ροή δεδομένων ή η αλληλουχία των διαδικασιών.

Συνοπτικά, είναι θεμελιώδη στην αντικειμενοστραφή μοντελοποίηση, παρέχοντας μια δομημένη και λεπτομερή άποψη του συστήματος. Διαδραματίζουν ζωτικό ρόλο στην κατανόηση, το σχεδιασμό και την τεκμηρίωση των στατικών πτυχών των συστημάτων λογισμικού, ιδιαίτερα όσον αφορά τις τάξεις και τις αλληλεπιδράσεις τους.

7.3 Sequence Diagrams

Τα Sequence Diagrams χρησιμοποιούνται για τη μοντελοποίηση της ροής της λογικής μέσα σε ένα σύστημα με οπτικό τρόπο, δείχνοντας πώς τα αντικείμενα αλληλοεπιδρούν με άλλα σε ένα συγκεκριμένο σενάριο μιας περίπτωσης χρήσης. Δίνουν έμφαση στη χρονική πτυχή των αλληλεπιδράσεων αντικειμένων, συλλαμβάνοντας την αλληλουχία των μηνυμάτων που στέλνονται και λαμβάνονται από τα αντικείμενα της εφαρμογής.

Είναι ιδιαίτερα χρήσιμα στη μοντελοποίηση της συμπεριφοράς ενός συστήματος ή μέρους ενός συστήματος. Βοηθούν στην οπτικοποίηση του τρόπου με τον οποίο τα αντικείμενα αλληλοεπιδρούν σε ένα δεδομένο σενάριο, το οποίο είναι κρίσιμο τόσο για τις φάσεις ανάλυσης όσο και για τις φάσεις σχεδιασμού της ανάπτυξης της εφαρμογής. Βοηθά στον προσδιορισμό των ευθυνών των διαφορετικών αντικειμένων και στον τρόπο με τον οποίο συνεργάζονται για να εκτελέσουν μια λειτουργία.

Πλεονεκτήματα:

Σαφήνεια στην αλληλεπίδραση: Παρέχει μια σαφή κατανόηση του τρόπου με τον οποίο τα αντικείμενα αλληλοεπιδρούν μέσω μηνυμάτων με την πάροδο του χρόνου.

Επεξεργασία Περίπτωσης Χρήσης: Χρήσιμο για την επεξεργασία περιπτώσεων χρήσης περιγράφοντας τη ροή των γεγονότων σε μία περίπτωση χρήσης.

Δυναμική προβολή: Προσφέρει μια δυναμική προβολή ενός συστήματος, σε αντίθεση με τα στατικά διαγράμματα όπως τα διαγράμματα κλάσεων.

Μειονεκτήματα:

Πολυπλοκότητα σε μεγάλα συστήματα: Για πολύπλοκα συστήματα με πολλά αντικείμενα, τα διαγράμματα ακολουθίας μπορεί να γίνουν υπερβολικά περίπλοκα και δύσκολο να ακολουθηθούν.

Περιορισμένο εύρος: Κάθε διάγραμμα αντιπροσωπεύει συνήθως μόνο ένα σενάριο ή μέρος μιας πολύπλοκης διαδικασίας, περιορίζοντας το εύρος της.

Συνοπτικά, τα Sequence Diagrams είναι ένα ισχυρό εργαλείο για την οπτικοποίηση της αλληλουχίας των μηνυμάτων που ανταλλάσσονται μεταξύ αντικειμένων και χρηστών σε ένα σύστημα. Παρέχουν έναν αποτελεσματικό τρόπο μετάδοσης πολύπλοκων προτύπων αλληλεπίδρασης και χρησιμοποιούνται ευρέως στις φάσεις ανάλυσης και σχεδίασης έργων ανάπτυξης λογισμικού.

7.4 Activity diagrams

Τα Activity Diagrams χρησιμοποιούνται για τη μοντελοποίηση της ροής εργασιών και των εσωτερικών ροών διεργασιών μέσα σε ένα σύστημα. Είναι εξαιρετικά για την απεικόνιση της δυναμικής συμπεριφοράς ενός συστήματος, απεικονίζοντας τη σειρά και τις συνθήκες για τον συντονισμό διαφόρων δραστηριοτήτων.

Χρησιμοποιούνται ευρέως στη μοντελοποίηση επιχειρηματικών διαδικασιών για την καταγραφή των επιχειρηματικών λειτουργιών και ροών εργασίας. Βοηθούν στη λεπτομέρεια των λειτουργιών μέσα σε ένα σύστημα, ιδιαίτερα χρήσιμο σε σύνθετα σενάρια που περιλαμβάνουν

πολλαπλές οντότητες, και με αποτελεσματικό τρόπο συμβάλουν στην οπτικοποίηση διαδικασιών που συμβαίνουν παράλληλα.

Πλεονεκτήματα:

Επισκόπηση υψηλού επιπέδου: Παρέχει μια προβολή υψηλού επιπέδου των διαδικασιών και των ροών εργασίας του συστήματος.

Ευελιξία: Μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς, από τη μοντελοποίηση επιχειρηματικών διαδικασιών έως την απεικόνιση της λειτουργίας του συστήματος.

Εκπροσώπηση απόφασης και ταυτόχρονης παρουσίασης: Αντιπροσωπεύει αποτελεσματικά σημεία λήψης αποφάσεων και ταυτόχρονες διαδικασίες.

Μειονεκτήματα:

Πολυπλοκότητα με μεγάλες διεργασίες: Μπορεί να γίνει υπερβολικά περίπλοκη και δυσανάγνωστη όταν απεικονίζονται μεγάλες διεργασίες με πολλές δραστηριότητες.

Έλλειψη λεπτομέρειας για υλοποίηση: Αν και είναι ιδανικά για σχεδιασμό υψηλού επιπέδου, ενδέχεται να μην παρέχουν αρκετές λεπτομέρειες για την πραγματική εφαρμογή.

Συνοπτικά, τα Activity Diagrams είναι ένα ευέλικτο εργαλείο για τη μοντελοποίηση των δυναμικών πτυχών των συστημάτων, ιδιαίτερα χρήσιμο για την απεικόνιση ροών εργασίας, επιχειρηματικών διαδικασιών και πολύπλοκων λειτουργικών διαδικασιών. Προσφέρουν μια οπτική αναπαράσταση διαδοχικών και παράλληλων δραστηριοτήτων, σημείων απόφασης και της ροής ελέγχου σε ένα σύστημα.

7.5 Σύγκριση της χρησιμότητας των διαγραμμάτων

Η σύγκριση της χρησιμότητας των Activity, Sequence, Class, and Use Case Diagrams στο πλαίσιο της ανάπτυξης μιας εξατομικευμένης ταξιδιωτικής εφαρμογής περιλαμβάνει την κατανόηση των συγκεκριμένων πτυχών που αντιμετωπίζει καλύτερα κάθε διάγραμμα στη διαδικασία ανάπτυξης λογισμικού. Κάθε διάγραμμα εξυπηρετεί διαφορετικούς σκοπούς και είναι χρήσιμο σε διάφορα στάδια ανάπτυξης.

Τελικά, η επιλογή εξαρτάται από το συγκεκριμένο στάδιο ανάπτυξης και την πτυχή του συστήματος στο οποίο εργάζεται. Σε μια ολοκληρωμένη διαδικασία ανάπτυξης, όλα αυτά τα διαγράμματα θα έπαιζαν ζωτικούς ρόλους σε διαφορετικά στάδια στην ανάπτυξη μιας εξατομικευμένης ταξιδιωτικής εφαρμογής.

Στο πλαίσιο της ανάπτυξης μιας εφαρμογής με JavaScript ή οποιαδήποτε γλώσσα προγραμματισμού για αυτό το θέμα, αυτά τα διαγράμματα UML μπορούν να παρέχουν σημαντικές πληροφορίες τόσο για τη δομή όσο και για τη λειτουργία της εφαρμογής.

Δομικές πληροφορίες (class diagrams):

Συνάφεια με τη JavaScript: Παρόλο που η JavaScript είναι μια δυναμικά κατασκευασμένη γλώσσα και δεν χρησιμοποιεί κλάσεις με την παραδοσιακή έννοια (ειδικά σε παλαιότερες εκδόσεις), η σύγχρονη JavaScript (ES6 και μεταγενέστερη) υποστηρίζει σύνταξη κλάσης. Τα διαγράμματα κλάσεων μπορούν να βοηθήσουν στην οραματισμό αντικειμένων, των ιδιοτήτων τους, των μεθόδων και των σχέσεων μεταξύ τους.

Οφέλη: Προσφέρει ένα σχέδιο για τη δημιουργία δομημένου και οργανωμένου κώδικα, ο οποίος είναι ιδιαίτερα χρήσιμος σε μεγαλύτερα έργα JavaScript ή όταν χρησιμοποιείτε πλαίσια/βιβλιοθήκες που δίνουν έμφαση σε στοιχεία (όπως το React).

Λειτουργικές πληροφορίες (use case , activity and sequence diagrams):

Χρήση Use Case Diagrams: Παρέχετε μια προβολή υψηλού επιπέδου των λειτουργιών που πρέπει να προσφέρει η εφαρμογή στους χρήστες της, καθοδηγώντας την ανάπτυξη λειτουργιών που αντιμετωπίζουν οι χρήστες στο JavaScript.

Χρήση Sequence Diagrams: Ιδιαίτερα χρήσιμα για την κατανόηση της ροής των αλληλεπιδράσεων εντός της εφαρμογής, όπως το πώς αλληλεπιδρούν διαφορετικές λειτουργίες και αντικείμενα JavaScript, η ακολουθία κλήσεων AJAX, ο χειρισμός συμβάντων κ.λπ.

Χρήση Activity and Sequence Diagrams: Βοήθεια στη χαρτογράφηση των ροών εργασίας, ιδιαίτερα χρήσιμη σε πολύπλοκες εφαρμογές JavaScript όπου πολλές διεργασίες ενδέχεται να εκτελούνται ταυτόχρονα (π.χ. ασύγχρονες λειτουργίες, προγραμματισμός βάσει συμβάντων).

Η χρησιμότητα αυτών των διαγραμμάτων υπερβαίνει συγκεκριμένες γλώσσες προγραμματισμού ή τεχνολογίες. Είτε η εφαρμογή αναπτύσσεται σε JavaScript, Python, Java ή οποιαδήποτε άλλη γλώσσα, αυτά τα διαγράμματα μπορεί να είναι εξίσου διορατικά.

Συνοπτικά, τα διαγράμματα UML είναι ένα πολύτιμο πλεονέκτημα στη διαδικασία ανάπτυξης, παρέχοντας κρίσιμες πληροφορίες για τη δομή και τη λειτουργία μιας εφαρμογής. Χρησιμεύουν ως οδηγός για τους προγραμματιστές για τη δημιουργία καλά δομημένων, αποδοτικών και αποτελεσματικών εφαρμογών, ανεξάρτητα από τη γλώσσα προγραμματισμού που χρησιμοποιείται, συμπεριλαμβανομένου του JavaScript.

8. Βιβλιογραφία

1. Alogogianni, E., & Virvou, M. (2022). Undeclared Work Prediction Using Machine Learning: Dealing with the Class Imbalance and Class Overlap Problems. In 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-8). IEEE.
2. Ammann, P. and Offutt, J., 2016. Introduction to software testing. Cambridge University Press.
3. Aniche, M., 2022. Effective Software Testing: A developer's guide. Simon and Schuster.
4. Ardito, L., Coppola, R., Malnati, G. and Torchiano, M., 2020. Effectiveness of Kotlin vs. Java in android app development tasks. Information and Software Technology, 127, p.106374.
5. Arlow, J. and Neustadt, I., 2005. UML 2 and the unified process: practical object-oriented analysis and design. Pearson Education.
6. Bala, K., Sharma, S. and Kaur, G., 2015. A study on smartphone based operating system. International Journal of Computer Applications, 121(1).
7. Banker, K., Garrett, D., Bakkum, P., & Verch, S. (2016). MongoDB in action: covers MongoDB version 3.0. Simon and Schuster.
8. Baumeister, H., Koch, N. and Mandel, L., 1999, October. Towards a UML extension for hypermedia design. In International Conference on the Unified Modeling Language (pp. 614-629). Berlin, Heidelberg: Springer Berlin Heidelberg.
9. Bavota, G., Gravino, C., Oliveto, R., De Lucia, A., Tortora, G., Genero, M. and Cruz-Lemus, J.A., 2015. A fine-grained analysis of the support provided by UML class diagrams and ER diagrams during data model maintenance. Software & Systems Modeling, 14, pp.287-306.
10. Bhattacherjee, A. and Lin, C.P., 2015. A unified model of IT continuance: three complementary perspectives and crossover effects. European Journal of Information Systems, 24(4), pp.364-373.
11. Black, P.E., Okun, V. and Yesha, Y., 2000, September. Mutation operators for specifications. In Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering (pp. 81-88). IEEE.
12. Botturi, G., Ebeid, E., Fummi, F. and Quaglia, D., 2013, September. Model-driven design for the development of multi-platform smartphone applications. In Proceedings of the 2013 Forum on specification and Design Languages (FDL) (pp. 1-8). IEEE.
13. Bowman, S.R., Gauthier, J., Rastogi, A., Gupta, R., Manning, C.D. and Potts, C., 2016. A fast unified model for parsing and sentence understanding. arXiv preprint arXiv:1603.06021.
14. Briand, L.C., Labiche, Y., Di Penta, M. and Yan-Bondoc, H., 2005. An experimental investigation of formality in UML-based development. IEEE Transactions on Software Engineering, 31(10), pp.833-849.
15. Chansuwath, W. and Senivongse, T., 2016, June. A model-driven development of web applications using AngularJS framework. In 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS) (pp. 1-6). IEEE.
16. Chrysafiadi, K., Kamitsios, M., & Virvou, M. (2023). Fuzzy-based dynamic difficulty adjustment of an educational 3D-game. Multimedia Tools and Applications, 1-25.
17. Chrysafiadi, K., Papadimitriou, S., & Virvou, M. (2022). Cognitive-based adaptive scenarios in educational games using fuzzy reasoning. Knowledge-Based Systems, 250, 109111.
18. Chrysafiadi, K., Virvou, M., & Tsihrintzis, G. A. (2022). A fuzzy-based mechanism for automatic personalized assessment in an e-learning system for computer programming. Intelligent Decision Technologies, 1-16.
19. Chrysafiadi, K., Virvou, M., Tsihrintzis, G. A., & Hatzilygeroudis, I. (2023). An Adaptive Learning Environment for Programming Based on Fuzzy Logic and Machine Learning. International Journal on Artificial Intelligence Tools.
20. Ciccozzi, F., Malavolta, I. and Selic, B., 2019. Execution of UML models: a systematic review of research and practice. Software & Systems Modeling, 18, pp.2313-2360.
21. Conallen, J., 2003. Building Web applications with UML. Addison-Wesley Professional.

22. Dennis, A., Wixom, B. and Tegarden, D., 2015. Systems analysis and design: An object-oriented approach with UML. John wiley & sons.
23. Dzidek, W.J., Arisholm, E. and Briand, L.C., 2008. A realistic empirical evaluation of the costs and benefits of UML in software maintenance. IEEE Transactions on software engineering, 34(3), pp.407-432.
24. Freitas, F. and Maia, P.H.M., 2016, November. Justmodeling: An mde approach to develop android business applications. In 2016 VI Brazilian Symposium on Computing Systems Engineering (SBESC) (pp. 48-55). IEEE.
25. Gilski, P. and Stefanski, J., 2015. Android os: A review. Tem Journal, 4(1), p.116.
26. Harkin, D. and Molnar, A., 2021. Operating-system design and its implications for victims of family violence: the comparative threat of smart phone spyware for Android versus iPhone users. Violence against women, 27(6-7), pp.851-875.
27. Hu, H., Huang, Y., Chen, Q., zhuo, T.Y. and Chen, C., 2023. A First Look at On-device Models in iOS Apps. ACM Transactions on Software Engineering and Methodology.
28. Irwansyah, F.S., Yusuf, Y.M., Farida, I. and Ramdhani, M.A., 2018, January. Augmented reality (AR) technology on the android operating system in chemistry learning. In IOP conference series: Materials science and engineering (Vol. 288, p. 012068). IOP Publishing.
29. Jürjens, J., 2002, September. UMLsec: Extending UML for secure systems development. In International Conference on The Unified Modeling Language (pp. 412-425). Berlin, Heidelberg: Springer Berlin Heidelberg.
30. Ko, M., Seo, Y.J., Min, B.K., Kuk, S. and Kim, H.S., 2012, May. Extending UML meta-model for android application. In 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (pp. 669-674). IEEE.
31. Koç, H., Erdoğan, A.M., Barjakly, Y. and Peker, S., 2021, March. UML diagrams in software engineering research: a systematic literature review. In Proceedings (Vol. 74, No. 1, p. 13). MDPI.
32. Koch, N., Baumeister, H., Hennicker, R. and Mandel, L., 2000, October. Extending uml to model navigation and presentation in web applications. In Proceedings of Modelling Web Applications in the UML Workshop. York, England.
33. Kochan, S.G., 2011. Programming in objective-C. Addison-Wesley Professional.
34. Kuzniarz, L., Staron, M. and Wohlin, C., 2004, June. An empirical study on using stereotypes to improve understanding of UML models. In Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004. (pp. 14-23). IEEE.
35. Martinez, M. and Lecomte, S., 2017, May. Towards the quality improvement of cross-platform mobile applications. In 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft) (pp. 184-188). IEEE.
36. Masruroh, S.U. and Saputra, I., 2016, April. Performance evaluation of instant messenger in Android operating system and iPhone operating system. In 2016 4th International Conference on Cyber and IT Service Management (pp. 1-6). IEEE.
37. Min, B.K., Ko, M., Seo, Y., Kuk, S. and Kim, H.S., 2011, November. A UML metamodel for smart device application modeling based on Windows Phone 7 platform. In TENCON 2011-2011 IEEE Region 10 Conference (pp. 201-205). IEEE.
38. Narawita, C.R. and Vidanage, K., 2016, September. UML generator-an automated system for model driven development. In 2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer) (pp. 250-256). IEEE.
39. Nassar, M., 2003, October. VUML: a Viewpoint oriented UML Extension. In 18th IEEE International Conference on Automated Software Engineering, 2003. Proceedings. (pp. 373-376). IEEE.
40. Nguyen, P.H., Kramer, M., Klein, J. and Le Traon, Y., 2015. An extensive systematic review on the model-driven development of secure systems. Information and Software Technology, 68, pp.62-81.
41. Panagoulias, D. P., Virvou, M., & Tsihrintzis, G. A. (2022). A microservices-based iterative development approach for usable, reliable and explainable AI-infused medical applications using RUP. In 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI) (pp. 1028-1035). IEEE.

42. Parada, A., Marques, M. and de Brisolara, L.B., 2015. Automating mobile application development: UML-based code generation for Android and Windows Phone. Revista de Informática Teórica e Aplicada, 22(2), pp.31-50.
43. Perego, G. and Pezzetti, S., 2013. Un approccio model driven per lo sviluppo di applicazioni mobili native.
44. Politou, E., Alepis, E., Virvou, M., & Patsakis, C. (2022). Privacy and Data Protection Challenges in the Distributed Era (Vol. 26, pp. 1-185). Springer.
45. Ribeiro, A., Ferreira, J.F. and Mendes, A., 2021, December. Ecoandroid: An android studio plugin for developing energy-efficient java mobile applications. In 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) (pp. 62-69). IEEE.
46. Rumpe, B., 2016. Modeling with UML (pp. 1-281). Cham: Springer.
47. Sabraoui, A., El Koutbi, M. and Khriss, I., 2012, November. Gui code generation for android applications using a mda approach. In 2012 IEEE International Conference on Complex Systems (ICCS) (pp. 1-6). IEEE.
48. Safdar, S.A., Iqbal, M.Z. and Khan, M.U., 2015. Empirical evaluation of UML modeling tools–a controlled experiment. In Modelling Foundations and Applications: 11th European Conference, ECMFA 2015, Held as Part of STAF 2015, LAquila, Italy, July 20-24, 2015. Proceedings 11 (pp. 33-44). Springer International Publishing.
49. Sarkar, A., Goyal, A., Hicks, D., Sarkar, D. and Hazra, S., 2019, December. Android application development: A brief overview of android platforms and evolution of security systems. In 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 73-79). IEEE.
50. Sarkar, D., Bhalla, M. and Singal, S.M., 2017, January. Enhancing unified process workflows using UML. In 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence (pp. 788-792). IEEE.
51. Shaw, H., Ellis, D.A., Kendrick, L.R., Ziegler, F. and Wiseman, R., 2016. Predicting smartphone operating system from personality and individual differences. Cyberpsychology, Behavior, and Social Networking, 19(12), pp.727-732.
52. Soldani, D. and Manzalini, A., 2015. Horizon 2020 and beyond: On the 5G operating system for a true digital society. IEEE Vehicular Technology Magazine, 10(1), pp.32-42.
53. Sunitha, E.V. and Samuel, P., 2019. Automatic code generation from UML state chart diagrams. IEEE Access, 7, pp.8591-8608.
54. Tiwari, R.G., Srivastava, A.P., Bhardwaj, G. and Kumar, V., 2021, April. Exploiting UML diagrams for test case generation: a review. In 2021 2nd international conference on intelligent engineering and management (ICIEM) (pp. 457-460). IEEE.
55. Tufail, H., Azam, F., Anwar, M.W. and Qasim, I., 2018, November. Model-driven development of mobile applications: A systematic literature review. In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) (pp. 1165-1171). IEEE.
56. Unhelkar, B., 2017. Software engineering with uml. CRC Press.
57. Usman, M., Iqbal, M.Z. and Khan, M.U., 2014, December. A model-driven approach to generate mobile applications for multiple platforms. In 2014 21st Asia-Pacific Software Engineering Conference (Vol. 1, pp. 111-118). IEEE.
58. Uttarwar, P., Tidke, R.P., Dandwate, D.S. and Tupe, U.J., 2021. A Literature Review on Android-A Mobile Operating system. International Research Journal of Engineering and Technology, 8(1), pp.1-6.
59. Utting, M. and Legeard, B., 2007. Selecting Your Tests, Practical Model-Based Testing.
60. Utting, M. and Legeard, B., 2010. Practical model-based testing: a tools approach. Elsevier.
61. Virvou, M. (2022). The emerging era of human-AI interaction: Keynote address. In 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA) (pp. 1-10). IEEE.
62. Virvou, M. (2023) Artificial Intelligence and User Experience in reciprocity: Contributions and state of the art. Intelligent Decision Technologies, 1-53.
63. Virvou, M., Alepis, E., Tsihrintzis, G. A., & Jain, L. C. (2020). Machine learning paradigms: advances in learning analytics (pp. 1-5). Springer International Publishing.

64. Virvou, M., Tsihrintzis, G. A., Phillips-Wren, G., & Jain, L. C. (2023) Special collection of invited original research papers on “Contributions by women in theory and applications of artificial intelligence”. Intelligent Decision Technologies, 1-4.
65. Yamacli, S., 2018. Beginner's Guide to iOS 12 App Development Using Swift 4: Xcode, Swift and App Design Fundamentals. CreateSpace Independent Publishing Platform.
66. Zander, J., Schieferdecker, I. and Mosterman, P.J. eds., 2017. Model-based testing for embedded systems. CRC press.