ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Εφαρμογής για Αλληλεπίδραση και Ανταλλαγή Απόψεων μεταξύ Φοιτητών

Διπλωματική Εργασία

Τσαπάλας Δημήτριος-Νικόλαος

Επιβλέπων: Βασιλακόπουλος Μιχαήλ

Ιανουάριου 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Εφαρμογής για Αλληλεπίδραση και Ανταλλαγή Απόψεων μεταξύ Φοιτητών

Διπλωματική Εργασία

Τσαπάλας Δημήτριος-Νικόλαος

Επιβλέπων: Βασιλακόπουλος Μιχαήλ

Ιανουάριου 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Development of an Application for Interaction and Exchange of Views between Students

Diploma Thesis

Tsapalas Dimitrios-Nikolaos

Supervisor: Vassilakopoulos Michael

January 2025

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Βασιλακόπουλος Μιχαήλ**
Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Φεύγας Αθανάσιος**
Μέλος Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Θάνος Γεώργιος**
Μέλος Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Θέλω να ευχαριστήσω την οικογένειά μου για την στήριξή τους όλα αυτά τα χρόνια που ήμουν φοιτητής. Η αγάπη και η υπομονή τους με βοήθησαν να ξεπεράσω πολλές δυσκολίες και να συνεχίσω την πορεία μου. Ευχαριστώ τους φίλους μου για τις όμορφες στιγμές που περάσαμε μαζί και για το οτι ηταν παντα εκει οταν τους χρειαζόμουν. Ευχαριστώ επίσης τους καθηγητές μου που με βοήθησαν με τις γνώσεις τους. Αυτά τα χρόνια ως φοιτητής ήταν γεμάτα εμπειρίες και προκλήσεις.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Τσαπάλας Δημήτριος-Νικόλαος

Διπλωματική Εργασία

Ανάπτυξη Εφαρμογής για Αλληλεπίδραση και Ανταλλαγή Απόψεων μεταξύ Φοιτητών

Τσαπάλας Δημήτριος-Νικόλαος

Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας σύγχρονης, ασφαλούς και φιλικής προς τον χρήστη εφαρμογής κοινωνικής δικτύωσης, με στόχο να διευκολύνει την επικοινωνία, τη συνεργασία και την ανταλλαγή πληροφοριών μεταξύ χρηστών. Η εργασία εξετάζει τη χρήση τεχνολογιών όπως το React για την ανάπτυξη της web έκδοσης και το React Native για την ανάπτυξη της κινητής εφαρμογής, καθώς και τις τεχνολογίες Back-End όπως Node.js , Express.js και MongoDB για τη διαχείριση των δεδομένων και την επικοινωνία με την εφαρμογή. Η μεθοδολογία περιλαμβάνει τη χρήση JWT tokens και bcrypt για την ασφαλή διαχείριση των χρηστών και των κωδικών πρόσβασης, ενώ για την επικοινωνία σε πραγματικό χρόνο χρησιμοποιήθηκε το socket.io. Η εφαρμογή υποστηρίζει επαλήθευση των λογαριασμών και ανάκτηση κωδικού πρόσβασης μέσω της αποστολής 6-ψήφιων κωδικών μέσω email, προσφέροντας μια επιπλέον διάσταση ασφάλειας. Τα κύρια αποτελέσματα περιλαμβάνουν τη δημιουργία μιας λειτουργικής και ασφαλούς πλατφόρμας κοινωνικής δικτύωσης που υποστηρίζει βασικές λειτουργίες όπως η δημιουργία προφίλ, η δημοσίευση περιεχομένου, τα σχόλια και τα μηνύματα, καθώς και την επικοινωνία σε πραγματικό χρόνο. Παράλληλα, η πλατφόρμα παρέχει δυνατότητες επέκτασης για μελλοντικές βελτιώσεις, όπως η υποστήριξη ρόλων χρηστών (διαχειριστές και απλοί χρήστες), η δυνατότητα ανάρτησης βίντεο, η δημιουργία ομάδων και η διοργάνωση εκδηλώσεων.

Λέξεις-κλειδιά:

Ανάπτυξη web και κινητής εφαρμογής, React, React Native, Express, Node, MongoDB

Diploma Thesis

Development of an Application for Interaction and Exchange of Views between Students

Tsapalas Dimitrios-Nikolaos

Abstract

This thesis focuses on the development of a modern, secure, and user-friendly social networking application aimed at facilitating communication, collaboration, and information exchange among users. The work explores the use of technologies such as React for the development of the web version and React Native for the development of the mobile application, as well as Back-end technologies like Node.js, Express.js, and MongoDB for data management and communication with the application. The methodology includes the use of JWT tokens and bcrypt for secure management of users and passwords, while real-time communication is enabled using socket.io. The application supports account verification and password recovery through the sending of 6-digit codes via email, providing an additional layer of security. The main results include the creation of a functional and secure social networking platform that supports basic features such as profile creation, content publishing, comments, and messaging, as well as real-time communication. Additionally, the platform provides extensibility for future improvements, such as support for user roles (admins or users), the ability to upload videos, create groups and events.

Keywords:

Development of web and mobile application, React, React Native, Express, Node, MongoDB

Πίνακας περιεχομένων

Κεφάλαιο 1

Εισαγωγή

Η ραγδαία ανάπτυξη της τεχνολογίας και του διαδικτύου έχει μετασχηματίσει τον τρόπο με τον οποίο οι άνθρωποι επικοινωνούν, συνεργάζονται και μοιράζονται πληροφορίες. Οι κοινωνικές εφαρμογές αποτελούν πλέον κομμάτι της καθημερινότητας μας, επιτρέποντας μας, να αλληλεπιδρούμε με άλλους ανθρώπους σε όλο τον κόσμο.

1.1 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη μιας cross-platform εφαρμογής, η οποία θα προσφέρει στους χρήστες τη δυνατότητα να δημιουργούν, να αλληλεπιδρούν και να διαμοιράζονται περιεχόμενο μέσω ενός σύγχρονου και ασφαλούς περιβάλλοντος. Η εφαρμογή θα υλοποιηθεί με χρήση της τεχνολογίας React για την ανάπτυξη του web τμήματος και της τεχνολογίας React Native για την ανάπτυξη του mobile τμήματος.

Βασικά προβλήματα που στοχεύει να λύσει η εφαρμογή περιλαμβάνουν:

- **Ασφαλής επικοινωνία και διαχείριση δεδομένων**: Ενσωμάτωση τεχνολογιών ασφάλειας, όπως JWT tokens για την αυθεντικοποίηση και bcrypt για την κρυπτογράφηση, με στόχο την προστασία των προσωπικών δεδομένων.

- **Ευκολία χρήσης**: Δημιουργία ενός απλού και φιλικού προς τον χρήστη interface.

- **Διαθεσιμότητα σε διαφορετικές πλατφόρμες**: Εξασφάλιση της πρόσβασης στην εφαρμογή τόσο από προγράμματα περιήγησης όσο και από κινητές συσκευές.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής εργασίας συνοψίζεται ως εξής:

1. **Μελέτη υπαρχουσών τεχνολογιών**: Ανάλυση των τεχνολογιών που σχετίζονται με την ανάπτυξη cross-platform εφαρμογών, συμπεριλαμβανομένων των React, React Native, Node.js & Express.js και MongoDB.

2. **Ανάπτυξη του Back-end**: Υλοποίηση ασφαλούς Back-end συστήματος με χρήση τεχνολογιών JWT και brcypt για τη διαχείριση και την προστασία των προσωπικών δεδομένων των χρηστών.

3. **Ανάπτυξη του Front-End** : Δημιουργία δυναμικού και φιλικού προς τον χρήστη interface για κινητές συσκευές και web browsers.

4. **Λειτουργίες**: Ενσωμάτωση λειτουργιών όπως δημιουργία δημοσιεύσεων, σχολιασμός, likes και μηνύματα.

5. **Επικοινωνία σε πραγματικό χρόνο**: Ανάπτυξη συστήματος ανταλλαγής μηνυμάτων σε πραγματικό χρόνο με την χρήση του socket-io.

6. **Αξιολόγηση και βελτιστοποίηση**: Δοκιμή της εφαρμογής και βελτιστοποίηση της.

7. **Επέκταση**: Παρουσίαση μιάς ολοκληρωμένης εφαρμογής, ιδανική για επέκταση και μελλοντική ανάπτυξη.

1.2 Οργάνωση του τόμου

Ο τόμος της διπλωματικής εργασίας είναι οργανωμένος σε επτά κεφάλαια. Στο Κεφάλαιο 2 παρουσιάζονται αντίστοιχα ή παρόμοια συστήματα που έχουν αναπτυχθεί και χρησιμοποιούνται στον τομέα των κοινωνικών δικτύων. Εξηγείται πώς το σύστημά μας διαφοροποιείται, αναδεικνύοντας τα χαρακτηριστικά που προσφέρει. Στο Κεφάλαιο 3 γίνεται ανάλυση των τεχνολογιών που χρησιμοποιούνται για την ανάπτυξη ιστοσελίδων και εφαρμογών για κινητά τηλέφωνα. Συγκεκριμένα, αναφέρονται οι γλώσσες και τα εργαλεία όπως HTML, CSS και JavaScript, οι τεχνολογίες Front-End όπως React, Angular και Vue, και οι τεχνολογίες Back-End όπως Node.js και Express. Επιπλέον, παρουσιάζονται οι βάσεις δεδομένων MongoDB, PostgreSQL και Firebase, καθώς και οι τεχνολογίες που χρησιμοποιούνται

για την ανάπτυξη εφαρμογών για κινητά, όπως το React Native και οι native γλώσσες προγραμματισμού. Το Κεφάλαιο 4 περιγράφει αναλυτικά τα βήματα ανάπτυξης της εφαρμογής. Ξεκινά με τη δημιουργία της βάσης δεδομένων, παρουσιάζοντας την αρχιτεκτονική της και τη σύνδεσή της με τον server. Παρουσιάζεται ο κώδικας του server, καθώς και τα πακέτα που χρησιμοποιήθηκαν. Περιγράφεται η σύνδεση του Back-end με το Front-End καθώς και η χρήση τεχνολογιών για την επίτευξη της επικοινωνίας σε πραγματικό χρόνο. Επιπλέον, περιγράφονται τα βήματα ανάπτυξης του mobile τμήματος της εφαρμογής. Στο Κεφάλαιο 5 εξετάζονται οι βασικές σελίδες και οι λειτουργίες της εφαρμογής τόσο για κινητά όσο και για web. Στο Κεφάλαιο 6 παρουσιάζονται τα εργαλεία διαχείρισης κώδικα, όπως το Git, επίσης παρέχεται ο πλήρης κώδικας της εφαρμογής. Τέλος, το Κεφάλαιο 7 περιλαμβάνει τα συμπεράσματα της εφαρμογής, εστιάζοντας στις προκλήσεις που αντιμετωπίστηκαν και στις λύσεις που εφαρμόστηκαν, ενώ αναφέρονται επίσης μερικές από τις πιθανές μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Σχετικά Συστήματα

Σε αυτό το κεφάλαιο, θα παρουσιαστούν και θα αναλυθούν σχετικά συστήματα που συνδέονται με το σύστημά μας. Παράλληλα, θα γίνει αναφορά σε επιστημονικές μελέτες και ερευνητικά έργα που έχουν εξετάσει την απόδοση διαφορετικών τεχνολογιών ανάπτυξης. Ιδιαίτερη έμφαση θα δοθεί σε συγκριτικές αναλύσεις μεταξύ των τεχνολογιών που χρησιμοποιήσαμε και εναλλακτικών λύσεων, όπως η PHP, MySQL και Kotlin/Swift, που συναντώνται σε πολλές υπάρχουσες εφαρμογές κοινωνικής δικτύωσης. Στόχος μας είναι να αναδείξουμε τα πλεονεκτήματα του δικού μας συστήματος, σε σχέση με τις υπάρχουσες προσεγγίσεις, εξετάζοντας παράγοντες όπως η απόδοση και ο χρόνος ανάπτυξης.

2.1 Υπάρχοντα Κοινωνικά Δίκτυα

Οι εφαρμογές κοινωνικής δικτύωσης αποτελούν αναπόσπαστο κομμάτι της καθημερινότητας των χρηστών του διαδικτύου. Μεγάλες πλατφόρμες, όπως το Facebook, το Instagram και το X, κυριαρχούν στον χώρο, προσφέροντας πληθώρα λειτουργιών επικοινωνίας, διαμοιρασμού περιεχομένου και κοινωνικής αλληλεπίδρασης. Ωστόσο, η κυριαρχία αυτών των εφαρμογών δεν σημαίνει ότι δεν υπάρχει χώρος για νέες προσεγγίσεις.

Η ανάπτυξη μιας νέας πλατφόρμας κοινωνικής δικτύωσης μπορεί να εστιάσει σε συγκεκριμένες βελτιώσεις, όπως η απόδοση του συστήματος αλλά και ο χρόνος ανάπτυξης.

Χαρακτηριστικά	**Υπάρχουσες Προσεγγίσεις**	**Προτεινόμενη Λύση**
Τεχνολογίες	PHP, Python	Node.js
Βάση Δεδομένων	MySQL	MongoDB
Mobile	Kotlin, Swift	React Native
Χρόνος Ανάπτυξης	Αργός, λόγω διαφορετικών τεχνολογιών για κάθε πλατφόρμα.	Γρήγορος, καθώς χρησιμοποιείται μόνο η JavaScript.

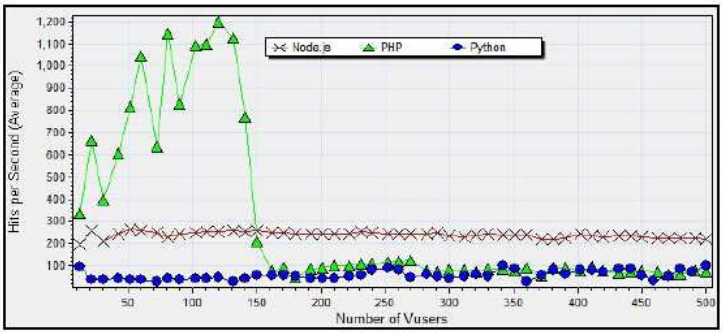
Πίνακας 2.1: Σύγκριση υπαρχουσών και προτεινόμενων λύσεων

Στην επόμενη ενότητα, θα παρουσιαστούν επιστημονικές μελέτες που αναλύουν και συγκρίνουν διαφορετικές τεχνολογικές προσεγγίσεις. Μέσα από αυτή τη σύγκριση, σκόπος είναι να αναδείξουμε τα πλεονεκτήματα του δικού μας σύστήματος.

2.2 Επιστημονικές Μελέτες

2.2.1 Σύγκριση Node.js με PHP και Python

Στην παρούσα μελέτη [1], πραγματοποιήθηκαν δοκιμές για τη σύγκριση της απόδοσης των τεχνολογιών Node.js, PHP και Python. Ο στόχος ήταν να αξιολογηθεί η ικανότητα τους να επεξεργάζονται αιτήματα από πολλούς χρήστες ταυτόχρονα. Τα αποτελέσματα αυτής της αξιολόγησης απεικονίζονται στο παρακάτω σχήμα:



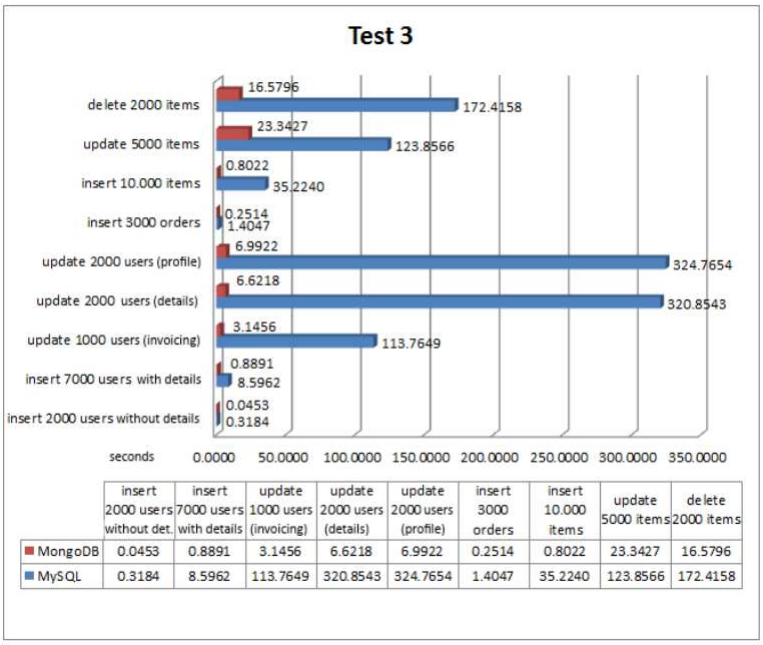
Σχήμα 2.1: Αιτήματα ανά δευτερόλεπτο

Το σχήμα παρουσιάζει τον αριθμό των αιτημάτων που μπορεί να επεξεργαστεί κάθε διακομιστής ανά δευτερόλεπτο. Σε περιβάλλον με λίγους χρήστες, η PHP ανταποκρίνεται ικανοποιητικά στην επεξεργασία των αιτημάτων. Ωστόσο, καθώς ο αριθμός των χρηστών αυξάνεται, η απόδοσή της μειώνεται απότομα. Αντίθετα, το Node.js διατηρεί σταθερή απόδοση αξιοποιώντας την αρχιτεκτονική του, που του επιτρέπει την ταυτόχρονη επεξεργασία πολλαπλών αιτημάτων. Επιπλέον, η PHP εμφανίζει σημαντική αύξηση στον χρόνο απόκρισης υπό υψηλό φόρτο χρηστών, ενώ το Node.js διατηρεί έναν πιο ομαλό ρυθμό αύξησης του χρόνου απόκρισης.

2.2.2 Σύγκριση MongoDB με MySQL

Η επιλογή βάσης δεδομένων εξαρτάται από τις απαιτήσεις της εκάστοτε εφαρμογής, καθώς κάθε τεχνολογία έχει τα δικά της πλεονεκτήματα και μειονεκτήματα. Η μελέτη [2] που παρουσιάζεται παρακάτω συγκρίνει τη MongoDB και τη MySQL σε ένα περιβάλλον εφαρμογής που υποστηρίζει μεγάλο αριθμό χρηστών και εκτελεί ταυτόχρονες λειτουργίες. Σύμφωνα με τη μελέτη, οι μη σχεσιακές βάσεις δεδομένων, όπως η MongoDB, προσφέρουν μεγαλύτερη ευελιξία, είναι βελτιστοποιημένες για την αποθήκευση μεγάλων όγκων δεδομένων και επιτρέπουν την εκτέλεση πολλαπλών λειτουργιών από χιλιάδες χρήστες ταυτόχρονα, αυξάνοντας έτσι την απόδοση του συστήματος.

Για τη σύγκριση, αναπτύχθηκαν δύο εκδόσεις της ίδιας εφαρμογής: μία με MySQL και μία με MongoDB. Στο παρακάτω γράφημα παρουσιάζονται τα αποτελέσματα των μετρήσεων απόδοσης.



	insert 2000 users without det.	insert 7000 users with details	update 1000 users (invoicing)	update 2000 users (details)	update 2000 users (profile)	insert 3000 orders	insert 10.000 items	update 5000 items	delete 2000 items
MongoDB	0.0453	0.8891	3.1456	6.6218	6.9922	0.2514	0.8022	23.3427	16.5796
MySQL	0.3184	8.5962	113.7649	320.8543	324.7654	1.4047	35.2240	123.8566	172.4158

Σχήμα 2.2: Σύγκριση της Ταχύτητας των λειτουργιών Insert, Update, Delete μεταξύ της MongoDB και MySQL

Τα αποτελέσματα δείχνουν ότι η MongoDB ολοκληρώνει τις λειτουργίες Insert, Update και Delete ταχύτερα σε σύγκριση με τη MySQL.

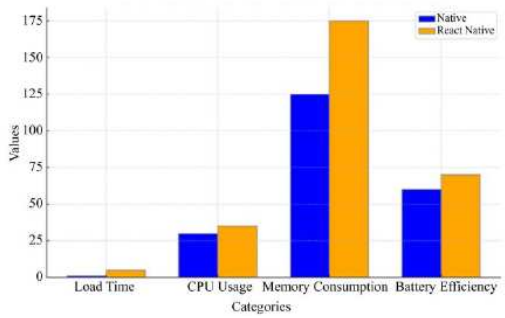
2.2.3 Σύγκριση React Native με Native (Kotlin & Swift)

Στην παρούσα μελέτη [3] αναπτύχθηκε μια εφαρμογή για κινητές συσκευές με δύο διαφορετικούς τρόπους: χρησιμοποιώντας React Native και Native ανάπτυξη.

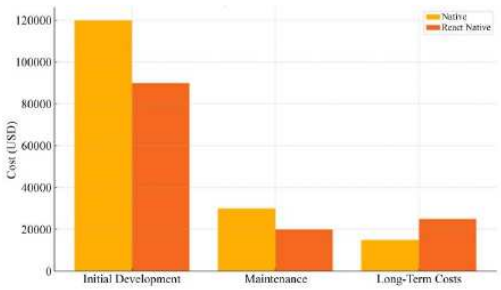
Η εφαρμογή περιλαμβάνει τα παρακάτω χαρακτηριστικά:

- Ταυτοποίηση χρηστών
- Αποθήκευση δεδομένων
- Ειδοποιήσεις
- Πρόσβαση στην κάμερα
- Λειτουργία εκτός σύνδεσης

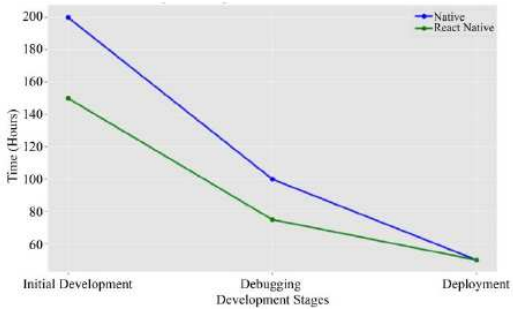
Τα παρακάτω σχήματα απεικονίζουν τις μετρήσεις σχετικά με την απόδοση των δύο εφαρμογών, το κόστος ανάπτυξης, καθώς και τον χρόνο που απαιτήθηκε για την υλοποίησή τους.



(α’)



(β’)



(γ’)

Σχήμα 2.3: (α’) Απόδοση εφαρμογών, (β’) Κόστος ανάπτυξης, (γ’) Χρόνος ανάπτυξης

Απόδοση

- Οι Native εφαρμογές είχαν ταχύτερους χρόνους φόρτωσης.
- Κατανάλωναν λιγότερη μνήμη και CPU, ιδιαίτερα σε απαιτητικές λειτουργίες, όπως η επεξεργασία εικόνας.
- Ήταν πιο αποδοτικές σε κατανάλωση μπαταρίας.

Κόστος

- Το React Native είχε 30% χαμηλότερο κόστος ανάπτυξης λόγω της κοινής βάσης κώδικα.
- Η συντήρηση ήταν ευκολότερη και φθηνότερη καθώς οι ενημερώσεις εφαρμόζονταν και στις δύο πλατφόρμες ταυτόχρονα.

Χρόνος

- Η React Native επέτρεψε 70% ταχύτερη ανάπτυξη σε σχέση με τη native.

Κεφάλαιο 3

Βασικές Τεχνολογίες Ανάπτυξης Ιστοσελίδων και Κινητών Εφαρμογών

Σε αυτό το κεφάλαιο θα γνωρίσουμε τις βασικές τεχνολογίες που χρησιμοποιούνται για τη δημιουργία ιστοσελίδων και κινητών εφαρμογών. Θα εξετάσουμε γλώσσες όπως HTML, CSS και JavaScript, καθώς και τις αρχές που διέπουν το Front-End και το Back-End. Επιπλέον, θα αναλύσουμε τη χρήση βάσεων δεδομένων, Frameworks, μεθόδων HTTP και κωδικών κατάστασης. Θα εξερευνήσουμε επίσης την έννοια των APIs, τον ρόλο του DOM και, τέλος, τις τεχνολογίες για την κατασκευή mobile εφαρμογών, όπως το React Native, οι Native γλώσσες (Kotlin & Swift) και WebView.

3.1 HTML (HyperText Markup Language)

Η HTML είναι μια γλώσσα σήμανσης που χρησιμοποιείται για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Χρησιμοποιεί ένα σύστημα στοιχείων και χαρακτηριστικών για να ορίσει τη διάταξη και το περιεχόμενο μιας σελίδας. Τα στοιχεία HTML αναπαρίστανται με ετικέτες, τις οποίες οι περιηγητές (browsers) ερμηνεύουν για να εμφανίσουν στην σελίδα. Η γλώσσα έχει εξελιχθεί μέσα από πολλές εκδόσεις, με την HTML5 να είναι το τρέχον πρότυπο. Η HTML λειτουργεί σε συνδυασμό με την CSS και την JavaScript [4].

3.1.1 Βασικά Στοιχεία HTML

- `<DOCKTYPE html>` : Δηλώνει ότι το έγγραφο είναι HTML5
- `<html>` : Περιέχει όλο το περιέχομενο του HTML εγγράφου
- `<head>` : Περιέχει πληροφορίες σχετικά με το έγγραφο, όπως ο τίτλος, σύνδεσμοι CSS κλπ
- `<meta charset="UTF-8">` : Δηλώνει την κωδικοποίηση χαρακτήρων του εγγράφου (UTF-8)
- `<title>` : Ορίζει τον τίτλο της σελίδας που εμφανίζεται στη γραμμή τίτλου του προγράμματος περιήγησης
- `<link>` : Χρησιμοποιείται για σύνδεση με εξωτερικά αρχεία, όπως CSS
- `<body>` : Το κύριο μέρος της σελίδας που περιέχει το περιεχόμενο που εμφανίζεται στον χρήστη
- `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` : Ετικέτες κεφαλίδων για τίτλους, από το πιο μεγάλο `h1` μέχρι το πιο μικρό `h6`
- `<p>` : Δημιουργεί μια παράγραφο
- `<a>` : Δημιουργεί έναν υπερσύνδεσμο (link)
- `<img>` : Εισάγει μια εικόνα στην σελίδα
- `<ul>` : Δημιουργεί μια μη-ταξινομημένη λίστα
- `<ol>` : Δημιουργεί μια αριθμημένη λίστα
- `<form>` : Δημιουργεί μια φόρμα για την εισαγωγή δεδομένων από τον χρήστη
- `<button>` : Δημιουργεί ένα κουμπί
- `<div>` : Δημιουργεί ένα γενικό μπλοκ για οργάνωση του περιεχομένου
- `<br>` : Δημιουργεί μια οριζόντια γραμμή

3.2 CSS (Cascading Style Sheets)

Η CSS είναι μια γλώσσα που χρησιμοποιείται για να διαμορφώσει την εμφάνιση ενός HTML εγγράφου. Μέσω της CSS μπορούμε να καθορίσουμε τη διάταξη, τα χρώματα, τις γραμματοσειρές, τις αποστάσεις και πολλές άλλες οπτικές παραμέτρους της ιστοσελίδας [5].

3.3 JavaScript

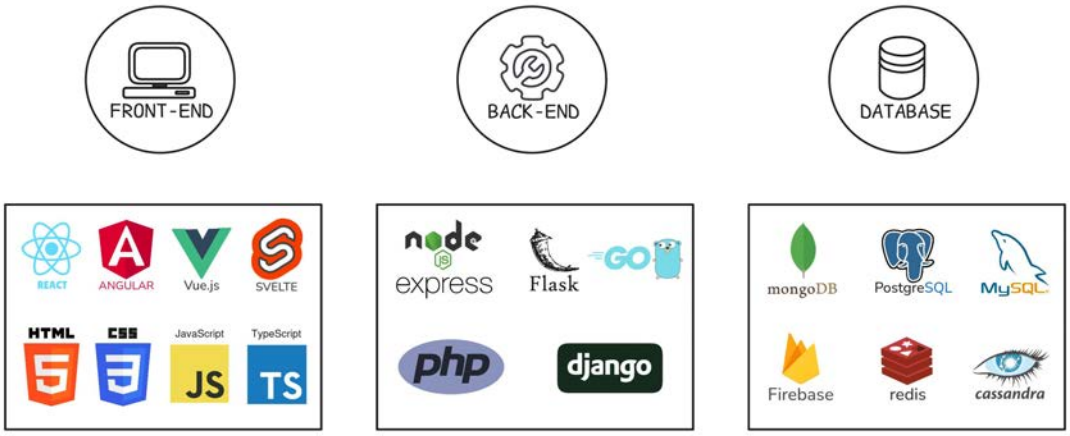
Η JavaScript ειναι μια γλώσσα προγραμματισμού δυναμικής τυποποίησης και είναι απο τις πιο δημοφιλείς στον κόσμο του web development. Πρόκειται για μια γλώσσα υψηλού επιπέδου, που εκτελείται σε πραγματικό χρόνο απο τον browser, επιτρέποντας την δημιουργία δυναμικών ιστοσελίδων. Χρησιμοποιείται σε συνδυασμό με την HTML και την CSS, παρέχοντας λειτουργίες όπως η δυναμική ενημέρωση του περιεχομένου της σελίδας και η αλληλεπίδραση με το Back-End. Χρησιμοποιεί τεχνικές όπως promises και async/await για να διαχειριστεί ασύγχρονες εργασίες χωρίς να μπλοκάρει τη λειτουργία του προγράμματος[6].

3.3.1 TypeScript

Η TypeScript είναι ένα υπερσύνολο της JavaScript, η οποία προσφέρει επιπλέον δυνατότητες για την ανάπτυξη εφαρμογών. Η TypeScript σε αντίθεση με την JavaScript, υποστηρίζει στατική τυποποίηση, δηλαδή οι προγραμματιστές μπορούν να καθορίσουν τύπους δεδομένων για τις μεταβλητές, τις παραμέτρους και τις συναρτήσεις. Αυτό σημαίνει ότι ο τύπος των δεδομένων ελέγχεται κατά τη διάρκεια της ανάπτυξης και όχι κατά την εκτέλεση της εφαρμογής, μειώνοντας έτσι τα πιθανά σφάλματα. Επίσης η Typescript κάνει τον κώδικα πιο κατανοητό και οργανωμένο [7].

3.4 FullStack (Πλήρης Στοίβα)

Η FullStack ανάπτυξη αναφέρεται στη δημιουργία εφαρμογών που καλύπτουν τόσο την πλευρά του πελάτη (Front-End) όσο και την πλευρά του διακομιστή (Back-End). Ένας Full Stack Developer ασχολείται με το σύνολο της ανάπτυξης ενός εργου, αναλαμβάνοντας τη σχεδίαση του οπτικού περιβάλλοντος στο Front-End, καθώς και τη διαχείριση της λογικής και των δεδομένων της εφαρμογής στο Back-end [8].



Σχήμα 3.1: Τεχνολογίες Full-Stack

3.4.1 Front-End (Εμπρόσθιο τμήμα)

React

Το React είναι ενα JavaScript Framework ανοιχτού κώδικα για τη δημιουργία διεπαφών χρηστη (User Interface). Αναπτύχθηκε απο την Facebook και επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα Components που ενημερώνονται καθώς τα δεδομένα αλλάζουν. Τα Components μπορούν να περιέχουν για παράδειγμα κάποια buttons που ο χρηστης μπορεί να χρησιμοποιήσει για να αλληλεπιδράσει με αλλους χρήστες ή ένα μέρος ολόκληρης της σελίδας. Το React, με τη βοήθεια της βιβλιοθήκης react-router-dom, επιτρέπει την πλοήγηση μεταξύ διαφορετικών σελίδων μέσα στην εφαρμογή χωρίς να απαιτείται πλήρης ανανέωση της σελίδας. Αντί να φορτώνεται εκ νέου ολόκληρη η εφαρμογή, αλλάζει μόνο το περιεχόμενο [9].

Angular

Το Angular είναι ενα Framework ανοιχτού κώδικα για την ανάπτυξη διαδικτυακών εφαρμογών που αναπτύχθηκε απο την Google. Χρησιμοποιεί TypeScript αντί για JavaScript που βοηθά στον εντοπισμό λαθών κατά τη φάση ανάπτυξης. Αποτελεί μία αρκετά περίπλοκλη βιβλιοθήκη, ως προς την εκμάθηση της [10].

Vue

Το Vue είναι ένα Framework ανοιχτού κώδικα που δημιουργήθηκε το 2014. Η αρχική ιδέα ήταν να είναι μια βιβλιοθήκη εύκολη στην χρήση της, συνδυάζοντας τις ήδη υπάρχουσες

React και Angular. Αποτελεί ιδανική λύση για κάποιον με μικρή εμπειρία στην σχεδίαση ιστοσελίδων. Οι αναλυτικές οδηγίες (documentation) έχουν συμβάλει στην υιοθέτηση του απο την κοινότητα [11].

Svelte

Το Svelte είναι ένα Framework το οποίο διαφέρει από άλλα Frameworks όπως το React, καθώς το μεγαλύτερο μέρος της δουλειάς του γίνεται κατά τη μεταγλώττιση (compile time), αντί να εκτελεί λειτουργίες στο runtime. Αυτό σημαίνει ότι ο κώδικας μετατρέπεται σε JavaScript κατά τη διαδικάσια ανάπτυξης, με αποτέλεσμα οι εφαρμογές να ειναι ελαφρύτερες και ταχύτερες. Συνδυάζεται με το SvelteKit, το οποίο παρέχει λειτουργίες όπως routing, server-side rendering (SSR) και preloading [12].

3.4.2 Back-End (Οπίσθιο τμήμα)

NodeJS

Το Node.js είναι ένα περιβάλλον εκτέλεσης, το οποίο επιτρέπει την εκτέλεση JavaScript εκτός του προγράμματος περιήγησης (browser). Χρησιμοποιείται κυρίως για την ανάπτυξη server-side εφαρμογών, όπως API requests και επικοινωνία με βάσεις δεδομένων. Το Node.js είναι ιδανικό για εφαρμογές Back-End, καθώς χρησιμοποιεί τον κινητήρα V8 της Google για γρήγορη εκτέλεση του κώδικα. Χάρη στην ασύγχρονη επεξεργασία και το event-driven μοντέλο το Node.js μπορεί να χειρίζεται ταυτόχρονα πολλές συνδέσεις και να εκτελεί εργασίες σε εξαιρετική απόδοση[13].

Στο Node.js το event-driven μοντέλο σημαίνει ότι το σύστημα περιμένει για συμβάντα και αντί να μπλοκάρει τη λειτουργία του κώδικα – περιμένοντας για κάποια δεδομένα – συνεχίζει να επεξεργάζεται άλλα αιτήματα και μόλις το συμβάν (π.χ απάντηση απο τη βάση δεδομένων) είναι έτοιμο, το σύστημα το “αντιλαμβάνεται” και το χειρίζεται άμεσα.

ExpressJS

Το Express.js είναι ένα δημοφιλές Framework για το Node.js που κάνει τη δημιουργία εφαρμογών πολύ πιο εύκολη και οργανωμένη [14]. Οι βασικές δυνατότητες του Express.js περιλαμβάνουν:

- Μια πιο απλή μέθοδο για την διαχείριση των routes (διαδρομών). Μπορούμε να ορίσουμε ποιες συναρτήσεις θα εκτελούνται όταν ο χρήστης επισκέπτεται μια συγκεκριμένη διεύθυνση URL.

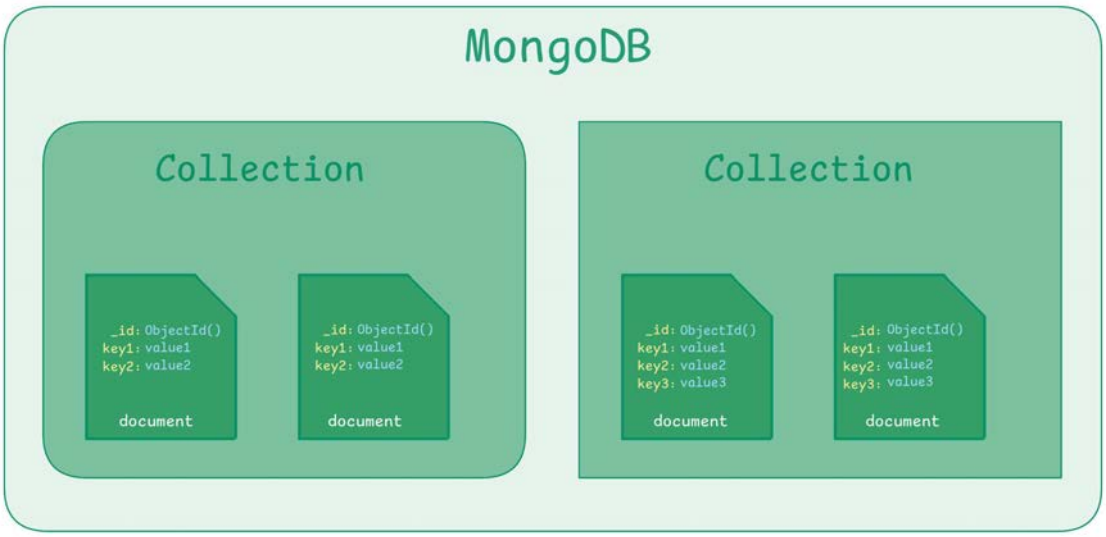
- Κάνει την διαχείριση των αιτημάτων HTTP (GET, POST, PUT, DELETE) πολύ πιο απλή, με την χρήση `app.get` , `app.post` , `app.put` , `app.delete` κ.λπ.

- Υποστηρίζει την χρήση ενδιάμεσων συναρτήσεων πριν ή μετά την επεξεργασία ενός αιτήματος. Αυτό είναι ιδιαίτερα χρήσιμο για λειτουργίες όπως ο έλεγχος ταυτότητας (authentication) ή για Διαχείριση Σφαλμάτων (error handling).

3.4.3 DataBase (Βάση Δεδομένων)

MongoDB

Η MongoDB είναι μια μη σχεσιασκή βάση δεδομένων NoSQL που χρησιμοποιεί ένα μη σχεσιακό μοντέλο αποθήκευσης δεδομένων, βασισμένο σε συλλογές (collections) και έγγραφα (documents). Αντι για τους πίνακες, τα δεδομένα αποθηκέυονται σε μορφή BSON (πάροιμοιο με το JSON). Αυτή η δομή την καθιστά ιδιαίτερα δημοφιλή για έργα όπου τα δεδομένα μπορεί να αλλάζουν συχνά [15].



Σχήμα 3.2: Δομή της Βάσης δεδομένων MongoDB

[16]

Βασικοί τελεστές της MongoDB

updateOne / updateMany:

- `$set`: Χρησιμοποιείται για να ενημερώσει συγκεκριμένα πεδία σε ένα έγγραφο.
- `$inc`: Αυξάνει ή μειώνει την τιμή ενός αριθμητικού πεδίου κατά συγκεκριμένη ποσότητα.
- `$push`: Προσθέτει ένα στοιχείο σε έναν πίνακα μέσα σε ένα έγγραφο.
- `$pull`: Αφαιρεί ένα στοιχείο από έναν πίνακα μέσα σε ένα έγγραφο.

find:

- `$and`: Επιστρέφει τα έγγραφα που ικανοποιούν όλες τις συνθήκες που του δίνονται (όλα τα `and`).
- `$or`: Επιστρέφει τα έγγραφα που ικανοποιούν τουλάχιστον μία από τις συνθήκες που δίνονται.
- `$in`: Επιστρέφει τα έγγραφα όπου η τιμή ενός πεδίου περιλαμβάνεται σε έναν πίνακα τιμών.
- `$all`: Επιστρέφει τα έγγραφα όπου το πεδίο περιέχει όλες τις τιμές που δίνονται σε έναν πίνακα. Αυτό σημαίνει ότι το πεδίο πρέπει να περιέχει όλα τα στοιχεία του πίνακα.

Παρακάτω παρουσιάζονται παραδείγματα για την Δημιουργία, Αναζήτηση, Επεξεργασία και Διαγραφή περιεχομένου στην MongoDB:

Σχήμα 3.3: Παράδειγμα εντολών στην MongoDB για CRUD (Create, Read, Update, Delete)

PostgreSQL

Η PostgreSQL είναι μία σχεσιακή βάση δεδομένων (RDBMS), χρησιμοποιεί πίνακες, γραμμές και στήλες για την αναπαράσταση των δεδομένων. Υποστηρίζει σύνθετα ερωτήματα, ξένα κλειδιά, και αναζήτηση κειμένου. Με την χρήση ξένων κλειδιών μπορούμε να διασυνδέσουμε πίνακες μεταξύ τους. Παρέχει υποστήριξη ACID (Atomicity, Consistency, Isolation, Durability), δηλαδή εξασφαλίζει ότι οι συναλλαγές εκτελούνται αξιόπιστα και με ασφάλεια. Χρησιμοποιεί MVCC (Multi-Vesion Concurrency Control) για να επιτρέπει πολλαπλές ταυτόχρονες συναλλαγές [17].

Firebase

Η Firebase είναι μια πλατφόρμα της Google που χρησιμοποιείται συνήθως για την ανάπτυξη εφαρμογών για κινητά. Παρέχει ένα σύνολο εργαλείων που περιλαμβάνουν αποθήκευση δεδομένων, αυθεντικοποίηση χρηστών και άλλα. Με την Firebase, οι προγραμματιστές επικεντρώνονται στην δημιουργία User Interface, χωρίς να ανησυχούνε για τις τεχνικές λεπτομέρειες του Back-End [18]. Παρακάτω θα δούμε μερικά κύρια χαρακτηριστικά που προσφέρει η Firebase:

- **Realtime Database:** Η βάση δεδομένων που προσφέρει είναι NoSQL και λειτουργεί σε πραγματικό χρόνο. Αυτό σημαίνει ότι, κάθε αλλαγή που πραγματοποιείται στα δεδομένα συγχρονίζεται άμεσα σε όλους τους χρήστες της εφαρμογής.

- **Authentication:** Παρέχει μια εύκολη λύση για την αυθεντικοποίηση χρηστών, με υποστήριξη για email, κωδικούς πρόσβασης και αριθμούς τηλεφώνου.

- **Cloud Storage:** Προσφέρει αποθηκευτικό χώρο για περιεχόμενο που δημιουργείται από τους χρήστες, όπως φωτογραφίες και βίντεο.

3.4.4 Full Stack Frameworks (Πλαίσια πλήρους στοίβας)

Τα τελευταία χρόνια, η ανάπτυξη εφαρμογων ιστού έχει στραφεί προς τα Full Stack Frameworks που επιτρέπουν στους προγραμματιστές να δημιουργούν υψηλής απόδοσης εφαρμογές. Το Next.js και το Remix.js είναι δύο τέτοια Frameworks που έχουν κερδίσει την δημοτικότητα, και έχουν σχεδιαστεί για χρήση με το React [19].

Σχήμα 3.4: Τεχνολογίες Full-Stack

Τα πλαίσια πλήρους στοίβας προσφέρουν εργαλεία όπου:

- **Κάνουν την εφαρμογή πιο γρήγορη:** Οι σελίδες φορτώνουν ταχύτερα όταν ο χρήστης επισκέπτεται έναν ιστότοπο.
- **Διευκολύνουν την ανάπτυξη:** Παρέχουν έτοιμα εργαλεία για καλύτερη οργάνωση του κώδικα, μειώνοντας τον χρόνο ανάπτυξης.

3.5 HTTP Methods

Οι HTTP μέθοδοι είναι εντολές που χρησιμοποιούνται για την αλληλεπίδραση ενός πελάτη (client) με έναν διακομιστή (server). Αυτές οι μέθοδοι καθορίζουν το είδος της ενέργειας που επιθυμεί ο πελάτης, όπως η Ανάκτηση δεδομένων, η Αποστολή νέων δεδομένων, η Ενημέρωση των δεδομένων, η Διαγραφή δεδομένων. [20].

Οι κύριες HTTP μέθοδοι είναι οι εξής :

HTTP Methods	Meaning
GET	Χρησιμοποιείται για την Ανάκτηση δεδομένων
POST	Χρησιμοποιείται για τη Δημιουργία νέων δεδομένων
PUT	Χρησιμοποιείται για την Ενημέρωση δεδομένων
DELETE	Χρησιμοποιείται για τη Διαγραφή δεδομένων

Πίνακας 3.1: Βασικοί μέθοδοι HTTP

3.6 HTTP Status Codes

Οι HTTP Status Codes (Κωδικοί Κατάστασης) είναι αριθμητικοί κωδικοί που επιστρέφονται από τον διακομιστή (server) όταν γίνεται ένα αίτημα από εναν πελάτη (client). Οι

κωδικοί αυτοί χρησιμοποιούνται για να ενημερώσουν τον πελάτη για την κατάσταση της αιτησης [21].

Οι κύριοι Κωδικοί Κατάστασης είναι οι εξής:

Status Code	**Meaning**
200 (OK)	Επιτυχές αίτημα.
201 (CREATED)	Δημιουργία νέου πόρου.
202 (ACCEPTED)	Το αίτημα έγινε δεκτό, αλλά δεν έχει γίνει επεξεργασία ακόμα.
204 (NO CONTENT)	Επιτυχές αίτημα χωρίς περιέχομενο.
301 (MOVED PERMANENTLY)	Ο πόρος έχει μετακινηθεί μόνιμα σε μια νέα διευθυνση.
400 (BAD REQUEST)	Λανθασμένο αίτημα.
401 (UNAUTHORIZED)	Μη εξουσιοδοτημένος χρήστης.
403 (FORBIDDEN)	Απαγόρευση πρόσβασης.
404 (NOT FOUND)	Ο πόρος δεν βρέθηκε.
500 (INTERNAL SERVER ERROR)	Σφάλμα στον διακομιστή.

Πίνακας 3.2: Βασικοί κωδικοί κατάστασης

3.7 JSX (JavaScript Syntax Extension)

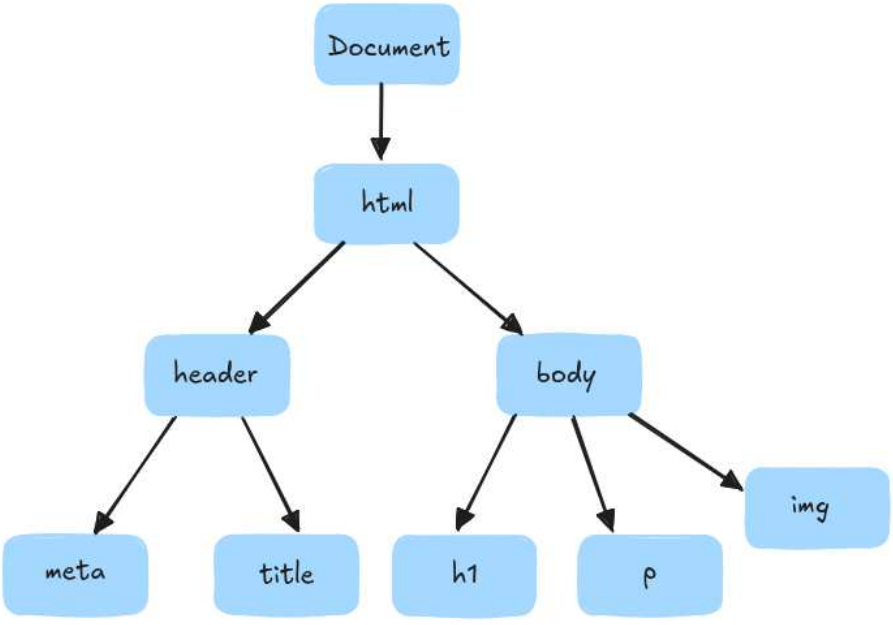
JSX είναι μια επέκταση της σύνταξης JavaScript. Επιτρέπει στους προγραμματιστές να γράφουν όμοιο κώδικα με την HTML απευθείας μέσα στη JavaScript. Στο παρακάτω σχήμα μπορούμε να παρατηρήσουμε τις ομοιότητες της JSX με την HTML [22].

```jsx
import React from "react";

const App = () => {
  const handleClick = () => {
    alert("Button clicked!");
  };

  return (
    <div>
      <h1>University of Thessaly</h1>
      <p>Electrical and Computer Engineer</p>
      <button onClick={handleClick}>Submit</button>
    </div>
  );
};

export default App;
```

Σχήμα 3.5: Παράδειγμα JSX στην React

3.8 DOM (Document Object Model)

Το DOM είναι ένας τρόπος για να αναπαραστήσουμε το HTML έγγραφο ως ένα ιεραρχικό δέντρο με αντικείμενα. Κάθε στοιχείο στο έγγαφο, όπως τίτλοι, παράγραφοι, εικόνες κλπ , μετατρέπεται σε κόμβο στο δέντρο DOM. [23]



Σχήμα 3.6: Ιεραρχικό δέντρο DOM

Στο επάνω μέρος του δέντρου βρίσκεται ο κόμβος εγγράφου (document node) που αντιπροσωπεύει ολόκληρο το έγγραφο HTML. Από εκεί, το δέντρο διακλαδίζεται σε κόμβους στοιχείων όπως `<meta>` `<title>` και `<h1>` `<p>` `<img>`.

Ο βασικός σκοπός του DOM είναι ότι επιτρέπει στους προγραμματιστές να αλληλεπιδρούν με την HTML από την JavaScript για να τροποποιούν τη σελίδα δυναμικά.

Virtual DOM

Στο React, η διαχείριση των στοιχείων της σελίδας γίνεται μέσω του Virtual DOM. Το Virtual DOM είναι μια αναπαράσταση του πραγματικού DOM και χρησιμοποιείται για να κάνει το React πιο αποδοτικό στην ανανέωση του περιεχομένου. Αντί να ανανεώνει ολόκληρη την σελίδα, το React συγκρίνει το Virtual DOM με το πραγματικό DOM και κάνει μόνο τις απαραίτητες αλλαγές [24].

3.9 APIs (Application Programming Interfaces)

Τα APIs είναι σύνολα κανόνων και πρωτοκόλλων που επιτρέπουν σε διαφορετικές εφαρμογές να επικοινωνούς μεταξύ τους. Ένα API παρέχει έναν τυποποιημένο τρόπο για να στείλεις αιτήματα σε μια υπηρεσία και να λάβεις δεδομένα.

Στην React τα APIs χρησιμοποιούνται συχνά για την αλληλεπίδραση με εξωτερικές υπηρεσίες ή την απόκτηση δεδομένων, όπως την ανάκτηση δεδομένων απο έναν διακομιστή ή την αποστολή δεδομένων σε αυτον.[25].

3.9.1 REST API (Representational State Transfer)

Ένα από τα πιο δημοφιλή APIs είναι το REST (Representational State Transfer). Το REST βασίζεται σε HTTP αιτήματα (GET, POST, PUT, DELETE).

Τα βασικά χαρακτηριστικά του REST περιλαμβάνουν:

- **Stateless**: Κάθε αίτημα από έναν πελάτη προς τον διακομιστή περιέχει όλες τις απαραίτητες πληροφορίες για να κατανοηθεί και να επεξεργαστεί.
- **Client-Server**: Διαχωρισμός μεταξύ πελάτη και διακομιστή. Ο πελάτης κάνει αιτήματα και ο διακομιστής απαντά.
- **Cacheable**: Οι απαντήσεις μπορούν να αποθηκευτούν σε προσωρινή μνήμη για να μειώσουν τον χρόνο απόκρισης.
- **Uniform Interface**: Ο τρόπος με τον οποίο γίνονται τα αιτήματα και οι απαντήσεις είναι καθορισμένος και κατανοητός.

3.9.2 GraphQL

Το GraphQL είναι μια εναλλακτική λύση που αναπτύχθηκε από τη Meta. Επιτρέπει στους πελάτες να ζητούν ακριβώς τα δεδομένα που χρειάζονται, αποφεύγοντας τη μεταφορά περιττών δεδομένων [26].

Μερικά από τα βασικά χαρακτηριστικά του GraphQL είναι τα εξής:

- **Αποφυγή Υπερβολικών Αιτημάτων (Over-fetching):** Ο πελάτης ζητά μόνο τα δεδομένα που χρειάζεται, μειώνοντας τη χρήση του δικτύου.

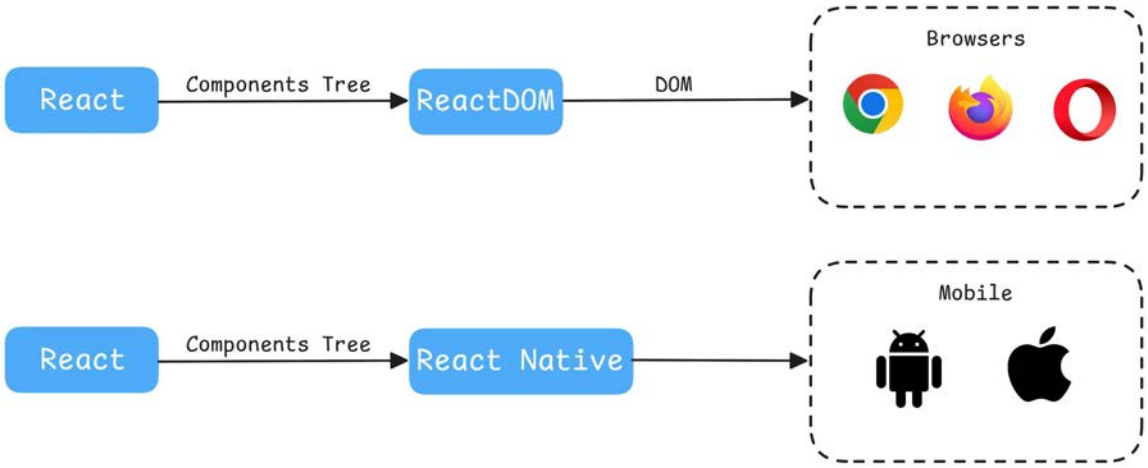
- **Ευκολία στη Λήψη Δεδομένων (Under-fetching):** Όλα τα απαραίτητα δεδομένα μπορούν να ληφθούν με ένα μόνο αίτημα, αντί για πολλαπλά αιτήματα σε διαφορετικά endpoints.

- **Ενημερώσεις σε Πραγματικό Χρόνο (Real-time):** Με τη χρήση subscriptions, το GraphQL μπορεί να στέλνει ενημερώσεις όταν αλλάζουν τα δεδομένα σε πραγματικό χρονο.

3.10 Mobile Development (Ανάπτυξη Κινητών Εφαρμογών)

3.10.1 React Native

Η ανάπτυξη εφαρμογών για κινητά τηλέφωνα απαιτεί τη χρήση ξεχωριστών γλωσσών προγραμματισμού, όπως Java / Kotlin για Android και Objective-C / Swift για iOS. Για να αντιμετωπίσει αυτό το πρόβλημα, η Meta ανέπτυξε το React Native, ένα Framework που επιτρέπει τη δημιουργία εφαρμογών για κινητές συσκευές τόσο για Android όσο και για iOS, χρησιμοποιώντας μία μόνο γλώσσα προγραμματισμού, τη JavaScript.

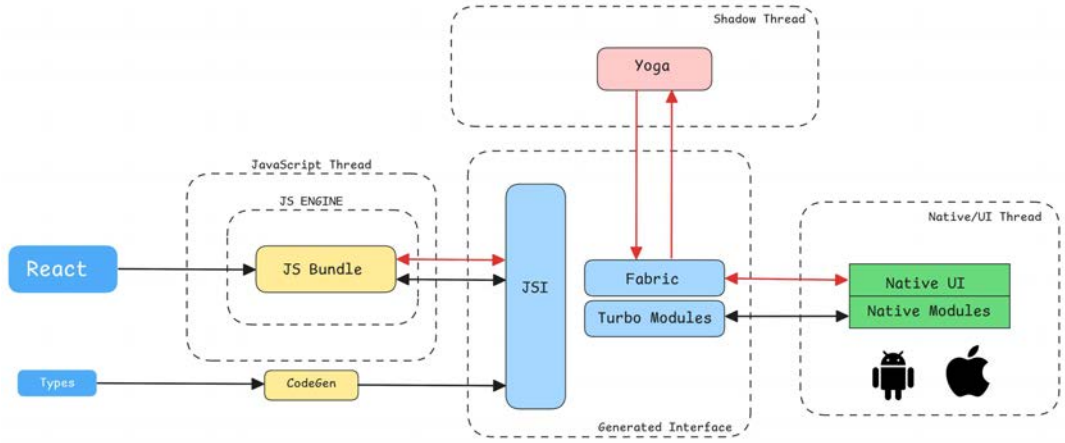


Σχήμα 3.7: Ροή ReactDOM και React Native[27]

Η React Native λειτουργεί ως γέφυρα μεταξύ του JavaScript κώδικα και των Native στοιχείων του εκάστοτε λειτουργικού συστήματος. Αντί για HTML και CSS, χρησιμοποιεί συγκεκριμένα στοιχεία (Components) τα οποία μεταφράζονται στα αντίστοιχα Native στοιχεία της πλατφόρμας.

Παρακάτω παρατίθενται τα πιο σημαντικά στοιχεία που συναντώνται σε μια εφαρμογή React Native [28] :

- `<View>`: Το βασικό στοιχείο κάθε εφαρμογής. Αντιστοιχεί στο στοιχείο `<div>` του HTML.
- `<ScrollView>`: Ένα στοιχείο που επιτρέπει στους χρήστες να κάνουν κύλιση περιεχομένου που δεν χωρά στην οθόνη.
- `<Text>`: Χρησιμοποιείται για την εμφάνιση κειμένου. Αντίστοιχο του HTML `<p>`.
- `<Image>`: Εμφανίζει εικόνες από διάφορες πηγές, όπως δίκτυο ή τοπικά αρχεία.
- `<TextInput>`: Επιτρέπει στους χρήστες να εισάγουν κείμενο μέσω πληκτρολογίου.
- `<Button>`: Δημιουργία κουμπιών.
- `<FlatList>`: Χρησιμοποιείται για την απόδοση μεγάλων λιστών δεδομένων.



Σχήμα 3.8: Αρχιτεκτονική React Native[29]

3.10.2 Native Γλώσσες

Παρόλο που frameworks όπως το React Native διευκολύνουν την ανάπτυξη εφαρμογών για κινητές συσκευές, οι Native γλώσσες προγραμματισμού, όπως η Kotlin και η Swift, παραμένουν οι πιο δημοφιλείς επιλογές. [30]. Οι βασικοί λόγοι είναι οι εξής:

- **Απόδοση:** Οι εφαρμογές εκτελούνται ταχύτερα, καθώς ο κώδικας μεταγλωττίζεται απευθείας σε Native κώδικα της πλατφόρμας. Αυτο σημαίνει οτι δεν απαιτείται γέφυρα όπως στην React Native για μετάφραση της JavaScript σε Native στοιχεία.

- **Πρόσβαση στα χαρακτηριστικά της πλατφόρμας:** Οι Native γλώσσες παρέχουν πλήρη πρόσβαση σε όλα τα APIs και τις βιβλιοθήκες του λειτουργικού συστήματος. Σε ορισμένες περιπτώσεις, λειτουργίες που απαιτούν πλήρη εκμετάλλευση των δυνατοτήτων της πλατφόρμας δεν μπορούν να υλοποιηθούν με React Native λόγω περιορισμών στις διαθέσιμες βιβλιοθήκες ή τη γέφυρα.

- **Υποστήριξη:** Η Kotlin υποστηρίζεται απο την Google για την ανάπτυξη εφαρμογών Android, ενώ η Swift αποτελεί την επίσημη γλώσσα της Apple για εφαρμογές iOS.

3.10.3 WebView

Το Webview είναι ένας ακόμη τρόπος ανάπτυξης εφαρμογών για κινητές συσκευές. Ενσωματώνει έναν browser μέσα στην εφαρμογή, και επιτρέπει την φόρτωση μιας ιστοσελίδας. Η διαδικασία, λειτουργεί ως εξης:

- **Ανάπτυξη web εφαρμογής:** Πρώτα αναπτύσσουμε μια εφαρμογη ιστού με HTML, CSS και JavaScript.

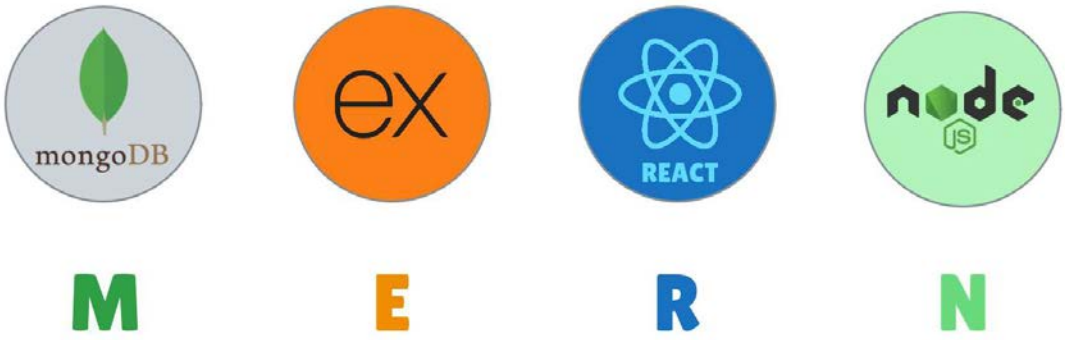
- **Πακετάρισμα:** Και στην συνέχεια την ”πακετάρουμε” με εργαλεία όπως το Cordova ή Capacitor για να την μετατρέψουμε σε εφαρμογή κινητού.

Τα πλεονεκτήματα που προσφέρει αυτή η προσεγγιση είναι η γρήγορη ανάπτυξη, ειδικά όταν έχουμε ήδη μια έτοιμη εφαρμογή ιστού και θέλουμε να την μετατρέψουμε σε mobile εφαρμογή χρησιμοποιώντας τον ίδιο ακριβώς κώδικα. Ωστόσο, το κύριο μειονέκτημα είναι η χαμηλότερη απόδοση σε σύγκριση με Native εφαρμογές [31].

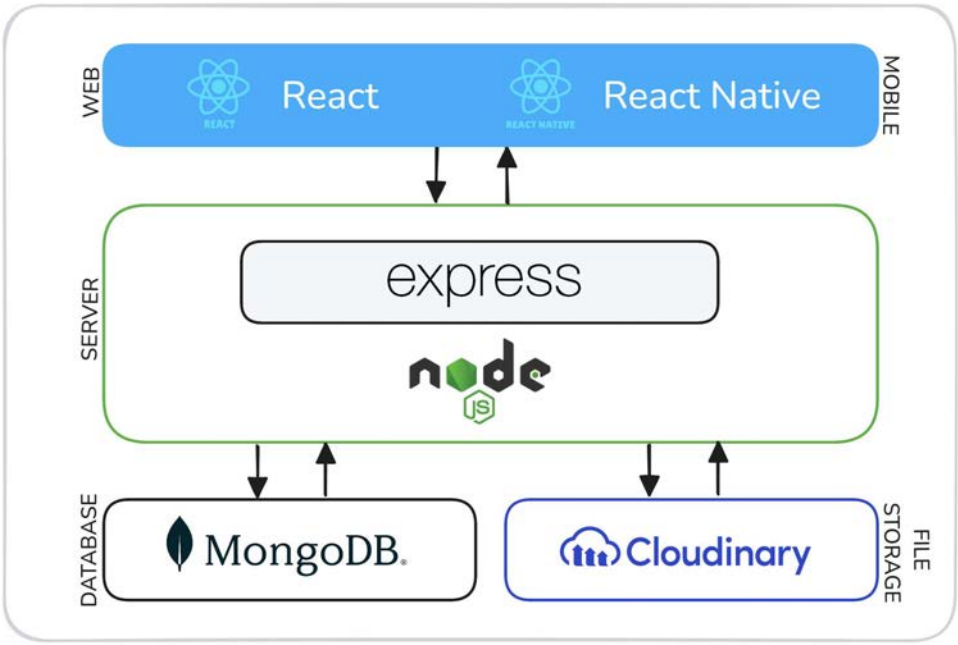
Κεφάλαιο 4

Ανάπτυξη της Εφαρμογής με MERN και React Native

Στο κεφάλαιο αυτό παρουσιάζονται τα βήματα για την ανάπτυξη της εφαρμογής χρησιμοποιώντας την MERN στοίβα. Το MERN είναι ένα απο τα πιο δημοφιλή σύνολα εργαλείων για την ανάπτυξη εφαρμογών. Το όνομα της στοίβας προέρχεται απο τις τέσσερις βασικές τεχνολογίες που το απαρτίζουν [32].



Σχήμα 4.1: Οι τεχνολογίες του MERN Stack



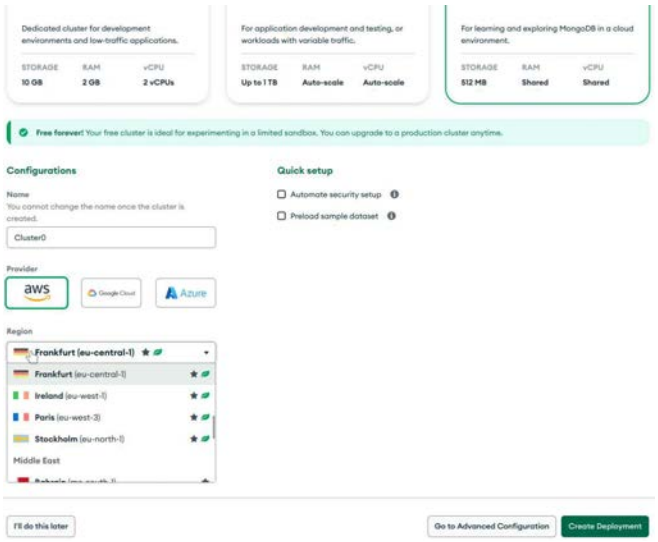
Σχήμα 4.2: Η Αρχιτεκτονική της Εφαρμογής

4.1 MongoDB

Για την υλοποίηση αυτής της εφαρμογής επιλέξαμε το MongoDB Atlas, μια cloud-based βάση δεδομένων. Στην παρούσα εφαρμογή χρησιμοποιείται για την αποθήκευση και διαχείριση δεδομένων που αφορούν χρήστες, δημοσιεύσεις και μηνύματα.

Αφού δημιουργήσουμε έναν λογαριασμό στον ιστότοπο του MongoDB Atlas, επιλέγουμε “New Project”, όπου δίνουμε ένα όνομα στο project και πατάμε “Create Project” [33].

Στη συνέχεια πατάμε “Create a Cluster” όπου μεταβαίνουμε στην παρακάτω σελίδα:



Σχήμα 4.3: Δημιουργία Cluster

Επιλέγουμε Cloud Provider (AWS, Google Cloud, Azure) και Region (π.χ. Frankfurt) και ολοκληρώνουμε με την επιλογή “Create Deployment”. Μετά τη δημιουργία της βάσης, θα εμφανιστεί ένα παράθυρο στο οποίο ορίζουμε:

- Τους χρήστες που θα έχουν πρόσβαση στην βάση δεδομένων
- Τις IP διευθύνσεις που θα επιτρέπεται να συνδεθούν στην βάση δεδομένων

Τέλος, το MongoDB Atlas παρέχει ένα URI, το οποίο θα χρησιμοποιηθεί για την σύνδεση του Node.js server με την βάση δεδομένων:

```
mongodb+srv://<username>:<password>@cluster.mongodb.net/
<database>?retryWrites=true&w=majority
```

4.1.1 Collections

Έχουμε ξανά αναφέρει στο Κεφάλαιο 3, ότι η MongoDB αποθηκεύει τα δεδομένα σε collections και documents. Σε αυτή την εφαρμογή έχουμε τέσσερα collections: `users`, `posts`, `messages`, `conversations`.

User Collection

- `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για κάθε χρήστη.
- `name`: Το όνομα του χρήστη.

- `username`: Το ψευδώνυμο του χρήστη.
- `email`: Η διεύθυνση email του χρήστη.
- `password`: Ο κωδικός πρόσβασης του χρήστη.
- `profilePic`: URL για την εικόνα προφίλ του χρήστη.
- `followers`: Μία λίστα με τα `_id` των χρηστών που ακολουθούν τον συγκεκριμένο χρήστη.
- `following`: Μια λίστα με τα `_id` των χρηστών που ακολουθεί ο συγκεκριμένος χρήστης.
- `bio`: Σύντομο κείμενο του χρήστη.
- `isFrozen`: Boolean τιμή που δηλώνει εάν ο λογαριασμός του χρήστη έχει παγώσει.
- `verificationCode`: Ένας 6-ψήφιος κωδικός επιβεβαίωσης που αποστέλλεται στο email του χρήστη κατά τη διαδικασία εγγραφής, για να επαληθευτεί ο λογαριασμός του.
- `verificationCodeExpiry`: Η ημερομηνία και ώρα λήξης του κωδικού επιβεβαίωσης. Μετά από αυτήν την ώρα, ο 6-ψήφιος κωδικός παύει να ισχύει.
- `isVerified`: Μια Boolean τιμή που δηλώνει εάν ο λογαριασμός του χρήστη έχει επαληθευτεί.
- `resetPasswordCode`: Ένας 6-ψήδιος κωδικός επαναφοράς κωδικού πρόσβασης που αποστέλλεται στο email του χρήστη όταν ζητάει επαναφορά του κωδικού του.
- `resetPasswordCodeExpiry`: Η ημερομηνία και ώρα λήξης του κωδικού επαναφοράς. Μετά από αυτήν την ώρα, ο 6-ψήφιος κωδικός παύει να ισχύει.

Post Collection

- `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για κάθε δημοσίευση.
- `postedBy`: Το `_id` του χρήστη που δημιούργησε τη δημοσίευση.
- `text`: Το κείμενο της δημοσίευσης (500 χαρακτήρες).

- `img`: URL της εικόνας για τη δημοσίευση.
- `likes`: Λίστα με τα `_id` των χρηστών που έχουν κάνει like στη δημοσίευση.
- `replies`: Λίστα με τα σχόλια που έχουν προστεθεί στη δημοσίευση. Κάθε σχόλιο περιλαμβάνει:
  - `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για το σχόλιο.
  - `userId`: Το `_id` του χρήστη που δημιούργησε το σχόλιο.
  - `text`: Το περιεχόμενο του σχόλιου.
  - `userProfilePic`: URL της εικόνας προφίλ του χρήστη που σχολίασε.
  - `username`: Το όνομα χρήστη που του ανήκει το σχόλιο.

Messages Collection

- `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για κάθε μήνυμα.
- `conversationId`: Το `_id` της συνομιλίας στην οποία ανήκει το μήνυμα.
- `sender`: Το `_id` του χρήστη που έστειλε το μήνυμα.
- `text`: Το περιεχόμενο του μηνύματος.
- `seen`: Boolean τιμή που δηλώνει αν ο παραλήπτης έχει διαβάσει το μήνυμα.
- `img`: URL της εικόνας που στέλνει ο χρήστης σε μήνυμα.

Conversations Collection

- `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για κάθε συνομιλία.
- `participants`: Λίστα με τα `_id` των δύο χρηστών που συμμετέχουν στη συνομιλία.
- `lastMessage`: Πληροφορίες για το τελευταίο μήνυμα της συνομιλίας:
  - `_id`: Ένας μοναδικός αναγνωριστικός αριθμός για το τελευταίο μήνυμα στην συνομιλία.
  - `text`: Το περιεχόμενο του τελευταίου μηνύματος.

- `sender`: Το `_id` του χρήστη που έστειλε το τελευταίο μήνυμα.
- `seen`: Βοολεαν τιμή που δηλώνει αν το τελευταίο μήνυμα έχει διαβαστεί.

4.2 NodeJS & ExpressJS

Η υλοποίηση του Back-End γίνεται με τη χρήση του Node.js και του Express.js. Αρχικά δημιουργούμε έναν φάκελο και στη συνέχεια με την παρακάτω εντολή

```
~$ npm init -y
```

δημιουργείται το αρχείο `package.json`, αυτό το αρχείο περιλαμβάνει τις εξαρτήσεις που χρησιμοποιούνται στην εφαρμογή, αλλά και άλλες σημαντικές πληροφορίες, όπως το entry point, και εντολές που διευκολύνουν τη λειτουργία της εφαρμογής [34].
Στη συνέχεια εγκαθιστούμε το Express.js:

```
~$ npm install express
```

Έπειτα δημιουργούμε ένα αρχείο για τον server:

Σχήμα 4.4: Server με Express

Στην εφαρμογή αυτή, οι δύο αυτές τεχνολογίες χρησιμοποιούνται για την ανάπτυξη του server, ο οποίος διαχειρίζεται αιτήματα χρηστών μέσω REST API διαδρομών, επιτρέποντας την αποτελεσματική επικοινωνία μεταξύ του διακομιστή και του χρήστη.
Για να εκκινήσουμε τον server χρησιμοποιούμε την εντολή:

```
~$ node server.js
```

4.2.1 Δημιουργία routes και controllers

Οι `routes` καθορίζουν την πορεία των αιτημάτων του χρήστη, ενώ οι `controllers` διαχειρίζονται τη λογική των αιτημάτων αυτών. Κάθε αίτηση του χρήστη στον διακομιστή, γίνεται μέσω των διευθύνσεων URL. Στο Σχέδιο 4.5, έχουμε τις κύριες διαδρομές της εφαρμογής, που αφορούν τους χρήστες, τις δημοσιεύσεις και τα μηνύματα. Και στο Σχέδιο 4.6, έχουμε τις διαδρομές για τις διάφορες ενέργειες του χρήστη (π.χ εγγραφή, σύνδεση, ενημέρωση προφίλ κλπ). Κάθε route αντιστοιχεί σε ένα συγκεκριμένο `function` του controller (π.χ `signupUser`, `loginUser`, `updateUser`).

```
10  app.use("/api/users", userRoutes);
11  app.use("/api/posts", postsRoutes);
12  app.use("/api/messages", messagesRoutes);
```

Σχήμα 4.5: Διαδρομές (routes) στον server

```
userRoutes.js M ×
backend > routes > userRoutes.js > ...
1   import express from 'express';
2   import {
3     followUnFollowUser,
4     getUserProfile,
5     loginUser,
6     logoutUser,
7     signupUser,
8     updateUser,
9     getSuggestedUsers,
10    freezeAccount,
11  } from '../controllers/userController.js';
12
13  const router = express.Router();
14
15  // METHODS
16  router.get('/profile/:query', getUserProfile);
17  router.get('/suggested', getSuggestedUsers);
18
19  router.post('/signup', signupUser);
20  router.post('/login', loginUser);
21  router.post('/logout', logoutUser);
22  router.post('/follow/:id', followUnFollowUser);
23
24  router.put('/update/:id', updateUser);
25  router.put('/freeze', freezeAccount);
26
27  export default router;
```

Σχήμα 4.6: Διαδρομές για τον χρήστη (userRoutes)

4.2.2 Διαδρομές Εφαρμογής

Παρακάτω παρουσιάζονται αναλυτικά όλες οι διαδρομές της εφαρμογής. Κάθε διαδρομή εξυπηρετεί μια συγκεκριμένη λειτουργία της εφαρμογής, όπως η διαχείριση χρηστών, η ανταλλαγή μηνυμάτων, η δημιουργία και διαχείριση αναρτήσεων, καθώς και οι διαδικασίες επιβεβαίωσης και ανάκτησης κωδικού πρόσβασης.

Διαδρομή	**Μέθοδος HTTP**	**Σκοπός**
/api/users/profile/{_id}	GET	Επιστρέφει δεδομένα για το προφίλ ενός χρήστη.
/api/users/suggested	GET	Επιστρέφει προτεινόμενους χρήστες.
/api/users/signup	POST	Δημιουργεί έναν νέο λογαριασμό χρήστη.
/api/users/login	POST	Συνδέει τον χρήστη στο σύστημα.
/api/users/logout	POST	Αποσυνδέει τον χρήστη από το σύστημα.
/api/users/follow/{_id}	POST	Επιτρέπει στον χρήστη να ακολουθήσει ή να σταματήσει να ακολουθεί κάποιον άλλο χρήστη.
/api/users/update/{_id}	PUT	Ενημερώνει τα στοιχεία του προφίλ ενός χρήστη.
/api/users/freeze	PUT	Απενεργοποιεί τον λογαριασμό προσωρινά.

Πίνακας 4.1: Διαδρομές Χρηστών

Διαδρομή	**Μέθοδος HTTP**	**Σκοπός**
/api/messages/conversations	GET	Επιστρέφει τις συνομιλίες του χρήστη
/api/messages/{_id}	GET	Επιστρέφει τα μηνύματα μιας συγκεκριμένης συνομιλίας
/api/messages	POST	Στέλνει ένα νέο μήνυμα σε μια συνομιλία

Πίνακας 4.2: Διαδρομές Μηνυμάτων

Διαδρομή	**Μέθοδος HTTP**	**Σκοπός**
/api/users/forgot-password	POST	Ξεκινά τη διαδικασία επαναφοράς κωδικού, στέλνοντας έναν 6-ψήφιο κωδικό στο email του χρήστη
/api/users/verifyResetCode	POST	Επιβεβαιώνει τον 6- ψηφιο κωδικό επαναφοράς
/api/users/resendResetPasswordCode	POST	Στέλνει νέο 6-ψήφιο κωδικό επαναφοράς
/api/users/resetPassword	POST	Αλλάζει τον κωδικό πρόσβασης του χρήστη

Πίνακας 4.3: Διαδρομές Ανάκτησης Κωδικού Πρόσβασης

Διαδρομή	**Μέθοδος HTTP**	**Σκοπός**
/api/posts/feed	GET	Επιστρέφει τις αναρτήσεις από τους χρήστες που ακολουθεί ο συνδεδεμένος χρήστης.
/api/posts/{_id}	GET	Επιστρέφει τα δεδομένα μιας ανάρτησης.
/api/posts/user/{username}	GET	Επιστρέφει τις αναρτήσεις ενός χρήστη.
/api/posts/replies/{username}	GET	Επιστρέφει τα σχόλια που έχουν κάνει άλλοι χρήστες στον συγκεκριμένο χρήστη.
/api/posts/create	POST	Δημιουργεί μια νέα ανάρτηση.
/api/posts/delete/{_id}	DELETE	Διαγράφει μια ανάρτηση.
/api/posts/{postId}/reply/{replyId}	DELETE	Διαγράφει ένα σχόλιο σε μια ανάρτηση.
/api/posts/like/{_id}	PUT	Κάνει like ή αναιρεί το like σε μια ανάρτηση.
/api/posts/reply/{_id}	PUT	Προσθέτει σχόλιο σε μια ανάρτηση.

Πίνακας 4.4: Διαδρομές Αναρτήσεων

Διαδρομή	**Μέθοδος HTTP**	**Σκοπός**
/api/users/isVerified	GET	Ελέγχει αν ο χρήστης έχει επιβεβαιώσει το email του
/api/users/resendVerificationCode	GET	Στέλνει νέο 6-ψηφιο κωδικό επιβεβαίωσης στο email του χρήστη
/api/users/verify-email	POST	Επιβεβαιώνει το email του χρήστη με βάση εαν ο 6-ψηφιος κωδικός ειναι σωστός

Πίνακας 4.5: Διαδρομές Επιβεβαίωσης Χρήστη

4.2.3 Εγκατάσταση και ρύθμιση των πακέτων του Back-End.

Για να βελτιώσουμε τη λειτουργικότητα και τις δυνατότητες του Back-End, έχουμε εγκαταστήσει και χρησιμοποιήσει διάφορες βιβλιοθήκες και εργαλεία. Ας ξεκινήσουμε να εξετάζουμε αναλυτικά, ένα προς ένα, τον ρόλο και τη χρηση του καθενός.

dotenv

Το `dotenv` μας επιτρέπει να αποθηκεύουμε ευαίσθητες τιμές σε ένα ξεχωριστό αρχείο το οποίο ονομάζεται `.env`. Στην εφαρμογή μας, το αρχείο αυτό περιέχει το URI της βάσης δεδομένων, τη θύρα (`PORT`) του server και του client, στοιχεία για την αποθήκευση αρχείων, καθώς και κλειδιά κρυπτογράφησης. Για να λειτουργήσει η εφαρμογή, όλες αυτές οι απαιτούμενες τιμές πρέπει να έχουν συμπληρωθεί.[35].

mongoose

Το `Mongoose` είναι μία βιβλιοθήκη Object Data Modeling (ODM) που μας επιτρέπει να αλληλεπιδρούμε με τη βάση δεδομένων MongoDB μέσω αντικειμένων, αντί να γράφουμε απευθείας ερωτήματα στη βάση [36].

Για να συνδέσουμε τον server με τη βάση δεδομένων, δημιουργούμε ένα αρχείο το οποίο περιλαμβάνει τον παρακάτω κώδικα:

Σχήμα 4.7: Χρήση του mongoose για συνδεση του server με την βαση δεδομένων

Το `MONGODB_URI` είναι μια ευαίσθητη μεταβλητή που περιέχει το URI σύνδεσης που μας έδωσε το MongoDB Atlas. Για λόγους ασφάλειας, δεν το χρησιμοποιούμε απευθείας στον κώδικα αλλά το ανακτούμε από το αρχείο `.env`.

CORS

Το `CORS` (Cross-Origin Resource Sharing) είναι ένας μηχανισμός ασφάλειας που επιτρέπει ή περιορίζει την πρόσβαση σε πόρους της εφαρμογής από άλλες τοποθεσίες (domains). Στην εφαρμογή μας, χρησιμοποιείται για να διασφαλίσουμε ότι ο server δέχεται αιτήματα μόνο από τις δύο εξουσιοδοτημένες εφαρμογές: τη mobile έκδοση και την ιστοσελίδα.[37].

bcrypt

Η βιβλιοθήκη `bcrypt` χρησιμοποιείται για την ασφαλή κρυπτογράφηση και αποθήκευση κωδικών πρόσβασης. Όταν ένας χρήστης δημιουργεί λογαριασμό ή αλλάζει τον κωδικό του, αντί να αποθηκεύεται απευθείας ο κωδικός, εφαρμόζεται διαδικασία κρυπτογράφησης (hashing) και στη συνέχεια αποθηκεύεται η κρυπτογραφημένη μορφή του, διασφαλίζοντας έτσι την προστασία των δεδομένων. [38].

Το bcrypt προσθέτει έναν τυχαίο χαρακτήρα (salt) στον κωδικό πριν από την κρυπτογράφηση, αυτό διασφαλίζει ότι ακόμη και αν δύο χρήστες έχουν τον ίδιο κωδικό πρόσβασης, το hash που θα παραχθεί θα είναι διαφορετικό. Όταν ο χρήστης προσπαθεί να συνδεθεί, ο κωδικός που εισάγει περνάει ξανά από κρυπτογράφηση και ο παραγόμενος hash συγκρίνεται με αυτόν που είναι αποθηκευμένος στην βάση.

nodemailer

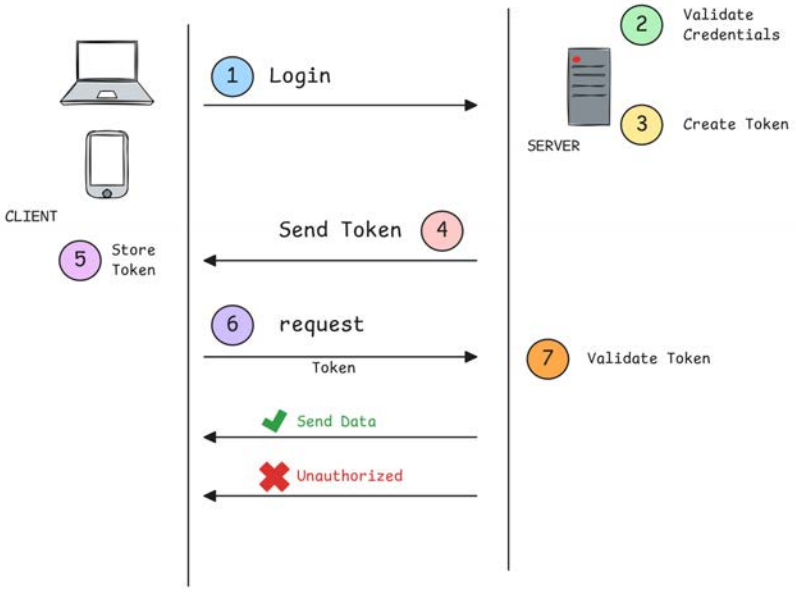
Το `nodemailer` είναι μια βιβλιοθήκη που επιτρέπει την αποστολή email από τον server στους χρήστες μέσω του πρωτοκόλλου SMTP. Στην εφαρμογή μας, χρησιμοποιείται για την αποστολή 6-ψήφιων κωδικών επαλήθευσης. Για την αποστολή των email, έχει διαμορφωθεί ένας λογαριασμός Yahoo, μέσω του οποίου αποστέλλονται οι κωδικοί στους χρήστες [39].

cookie-parser

Το `cookie-parser` είναι μια βιβλιοθήκη που μας επιτρέπει να αναλύουμε και να επεξεργαζόμαστε τα cookies. Τα cookies είναι δεδομένα που αποθηκεύονται στον browser του χρήστη και χρησιμοποιούνται για να θυμούνται διάφορες πληροφορίες. Στην εφαρμογή μας, το cookie-parser χρησιμοποιήθηκε για την αποθήκευση του JSON token ώστε να ελέγχουμε αν ο χρήστης είναι συνδεδεμένος προκειμένου να του επιτρέψουμε ή όχι την πρόσβαση σε προστατευμένες περιοχές της εφαρμογής. [40].

jsonwebtoken

Το `jsonwebtoken` είναι ένα πρωτόκολλο για την αυθεντικοποίηση των χρηστών, που μας επιτρέπει επίσης να προστατεύουμε τα routes. Όταν ένας χρήστης εγγράφεται ή συνδέεται στην εφαρμογή, δημιουργείται ένα token, το οποίο αποθηκεύεται ως cookie στον browser για την εφαρμογή web, και τοπικά για την εφαρμογή mobile. Το token περιλαμβάνει το `userId` του χρήστη και είναι ασφαλισμένο με ένα μυστικό κλειδί `JWT Secret`. Κατά τη διάρκεια των αιτημάτων που αποστέλλει ο client προς τον server, το token συμπεριλαμβάνεται αυτόματα, επιτρέποντας την επαλήθευση της ταυτότητας του χρήστη [41].



Σχήμα 4.8: Διαδικασία Αυθεντικοποίησης

Παρακάτω παρουσιάζεται η συνάρτηση που έχουμε υλοποιήσει για την παραγωγή του token.

```
import jwt from 'jsonwebtoken';

const generateToken = (userId, res, platform) => {
  // generate token
  const token = jwt.sign({ userId }, process.env.JWT_SECRET, {
    expiresIn: '7d',
  });

  // save token as cookies for web
  if (platform === 'web') {
    res.cookie('jwt-token', token, {
      httpOnly: true,
      maxAge: 7 * 24 * 60 * 60 * 1000,
      sameSite: 'strict',
    });
  }

  return token;
};
export default generateToken;
```

Σχήμα 4.9: Δημιουργία JWT token

Για την προστασία συγκεκριμένων διαδρομών, χρησιμοποιούμε ενδιάμεσες συναρτήσεις (`middlewares`) που ελέγχουν την εγκυρότητα του token. Αυτές οι συναρτήσεις επαληθεύουν εάν το token υπάρχει και είναι έγκυρο. Αν το token είναι έγκυρο, επιτρέπεται η πρόσβαση σε συγκεκριμένες λειτουργίες της εφαρμογής. Αν το token λείπει ή είναι μη έγκυρο, η πρόσβαση απορρίπτεται.

Όταν ο client κάνει αίτηση σε "ποστατευμένες" διαδρομές, εκτελείται πρώτα η ενδιάμεση συνάρτηση (middleware).

```
router.get('/profile/:query', getUserProfile);
router.get('/suggested', protectRoute, getSuggestedUsers);

router.post('/signup', signupUser);
router.post('/login', loginUser);
router.post('/logout', logoutUser);
router.post('/follow/:id', protectRoute, followUnFollowUser);

router.put('/update/:id', protectRoute, updateUser);
router.put('/freeze', protectRoute, freezeAccount);
```

Σχήμα 4.10: Προστατευμένες διαδρομές

Παρακάτω παρουσιάζεται η ενδιάμεση συνάρτηση που έχουμε υλοποιήσει:

```
import User from '../models/userModel.js';
import jwt from 'jsonwebtoken';

const protectRoute = async (req, res, next) => {
  try {
    let token;

    // extract token from cookies for web & from header for mobile
    if (req.platform === 'web') {
      token = req.cookies['jwt-token'];
    } else if (req.platform === 'mobile') {
      const authHeader = req.headers.authorization;
      if (authHeader && authHeader.startsWith('Bearer ')) {
        token = authHeader.split(' ')[1];
      }
    }

    // check if token exists
    if (!token) {
      return res
        .status(401)
        .json({ message: 'Unauthorized - No token provided' });
    }

    // verify the token
    const decoded = jwt.verify(token, process.env.JWT_SECRET);

    // retrieve user data (no password)
    const user = await User.findById(decoded.userId).select('-password');

    if (!user) {
      return res.status(401).json({ message: 'Unauthorized - User not found' });
    }

    req.user = user;

    // proceed to the next function
    next();
  } catch (err) {
    res.status(401).json({ message: 'Unauthorized - Invalid token' });
    console.error('Error in protectRoute:', err.message);
  }
};

export default protectRoute;
```

Σχήμα 4.11: Συνάρτηση για προστασία διαδρομών

cloudinary

Το `Cloudinary` είναι μια cloud-based υπηρεσία που χρησιμοποιείται για την αποθήκευση πολυμέσων (π.χ εικόνες, βίντεο). Στην εφαρμογή μας, χρησιμοποιείται για την αποθήκευση των εικόνων που ανεβάζουν οι χρήστες, όπως φωτογραφίες προφίλ και δημοσιεύσεις. Κάθε εικόνα αποκτά ένα μοναδικό `URL`, το οποίο αποθηκεύεται στη βάση δεδομένων MongoDB για εύκολη πρόσβαση και φόρτωση. [42].

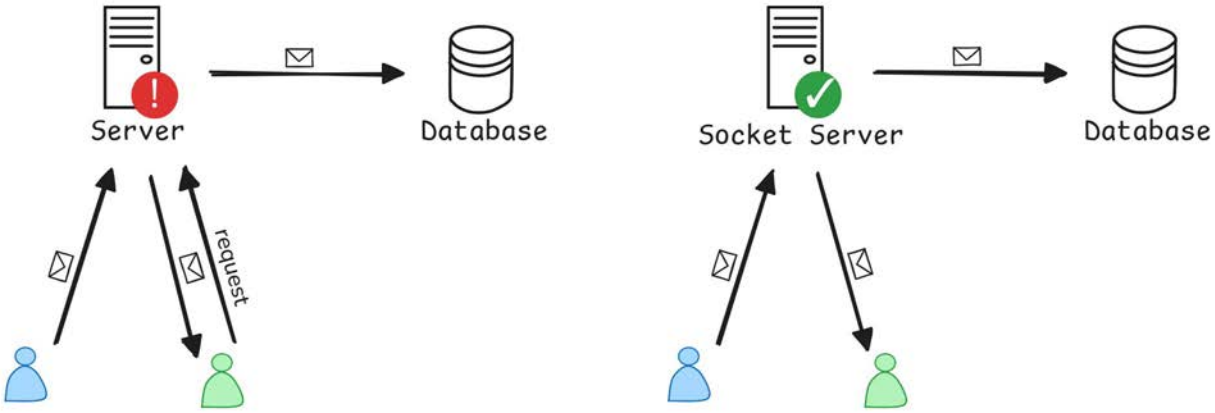
Για να μεταφορτώσουμε αρχεία στο Cloudinary δημιουργούμε λογαριασμό στην υπηρεσία [43], και στην συνέχεια μας παρέχονται τρεις κωδικοί, τους οποίους αποθηκεύουμε στο αρχείο `.env` για λόγους ασφάλειας.

```
cloudinary.config({
  cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
  api_key: process.env.CLOUDINARY_API_KEY,
  api_secret: process.env.CLOUDINARY_API_SECRET,
});
```

Σχήμα 4.12: Ρύθμιση του Cloudinary

socket.io

Το `socket.io` είναι μια βιβλιοθήκη που επιτρέπει την επικοινωνία σε πραγματικό χρόνο μεταξύ του server και του client. Σε αντίθεση με τα HTTP requests, όπου ο client πρέπει να αποστέλλει συνεχώς αιτήματα για να λαμβάνει νέες πληροφορίες, το Socket.io διατηρεί μια συνεχή σύνδεση. Αυτό εξαλείφει την ανάγκη για επαναλειλημμένα requests, μειώνοντας το φορτίο στον server και προστατεύοντας τον από πιθανή υπερφόρτωση ή κατάρρευση (server downtime) [44].



Σχήμα 4.13: Διαδικάσια αποστολής μηνυμάτων πριν και μετά την χρήση socket.io

Στην εφαρμογή, οταν ένας χρήστης στέλνει ένα μήνυμα σε έναν άλλο χρήστη, αυτό αποστέλλεται στον server μέσω του Socket.io. Ο server αποθηκεύει το μήνυμα στη βάση δεδομένων και το προωθεί άμεσα στον παραλήπτη, χωρίς αυτός να χρειαστεί να στείλει κάποιο αίτημα στον server.

Όταν ένας χρήστης συνδέεται στην εφαρμογή, ο server δημιουργεί μια νέα σύνδεση και ένα μοναδικό `socketId`, το οποίο αποθηκεύεται. Οι υπόλοιποι συνδεδεμένοι χρήστες ενημερώνονται για τη νέα σύνδεση. Όταν ο χρήστης αποσυνδεθεί, το `socketId` διαγράφεται και οι υπόλοιποι χρήστες ενημερώνονται ότι ο συγκεκριμένος χρήστης αποσυνδέθηκε.

Βασικές εντολές:

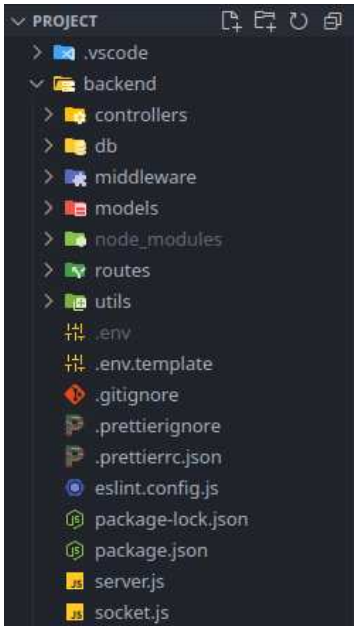
- `socket.on( )`: Χρησιμοποιείται από τον server και τον client για την ακρόαση συγκεκριμένων events.
- `io.emit( )`: Ο server στέλνει ένα event σε όλους τους συνδεδεμένους χρήστες.
- `io.to( socketId ).emit( )`: Ο server στέλνει ένα event μόνο σε έναν συγκεκριμένο χρήστη.

Παρακάτω παρουσιάζεται η υλοποίηση της Real-Time επικοινωνίας στον server:

```
// listens for incoming connections
io.on('connection', (socket) => {
  console.log('user connected', socket.id);
  const userId = socket.handshake.query.userId;

  // emits the connected users to all online clients
  if (userId != 'undefined') userSocketMap[userId] = socket.id;
  io.emit('getOnlineUsers', Object.keys(userSocketMap));

  // mark messages status as seen
  socket.on('markMessagesAsSeen', async ({ conversationId, userId }) => {
    if (userId === socket.handshake.query.userId) {
      try {
        await Message.updateMany(
          { conversationId: conversationId, seen: false },
          { $set: { seen: true } },
        );
        await Conversation.updateOne(
          { _id: conversationId },
          { $set: { 'lastMessage.seen': true } },
        );
        // notifies the user that the message have marked as seen
        io.to(userSocketMap[userId]).emit('messagesSeen', { conversationId });
      } catch (error) {
        console.log(error);
      }
    }
  });

  // when user disconnected emit all online clients
  socket.on('disconnect', () => {
    console.log('user disconnected');
    delete userSocketMap[userId];
    io.emit('getOnlineUser', Object.keys(userSocketMap));
  });
});

export { io, server, app };
```

Σχήμα 4.14: Υλοποίηση Real-Time επικοινωνίας με socket.io στον server

4.2.4 Δομή των αρχείων του Back-End



Για λόγους καλύτερης οργάνωσης του Back-End στην εφαρμογή μας ακολουθήσαμε την εξής δομή αρχείων:

- /controllers → Περιέχει τις μεθόδους που διαχειρίζονται τη λογική κάθε route
- /db → Περιέχει αρχεία για τη σύνδεση με τη βάση δεδομένων
- /middleware → Περιέχει τις ενδιάμεσες συναρτήσεις που εκτελούνται πριν την επεξεργασία μιας αίτησης
- /models → Περιέχει τα μοντέλα της MongoDB, τα οποία χρησιμοποιούνται για να ορίσουμε τύπους δεδομένων για κάθε collection
- /routes → Περιέχει τις διαδρομές της εφαρμογής
- /utils → Περιέχει βοηθητικές συναρτήσεις

Σχήμα 4.15: Οργάνωση αρχείων Back-End

4.3 React

Για την ανάπτυξη της εφαρμογής Web, χρησιμοποιήσαμε το Vite, ένα εργαλείο που επιταχύνει τη διαδικασία του build. Το Vite προσφέρει ταχύτερη ανανέωση κώδικα μέσω της λειτουργία Hot Module Replacement (HMR) [45]. Η εγκατάσταση γίνεται με την εντολή:

```
~$ npm create vite@latest
```

Στη συνέχεια ακολουθούμε τις οδηγίες, επιλέγοντας το React ως framework και Javascript. Το Vite αποτελεί μια σύγχρονη εναλλακτική λύση του:

```
~$ npx create-react-app
```

Για να εκκινήσουμε την εφαρμογή εκτελούμε την παρακάτω εντολή:

```
~$ npm run dev
```

4.3.1 Σύνδεση του Web (Ιστού) με το Back-End

Η επικοινωνία μεταξύ του Front-End και του Back-End πραγματοποιείται μέσω της μεθόδου `fetch`, η οποία επιτρέπει στον client να στέλνει HTTP αιτήματα στον server [46]. Τα αιτήματα αυτά κατευθύνονται στα routes που έχουμε υλοποιήσει στο Back-End. Στο παρακάτω παράδειγμα, εκτελείται από τον client ένα `GET` αίτημα προς το route `/api/posts/${pid}` (όπου το `pid` είναι το `post id`). Ο server επεξεργάζεται το αίτημα και επιστρέφει τα δεδομένα σε μορφή JSON.

Στη συνέχεια, αυτά τα δεδομένα χρησιμοποιούνται για την ενημέρωση της Διεπαφής Χρήστη (User Interface). Εάν το `pid` αλλάξει, η συνάρτηση εκτελείται ξανά, αποστέλλοντας νέο αίτημα στον server.

```
useEffect(() => {
  // fetch post
  const getPost = async () => {
    setPosts([]);
    try {
      // send GET request to the server to fetch post by id
      const res = await fetch(`/api/posts/${pid}`);
      const data = await res.json();
      // error
      if (data.error) {
        showToast('Error', data.error, 'error');
        return;
      }
      setPosts([data]); // update UI data
    } catch (error) {
      showToast('Error', error.message, 'error');
    }
  };
  getPost();
}, [showToast, pid, setPosts]); // re-runs when post id change
```

Σχήμα 4.16: Παράδειγμα αποστολής αιτήματος GET στον server

4.3.2 Εγκατάσταση και ρυθμιση των πακέτων Frontent

ChakraUI

Το `Chakra UI` είναι μια βιβλιοθήκη που παρέχει έτοιμα components για εφαρμογές React, διευκολύνοντας τη δημιουργία μοντέρνων και ευέλικτων διεπαφών χρήστη. Στην εφαρμογή μας, χρησιμοποιήθηκε για τη διαμόρφωση του User Interface, προσφέροντας προσαρμόσιμα στοιχεία, όπως κουμπιά, φόρμες και ειδοποιήσεις. [47].

react-router-dom

Η βιβλιοθήκη react-router-dom επιτρέπει την πλοήγηση σε διαφορετικές σελίδες της εφαρμογής, χωρίς την ανάγκη ανανέωσης ολόκληρης της σελίδας. Αυτό επιτρέπει τη δημιουργία δυναμικών και διαδραστικών Single-Page Applications (SPA), όπου το περιέχομενο αλλάζει ανάλογα με το URL [48].

emoji-mart/react

Η βιβλιοθήκη emoji-mart χρησιμοποιείται στην εφαρμογή για να δώσει την δυνατοτητα στους χρήστες να επιλέγουν emoji για χρήση στις δημοσιεύσεις , τα σχόλια και τα μηνύματα τους [49].

react-icons

Η βιβλιοθήκη react-icons παρέχει έτοιμα εικονίδια που μπορούν να χρησιμοποιηθούν σε React εφαρμογές. Στην εφαρμογή μας, έχουν χρησιμοποιηθεί εικονίδια σε κουμπιά και σε μενού επιλογών [49].

4.3.3 Components

Η ανάπτυξη React εφαρμογών βασίζεται στη δημιουργία επαναχρησιμοποήσιμων components, τα οποία ειναι τα δομικά στοιχεία κάθε εφαρμογής. Μέσω αυτών, η React δίνει την δυνατότητα διαίρεσης της εφαρμογής σε περισσότερα μέρη [50]. Τα components ειναι JavaScript συναρτήσεις που δέχονται ως είσοδο δεδομένα μέσω των props και επιστέφουν JSX πίσω.

Τα props ειναι δεδομένα που διαβιβάζονται από εναν γονέα σε ένα παιδί.

```
function Parent() {
    const name = "tdimitr"
    return <Child name={name} />;
}

function Child({name}) {
    return <h1>Hello {name}</h1>;
}
```

Σχήμα 4.17: Παράδειγμα επικοινωνίας Γονέα-Παιδιού

4.3.4 useState , useEffect και Custom Hooks

useState

Το `useState` είναι ένα hook που επιτρέπει τη διαχείριση κατάστασης (*state*) ενός component. Παίρνει μια αρχική τιμή και επιστρέφει έναν πίνακα με την τρέχουσα κατάσταση και τη συνάρτηση που χρησιμοποιείται για την ενημέρωσή της [51].

```
const [state, setState] = useState(initialValue);
```

- `state`: Η τρέχουσα τιμή της κατάστασης
- `setState`: Η συνάρτηση που χρησιμοποιείται για την ενημέρωση της κατάστασης
- `initialValue`: Η αρχική τιμή της κατάστασης

Στην εφαρμογή μας, το useState χρησιμοποιείται για τη διαχείριση της κατάστασης σε διάφορα component, όπως η διαχείριση των δυναμικών δεδομένων στις σελίδες του χρήστη.

useEffect

Το `useEffect` στο React χρησιμοποιείται για την εκτέλεση κώδικα όταν αλλάζουν συγκεκριμένα δεδομένα (dependencies). Ο κώδικας συχνά περιλαμβάνει ενέργειες όπως η φόρτωση δεδομένων από έναν διακομιστή [52].

```
useEffect(() => {
  // Κώδικας που εκτελείται
}, [dependencies])
```

- `dependencies`: Αν δεν περιέχει εξαρτήσεις (δηλαδή `[]`), ο κώδικας εκτελείται μόνο μία φορά κατά την έναρξη της σελίδας. Αν περιέχει εξαρτήσεις, τότε ο κώδικας εκτελείται – εκτός από την έναρξη της σελίδας – και κάθε φορά που οι εξαρτήσεις αλλάζουν.

Στην εφαρμογή μας, το useEffect χρησιμοποιείται για την ανάκτηση δεδομένων από τον server καθώς και για την αυτόματη ανανέωση δεδομένων σε πραγματικό χρόνο όπου απαιτείται.

Custom Hooks

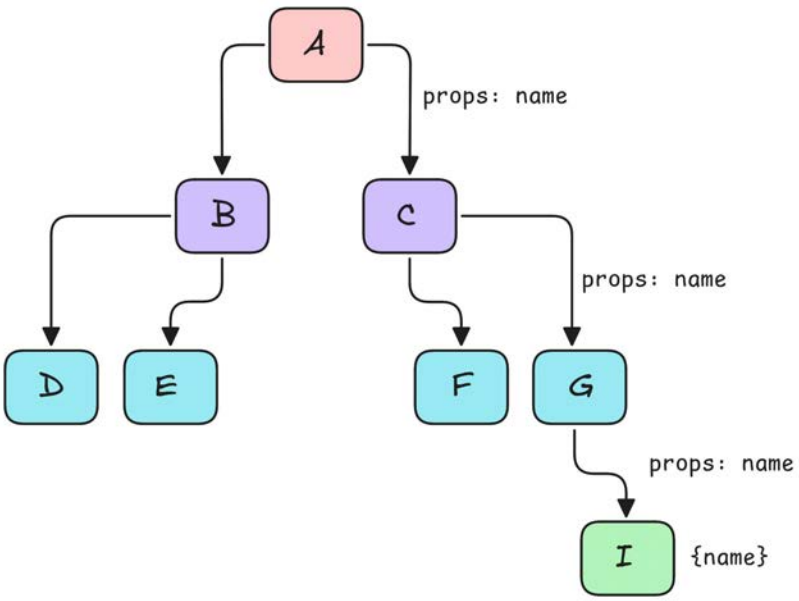
Τα `custom hooks` είναι συναρτήσεις που δημιουργούμε για να αποφύγουμε την επαναλαμβανόμενη χρήση του ίδιου κώδικα σε διαφορετικά components. Αντί να γράφουμε ξανά και ξανά την ίδια λογική, δημιουργούμε ένα custom hook που περιλαμβάνει αυτή τη λογική και το χρησιμοποιούμε σε πολλά μέρη της εφαρμογής μας [53].

4.3.5 recoil

Το `Recoil` είναι μια βιβλιοθήκη διαχείρισης κατάστασης για εφαρμογές React. Σχεδιάστηκε για να αντιμετωπίσει τα προβλήματα που προκύπτουν από τη διαχείριση κοινόχρηστης κατάστασης (*shared state*) [54].

Ένα από τα βασικά προβλήματα που λύνει το Recoil είναι το *Prop Drilling*. Το *prop drilling* συμβαίνει όταν μία κατάσταση (*state*) πρέπει να περάσει από έναν γονέα (*parent*) σε ένα απομακρυσμένο παιδί (*child*) μέσω `props`. Αυτό που συμβαίνει είναι:

- Τα ενδιάμεσα components αναγκάζονται να μεταφέρουν τα δεδομένα, ακόμα και αν δεν τα χρειάζονται, μόνο και μόνο επειδή κάποιο από τα παιδιά τους τα χρειάζεται.
- Όταν η κατάσταση αλλάξει, όλα τα ενδιάμεσα components ανανεώνονται (*re-render*), κάτι που σημαίνει ότι μειώνεται η απόδοση της εφαρμογής.



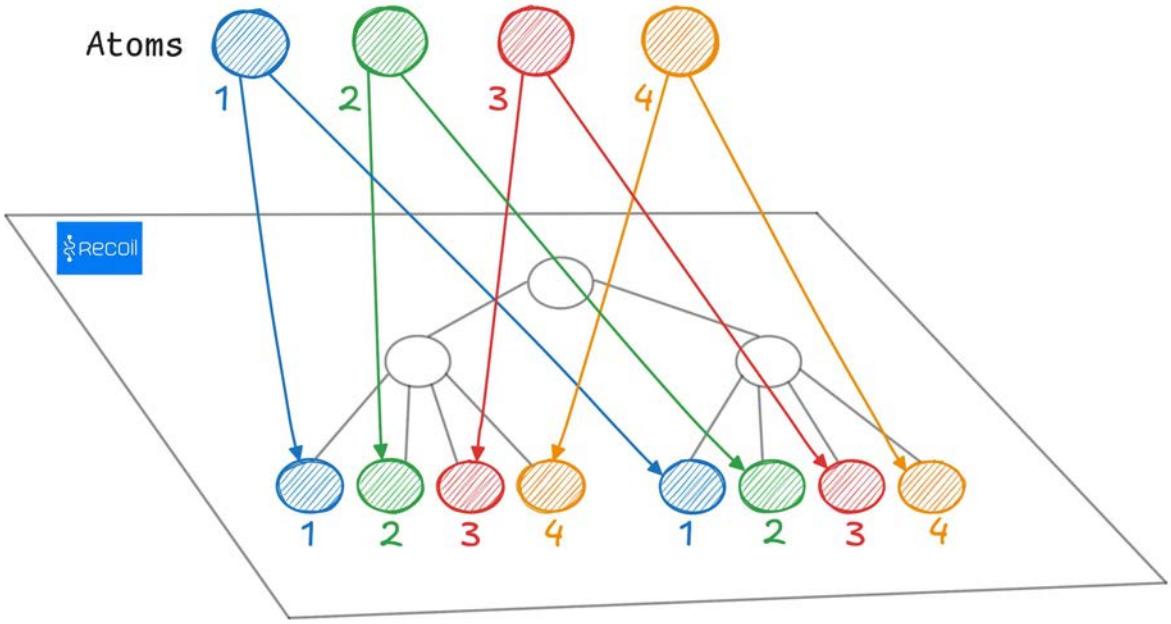
Σχήμα 4.18: Prop Drilling[55]

Η λύση του Recoil

Το Recoil εισάγει τα `atoms`, τα οποία λειτουργούν ως κεντρικές μονάδες αποθήκευσης (*global state*). Τα atoms είναι ανεξάρτητες μονάδες δεδομένων που μπορούν να διαβαστούν και να τροποποιηθούν από οποιοδήποτε component της εφαρμογής. Κάθε component που χρησιμοποιεί ένα atom εγγράφεται σε αυτό. Όταν η τιμή του atom αλλάξει, μόνο τα component που είναι συνδεδεμένα με αυτό το atom ενημερώνονται.

Αυτό σημαίνει ότι:

- Δεν απαιτείται η μεταφορά δεδομένων μέσω props
- Η εφαρμογή είναι πιο αποδοτική, καθώς μόνο τα απαραίτητα components ανανεώνονται.

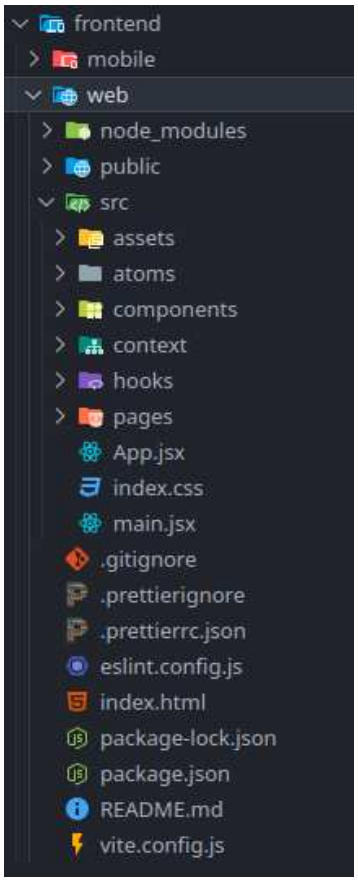


Σχήμα 4.19: Οπτική αναπαράσταση του Recoil [56]

```
postsAtom.js
frontend > web > src > atoms > postsAtom.js > ...
1  import { atom } from 'recoil';
2
3  const postsAtom = atom({
4    key: 'postAtom',
5    default: [],
6  });
7
8  export default postsAtom;
```

Σχήμα 4.20: Παράδειγμα δημιουργίας atom για δημοσιεύσεις

4.3.6 Δομή των αρχείων Web (Ιστού)



Σχήμα 4.21: Οργάνωση αρχείων web

Για λόγους καλύτερης οργάνωσης του Front-End στην εφαρμογή μας ακολουθήσαμε την εξής δομή αρχείων:

- /atoms → Χρησιμοποιείται για την αποθήκευση και διαχείριση της κατάστασης μέσω της βιβλιοθήκης recoil.
- /components → Περιέχει τα επαναχρησιμοποιήσιμα τμήματα της διεπαφής χρήστη (UI) όπως επικεφαλίδα, σχόλια, παράθυρο συνομιλίας, προτεινόμενοι χρήστες κλπ.
- /context → Χρησιμοποιείται για την επικοινωνία σε πραγματικό χρόνο με τον server. (socket.io)
- /hooks → Περιέχει custom hooks, δηλαδή συναρτήσεις που χρησιμοποιούνται σε διάφορα components.
- /pages → Περιέχει τις βασικές σελίδες της εφαρμογής όπως αρχική σελίδα, σελίδα προφίλ, σελίδα μηνυμάτων κλπ.

4.4 React Native

Για την ανάπτυξη της Mobile εφαρμογής, χρησιμοποιήσαμε το Expo Go, ένα Framework το οποίο προσφέρει μια εύκολη διαδικασία για τη δημιουργία και την εκτέλεση εφαρμογών React Native [57]. Η εγκατάσταση γίνεται με την εντολή:

```
~$ npx create-expo-app@latest <name> -template
```

Στη συνέχεια ακολουθούμε τις οδηγίες, επιλέγοντας Blank και JavaScript.
Για να εκκινήσουμε την εφαρμογή, χρησιμοποιούμε την εντολή:

```
~$ npm start
```

Στο Τερματικό (Terminal) θα εμφανιστεί ένας κωδικός QR. Σαρώνοντας τον με την εφαρμογή Expo Go του κινητού μας [58] [59], μπορούμε να εκτελέσουμε την εφαρμογή απευθείας στη συσκευή. Εναλλακτικά, για την εκτέλεση μπορούμε να χρησιμοποιήσουμε το Android Studio Emulator [60] για Android ή το Xcode [61] για iOS.

4.4.1 Expo Router

Το Expo Router είναι ένα εργαλείο για τη διαχείριση της πλοήγησης σε εφαρμογές React Native. Χρησιμοποιεί τη δομή των αρχείων και φακέλων για να ορίσει τις οθόνες της εφαρμογής, κάνοντάς την πλοήγηση και τη δημιουργία διαδρομών πιο απλή και οργανωμένη. [62]

- _layout.jsx : Κάθε φάκελος μπορεί να περιέχει ένα αρχείο _layout.jsx, το οποίο καθορίζει τη διάταξη για όλες τις οθόνες μέσα στον συγκεκριμένο φάκελο. Οι οθόνες του φακέλου κληρονομούν αυτόματα τη διάταξη που καθορίζεται από το _layout.jsx.

- index.jsx : Το αρχείο αυτό, αντιστοιχεί στην αρχική οθόνη κάθε φακέλου. π.χ., αν ο φάκελος `tabs` έχει `index.jsx`, τότε η διαδρομή `/tabs` θα φορτώσει αυτή την οθόνη.

- Δυναμικά αρχεία : Με τη χρήση αγκυλών, όπως `[username].jsx`, δημιουργούμε δυναμικές διαδρομές. π.χ., το αρχείο `/profile/[username].jsx` θα αντιστοιχεί σε μια διαδρομή όπως `/profile/john` και μπορούμε να προσπελάσουμε την παράμετρο `username` μέσω των `useGlobalSearchParams` ή `useLocalSearchParams` [63].

4.4.2 SecureStore

Στην εφαρμογή χρησιμοποιήσαμε το `SecureStore` για την αποθήκευση ευαίσθητων δεδομένων, όπως το token του χρήστη και άλλες πληροφορίες που πρέπει να παραμείνουν ασφαλείς. Το SecureStore είναι μια βιβλιοθήκη που παρέχεται από το Expo, η οποία αξιοποιεί τις δυνατότητες κρυπτογράφησης των λειτουργικών συστημάτων Android και iOS [64].

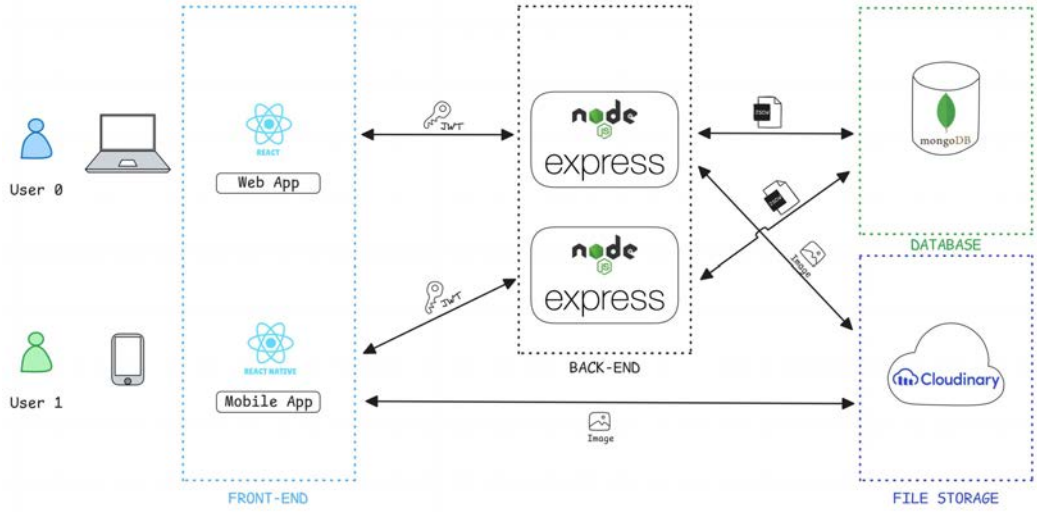
4.4.3 Expo Icons

Στην εφαρμογή mobile χρησιμοποιούμε τα εικονίδια που παρέχονται από το Expo μέσω της βιβλιοθήκης `@expo/vector-icons`. Αυτή η βιβλιοθήκη περιλαμβάνει μια μεγάλη ποικιλία από εικονίδια που έχουμε χρησιμοποιήσει σε διάφορα στοιχεία, όπως κουμπιά και μενού επιλογών. [65].

4.5 Σχεδιασμός της Εφαρμογής

Η εφαρμογή είναι πλήρως λειτουργική τόσο σε Web όσο και σε Mobile περιβάλλοντα, παρέχοντας μια ομοιόμορφη και ομαλή εμπειρία χρήσης ανεξάρτητα από τη συσκευή. Είναι βασισμένη στη MERN στοίβα και διαθέτει ασφαλείς μηχανισμούς επικοινωνίας μεταξύ πελάτη-διακομιστή. Η διαχείριση αρχείων πραγματοποιείται μέσω της Cloud υπηρεσίας Cloudinary, ενώ η αυθεντικοποίηση των χρηστών γίνεται με τη χρήση JSON Web Tokens (JWT).

Παρακάτω παρουσιάζεται η αρχιτεκτονική του συστήματος και η ροή των δεδομένων:



Σχήμα 4.22: Διάγραμμα Αρχιτεκτονικής Συστήματος

Πιο συγκεκριμένα, οι χρήστες είτε μέσω rowser είτε μέσω mobile εφαρμογής, αλληλεπιδρούν με το σύστημα υποβάλλοντας αιτήματα προς τον server, ο οποίος τα επεξεργάζεται και εκτελεί τις απαραίτητες λειτουργίες στη βάση δεδομένων.

Στον web client, η αυθεντικοποίηση βασίζεται σε cookies. Ο server αποθηκεύει το authentication token (JWT) ως cookie και το ελέγχει σε κάθε αίτημα. Η επεξεργασία του αιτήματος συνεχίζεται μόνο εφόσον το token είναι έγκυρο. Στον mobile client, το token αποθηκεύεται τοπικά στη συσκευή και αποστέλλεται μέσω των request headers. Ο server προσδιορίζει αν το αίτημα προέρχεται από τη mobile εφαρμογή ελέγχοντας μια ειδική τιμή στο request. Αν δεν υπάρχει, το αντιμετωπίζει ως αίτημα από web client.

Η αποθήκευση και διαχείριση εικόνων διαφοροποιείται μεταξύ των δύο τύπων clients. Στην περίπτωση του web client, η εικόνα αποστέλλεται πρώτα στον server, ο οποίος στη συνέχεια την ανεβάζει στο Cloudinary και αποθηκεύει το επιστρεφόμενο URL στη βάση δεδομένων. Αντίθετα, στον mobile client η εικόνα ανεβαίνει απευθείας στο Cloudinary από

τη συσκευή του χρήστη, λαμβάνεται το URL και αποστέλλεται στον server για αποθήκευση στη βάση δεδομένων.

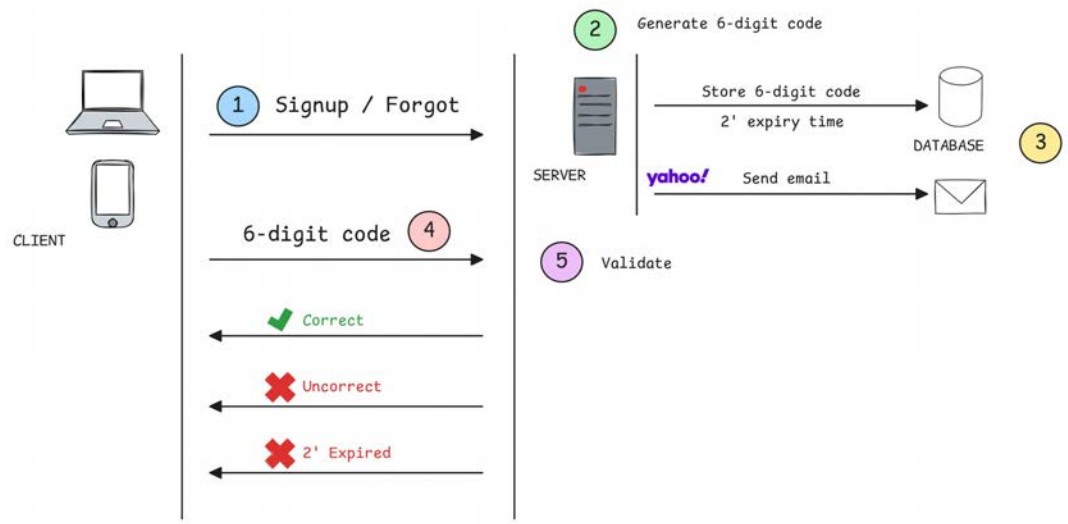
Μηχανισμός Επαλήθευσης

Για την επαλήθευση του λογαριασμού και την ανάκτηση πρόσβασης, η εφαρμογή χρησιμοποιεί έναν μηχανισμό 6-ψήφιων κωδικών ασφαλείας. Αυτοί οι κωδικοί δημιουργούνται και αποστέλλονται στον χρήστη μέσω email.

Όταν ένας χρήστης δημιουργεί λογαριασμό στην εφαρμογή, ξεκινά η διαδικασία επαλήθευσης του λογαριασμού του. Ενας 6-ψήφιος κωδικός δημιουργείται, ο οποίος αποθηκεύεται κρυπτογραφημένος στη βάση δεδομένων, μαζί με την ώρα δημιουργίας του. Ο χρήστης πρέπει να εισαγάγει τον 6-ψήφιο κωδικό μέσα σε 2 λεπτά. Αν παρέλθει αυτό το χρονικό όριο, το αίτημα επαλήθευσης απορρίπτεται και απαιτείται η δημιουργία νέου 6-ψήδιου κωδικού.

Μόλις ο χρήστης εισαγάγει σωστά τον 6-ψήφιο κωδικό εντός του επιτρεπόμενου χρόνου, το σύστημα ενημερώνει τη βάση δεδομένων ώστε ο λογαριασμός να σημειωθεί ως επαληθευμένος. Κατά τη διαδικασία σύνδεσης, το σύστημα ελέγχει εάν ο λογαριασμός έχει επαληθευτεί για να επιτρέψει την πρόσβαση στην αρχική σελίδα.

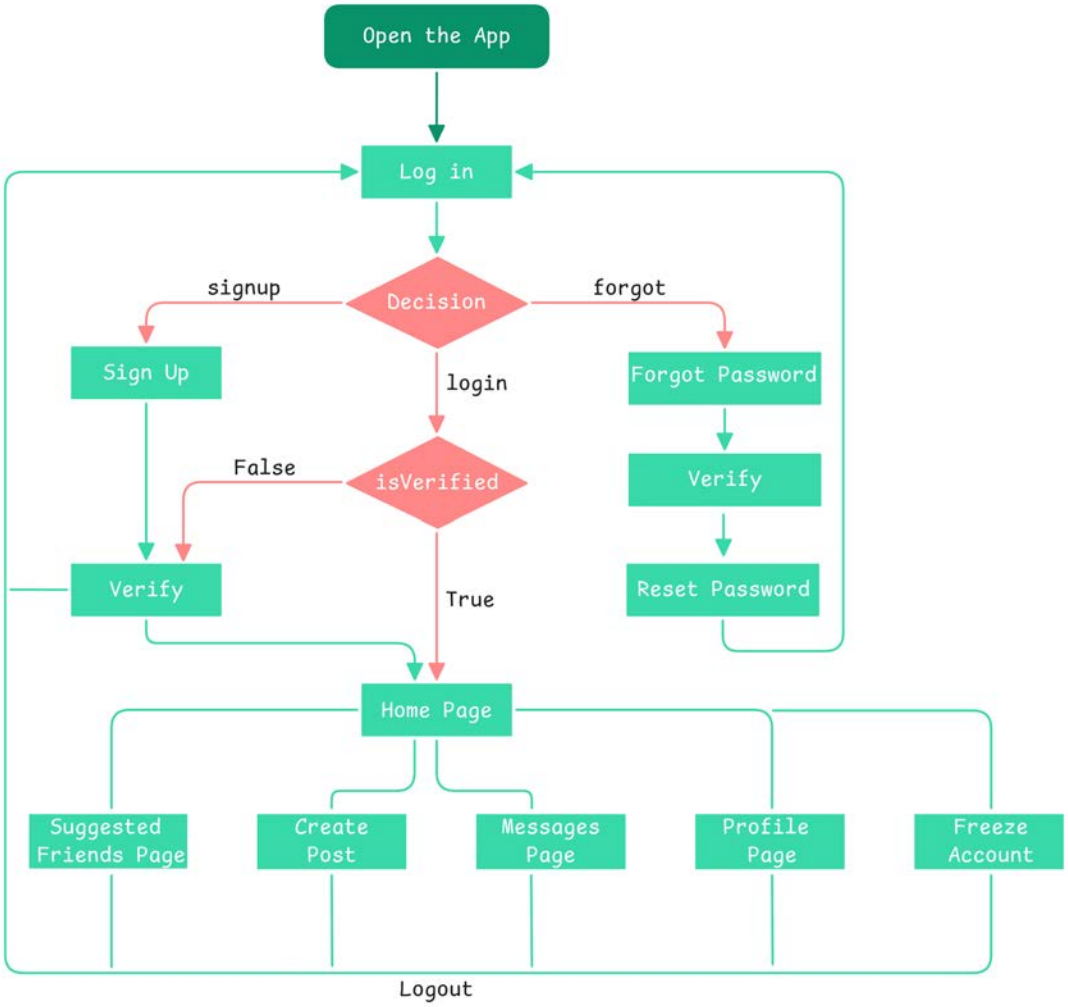
Ο ίδιος μηχανισμός εφαρμόζεται και στη διαδικασία ανάκτησης λογαριασμού. Ο χρήστης λαμβάνει έναν 6-ψήφιο κωδικό, ο οποίος έχει ισχύ έως 2 λεπτά. Μετά την πάροδο αυτού του διαστήματος, ο κωδικός καθίσταται άκυρος και ο χρήστης πρέπει να ζητήσει νέο.



Σχήμα 4.23: Μηχανισμός Επαλήθευσης και Μηχανισμός Ανάκτησης Λογαριασμου

4.6 Πλοήγηση στην Εφαρμογή

Η πλοήγηση στην εφαρμογή μας είναι σχεδιασμένη έτσι ώστε να παρέχει μια ομαλή εμπειρία στους χρήστες. Ξεκινώντας από τη διαδικασία σύνδεσης και εγγραφής μέχρι την πλήρη αλληλεπίδραση με άλλους χρήστες, ακολουθεί μια σαφώς καθορισμένη ροή.



Σχήμα 4.24: Διάγραμμα Πλοήγησης

Πρώτη Σελίδα

Όταν ο χρήστης ανοίγει την εφαρμογή, μεταφέρεται αρχικά στη σελίδα σύνδεσης. Από εκεί έχει τις εξής επιλογές:

- Σύνδεση (Log in): Εάν διαθέτει ήδη λογαριασμό, μπορεί να εισαγάγει τα διαπιστευτήριά του για να προχωρήσει.
- Δημιουργία Λογαριασμού (Sign Up): Αν δεν έχει λογαριασμό, μπορεί να εγγραφεί

εισάγοντας τα απαραίτητα στοιχεία.

- Ανάκτηση Κωδικού Πρόσβασης (Forgot Password): Αν έχει ξεχάσει τον κωδικό του, μπορεί να εισαγάγει το email του. Ένας 6-ψήφιος κωδικός θα σταλεί στο email του, και αν τον εισαγάγει σωστά, θα μεταφερθεί σε μια σελίδα αλλαγής κωδικού. Στην συνέχεια θα επιστρέψει πίσω στην σελίδα σύνδεσης.

Επιβεβαίωση Email & Ενεργοποίηση Λογαριασμού

Εάν ο χρήστης δημιουργήσει έναν νέο λογαριασμό, θα πρέπει να επιβεβαιώσει το email του (Email Verification).

- Αν προσπαθήσει να συνδεθεί χωρίς να έχει επιβεβαιώσει το email, θα μεταφερθεί αυτόματα στη σελίδα επιβεβαίωσης λογαριασμού.
- Μόνο αφού ολοκληρωθεί η επιβεβαίωση, θα μπορεί να αποκτήσει πλήρη πρόσβαση στην εφαρμογή.

Αρχική Σελίδα & Διαχείριση Περιεχομένου

Μόλις ο χρήστης συνδεθεί και ο λογαριασμός του είναι ενεργοποιημένος, θα μεταφερθεί στην Αρχική Σελίδα. Εκεί μπορεί να:

- Δει δημοσιεύσεις από άτομα που ακολουθεί.
- Αλληλεπιδράσει με δημοσιεύσεις μέσω likes και σχολίων.
- Ακολουθήσει νέους χρήστες μέσω της σελίδας ”Προτεινόμενοι Λογαριασμοί”.
- Δημιουργήσει μια νέα δημοσίευση και να μοιραστεί περιεχόμενο.
- Στείλει μηνύματα σε άλλους χρήστες μέσω του συστήματος συνομιλιών.

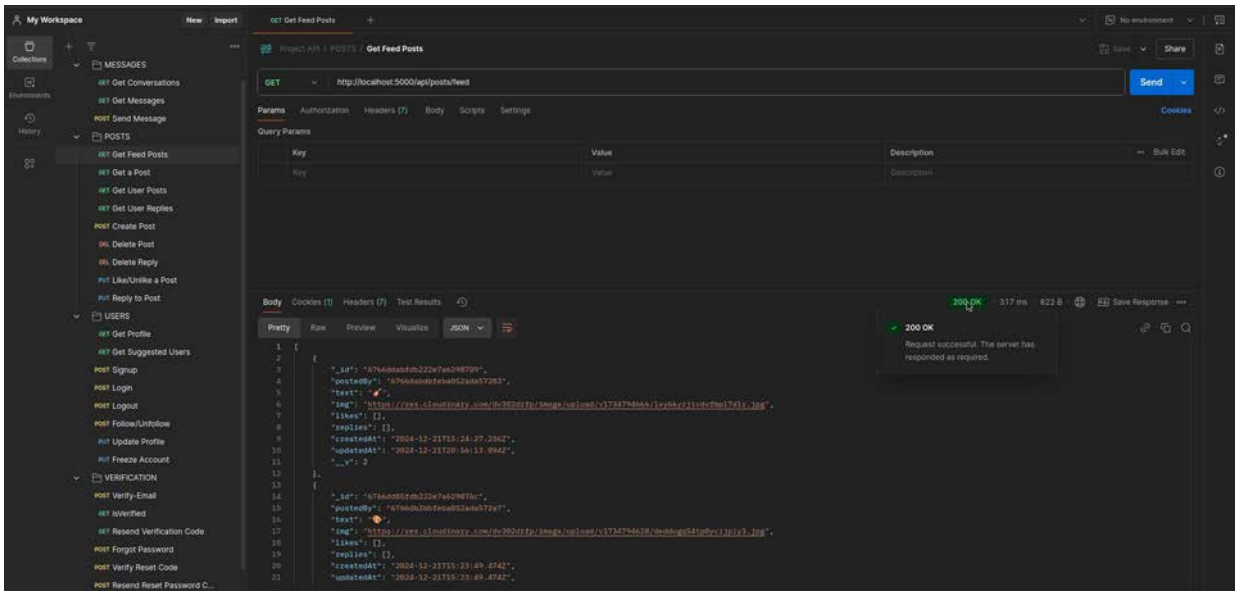
Σελίδα Προφίλ & Ρυθμίσεις

Ο χρήστης μπορεί να μεταβεί στη σελίδα προφίλ όπου θα βρει:

- Τις δικές του δημοσιεύσεις.
- Την επιλογή να επεξεργαστεί το προφίλ του (όνομα, εικόνα προφίλ κ.λπ.).

4.7 Postman

Το Postman είναι ένα εργαλείο που χρησιμοποιείται για την αποστολή HTTP αιτημάτων (requests). Επιτρέπει στους προγραμματιστές να στέλνουν αιτήματα HTTP στον server και να βλέπουν την απάντηση (responses). Αυτο μας επιτρέπει να ελέγξουμε την ορθη λειτουργία των API routes στο Back-End χωρίς να απαιτείται να κάνουμε εκτελέσεις μέσω της διεπαφής χρηστη στο Front-End [66].



Σχήμα 4.25: Παράδειγμα αίτηματος GET στον Server μέσω του Postman

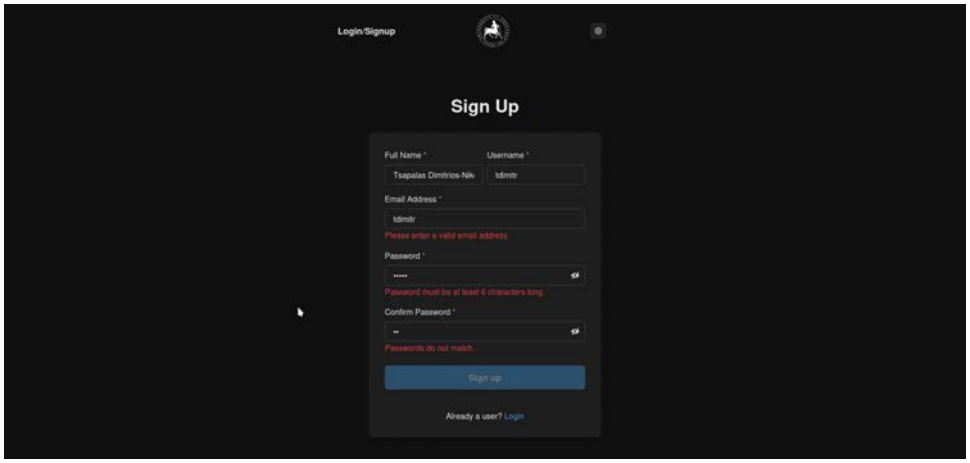
Κεφάλαιο 5

Σελίδες και Χαρακτηριστικά της Εφαρμογής

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε τις κύριες σελίδες της εφαρμογης και τις βασικές λειτουργίες που υποστηρίζει. Η εφαρμογή αποτελείται απο διάφορες σελίδες, όπως η σελίδα συνδεσης και εγγραφης (Login / Signup), η αρχική σελίδα (Home), η σελίδα προφιλ (Profile) και άλλες.

5.1 Κύριες Σελίδες Εφαρμογής Web (Ιστού)

5.1.1 Σελίδα Εγγραφης



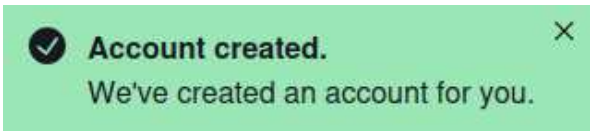
Σχήμα 5.1: Σελίδα Εγγραφης

Η σελίδα Εγγραφής παρέχει μια σειρά από ελέγχους, που περιλαμβάνουν:

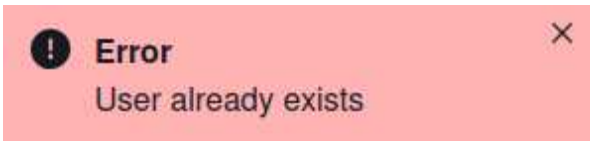
- **Εγκυρότητα Email:** Το email που καταχωρείται πρέπει να είναι σε έγκυρη μορφή (δηλαδή πρέπει να έχει κατάληξη `@uth.gr`), εξασφαλίζοντας ότι οι χρήστες παρέχουν μια σωστή διεύθυνση email.

- **Κωδικός πρόσβασης:** Ο κωδικός πρόσβασης πρέπει να έχει τουλάχιστον 6 χαρακτήρες και να περιέχει τουλάχιστον ένα γράμμα.

Αφού συμπληρώσουμε τα στοιχεία μας και πατήσουμε το κουμπί *"Sign up"*, θα εμφανιστεί στην οθόνη μας, μία από τις παρακάτω ειδοποιήσεις:



(α′)

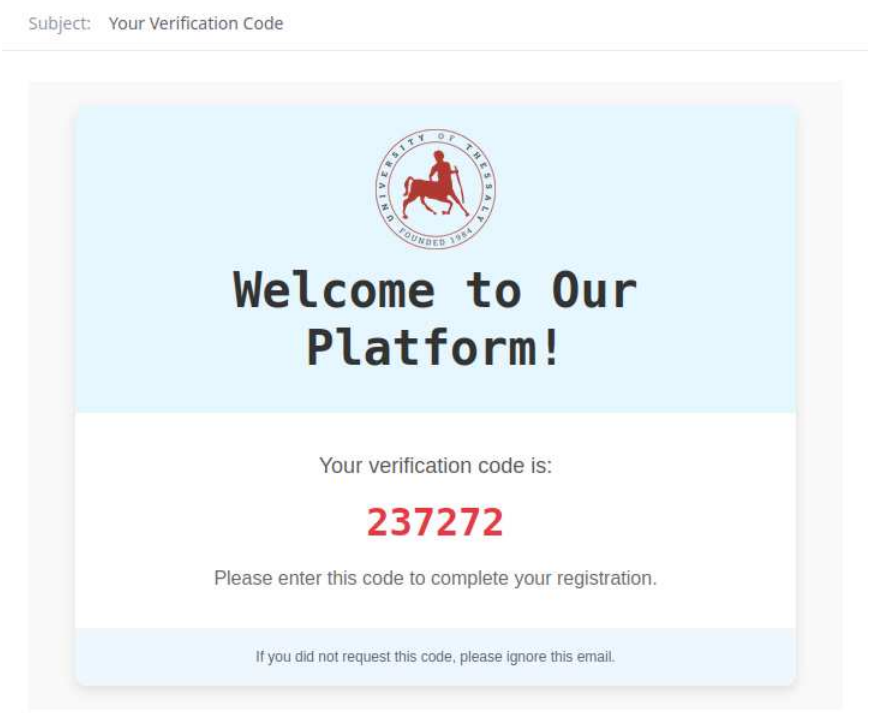


(β′)

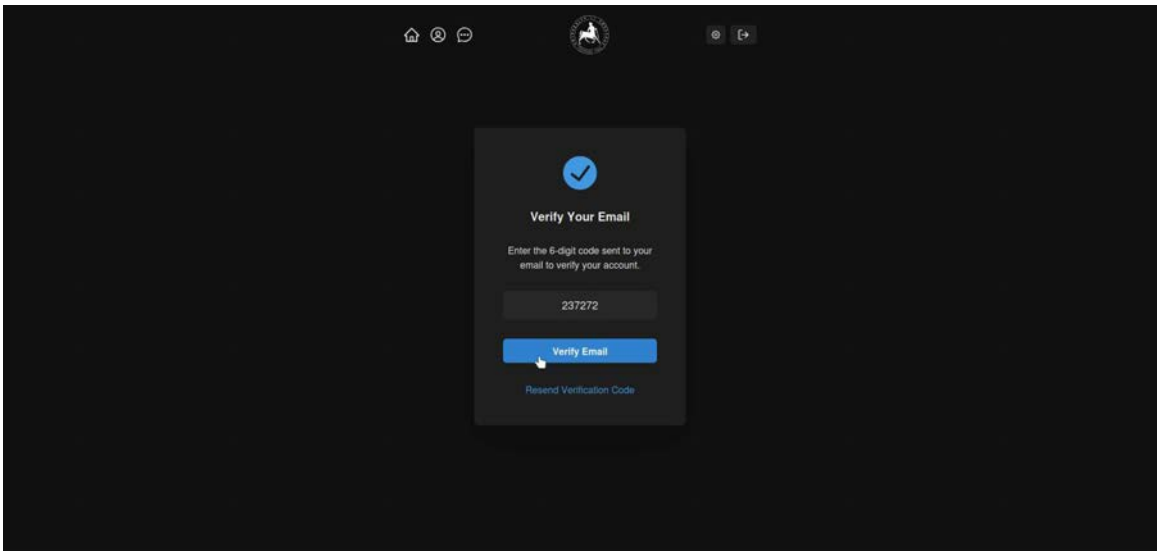
Σχήμα 5.2: (α) Επιτυχία δημιουργίας λογαριασμού (β) Αποτυχία δημιουργίας λογαριασμού

Αν η δημιουργία του λογαριασμού μας απέτυχε, τότε θα εμφανιστεί μήνυμα λάθους στην οθόνη το οποίο π.χ μπορεί να αναγράφει οτι ο χρήστης υπάρχει ήδη ή ότι υπήρξε κάποιο πρόβλημα στην βάση.

Διαφορετικά, έαν η δημιουργία του λογαριασμού ήταν επιτυχής θα εμφανιστεί μήνυμα που θα αναγράφει ότι ο λογαριασμός μας δημιουργθήκε, και στην συνέχεια θα μεταφερθούμε στην παρακάτω σελίδα στην οποία θα πρέπει να συμπληρώσουμε τον 6-ψηφιο κωδικό που μας έχει αποσταλλέι στο email.



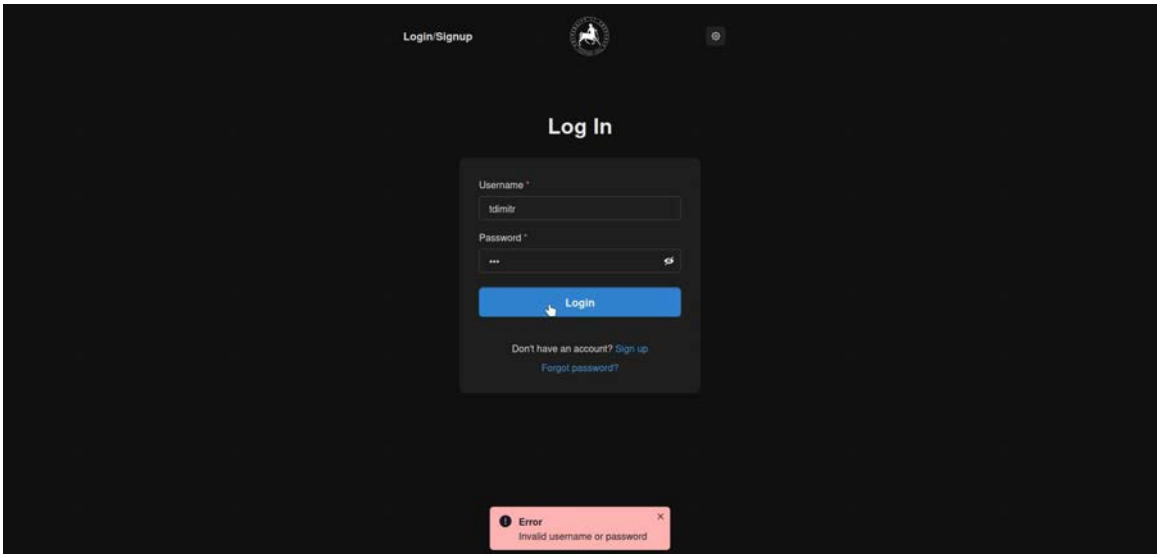
Σχήμα 5.3: Το email που μας έχει αποσταλλέι με τον 6-ψηφιο κωδικο



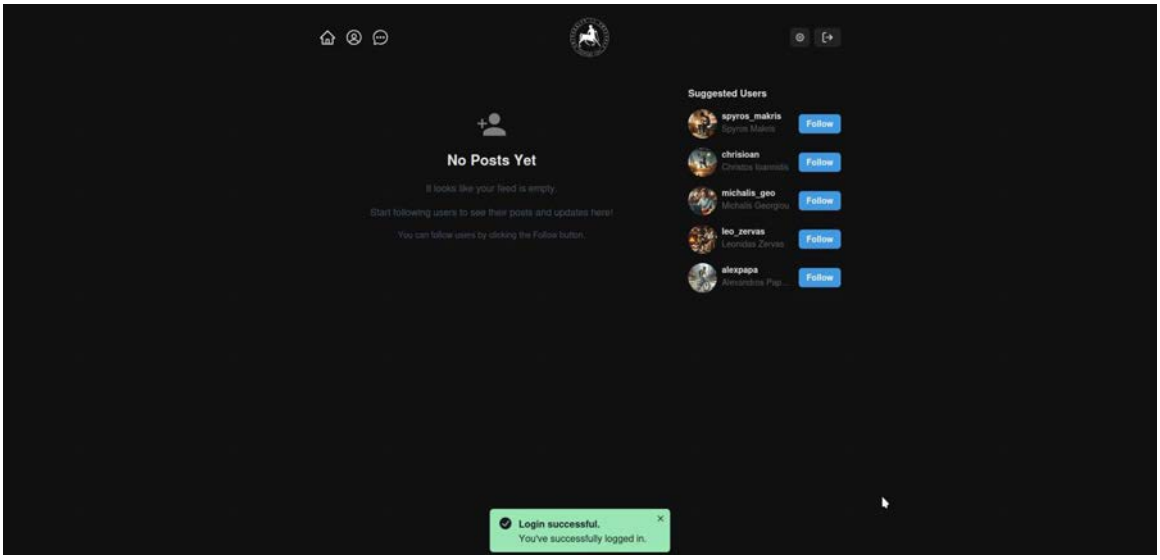
Σχήμα 5.4: Σελίδα επιβεβαίωσης λογαριασμού

Ο κωδικός αυτός έχει ισχύ για 2 λεπτά, μετά τα οποία λήγει. Εάν δεν προλάβουμε να τον χρησιμοποιήσουμε ή δεν λάβουμε τον κωδικό, υπάρχει η δυνατότητα να ζητήσουμε έναν νέο, μέσω της λειτουργίας “Resend Verification Code”. Αφού συμπληρώσουμε τον κωδικό, ο λογαριασμός μας ενεργοποιείται, και πλέον μπορούμε να περιηγηθούμε στην εφαρμογή.

5.1.2 Σελίδα Σύνδεσης



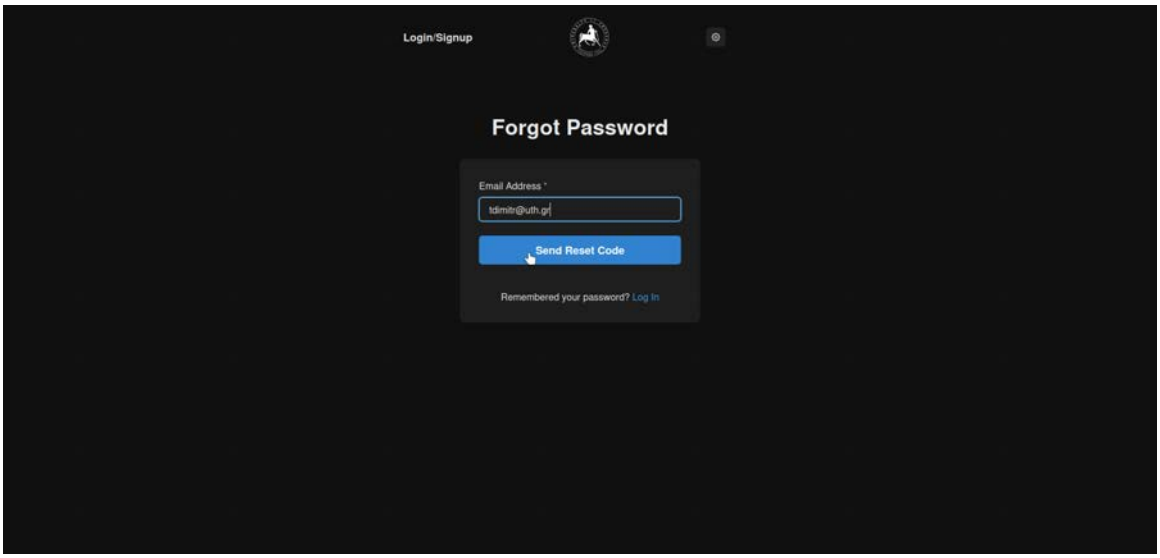
Σχήμα 5.5: Αποτυχία σύνδεσης. Λάθος ψευδώνυμο ή κωδικός



Σχήμα 5.6: Επιτυχία σύνδεσης.

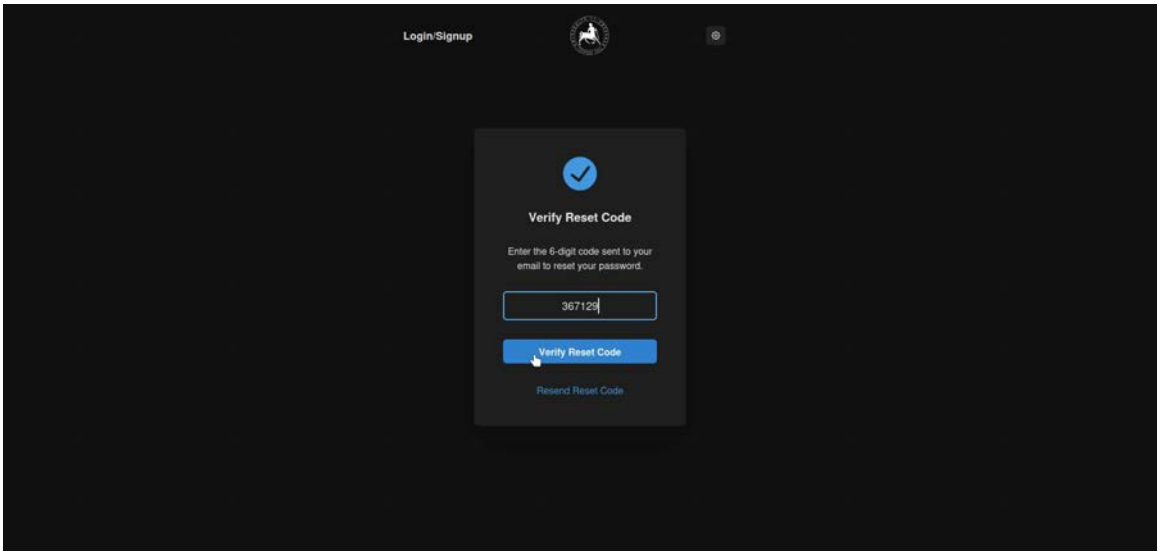
5.1.3 Επαναφορά Κωδικού Πρόσβασης

Σε περίπτωση που ξεχάσουμε τον κωδικό πρόσβασης του λογαριασμού μας, μπορούμε να τον επαναφέρουμε μέσω της επιλογής *"Forgot Password?"*, όπου θα μας ζητηθεί να εισάγουμε το email μας.



Σχήμα 5.7: Συμπλήρωση του Email για επαναφορά κωδικού πρόσβασης

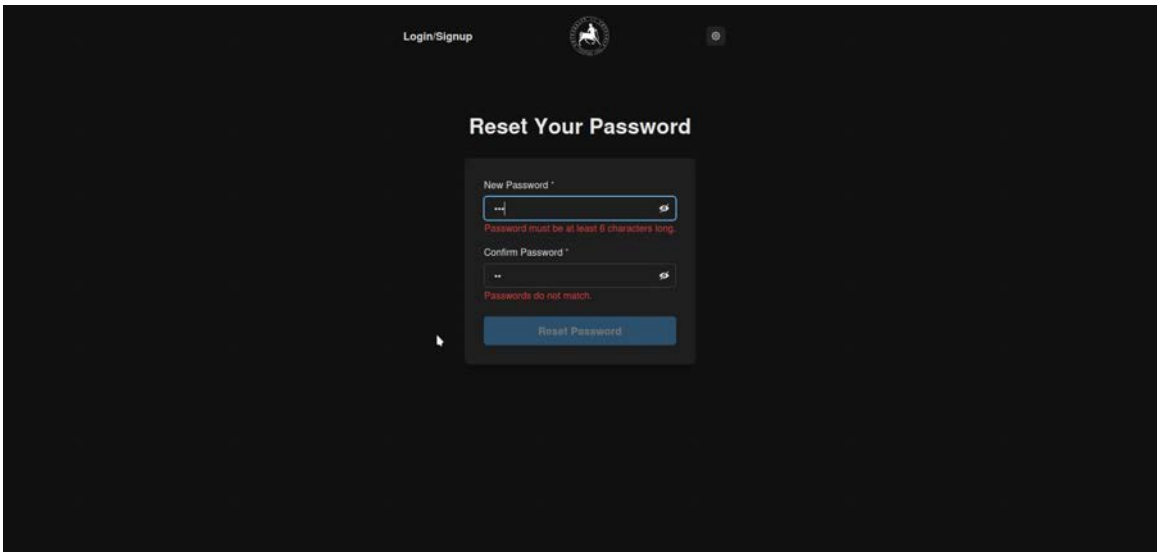
Εάν το email υπάρχει στην βάση δεδομένων τότε, θα λάβουμε έναν 6-ψήφιο κωδικό επιβεβαίωσης, τον οποίο θα πρέπει να συμπληρώσουμε στην παρακάτω σελίδα :



Σχήμα 5.8: Σελίδα επιβεβαίωσης, επαναφοράς κωδικού πρόσβασης

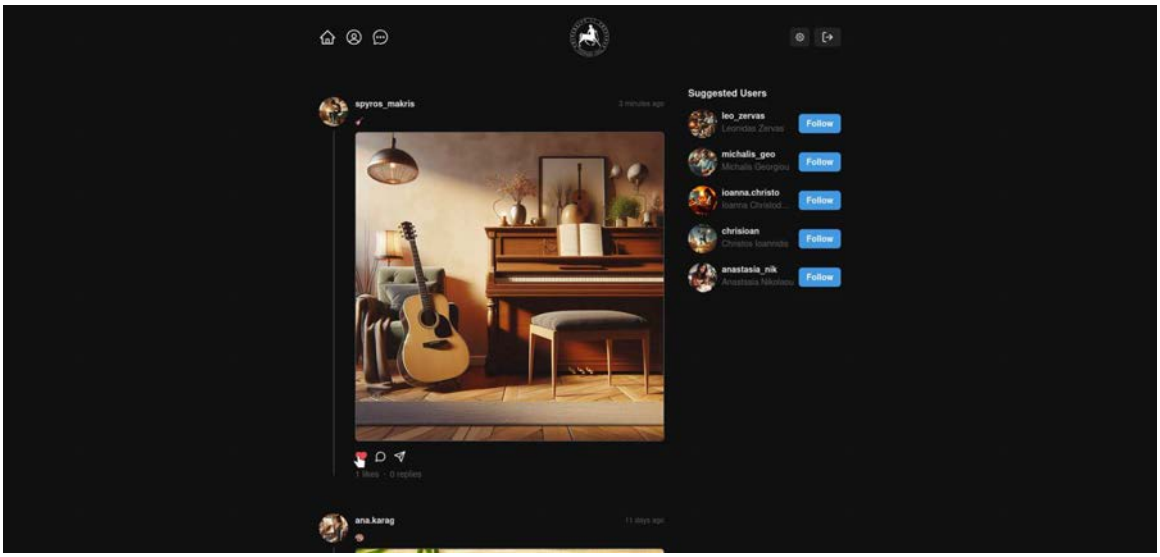
Ο κωδικός αυτός έχει ισχύ για 2 λεπτά, μετά τα οποία λήγει. Εάν δεν προλάβουμε να τον χρησιμοποιήσουμε ή δεν λάβουμε τον κωδικό, υπάρχει η δυνατότητα να ζητήσουμε έναν νέο μέσω της λειτουργίας “Resent Reset Code”.

Αφού εγκριθεί ότι ο 6-ψηφιος κωδικός είναι σωστός, μεταβαίνουμε στην παρακάτω σελίδα όπου ορίζουμε έναν νέο κωδικό πρόσβασης για τον λογαριασμο μας.



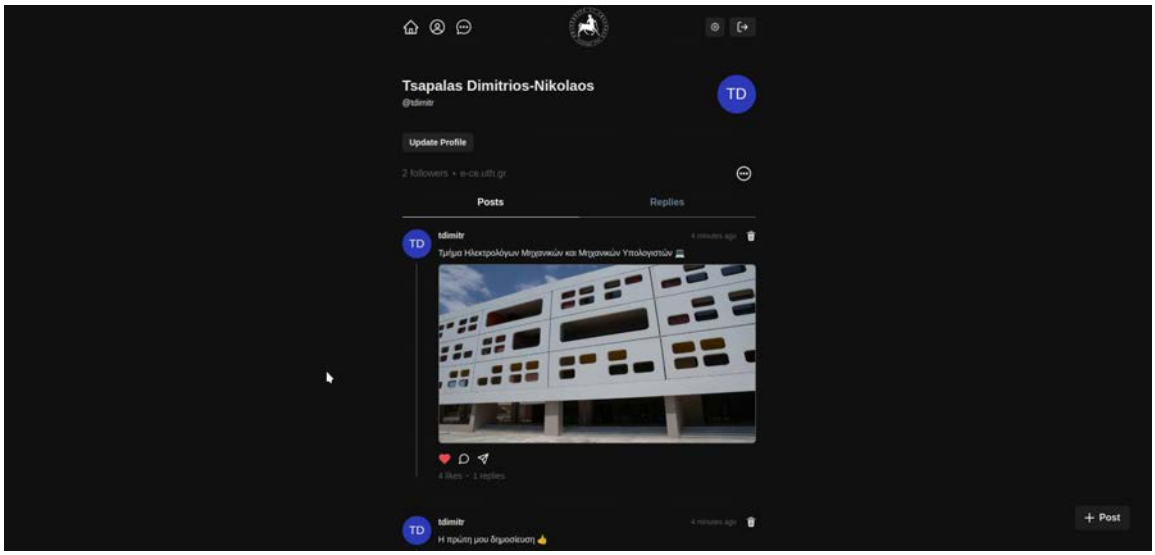
Σχήμα 5.9: Αλλαγή κωδικού πρόσβασης

5.1.4 Αρχική Σελίδα



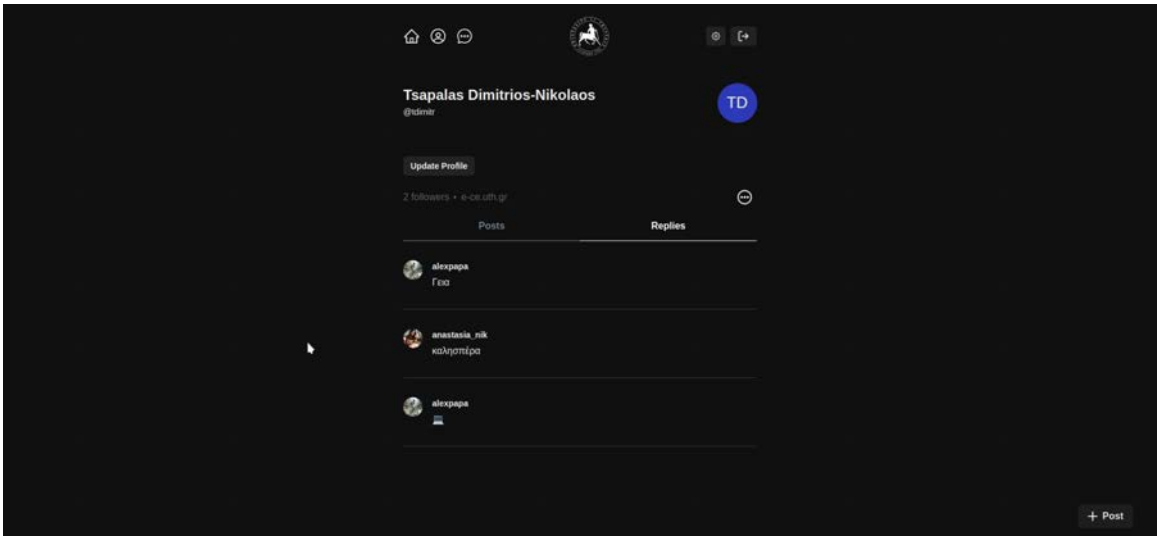
Σχήμα 5.10: Αρχική σελίδα

5.1.5 Σελίδα Προφίλ



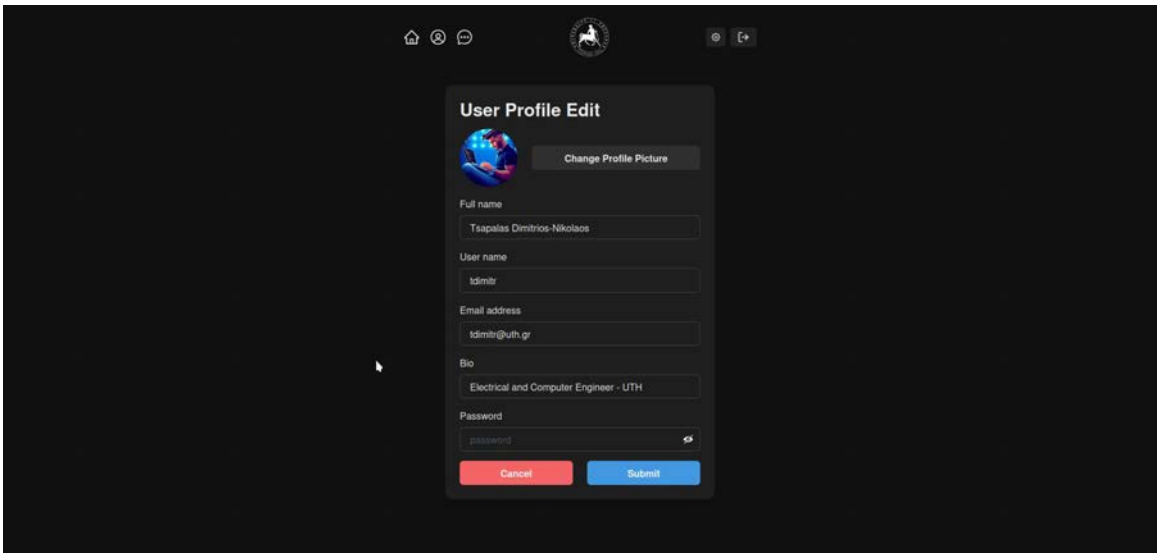
Σχήμα 5.11: Σελίδα προφίλ

5.1.6 Σελίδα Απαντήσεων



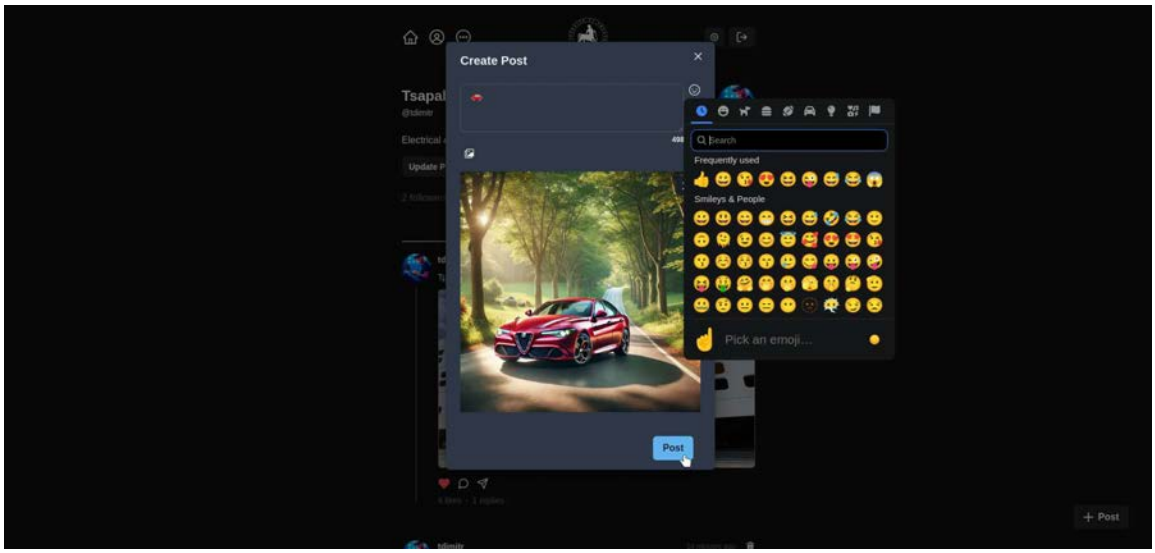
Σχήμα 5.12: Σελίδα απαντήσεων

5.1.7 Σελίδα Επεξεργασίας Προφίλ



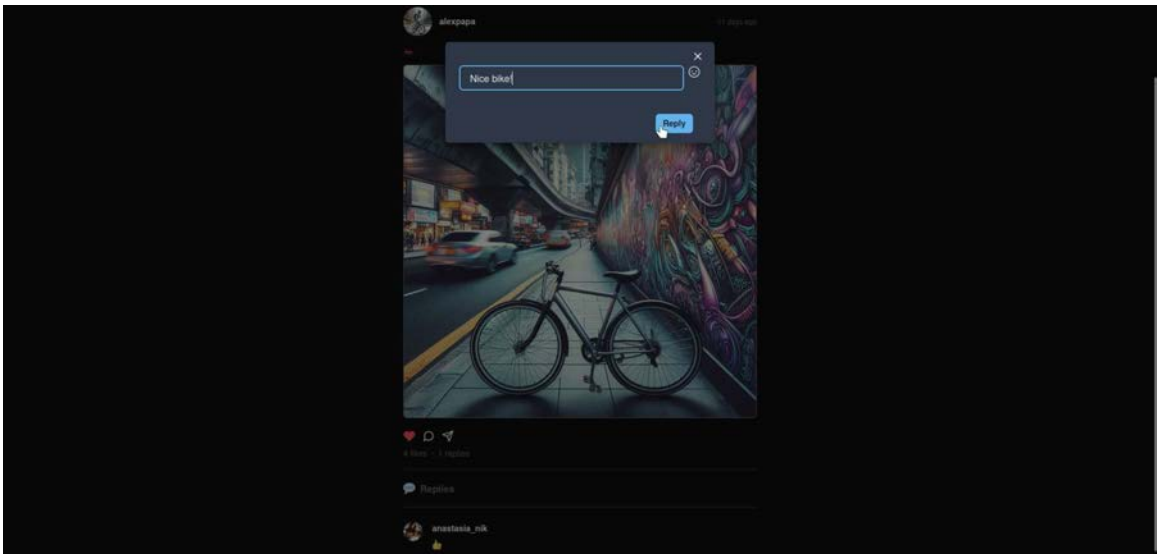
Σχήμα 5.13: Σελίδα αλλαγής φωτογραφίας προφιλ και προσωπικών στοιχείων

5.1.8 Δημοσίευση Κειμένου και Φωτογραφικού Υλικού



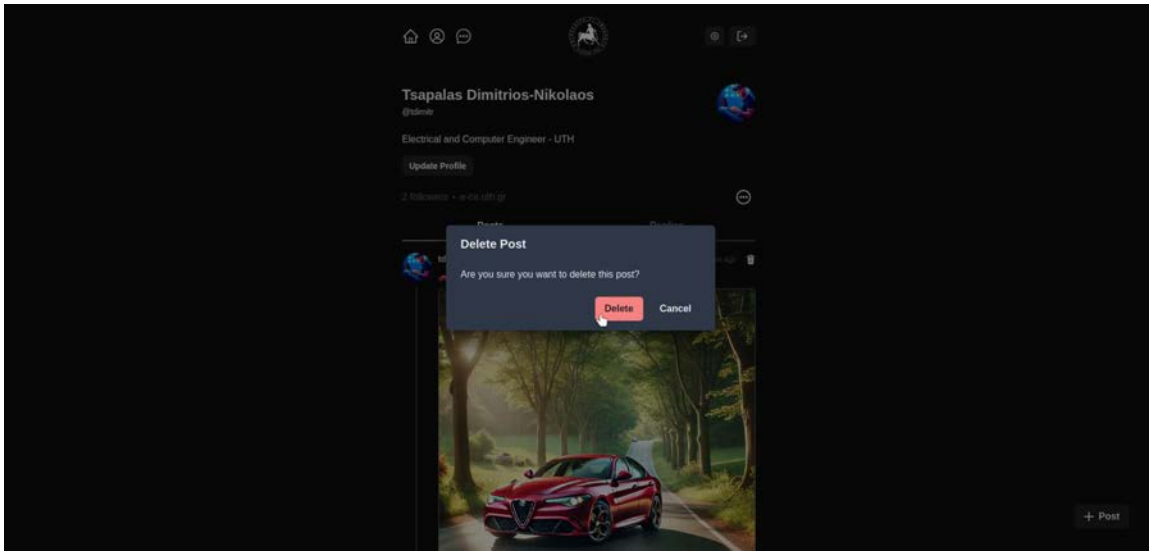
Σχήμα 5.14: Δημιουργία δημοσιεύσεων

5.1.9 Απάντηση σε Δημοσίευση



Σχήμα 5.15: Απάντηση σε δημοσιεύσεις άλλων χρηστών

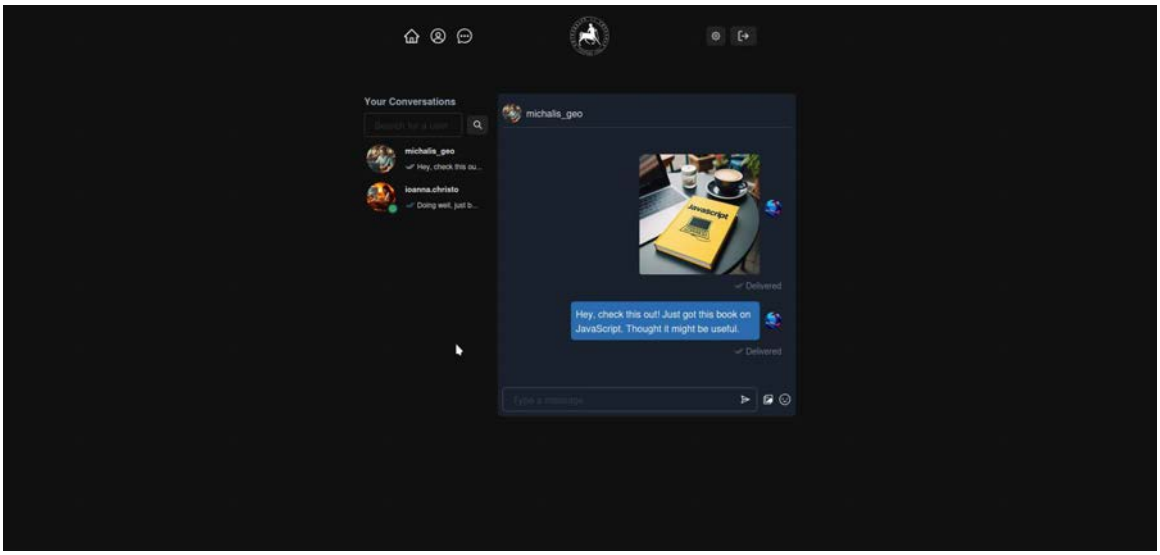
5.1.10 Διαγραφή Δημοσίευσης και Διαγραφή Απαντήσεων



Σχήμα 5.16: Διαγραφή Δημοσίευσης

Η διαδικασία διαγραφής απαντήσεων σε μια δημοσίευση, πραγματοποιείται με τον ίδιο ακριβώς τρόπο όπως φαίνεται στην παραπάνω εικόνα. Ο χρήστης έχει τη δυνατότητα να διαγράψει τα σχόλια που έχει υποβάλει σε οποιαδήποτε δημοσίευση.

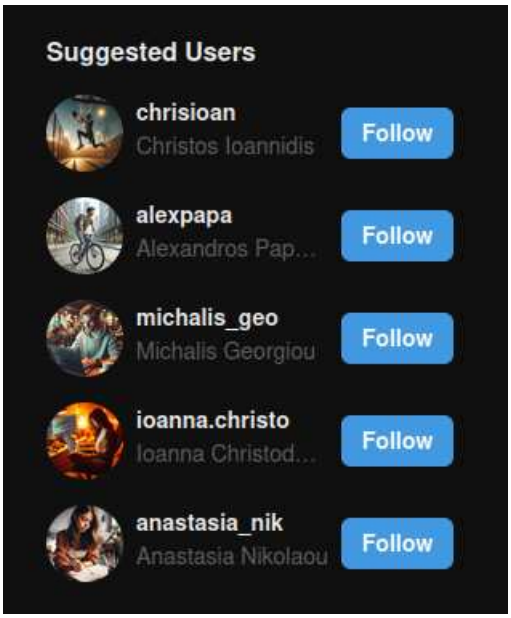
5.1.11 Σελίδα Συζητήσεων



Σχήμα 5.17: Συνομιλία με άλλους χρηστες

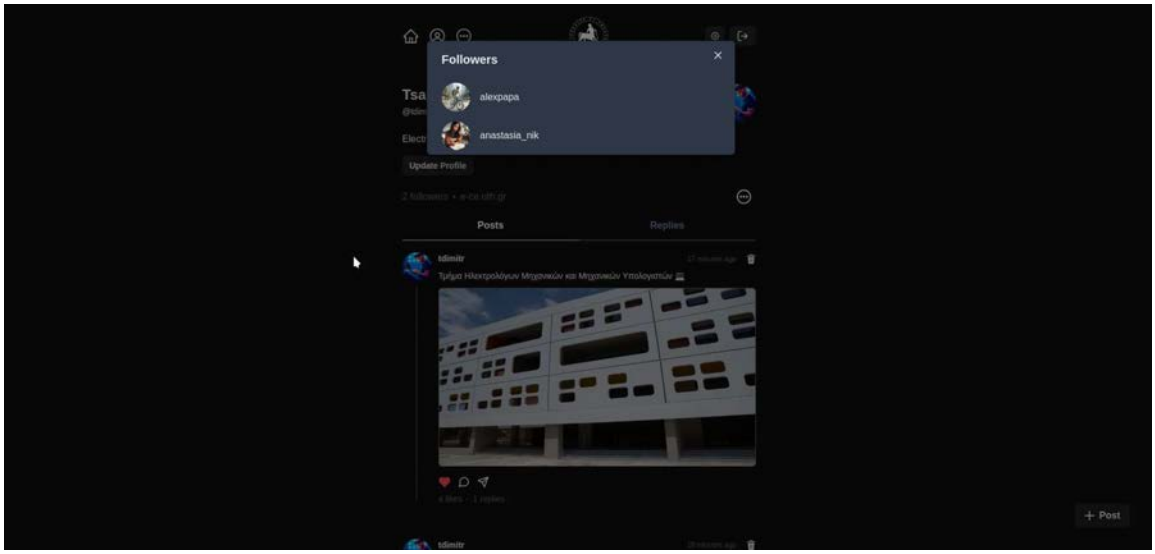
5.2 Προσθετες Λειτουργίες

5.2.1 Προτεινόμενοι Χρήστες



Σχήμα 5.18: Προτεινόμενοι χρήστες

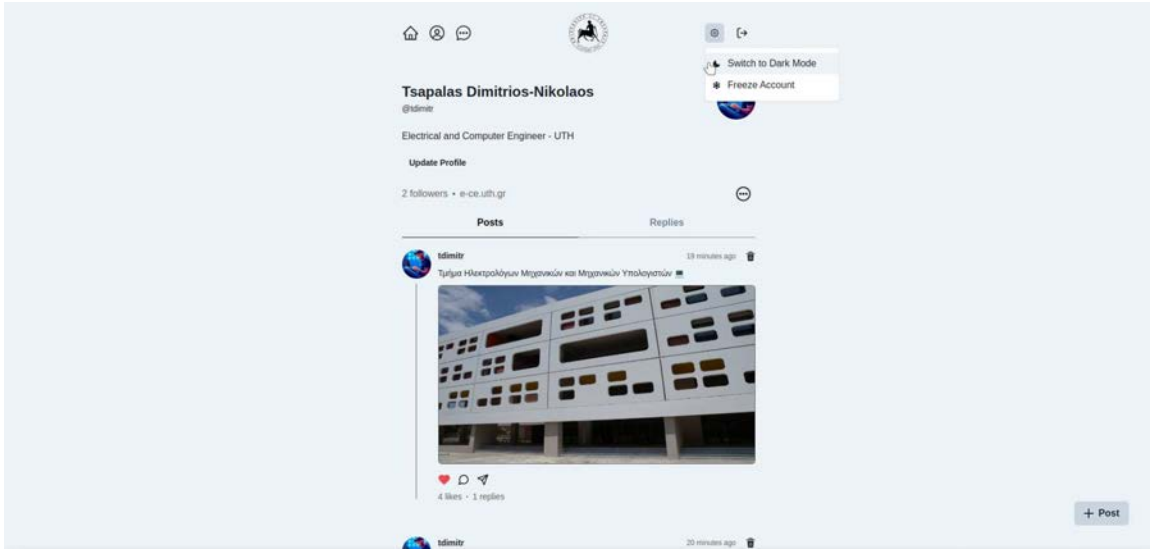
5.2.2 Λίστα Ακολούθων



Σχήμα 5.19: Λίστα ακολούθων στην σελίδα Προφιλ

5.2.3 Εναλλαγή Θέματος

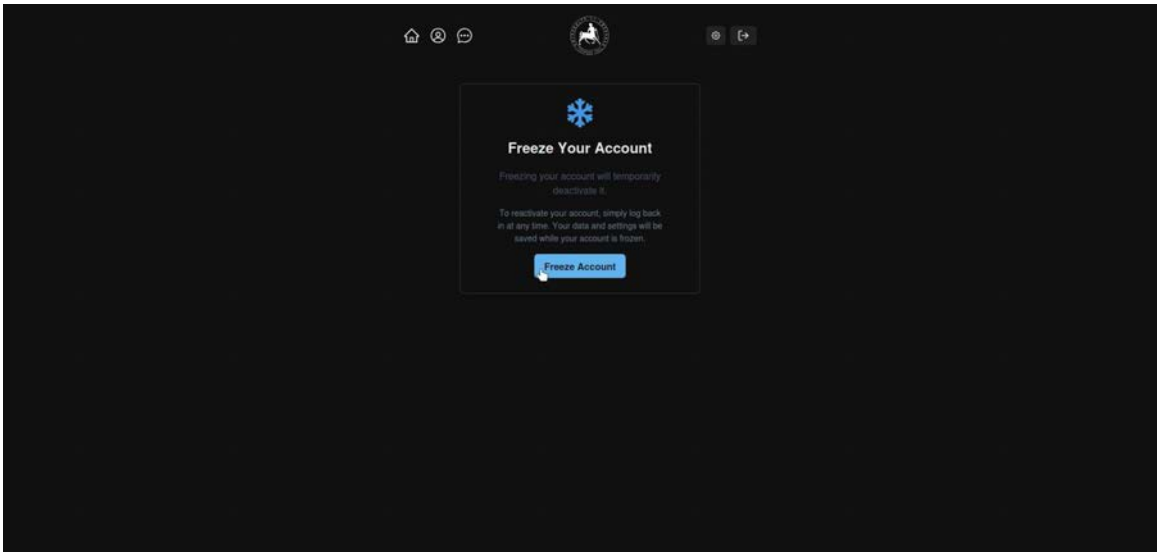
Η εφαρμογή παρέχει την δυνατότητα αλλαγής του θέματος εμφάνισης μέσω των ρυθμίσεων, επιτρέποντας στους χρήστες να εναλλάσουν σε σκοτεινό (dark mode) ή φωτεινό θέμα (light mode).



Σχήμα 5.20: Επιλογή για ενναλάγη θέματος απο σκοτεινό σε φωτεινό (και αντίστροφα)

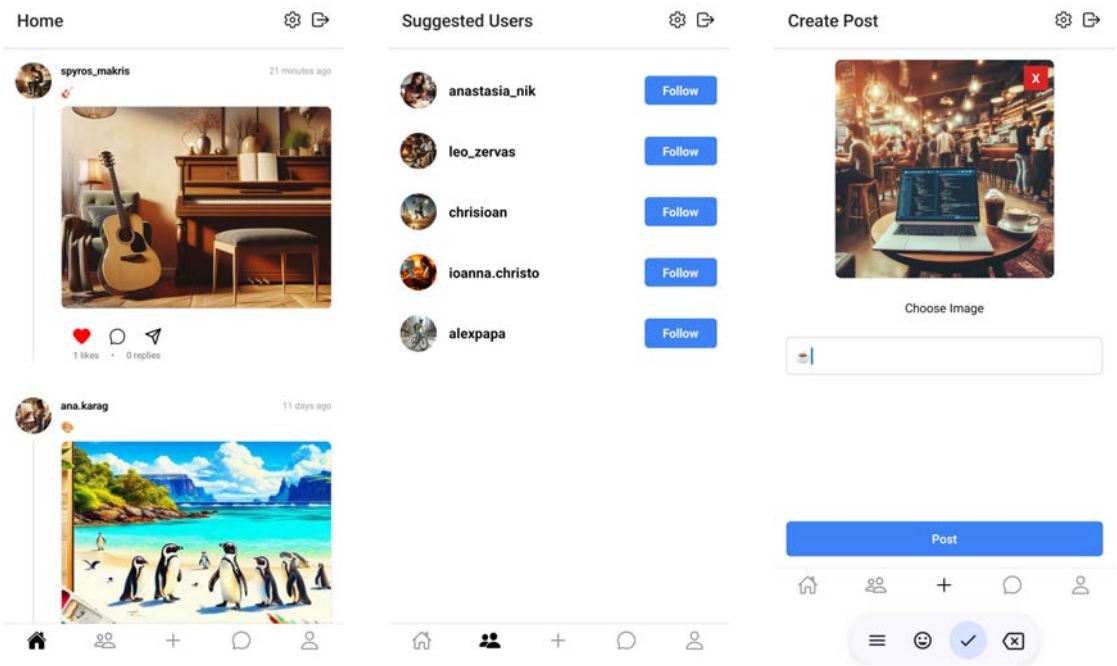
5.2.4 Προσωρινή Διαγραφή του Λογαριασμού

Η εφαρμογή παρέχει τη δυνατότητα προσωρινής διαγραφής του λογαριασμού. Τα δεδομένα του χρήστη παραμένουν αποθηκευμένα και οποτεδήποτε θελήσει, μπορει να συνδεθεί στην εφαρμογή ώστε να ενεργοποιήσει ξανά τον λογαριασμο του.

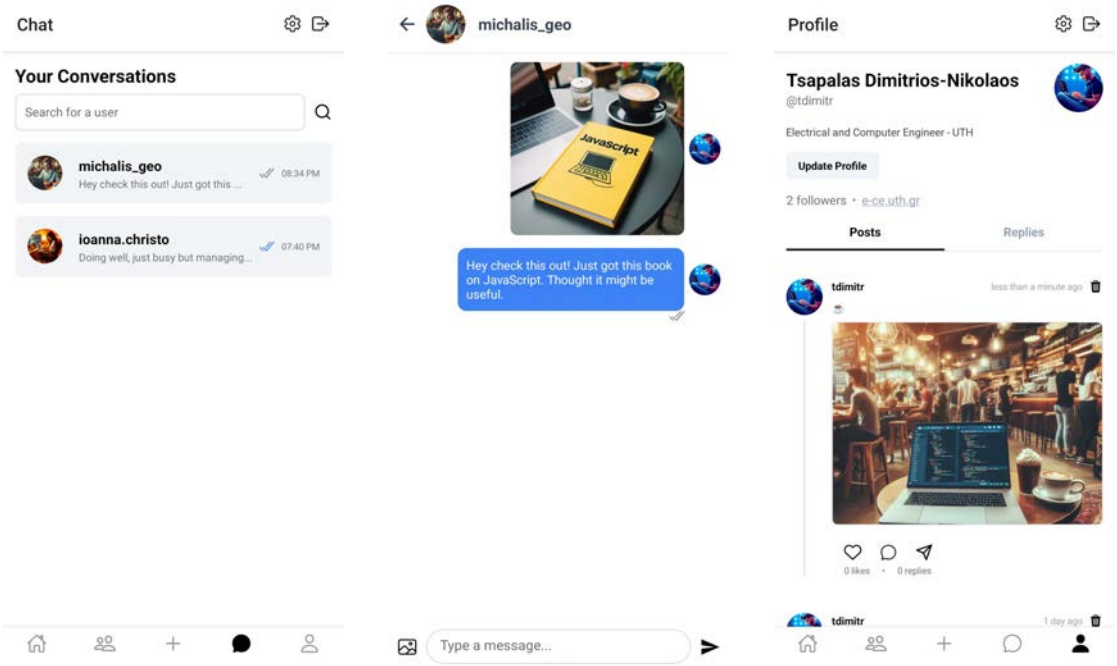


Σχήμα 5.21: Κλείδωμα του λογαριασμού

5.3 Κύριες Οθόνες Εφαρμογής Mobile



(α’) (β’) (γ’)



(δ’) (ε’) (ζ’)

Σχήμα 5.22: (α’) Αρχική Οθόνη, (β’) Οθόνη με Προτεινόμενους χρήστες προς ακολούθηση, (γ’) Οθόνη για δημιουργία δημοσίευσης, (δ’) Οθόνη Συζητήσεων, (ε’) Οθόνη Μηνυμάτων, (ζ’) Οθόνη Προφιλ.

Κεφάλαιο 6

Git και Source Code

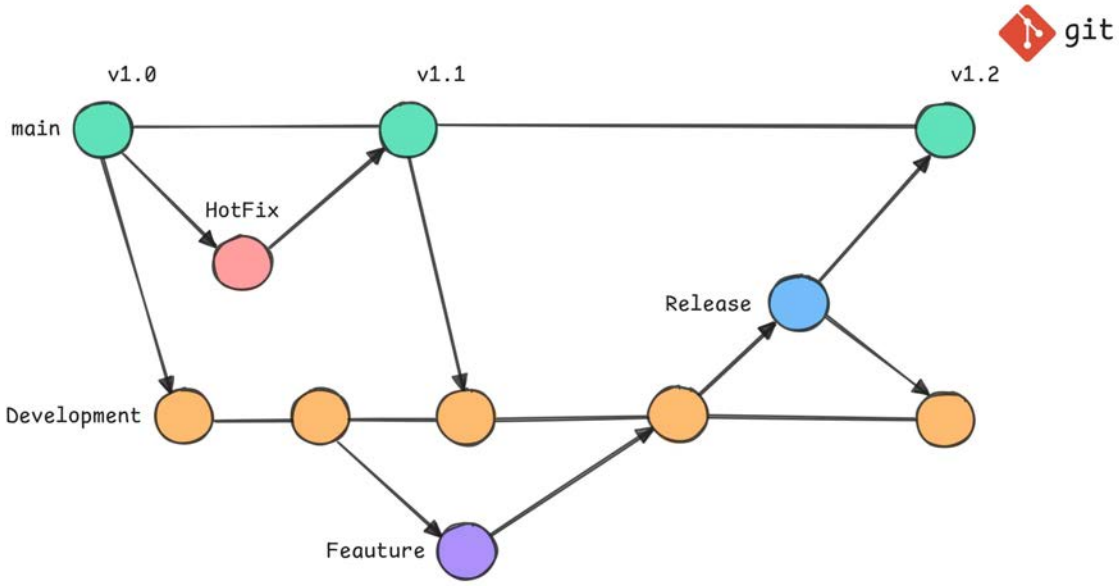
Στο παρόν κεφάλαιο εξετάζουμε το Git, ένα κατανεμημένο σύστημα ελέγχου εκδόσεων που χρησιμοποιείται στην ανάπτυξη λογισμικού. Θα παρουσιάσουμε τις βασικές έννοιες του Git, όπως τα commits και τα branches, καθώς και εντολές που χρησιμοποιούνται για τη διαχείριση του κώδικα. Στη συνέχεια, παρέχουμε οδηγίες για τη λήψη και εκτέλεση της εφαρμογής μας, συμπεριλαμβανομένων των βημάτων εγκατάστασης για το Back-End και το Front-End, τόσο για τον ιστό όσο και για τις κινητές συσκευές.

6.1 Git

Το Git είναι ένα Κατανεμημένο Σύστημα Ελέγχου Εκδόσεων (Distributed Version Control Software). Ο έλεγχος εκδόσεων επιτρέπει την αποθήκευση και την παρακολούθηση των αλλαγών στον κώδικα ενός έργου, χωρίς την αντικατάσταση ή απώλεια των προηγούμενων εκδόσεων. Η έννοια του "κατανεμημένου" σημαίνει ότι κάθε προγραμματιστής που συμμετέχει σε ένα έργο έχει στον υπολογιστή του ένα αρχείο (repository) αντίγραφο, το οποίο περιλαμβάνει όλες τις αλλαγές, τα commits και τα branches. Αυτό επιτρέπει στους προγραμματιστές να παρακολουθούν τις αλλαγές που έχουν γίνει από άλλα μέλη της ομάδας και να εργάζονται ανεξάρτητα.

Η χρήση συστήματος ελέγχου εκδόσεων είναι σημαντική, χωρίς αυτή υπάρχει κίνδυνος απώλειας δεδομένων και κώδικα. Το Git επιτρέπει στα μέλη της ομάδας να δημιουργούν "commits", δηλαδή να αποθηκεύουν την κατάσταση του έργου σε συγκεκριμένα σημεία ελέγχου. Έτσι μπορούν να προχωρούν στην ανάπτυξη της εφαρμογής, γνωρίζοντας ότι μπορούν οποτεδήποτε να επιστρέψουν σε κάποια προηγούμενη σταθερή κατάσταση.

Τα κλαδια (branches) στο Git είναι ξεχωριστές γραμμές ανάπτυξης μέσα στο ίδιο έργο. Είναι αντίγραφα του κώδικα και επιτρέπουν την ανάπτυξη νέων λειτουργιών ή την διόρθωση λαθών, χωρίς να επηρεάζονται οι κύριες εκδόσεις του κώδικα που ήδη λειτουργούν. [67].



Σχήμα 6.1: Παράδειγμα κλαδιών στο Git [68]

6.1.1 Εντολές Git

Το Git παρέχει εντολές για να διαχειριστούμε τα branches, να καταγράψουμε τις αλλαγές και να συγχρονίσουμε τον κώδικα με άλλα μέλη της ομάδας. Οι πιο κοινές εντολές που χρησιμοποιούνται στο Git είναι:

- `git init`: Δημιουργεί ένα νέο έργο Git στον τοπικό υπολογιστή.
- `git clone [url]`: Αντιγράφει ένα έργο από το GitHub στον τοπικό υπολογιστή.
- `git status`: Εμφανίζει την τρέχουσα κατάσταση του έργου, δηλαδή τα αρχεία που έχουν τροποποιηθεί ή προστεθεί και δεν έχουν καταχωρηθεί ακόμα με commit.
- `git add [αρχεία]`: Επιλέγει ποια αρχεία θα καταχωρηθούν στο επόμενο commit.
- `git commit -m "Μήνυμα"`: Αποθηκεύει τις αλλαγές σε ένα νέο αντίγραφο. Το μήνυμα πρέπει να περιγράφει την αλλαγή που έγινε, ώστε να είναι κατανοητό.
- `git log`: Εμφανίζει όλα τα commits που έχουν γίνει.

- `git branch`: Εμφανίζει όλους τους κλάδους του έργου ή δημιουργεί νέο κλάδο με `git branch [ονομα-κλαδιού]`.
- `git checkout [ονομα-κλαδιού]`: Μεταβαίνει σε ένα συγκεκριμένο κλάδο.
- `git merge [ονομα-κλαδιού]`: Συγχωνεύει τις αλλαγές από το `[ονομα-κλαδιού]` στον τρέχον κλάδο.
- `git pull`: Φέρνει τις αλλαγές από το GitHub στον τρέχον κλάδο.
- `git push`: Στέλνει τις τοπικές αποθηκεύσεις στο GitHub.

6.2 Source Code (Πηγαίος Κώδικας) & Οδηγίες Εγκατάστασης

Για να κατεβάσετε και να τρέξετε την εφαρμογή, ακολουθήστε τα παρακάτω βήματα.

Κατεβάστε τα αρχεία απο τον παρακάτω σύνδεσμο :
https://github.com/tdimitr/UTH-Social-Media-App

6.2.1 Back-End

1. Ανοίξτε το τερματικό και εκτελέστε τις παρακάτω εντολές:

```
cd backend
npm install
```

   για να κατεβάσετε τα απαιτούμενα πακέτα που αναφέρονται στο `package.json`.

2. Συμπληρώστε το αρχείο `.env.template` με τις απαραίτητες παραμέτρους και μετονομάστε το σε `.env`

3. Για να τρέξετε τον server, εκτελέστε την εντολή:

```
npm run dev
```

6.2.2 Front-End / Web (Ιστός)

1. Ανοίξτε το τερματικό και εκτελέστε τις παρακάτω εντολές:

```
cd frontend/web
npm install
```

   για να κατεβάσετε τα απαιτούμενα πακέτα που αναφέρονται στο `package.json`.

2. Για να τρέξετε την εφαρμογή στο web, εκτελέστε την εντολή:

```
npm run dev
```

6.2.3 Front-End / Mobile

1. Ανοίξτε το τερματικό και εκτελέστε τις παρακάτω εντολές:

```
cd frontend/mobile
npm install
```

   για να κατεβάσετε τα απαιτούμενα πακέτα που αναφέρονται στο `package.json`.

2. Συμπληρώστε στο αρχείο `src/context/uploadingImgToCloudinary.jsx` το `<CLOUD_NAME>` με το κωδικό που λάβατε απο το Cloudinary.

3. Για να τρέξετε την εφαρμογή στην κινητή συσκευή, ακολουθήστε τα παρακάτω βήματα:

   (α′) Εγκαταστήστε την εφαρμογή `Expo Go` από το Google Play Store ή το App Store.

   (β′) Εκτελέστε την εντολή:

```
npm start
```

   για να ξεκινήσει η εφαρμογή.

(γ′) Σαρώστε το QR code με την εφαρμογή `Expo Go` με την κινητή σας συσκευή για να τρέξετε την εφαρμογή.

Κεφάλαιο 7

Συμπεράσματα

Στο παρόν κεφάλαιο παρατίθενται συνοπτικά, τα συμπεράσματα που προέκυψαν από την εκπόνηση αυτής της διπλωματικής εργασίας, καθώς και προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής.

7.1 Σύνοψη και συμπεράσματα

Η παρούσα διπλωματική εργασία είχε ως στόχο την ανάπτυξη μίας εφαρμογής κοινωνικών δικτύων που βασίζεται στις σύγχρονες τεχνολογίες. Η υλοποίηση βασίστηκε στη MERN στοίβα και στο React Native για την ανάπτυξη μιας Cross-Platform εφαρμογής.
Τα βασικά αποτελέσματα της εργασίας συνοψίζονται ως εξής:

1. **Ανάπτυξη μιας ολοκληρωμένης και λειτουργικής εφαρμογής**: Δημιουργήθηκε μια εφαρμογή η οποία περιλαμβάνει κάποιες απο τις βασικές λειτουργίες παρόμοιων εφαρμογών κοινωνικής δικτύωσης, όπως δημιουργία δημοσιεύσεων, σχόλια, likes και ανταλλαγή μηνυμάτων.

2. **Ασφάλεια**: Δόθηκε ιδιαίτερη βαρύτητα στην προστασία των δεδομένων και στην ασφάλεια της εφαρμογής, χρησιμοποιώντας τεχνολογίες όπως το JSON Web Token (JWT) για την αυθεντικοποίηση, το bcrypt για την κρυπτογράφηση των κωδικών πρόσβασης και τέλος την αποστολή 6-ψήφιων κωδικών μέσω email για την επαλήθευση των λογαριασμών.

3. **Cross-Platform**: Η χρήση του React Native επέτρεψε στους χρήστες να έχουν πρόσβαση στην εφαρμογή όχι μόνο μέσω της ιστοσελίδας, αλλά και από συσκευές Android και iOS.

7.2 Μελλοντικές επεκτάσεις

Οι εφαρμογές κοινωνικών δίκτυων έχουν μεγάλο περιθώριο εξέλιξης. Ακόμα και οι μεγαλύτερες και πιο γνωστές πλατφόρμες, συνεχώς προσθέτουν νέες δυνατότητες για να διατηρήσουν το ενδιαφέρον των χρηστών τους. Αυτή η συνεχής εξέλιξη δείχνει πόσο σημαντικό είναι να προσαρμόζεται μια εφαρμογή στις νέες τάσεις και ανάγκες των χρηστών.

Παρακάτω ακολουθούν ορισμένα απο τα μελλοντικά σχέδια της εφαρμογής.

7.2.1 Ομάδες

Η δυνατότητα δημιουργίας ομάδων θα επιτρέψει στους χρήστες να σχηματίζουν φιλίες με βάση τα ενδιαφέροντα τους. Κάθε ομάδα μπορεί να έχει τις δικές της συζητήσεις και δραστηριότητες, δίνοντας τη δυνατότητα στους χρήστες να αλληλεπιδρούν με άλλους, που έχουν παρόμοια ενδιαφέροντα.

7.2.2 Ρόλοι

Η ενσωμάτωση ρόλων στην εφαρμογή θα επιτρέπει στους χρήστες να έχουν διαφορετικά επίπεδα πρόσβασης. Οι Διαχειριστές (Admins) θα μπορούν να ελέγχουν τη διαχείριση της πλατφόρμας, θα έχουν την δυνατότητα να επιβάλλουν κανόνες και να απομακρύνουν χρηστες. Ενώ απο την άλλη, οι Χρήστες (Users) θα έχουν την δυνατότητα να συμμετέχουν σε συζητήσεις να δημοσιεύουν περιεχόμενο ή να αλληλεπιδρούν με άλλους εντός των κανόνων.

7.2.3 Βίντεο και Live Streaming

Η προσθήκη υποστήριξης βίντεο και ζωντανής μετάδοσης θα επιτρέψει στους χρήστες να μοιράζονται ζωντανά τις εμπειρίες τους με αλλους.

7.2.4 Προσωποποιημένα Posts με Αλγορίθμους

Η εφαρμογή θα μπορούσε επίσης να χρησιμοποιεί αλγόριθμους για να προσαρμόζει το περιεχόμενο στις προτιμήσεις του κάθε χρήστη. Αυτό σημαίνει ότι οι χρήστες θα βλέπουν δημοσιεύσεις και πληροφορίες που είναι πιο πιθανό να τους ενδιαφέρουν, βασισμένες σε προηγούμενες αλληλεπιδράσεις τους.

7.2.5 Εκδηλώσεις

Με την προσθήκη Δημιουργίας Εκδηλώσεων, οι χρήστες θα έχουν τη δυνατότητα να οργανώσουν εκδηλώσεις και δραστηριότητες. Αυτο θα μπορούσε να περιλαμβάνει την οργάνωση ημερίδων, συνεδρίων, ή και συναντήσεων. Η εφαρμογή θα μπορούσε να προσφέρει εργαλεία για τη διαχείριση των εκδηλώσεων, όπως τη δημιουργία ημερολογίων, την αποστολή ειδοποιήσεων για νέες εκδηλώσεις και την καταγραφή συμμετοχών.

7.2.6 Λειτουργία Εκτός Σύνδεσης

Η υποστήριξη λειτουργίας εκτός σύνδεσης (offline mode) θα επιτρέψει στους χρήστες να χρησιμοποιούν λειτουργίες της εφαρμογής ακόμα και χωρίς σύνδεση στο διαδίκτυο. Αυτό είναι ιδιαίτερα σημαντικό για εφαρμογές σε κινητές συσκευές, καθώς η σύνδεση στο διαδίκτυο μπορεί να χαθεί προσωρινά λόγω αδύναμου σήματος. Με την υλοποίηση αυτής της δυνατότητας, οι χρήστες θα μπορούν να διαβάζουν προηγούμενα μηνύματα και να γράφουν νέες δημοσιεύσεις που θα αποστέλλονται αυτόματα όταν αποκατασταθεί η σύνδεση.

7.2.7 Πλαίσια Πλήρους Στοίβας

Η χρήση πλαισίων πλήρους στόιβας, όπως το Next.js θα μπορούσε να βελτιώσει την απόδοση της εφαρμογής, προσφέροντας ταχύτερη φόρτωση των σελίδων.

Βιβλιογραφία

[1] Kai Lei, Yining Ma, and Zhi Tan. Performance comparison and evaluation of web development technologies in php, python and node.js. In *IEEE 17th International Conference on Computational Science and Engineering*, Shenzhen, China, 2014.

[2] Cornelia Győrödi, Robert Győrödi, Ioana Andrada Olah, and Livia Bandici. A comparative study between the capabilities of mysql vs. mongodb as a back-end for an online platform. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 2016.

[3] Naveen Chikkanayakanahalli Ramachandrappa. A comparative analysis of native vs react native mobile app development. *International Journal of Computer Trends and Technology*, 2024.

[4] Html. `https://www.w3schools.com/html/`. Ημερομηνια Επισκεψης: 07-02-2025.

[5] Css. `https://www.w3schools.com/css/`. Ημερομηνια Επισκεψης: 07-02-2025.

[6] Javascript. `https://en.wikipedia.org/wiki/JavaScript`. Ημερομηνια Επισκεψης: 07-02-2025.

[7] Typescript. `https://www.typescriptlang.org/`. Ημερομηνια Επισκεψης: 07-02-2025.

[8] Fullstack development. `https://www.mongodb.com/resources/basics/full-stack-development`. Ημερομηνια Επισκεψης: 07-02-2025.

[9] React. `https://react.dev/learn`. Ημερομηνια Επισκεψης: 07-02-2025.

[10] Angular. https://angular.dev/overview. Ημερομηνια Επισκεψης: 07-02-2025.

[11] Vue. https://vuejs.org/guide/introduction.html. Ημερομηνια Επισκεψης: 07-02-2025.

[12] Svelte. https://svelte.dev/docs. Ημερομηνια Επισκεψης: 07-02-2025.

[13] nodejs. https://nodejs.org/. Ημερομηνια Επισκεψης: 07-02-2025.

[14] Expressjs. https://expressjs.com/. Ημερομηνια Επισκεψης: 07-02-2025.

[15] mongodb. https://www.mongodb.com/docs/. Ημερομηνια Επισκεψης: 07-02-2025.

[16] Εικόνα: Mongodb. https://www.pragimtech.com/blog/mongodb-tutorial/relational-and-non-relational-databases/. Ημερομηνια Επισκεψης: 07-02-2025.

[17] postgres. https://www.postgresql.org/docs/. Ημερομηνια Επισκεψης: 07-02-2025.

[18] firebase. https://firebase.google.com/docs. Ημερομηνια Επισκεψης: 07-02-2025.

[19] Next vs remix. https://zerotomastery.io/blog/remix-vs-next/. Ημερομηνια Επισκεψης: 07-02-2025.

[20] Http methods. https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/HTTP-methods. Ημερομηνια Επισκεψης: 07-02-2025.

[21] Http status. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status. Ημερομηνια Επισκεψης: 07-02-2025.

[22] Jsx. https://react.dev/learn/writing-markup-with-jsx. Ημερομηνια Επισκεψης: 07-02-2025.

[23] Dom. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Ημερομηνια Επισκεψης: 07-02-2025.

[24] Virtual dom. https://www.freecodecamp.org/news/what-is-the-virtual-dom-in-react/. Ημερομηνια Επισκεψης: 07-02-2025.

[25] Api. https://aws.amazon.com/what-is/api/. Ημερομηνια Επισκεψης: 07-02-2025.

[26] Graphql. https://graphql.org/learn/. Ημερομηνια Επισκεψης: 07-02-2025.

[27] Mikhail Sakhniuk and Adam Boduch. *React and React Native: Build cross-platform JavaScript and TypeScript apps for the web, desktop, and mobile*. Packt Publishing, 5th edition, 2024.

[28] React native components. https://reactnative.dev/docs/components-and-apis. Ημερομηνια Επισκεψης: 07-02-2025.

[29] Αρχιτεκτονική react native. https://medium.com/coox-tech/deep-dive-into-react-natives-new-architecture-fb67ae615ccd. Ημερομηνια Επισκεψης: 07-02-2025.

[30] Οφέλη Ανάπτυξης native Κινητών εφαρμογών. https://www.scalefocus.com/blog/14-benefits-of-native-mobile-app-development. Ημερομηνια Επισκεψης: 07-02-2025.

[31] Webview. https://developer.android.com/develop/ui/views/layout/webapps/webview. Ημερομηνια Επισκεψης: 07-02-2025.

[32] Mern stack. https://www.mongodb.com/resources/languages/mern-stack. Ημερομηνια Επισκεψης: 07-02-2025.

[33] Mongodb atlas. https://www.mongodb.com/products/platform/atlas-database. Ημερομηνια Επισκεψης: 07-02-2025.

[34] Nodejs and express. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. Ημερομηνια Επισκεψης: 07-02-2025.

[35] dotenv. https://www.npmjs.com/package/dotenv. Ημερομηνια Επισκεψης: 07-02-2025.

[36] mongoose. https://www.npmjs.com/package/mongoose. Ημερομηνια Επισκεψης: 07-02-2025.

[37] Cors. https://www.npmjs.com/package/cors. Ημερομηνια Επισκεψης: 07-02-2025.

[38] bcrypt. https://www.npmjs.com/package/bcrypt. Ημερομηνια Επισκεψης: 07-02-2025.

[39] nodemailer. https://www.npmjs.com/package/nodemailer. Ημερομηνια Επισκεψης: 07-02-2025.

[40] cookie-parser. https://www.npmjs.com/package/cookie-parser. Ημερομηνια Επισκεψης: 07-02-2025.

[41] Json web token. https://www.npmjs.com/package/jsonwebtoken. Ημερομηνια Επισκεψης: 07-02-2025.

[42] Cloudinary. https://www.npmjs.com/package/cloudinary. Ημερομηνια Επισκεψης: 07-02-2025.

[43] Δημιουργία cloudinary λογαριασμού. https://cloudinary.com/users/register_free. Ημερομηνια Επισκεψης: 07-02-2025.

[44] Socket.io. https://www.npmjs.com/package/socket.io. Ημερομηνια Επισκεψης: 07-02-2025.

[45] Vite. https://vite.dev/. Ημερομηνια Επισκεψης: 07-02-2025.

[46] fetch. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch. Ημερομηνια Επισκεψης: 07-02-2025.

[47] Chakra-ui. https://www.chakra-ui.com/. Ημερομηνια Επισκεψης: 07-02-2025.

[48] React router dom. https://reactrouter.com/. Ημερομηνια Επισκεψης: 07-02-2025.

[49] Emoji mart. https://www.npmjs.com/package/@emoji-mart/react. Ημερομηνια Επισκεψης: 07-02-2025.

[50] Components and props. https://react.dev/learn/passing-props-to-a-component. Ημερομηνια Επισκεψης: 07-02-2025.

[51] usestate. https://react.dev/reference/react/useState. Ημερομηνια Επισκεψης: 07-02-2025.

[52] useeffect. https://react.dev/reference/react/useEffect. Ημερομηνια Επισκεψης: 07-02-2025.

[53] Hooks. https://react.dev/reference/react/hooks. Ημερομηνια Επισκεψης: 07-02-2025.

[54] Recoil. https://recoiljs.org/. Ημερομηνια Επισκεψης: 07-02-2025.

[55] Εικόνα: Prop drilling. https://www.scaler.com/topics/react/prop-drilling-in-react/. Ημερομηνια Επισκεψης: 07-02-2025.

[56] Εικόνα: Recoil. https://medium.com/better-programming/recoiljs-the-future-of-react-state-management-ffb1345833b6. Ημερομηνια Επισκεψης: 07-02-2025.

[57] Expo installation. https://docs.expo.dev/router/installation/. Ημερομηνια Επισκεψης: 07-02-2025.

[58] Expo go: Google store. https://play.google.com/store/apps/details?id=host.exp.exponent&hl=en. Ημερομηνια Επισκεψης: 07-02-2025.

[59] Expo go: Apple store. https://apps.apple.com/us/app/expo-go/id982107779. Ημερομηνια Επισκεψης: 07-02-2025.

[60] Emulator. https://developer.android.com/studio/run/emulator. Ημερομηνια Επισκεψης: 07-02-2025.

[61] Xcode. https://developer.apple.com/xcode/. Ημερομηνια Επισκεψης: 07-02-2025.

[62] Expo router. https://docs.expo.dev/router/introduction/. Ημερομηνια Επισκεψης: 07-02-2025.

[63] Url parameters. https://docs.expo.dev/router/reference/url-parameters/. Ημερομηνια Επισκεψης: 07-02-2025.

[64] Securestore. https://docs.expo.dev/versions/latest/sdk/securestore/. Ημερομηνια Επισκεψης: 07-02-2025.

[65] Expo icons. https://icons.expo.fyi/Index. Ημερομηνια Επισκεψης: 07-02-2025.

[66] Postman. https://www.postman.com/. Ημερομηνια Επισκεψης: 07-02-2025.

[67] Git. https://git-scm.com/. Ημερομηνια Επισκεψης: 07-02-2025.

[68] Κλαδιά. https://www.geeksforgeeks.org/branching-strategies-in-git/. Ημερομηνια Επισκεψης: 07-02-2025.