# ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ


**Χρήση τεχνολογιών Progressive Web Apps για την υλοποίηση εφαρμογής φοιτητολογίου**

Πτυχιακή εργασία


**Δημήτριος Κατωμέρης**



Αθήνα, Ιανουάριος 2021

# ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

**Τριμελής Εξεταστική Επιτροπή**


**Τσερπές Κωνσταντίνος (Επιβλέπων)**
**Επίκουρος Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο**

**Βαρλάμης Ηρακλής**
**Επίκουρος Καθηγητής, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο**

**Νικολαΐδη Μάρα**
**Καθηγήτρια, Πληροφορική και Τηλεματική, Χαροκόπειο Πανεπιστήμιο**

Ο Δημήτριος Κατωμέρης

δηλώνω υπεύθυνα ότι:

**1)** Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

**2)** Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

# ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω ιδιαιτέρως τον καθηγητή μου κ. Τσερπέ Κωνσταντίνο για τη βοήθεια και την καθοδήγησή του κατά τη διάρκεια της πτυχιακής, αλλά ιδιαιτέρως για την προτροπή του να ασχοληθώ με τις τεχνολογίες με τις οποίες υλοποιήθηκε η εργασία, καθώς ανακάλυψα μια κλίση προς αυτές και άρχισα την επαγγελματική του καριέρα πάνω σε αυτές.

Θα ήθελα, επίσης, να ευχαριστήσω την τριμελής επιτροπή, τον κ. Ηρακλή Βαρλάμη και την κ. Μάρα Νικολαΐδου για τον χρόνο που θα διαθέσουν να με αξιολογήσουν.

Τέλος, ευχαριστώ την οικογένειά μου, τους φίλους μου και τους συναδέλφους μου για τη στήριξη τους κατά τη διάρκεια της εκπόνησης την πτυχιακής μου εργασίας.

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

# Περίληψη

Σκοπός της πτυχιακής εργασίας είναι η υλοποίηση ενός φοιτητολογίου για όλα τα τμήματα του Πανεπιστημίου μας, στα πρότυπα των τεχνολογιών γνωστές ως “Progressive Web Apps”. Με τη βοήθεια αυτών των τεχνολογιών, έχω ως στόχο να δημιουργήσω ένα περιβάλλον μοντέρνο, καινοτόμο και χρηστικό για τους φοιτητές, κάτι που αρμόζει σε μια εφαρμογή τη σήμερον ημέρα.

Η εφαρμογή δημιουργήθηκε πάνω σε καινούριες τεχνολογίες στο backend και στο frontend, με κύριο σκοπό την εύκολη πλοήγηση του χρήστη και τη μέγιστη χρησιμότητα. Επιπλέον, σκεπτόμενος την εποχή που ζούμε και τη ραγδαία αύξηση της χρήσης smartphones, ιδιαίτερα από τους νέους ανθρώπους, θέλησα να φτιάξω μια πλατφόρμα στην οποία οι φοιτητές θα είχαν την ίδια εμπειρία χρήσης είτε από ηλεκτρονικό υπολογιστή είτε από κινητό τηλέφωνο, κάτι το οποίο δεν είναι δυνατόν με το υπάρχων φοιτητολόγιο.

Κάνοντας την εφαρμογή Progressive Web App, οι χρήστες θα έχουν τη δυνατότητα να απολαύσουν μια εμπειρία χρήσης που δεν παρέχεται από τις συμβατικές Εφαρμογές Διαδικτύου. Μεταξύ αυτών είναι η δυνατότητα εγκατάστασης της εφαρμογής από έναν browser και όχι από κάποιο App Store του εκάστοτε κινητού, της χρήσης push notifications, τη χρήση ενός μέρους της εφαρμογής ακόμα και χωρίς πρόσβαση στο διαδίκτυο, αλλά ακόμα και στην ταχύτητα πρόσβασης της εφαρμογής.

Το φοιτητολόγιο φτιάχτηκε έτσι ώστε να μπορεί να εξυπηρετεί και τους φοιτητές αλλά και τη γραμματεία. Οι φοιτητές θα μπορούν μέσω της πλατφόρμας να βλέπουν τις βαθμολογίες τους στα μαθήματα, να δηλώνουν μαθήματα κάθε εξάμηνο, να κάνουν δηλώσεις για διάφορα έγγραφα που μπορεί να τους παρέχει η γραμματεία, καθώς και άλλες λειτουργίες. Η γραμματεία από την πλευρά της, θα μπορεί να κάνει εγγραφή σε νέους φοιτητές, να καταγράφει τους βαθμούς ενός μαθήματος, να επεξεργάζεται τις δηλώσεις των μαθητών και πολλά άλλα.

**Λέξεις κλειδιά:** Φοιτητολόγιο, Progressive Web Apps, μοντέρνες τεχνολογίες

# Abstract

The purpose of this thesis is to create a Student's Information System for the University's departments, based on the Progressive Web Apps technologies. With the help of these technologies, I'm aiming to create a platform that is modern, innovative and functional for the students, as it should be nowadays.

The application was created with the latest backend and frontend technologies with the ultimate purpose of an easy navigation and maximum functionality of the user. Moreover, since smartphones are widely used nowadays, especially by young people, I wanted to create a platform where students would have the same user experience on mobile or a computer, something not possible with the current Student's Information System.

By making the application a Progressive Web App, the users will have the capability to enjoy a user experience that is not provided by conventional Web Apps. Among those, is the ability to install the application by a web browser and not from a smartphone's app store, having push notifications, the partial usage of the app without an Internet connection, and even having faster response times.

The Student's Information System was made to facilitate both students and the department's secretariat. Using the platform, the students will be able to view their grades, enroll in classes every semester, request documents from the secretariat and many other services. The secretariat will be able to enroll first year students to the university, log grades, process students' requests and many more.



**Keywords:** Student's Information System, Progressive Web Apps, modern technologies

# ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

# ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

# ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

# ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

| | |
|---|---|
| PWA | Progressive Web App(s) |
| UI | User Interface |
| API | Application Programming Interface |
| REST | Representational state transfer |

# ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια τα έξυπνα τηλέφωνα έχουν εισχωρήσει για τα καλά στην καθημερινή ζωή των ανθρώπων και ειδικότερα των νέων ανθρώπων. Η χρήση τους αυξάνεται ολοένα και περισσότερο για πολλά πράγματα που παλαιότερα κάναμε με τον ηλεκτρονικό υπολογιστή. Πλέον πολλές ιστοσελίδες έχουν ως νούμερο 1 συσκευή επισκεψιμότητας τα κινητά τηλέφωνα.

Κάποιες ιστοσελίδες έχουν προσαρμοστεί σε αυτή τη νέα πραγματικότητα, φτιάχνοντας responsive για κινητά σελίδες, ενώ άλλες έχουν φτιάξει και applications για τα διάφορα λειτουργικά συστήματα κινητών. Παρόλα αυτά, πολλές πλατφόρμες δεν έχουν εναρμονιστεί ακόμα με αυτή τη νέα πραγματικότητα, και η πλοήγηση στις σελίδες τους είναι δύσκολη για τους χρήστες κινητών τηλεφώνων. Ένα απτό παράδειγμα είναι το Φοιτητολόγιο του πανεπιστημίου μας.

## Στόχος – Σκοπός

Έχοντας περάσει 4 χρόνια στη σχολή Πληροφορικής και Τηλεματικής, έζησα από πρώτο χέρι τις συνήθειες του μέσου φοιτητή και αυτό που παρατήρησα ήταν ότι οι περισσότεροι από εμάς χρησιμοποιούσαμε το κινητό μας για τα πάντα που θέλαμε να κάνουμε, και σε σχέση με τη φοίτησή μας στο Πανεπιστήμιο αλλά και στην κανονική μας ζωή. Η περίοδος μετά από μία εξεταστική, είναι μια περίοδος που πολλοί φοιτητές τσεκάρουν συνέχεια το φοιτητολόγιο για να δουν τους βαθμούς τους. Αυτό που παρατήρησα ήταν ότι πολλοί από εμάς είχαν πρόβλημα με την τωρινή εφαρμογή του φοιτητολογίου όταν ήμασταν στη σχολή και δεν είχαμε πρόσβαση στον υπολογιστή μας.

Στόχος, λοιπόν, της παρούσας εργασίας είναι με τη χρήση των τεχνολογιών Progressive Web Apps (PWA), να υλοποιηθεί μια καινούρια πλατφόρμα για το φοιτητολόγιο, η οποία να είναι mobile first, δηλαδή να έχει ως στόχο την απόλυτη εμπειρία χρήσης μέσω ενός κινητού τηλεφώνου, καθώς και να παρέχει περισσότερες δυνατότητες στον χρήστη. Πρώτη

μου σκέψη ήταν η δημιουργία μιας εφαρμογής για Android κινητά. Αυτό το σενάριο, όμως, εμφανίζει μερικά προβλήματα, καθότι δεν έχουν όλοι οι φοιτητές Android κινητό. Η επόμενη λύση θα ήταν να φτιαχτεί μια εφαρμογή με τεχνολογίες όπως η React Native, όπου δεν θα ήταν απαραίτητο να φτιαχτούν διαφορετικές εφαρμογές για κάθε λογισμικό. Ακόμα και αυτή η λύση, όμως, έχει ένα σημαντικό πρόβλημα. Το παρών Φοιτητολόγιο είναι αρκετά «απαρχαιωμένο» ακόμα και με πρόσβαση από τους υπολογιστές.

Σε αυτό το σημείο, μετά και από πρόταση του επιβλέπων καθηγητή μου, έμαθα για τις τεχνολογίες PWA. Μετά από μια μικρή έρευνα, αποφάσισα ότι η λύση αυτή είναι η καλύτερη και καλύπτει όλους τους στόχους μου, ενώ κι εγώ από την πλευρά μου θα μάθαινα κάποιες καινούριες τεχνολογίες, οι οποίες εκείνο τον καιρό (2017) φαινόταν να βρίσκονται σε μεγάλη ανάπτυξη, με πολλές εταιρίες κολοσσούς να τις υιοθετούν.

## Δομή εργασίας

Η πτυχιακή εργασία έχει χωριστεί σε 5 κεφάλαια. Στο 1ο Κεφάλαιο γίνεται η βιβλιογραφική επισκόπηση και η εισαγωγή στην έννοια των Progressive Web Apps, τα τεχνικά χαρακτηριστικά τους, στην ιστορία και στις ιστορίες επιτυχίας, αλλά και σε δύο εργαλεία που βοηθάνε στην ανάπτυξή τους. Στο 2ο Κεφάλαιο καλύπτεται η τεχνική υλοποίηση της εφαρμογής, οι γλώσσες προγραμματισμού και τεχνολογίες που χρησιμοποιήθηκαν. Επίσης περιγράφεται το REST API της εφαρμογής και ο σχεδιασμός της Βάσης Δεδομένων.

Το 3ο Κεφάλαιο είναι αφιερωμένο στο UI της εφαρμογής και τις δυνατότητες που προσφέρει, κυρίως, στους φοιτητές αλλά και στη γραμματεία και γενικά πώς φαίνεται η εφαρμογή στους χρήστες. Το 4ο Κεφάλαιο καλύπτει τις δυνατότητες που προσφέρει η εφαρμογή ως PWA, τον πιο σημαντικό στόχο της εφαρμογής, ενώ στο 5ο και τελευταίο Κεφάλαιο αναφέρονται τα συμπεράσματα που βγήκαν κατά την εκπόνηση της πτυχιακής, η σύγκριση με το τωρινό Φοιτητολόγιο, καθώς και μελλοντικές βελτιώσεις. Τέλος, ακολουθεί η βιβλιογραφία που χρησιμοποιήθηκε.

# ΚΕΦΑΛΑΙΟ 1
# PROGRESSIVE WEB APPS

## 1.1 Είδη Διαδικτυακών Εφαρμογών

Στο διαδίκτυο 2 είναι τα είδη των εφαρμογών που επικρατούν [20]:

- Κλασικές Διαδικτυακές Εφαρμογές (web apps)
- Εγγενείς εφαρμογές (Native Apps)

Οι κλασικές εφαρμογές είναι οι απλές και γνωστές εφαρμογές που υπήρχαν πάντα και, συνήθως, λειτουργούσαν για κινητά από την αρχή. Με τον ερχομό των έξυπνων τηλεφώνων, ήρθαν και οι Native Apps οι οποίες είναι φτιαγμένες ειδικά για συγκεκριμένες φορητές συσκευές και διανέμονται συνήθως μέσω κάποιου marketplace του εκάστοτε λειτουργικού συστήματος, με πιο διαδεδομένα το App Store του iOS και το Google Play του Android.

Τα 2 αυτά είδη έχουν αρκετές διαφορές μεταξύ των οποίων:

- Οι Native Apps είναι γραμμένες σε διαφορετικές γλώσσες προγραμματισμού, ανάλογα το λειτουργικό σύστημα, ενώ οι κλασικές εφαρμογές είναι όλες γραμμένες με τις κλασικές τεχνολογίες διαδικτύου
- Οι Native Apps μπορούν να αξιοποιήσουν εγγενείς λειτουργίες του κινητού όπως η κάμερα, το GPS, το επιταχυνσιόμετρο και άλλα, ενώ οι κλασικές εφαρμογές είναι περισσότερο περιορισμένες
- Οι Native Apps είναι, συνήθως, πιο γρήγορες, πιο φιλικές προς τον χρήστη, και γενικά προσφέρουν μια καλύτερη πλοήγηση σε σχέση με τις αντίστοιχες κλασικές εφαρμογές
- Τις Native Apps ο χρήστης πρέπει να τις κατεβάσει από κάποιο marketplace και να κάνει τις αναβαθμίσεις χειροκίνητα, σε αντίθεση με τις κλασικές οι οποίες είναι διαθέσιμες από οποιονδήποτε browser και οι αναβαθμίσεις γίνονται κατευθείαν στον server της εφαρμογής

Γενικά, και τα 2 είδη έχουν τα θετικά τους και τα αρνητικά, τόσο για τους χρήστες όσο και τις εταιρίες που καλούνται να τις αναπτύξουν. Αυτό που έχει επικρατήσει τα τελευταία χρόνια είναι να συνυπάρχουν και οι 2 και ο χρήστης να καλείται να χρησιμοποιήσει όποια του αρέσει και τον βολεύει.

## 1.2 Τι είναι Progressive Web Apps

Progressive Web App είναι ένα είδος εφαρμογής που διανέμεται στο διαδίκτυο και είναι υλοποιημένη με τις συνηθισμένες τεχνολογίες όπως η HTML, CSS και JavaScript. Σκοπός τους είναι να λειτουργούν σε οποιαδήποτε πλατφόρμα μέσω ενός browser, είτε αυτή είναι σταθερός υπολογιστής ή κινητό τηλέφωνο, ενώ στοχεύουν να περιορίσουν το κενό μεταξύ των κλασικών εφαρμογών και των Native Apps [15].

Οι PWA μπορούν να χαρακτηριστούν και ως «παιδιά» των web apps και των native apps, παίρνοντας από τα θετικά και από τα δύο, σε μια προσπάθεια να προσφέρουν μια πιο ολοκληρωμένη εμπειρία.

**Σύντομη διαδρομή στην ιστορία**

Το 2007, κατά τη διάρκεια της παρουσίασης του iPhone, ο Steve Jobs οραματίστηκε το μέλλον των εφαρμογών για κινητά που θα είχαν τα εξής χαρακτηριστικά [15]:

- Θα είναι διαθέσιμες στους browser, χωρίς να χρειάζεται να εγκαταστήσεις κάτι
- Θα ενσωματώνονται με λειτουργίες των κινητών όπως οι ειδοποιήσεις push (push notifications)
- Άμεση διανομή, απλώς «ανέβασέ» τες στο server σου
- Πολύ εύκολα αναβαθμίσιμες, απλώς άλλαξε τον κώδικα στον server σου

Το 2015, όμως, η Google έδωσε μια «επίσημη» ονομασία σε αυτό το όραμα και έβγαλε τις προδιαγραφές για την υλοποίησή τους.

## 1.3 Χαρακτηριστικά των PWA

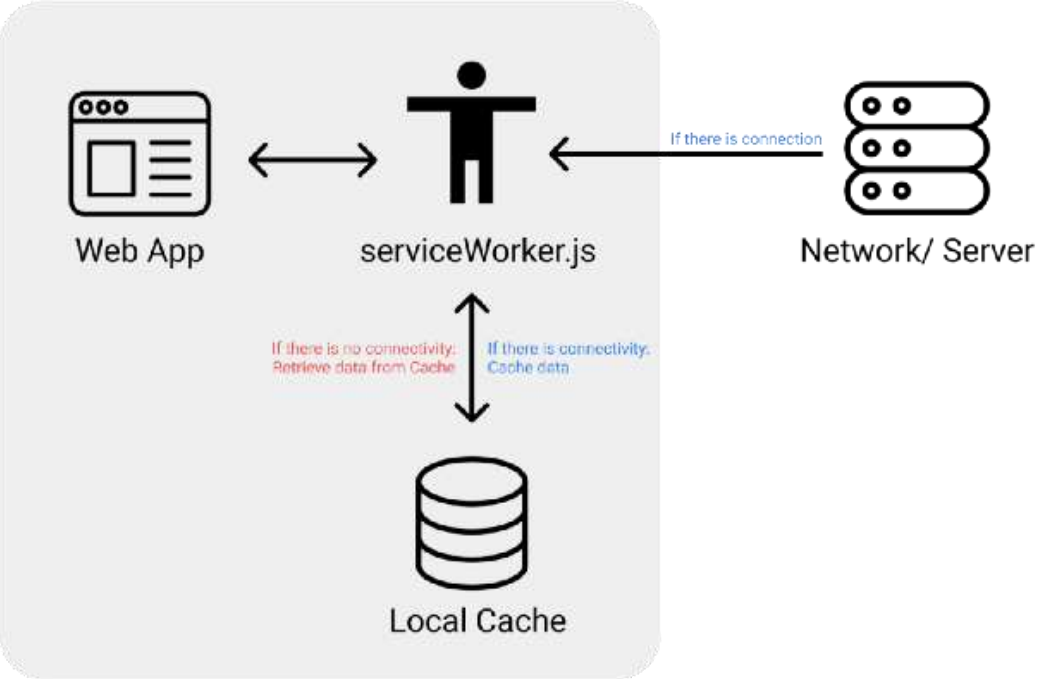
Τα κύρια χαρακτηριστικά των Progressive Web Apps, που τις κάνει να διαφέρουν από τα άλλα είδη διαδικτυακών εφαρμογών είναι τα παρακάτω [15] [25] [26]:

1. Προοδευτικές (progressive): δουλεύουν για κάθε χρήστη, ανεξαρτήτως ποιον browser χρησιμοποιούν
2. Responsive: Ανταποκρίνονται το ίδιο καλά σε όλα τα μεγέθη οθόνης, είτε σε κινητό, σε τάμπλετ, λάπτοπ ή σταθερό υπολογιστή
3. Γρηγορότερες μετά το πρώτο φόρτωμα: αφού έχει φορτωθεί η πρώτη σελίδα τα περιεχόμενα και τα στοιχεία της σελίδας δεν χρειάζεται να κατεβούν από την αρχή
4. Ανεξάρτητες από τη σύνδεση στο διαδίκτυο: όπως θα δούμε και πιο αναλυτικά αργότερα, οι service workers των PWA προσφέρουν τη χρήση της εφαρμογής χωρίς τη σύνδεση στο διαδίκτυο, ή με κακή πρόσβαση σε αυτό
5. Σαν Native App: η πλοήγηση και οι αλληλεπιδράσεις της εφαρμογής μοιάζουν πολύ με αυτές των Native Apps που βρίσκεις στο εκάστοτε marketplace της συσκευής
6. Ασφαλείς: λόγω της απαίτησής του να «σερβίρονται» μέσω του πρωτοκόλλου HTTPS, διασφαλίζεται η μέγιστη δυνατή ασφάλεια
7. Ανιχνεύσιμες: λόγω του ότι αναγνωρίζονται ως «εφαρμογές» στο manifest.json, αλλά και της εγγραφής των service workers, είναι εύκολα ανιχνεύσιμες από τις διάφορες μηχανές αναζήτησης
8. Μπορούν να εγκατασταθούν: προσφέρουν τη δυνατότητα εγκατάστασης της εφαρμογής, βάζοντας και το εικονίδιο της εφαρμογής στην αρχική οθόνη του χρήστη, χωρίς να χρειάζεται να τις εγκαταστήσεις από κάποιο marketplace

**Τεχνικά Χαρακτηριστικά**

Για να μπορεί μια εφαρμογή να χαρακτηριστεί ως PWA, πρέπει να πληροί κάποιες προϋποθέσεις, που αυτές είναι τεχνικά και οι διαφορές τους με τις κανονικές διαδικτυακές εφαρμογές [15] [25] [26]:.

## 1. Service Worker

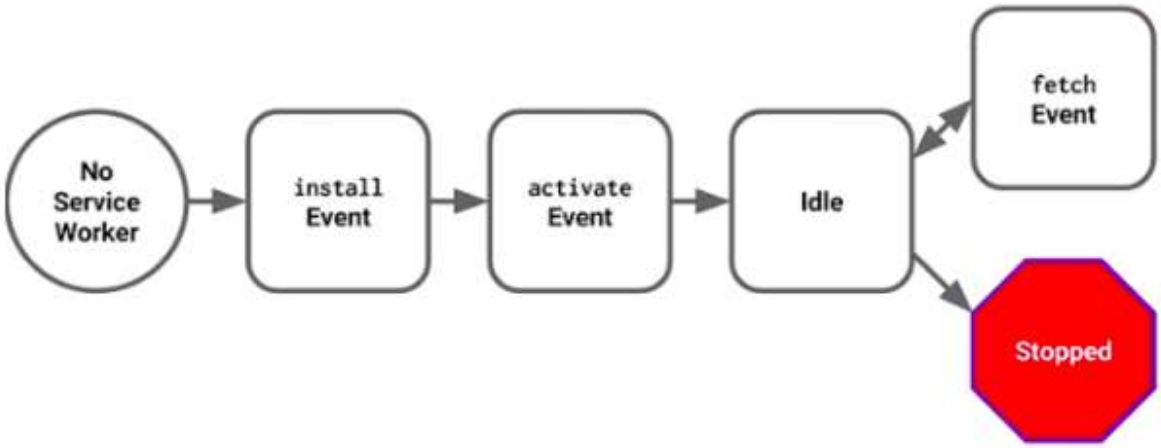



**Σχήμα 1:** Απεικόνιση της λειτουργίας των service workers

Οι service workers είναι ένας τύπος web worker, και είναι η «καρδιά» των Progressive Web Apps, αφού μέσω αυτών προσφέρονται πολλά ιδιαίτερα χαρακτηριστικά που είναι διαθέσιμα μόνο στις PWA. Στα τεχνικά, δεν είναι τίποτα παραπάνω από ένα αρχείο JavaScript, που τρέχει ξεχωριστά από το βασικό νήμα του browser και μπορεί να ανακόπτει network requests, να «κασάρει» πόρους και να παραδίδει push notifications. Στο Σχήμα 1 απεικονίζονται οι λειτουργίες των service workers.

Ένα ωραίο χαρακτηριστικό των service workers είναι ότι μπορούν να λάβουν ένα μήνυμα από τον server ακόμα και όταν η εφαρμογή δεν είναι ενεργή και αυτό τους επιτρέπει να στέλνουν ειδοποιήσεις στον χρήστη ακόμα και όταν η εφαρμογή δεν είναι ανοιχτή στον browser.

**Κύκλος ζωής (lifecycle)**




**Σχήμα 2:** Ο κύκλος ζωής ενός service worker

Στο Σχήμα 2 φαίνεται ο κύκλος ζωής των service workers Ο κύκλος ζωής των service workers έχει 3 βασικά στάδια [27]:

**Εγγραφή**
Για να μπορέσει να εγκατασταθεί ο service worker, θα πρέπει να τον εγγράψουμε στον κώδικά μας. Με αυτόν τον τρόπο, ο browser θα ξέρει πού βρίσκεται ο service worker και θα μπορέσει να αρχίσει την εγκατάσταση στο μπαγκράουντ.

**Εγκατάσταση**
Όταν ολοκληρωθεί η εγγραφή, τότε μπορεί να αρχίσει η εγκατάσταση. Αυτό θα γίνει αν ο browser θεωρήσει ότι αυτός είναι ένας καινούριος service worker, που είτε δεν είναι εγγεγραμμένος είτε έχει διαφορές με τον ήδη εγγεγραμμένο. Η εγκατάσταση θα πυροδοτήσει το **install** event, όπως βλέπουμε στην Εικόνα 1, μέσα στο οποίο μπορούμε να κάνουμε διάφορες λειτουργίες, όπως να «κασάρουμε» μέρη και στοιχεία της εφαρμογής ώστε αυτά να είναι άμεσα διαθέσιμα στον χρήστη την επόμενη επίσκεψή του.

```
// Listen for install event, set callback
self.addEventListener('install', function(event) {
    // Perform some task
});
```

**Εικόνα 1:** Κώδικας εγκατάστασης του service worker

**Ενεργοποίηση**
Όταν ολοκληρωθεί η εγκατάσταση, φτάνουμε στο τελευταίο στάδιο, αυτό άλλες ενεργοποίησης. Όταν ο service worker ενεργοποιηθεί, θα μπορεί να ελέγχει άλλες σελίδες που βρίσκονται στην αρμοδιότητά του και να ακούει events από άλλες σελίδες. Αυτό το σημείο είναι ένα καλό μέρος να «καθαριστούν» παλιά αποθηκευμένα στοιχεία που τώρα είναι ξεπερασμένα. Αυτό μπορεί να συμβεί κατά την πυροδότηση του **activate** event, που ο κώδικας του φαίνεται στην Εικόνα 2.

```
self.addEventListener('activate', function(event) {
  // Perform some task
});
```

**Εικόνα 2:** Κώδικας ενεργοποίησης του service worker

2. **Το αρχείο manifest**

   Το αρχείο manifest είναι η 2η απαραίτητη προϋπόθεση για κάθε PWA. Είναι ένα αρχείο JSON που περιέχει πληροφορίες για την εφαρμογή, όπως:

   - Το όνομα της εφαρμογής
   - Τα εικονίδια της εφαρμογής σε διάφορες αναλύσεις
   - Το URL που θα ανοίγει η εφαρμογή
   - Ο προεπιλεγμένος προσανατολισμός
   - Διάφορες ρυθμίσεις
   - Την επιλογή για λειτουργία προβολής, πχ. Πλήρης οθόνη

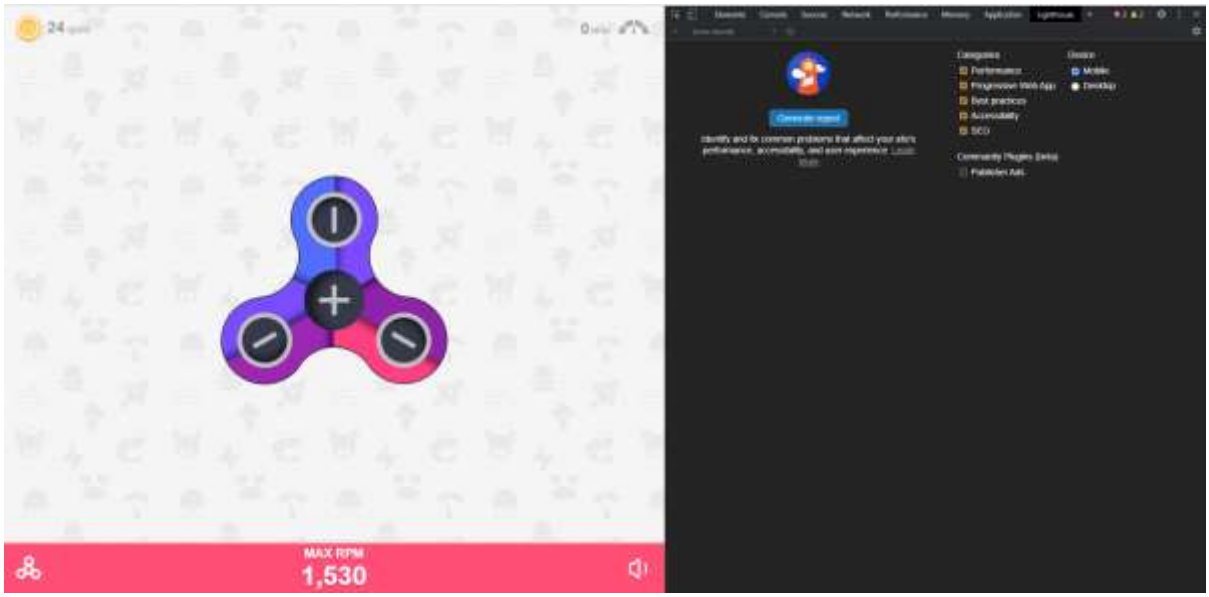
3. **HTTPS**

   Η 3η προϋπόθεση είναι ότι η εφαρμογή θα πρέπει να «σερβίρεται» μέσω του ασφαλούς πρωτόκολλου HTTPS. Αυτό συμβαίνει επειδή οι service workers έχουν τη δυνατότητα να ανακόπτουν τα network requests και να τροποποιούν τις απαντήσεις, για αυτό είναι ανάγκη να υπάρχει η μέγιστη ασφάλεια.

## 1.4 Lighthouse

Ένα σημαντικό εργαλείο που βοηθάει αρκετά τους developers, και όχι μόνο, κατά τη διάρκεια υλοποίησης μιας Progressive Web App, είναι το Lighthouse [14] από τη Google. Είναι ένα εργαλείο που βοηθάει στη βελτίωση της ποιότητας των διαδικτυακών εφαρμογών και σελίδων. Το Lighthouse προσφέρει μια σειρά ελέγχων για την ποιότητα, προσβασιμότητα, πληρότητα προϋποθέσεων PWA, SEO, ενδεδειγμένες πρακτικές και άλλα.

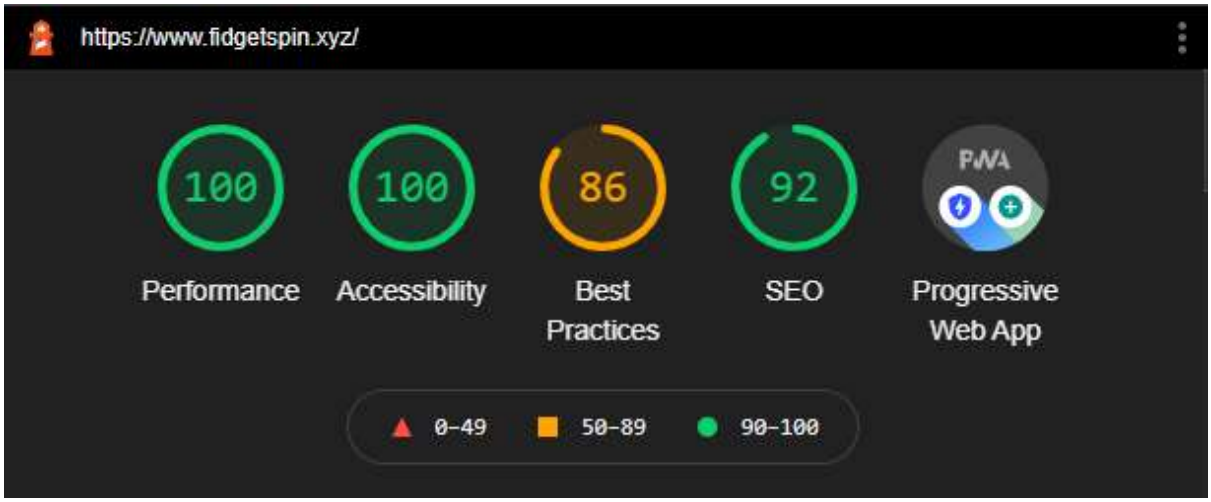
Το Lighthouse μπορεί να το τρέξει ο καθένας για οποιαδήποτε σελίδα στο Ίντερνετ μέσω του Chrome Dev Tools. Ας πάρουμε ως παράδειγμα μια γνωστή PWA, που βρίσκεται στο URL https://www.fidgetspin.xyz/, και ας τρέξουμε το lighthouse για να δούμε τα αποτελέσματα.




**Εικόνα 3:** Το εργαλείο Lighthouse στα Chrome Dev Tools

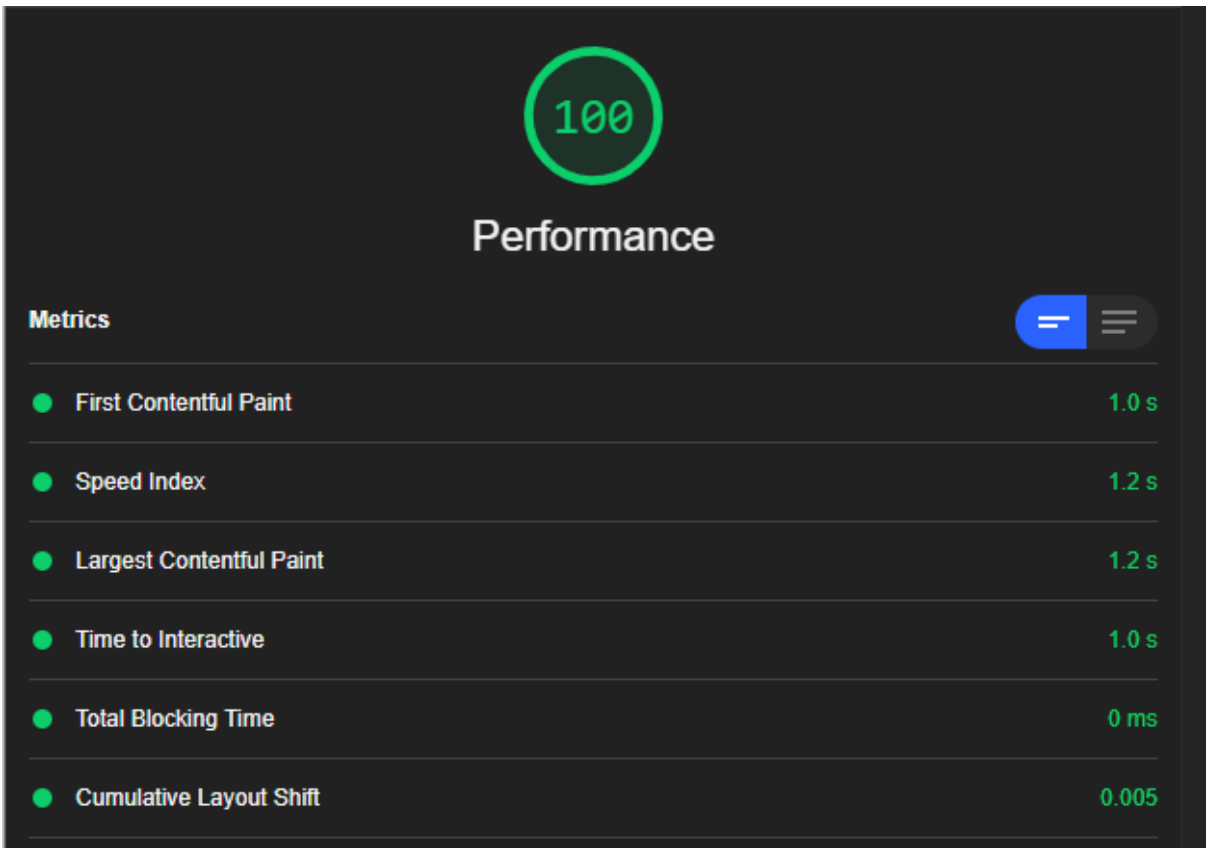
Όπως μπορούμε να δούμε στην Εικόνα 3, υπάρχει η επιλογή **Lighthouse** στο Chrome Dev Tools, όπου μπορούμε να δούμε τους ελέγχους που μπορούμε να τρέξουμε για μία σελίδα.

Παρακάτω είναι τα αποτελέσματα του ελέγχου:

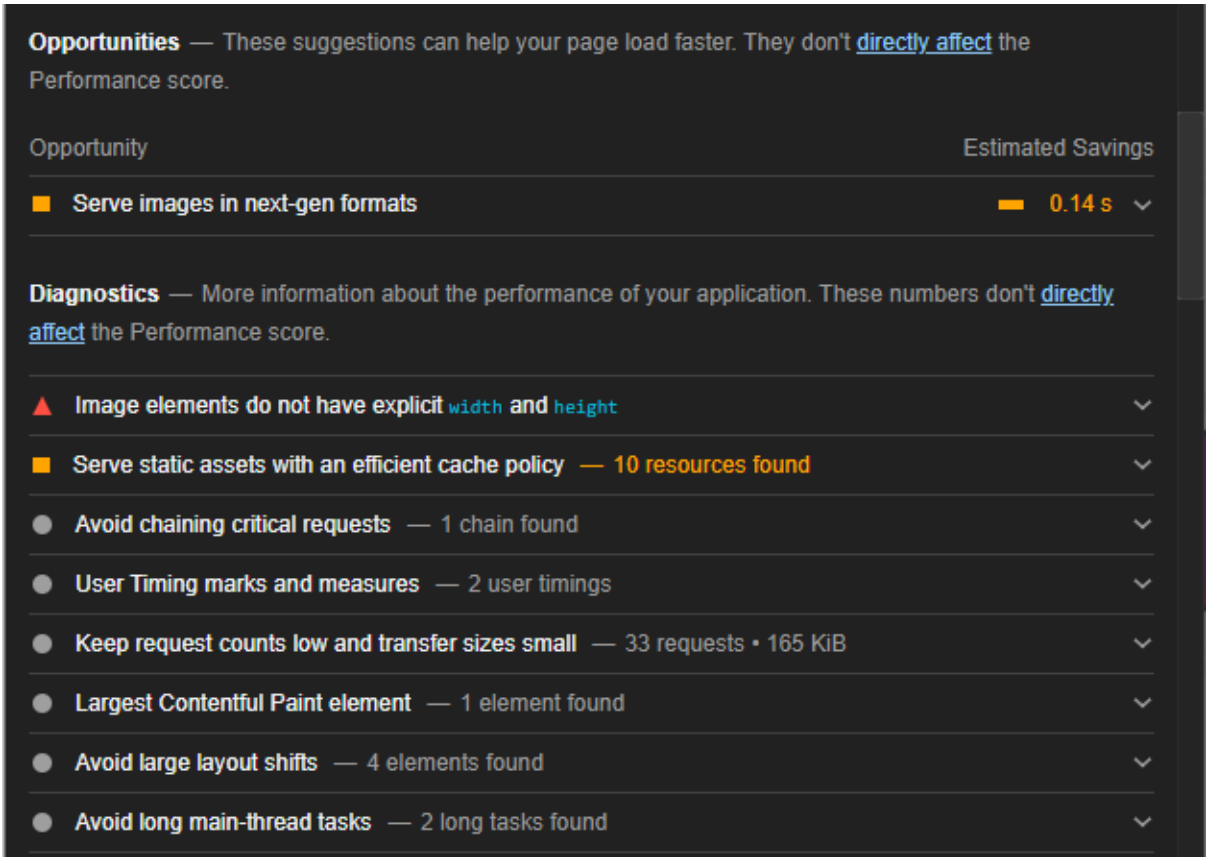



**Εικόνα 4:** Τα συνολικά αποτελέσματα του ελέγχου του Lighthouse

Στην Εικόνα 4, βλέπουμε τις βασικές κατηγορίες που τεστάρει το Lighthouse και τις αντίστοιχες βαθμολογίες και όπως βλέπουμε έχει αναγνωρίσει την εφαρμογή ως PWA.
Ακολουθούν αναλυτικά οι λεπτομέρειες για κάθε μία από τις λειτουργίες και τα αποτελέσματά τους στις Εικόνες 5, 7, 8 και 9.

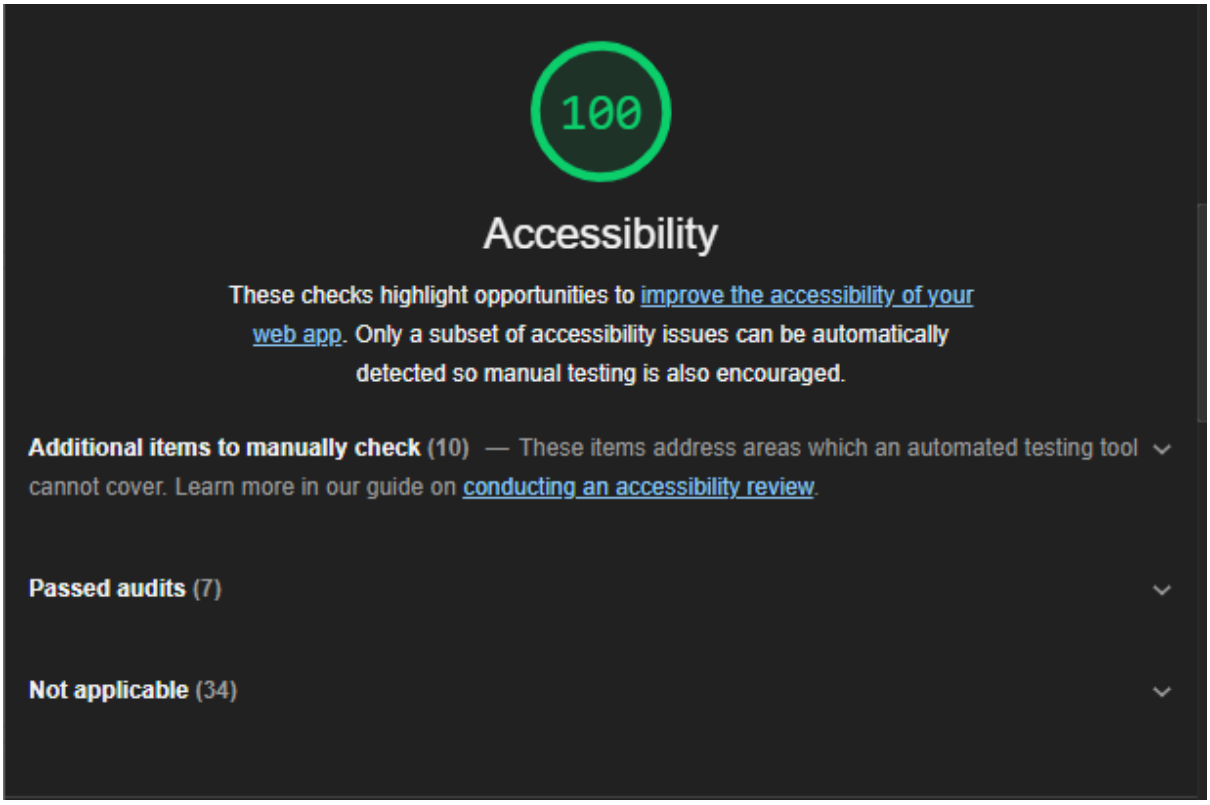



**Εικόνα 5:** Τα αποτελέσματα για το Performance της εφαρμογής

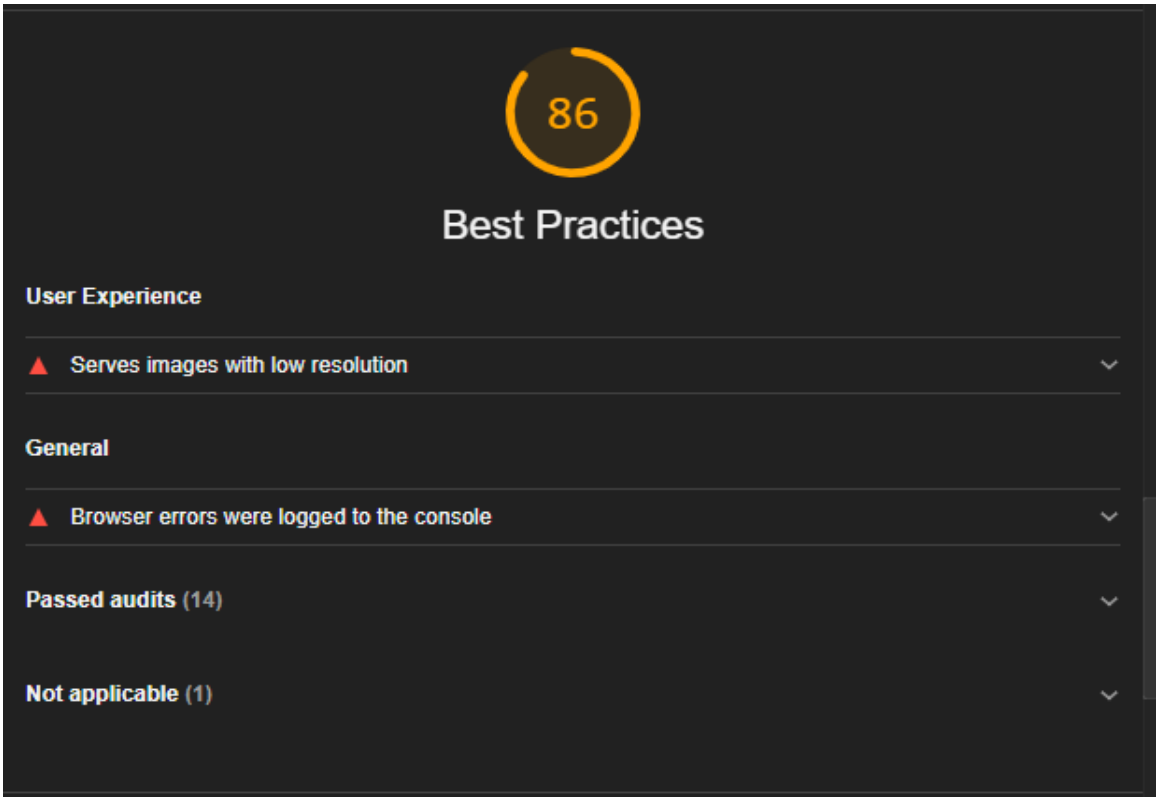



**Εικόνα 6:** Οι προτάσεις του Lighthouse για βελτίωση της απόδοσης της εφαρμογής

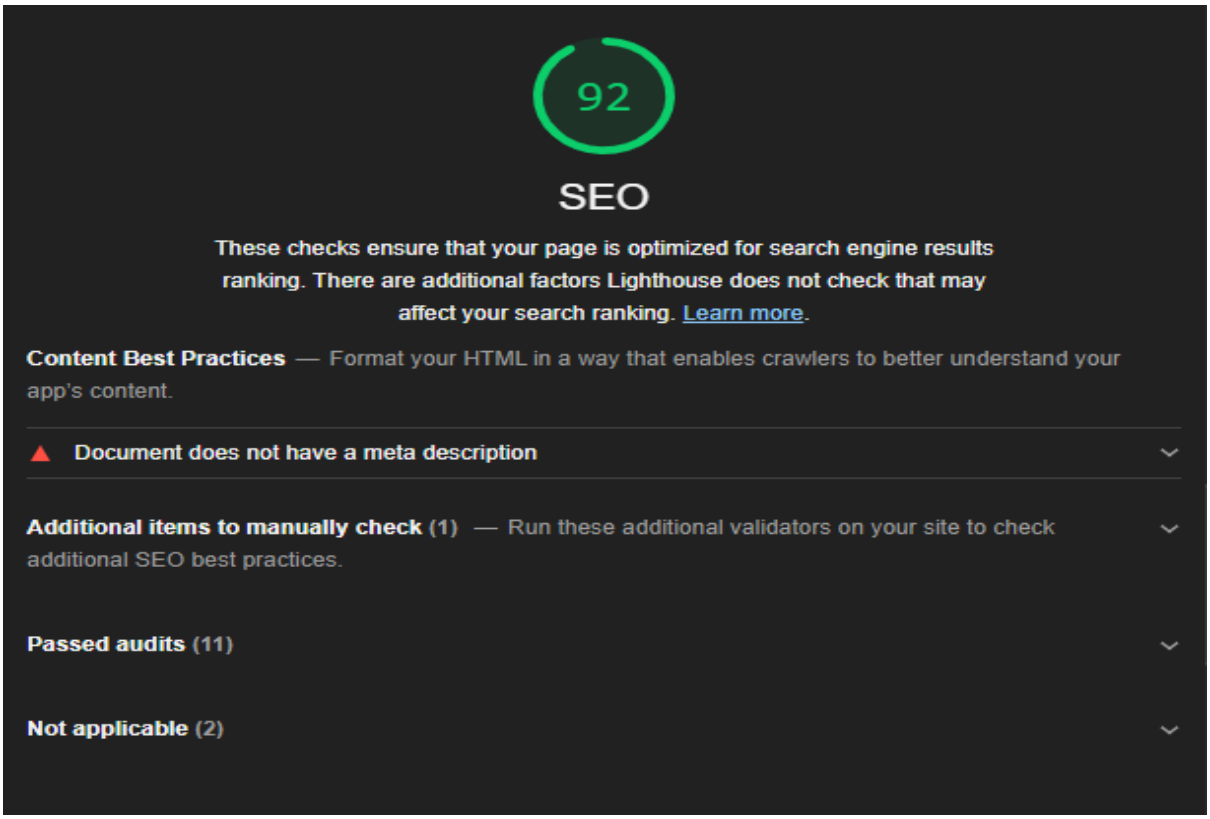
Στην Εικόνα 6 βλέπουμε μερικούς ελέγχους που δεν πέρασαν από το τεστ τους Lighthouse και θα ήταν καλό να αποφεύγονται.



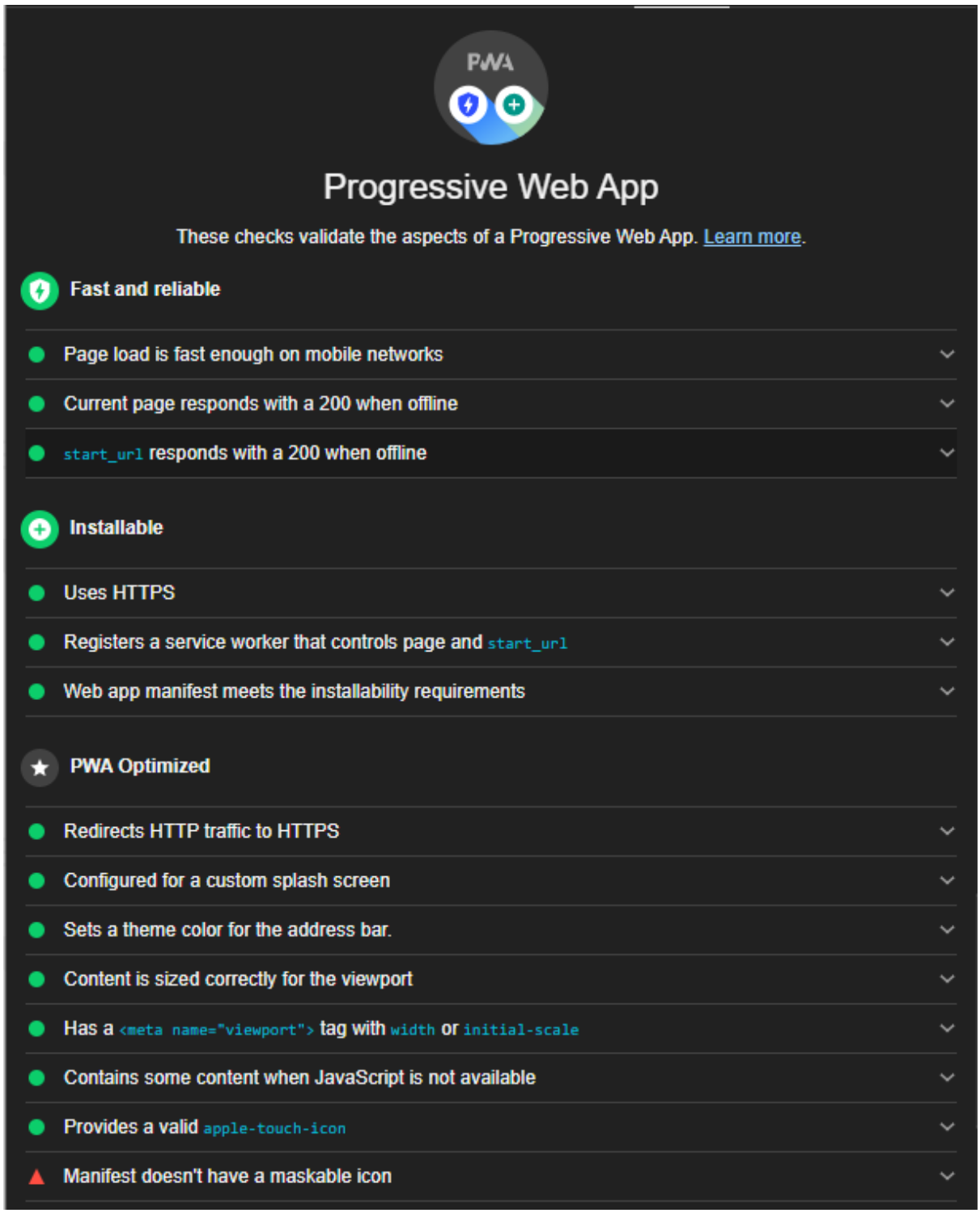

**Εικόνα 7:** Ο έλεγχος του Lighthouse για Accessibility




**Εικόνα 8:** Ο έλεγχος του Lighthouse για Best Practices




**Εικόνα 9:** Ο έλεγχος του Lighthouse για SEO




**Εικόνα 10:** Έλεγχοι πληρότητας προδιαγραφών PWA

Η Εικόνα 10 είναι και αυτή που μας ενδιαφέρει περισσότερο, αφού αφορά τους ελέγχους ως προς την πληρότητα των προδιαγραφών για να είναι PWA. Παρατηρούμε ότι ελέγχει για την ταχύτητα της εφαρμογής, για να μπορεί να φορτώσει γρήγορα ακόμα και σε αργά δίκτυα κινητής τηλεφωνίας, πιστοποιεί ότι χρησιμοποιείται το πρωτόκολλο HTTPS, ότι υπάρχει το αρχείο manifest, ότι ο service worker εγγράφεται, καθώς και άλλα λιγότερο σημαντικά πράγματα, που βρίσκονται στο manifest.

Συμπερασματικά, το Lighthouse είναι ένα πολύ καλό και χρήσιμο εργαλείο, που θα πρέπει να θεωρείται απαραίτητο κατά την υλοποίηση μιας Progressive Web App, καθώς μπορεί πολύ εύκολα να σε ενημερώσει αν πληροίς τις προϋποθέσεις των PWA και να σε βοηθήσει να τις λύσεις. Επίσης, είναι ένα πολύ καλό εργαλείο και κατά την ανάπτυξη κλασικών διαδικτυακών εφαρμογών, ενώ μπορεί να βοηθήσει και αυτούς που ασχολούνται με το SEO μιας σελίδας.

## 1.5 Google Workbox

Το Google Workbox [12] [13] είναι μία συλλογή από βιβλιοθήκες που βοηθάνε τους προγραμματιστές PWA εφαρμογών να γράψουν και να διαχειριστούν τους service workers, αλλά και στο «κασάρισμα» μέσω του CacheStorage API. Οι service workers και το CacheStorage API όταν χρησιμοποιούνται μαζί, δίνουν τη δυνατότητα να ελέγχεις πώς τα διάφορα αρχεία (HTML, CSS, JS, εικόνες, κτλ.) που χρησιμοποιείς στην εφαρμογή σου «έρχονται» από το δίκτυο ή την cache, ενώ ακόμα δίνει τη δυνατότητα να επιστρέφεις κασαρισμένο περιεχόμενο σε offline λειτουργία. Το Workbox μπορεί να χρησιμοποιηθεί μέσα στα αρχεία των service workers, μέσω JavaScript modules, εγκαθιστώντας μέσω του npm τα modules που χρειάζεσαι. Επειδή όμως τα JavaScript modules δεν δουλεύουν μέσα στα αρχεία των service workers, είναι απαραίτητη η χρήση ενός bundler που υποστηρίζει JavaScript modules για να κάνει μεταγλώττιση των service worker σε ένα αρχείο. Μερικοί από τους πιο δημοφιλής bundlers είναι ο webpack, ο Rollup και ο Parcel.

Μέσω αυτών των βιβλιοθηκών δίνεται η ευκολία στους προγραμματιστές να αποφύγουν να γράψουν από μόνοι τους μεγάλα αρχεία κώδικα για να υποστηρίξουν διάφορες δυνατότητες που προσφέρουν οι service workers. Το κασάρισμα των JS, JavaScript αρχείων αλλά και φωτογραφιών γίνονται πολύ εύκολα με τη βοήθεια του Workbox, ενώ δίνεται η δυνατότητα να κασαριστούν και τα Google Fonts. Επιπλέον της δυνατότητας να γίνεται caching όταν γίνονται requests στην εφαρμογή, το Workbox βοηθάει να γίνει precaching κατά την εγκατάσταση του service worker κι έτσι να υπάρχουν αρχεία στην cache όταν ο service worker πάρει τον έλεγχο της σελίδας. Μερικά πράγματα που καλό είναι να γίνουν precached είναι σημαντικά αρχεία CSS και JavaScript, η offline σελίδα και το URL της αρχικής σελίδας της εφαρμογής.

## 1.6 Οι δυνατότητες των PWA το 2021

Όπως έχει ήδη αναφερθεί, οι Progressive Web Apps γίνονται όλο και πιο δημοφιλής χρόνο με τον χρόνο, καταλαμβάνουν μεγαλύτερο μερίδιο στην αγορά και φυσικά, συνεχίζουν να εξελίσσονται συνέχεια και να προσθέτουν κι άλλες δυνατότητες, μέσω διαφόρων Web APIs, οι οποίες δεν είναι τόσο διαθέσιμές σε κανονικές Web εφαρμογές, παρά μόνο σε native εφαρμογές στο Android και στο iOS. Κάποιες από τις δυνατότητες των PWA αναφέρονται εκτενώς παρακάτω, ενώ στο WhatPwaCanDo.Today [7] υπάρχουν ακόμα κι άλλες δυνατότητες.

### 1.6.1 Web Authentication

Τα τελευταία χρόνια υπάρχει μια μεγάλη αύξηση στην ανάγκη για προστασία ενάντια σε κυβερνοεπιθέσεις αλλά και προστασία όταν κάποιος τρίτος αποκτά φυσική πρόσβαση σε κάποια συσκευή σου όπως το κινητό ή το λάπτοπ. Για αυτό και πια σχεδόν όλα τα νέα κινητά τηλέφωνα χρησιμοποιούν κάποιου είδους βιομετρικής λειτουργία για το άνοιγμα του κινητού, για το άνοιγμα και σύνδεση σε εφαρμογές με ευαίσθητες πληροφορίες όπως εφαρμογές ebanking, αλλά και για την έγκριση διάφορων λειτουργιών κάποιων εφαρμογών, όπως η έγκριση μιας συναλλαγής στο ebanking.

Η δυνατότητα λοιπόν πρόσβασης στις βιομετρικές λειτουργίες του κινητού για επαλήθευση κινήσεων είναι άκρως σημαντική και πλέον οι PWA μέσω του Web Authentication API [3] μπορούν να προσφέρουν στον χρήστη τη δυνατότητα να καταχωρήσει το δαχτυλικό του αποτύπωμα ή ένα εξωτερικό USB Security Key και μετέπειτα κάποιες κινήσεις να γίνονται με την επαλήθευση αυτών των στοιχείων.

### 1.6.2 Payment

Μια ακόμα δυνατότητα των PWA είναι η αυτόματη πληρωμή μέσω πιστωτικών/χρεωστικών καρτών ή Google Pay/Apple Pay, χωρίς να χρειάζεται η συμπλήρωση των στοιχείων της κάρτας για κάθε ξεχωριστή αγορά από το διαδίκτυο. Αυτό γίνεται χάρις στο Payment Request API [2] και με την αύξηση των διαδικτυακών αγορών τα τελευταία χρόνια, αυτό είναι μια ευχάριστη ευκολία.

### 1.6.3 Media Capture

Δύο πολύ σημαντικά και χρήσιμα κομμάτια hardware των κινητών συσκευών είναι η κάμερα και το μικρόφωνο. Τα δύο αυτά εργαλεία μπορούν να χρησιμοποιηθούν από ένα μεγάλο εύρος εφαρμογών και έχει πολλές δυνατότητες. Μερικές λειτουργίες που μπορεί να βοηθήσει αυτή η δυνατότητα των PWA είναι η εγγραφή και αποστολή βίντεο, φωτογραφιών ή ηχητικών μηνυμάτων, το σκανάρισμα διαφόρων εγγραφών ή κωδικών QR. Όλα αυτά επιτυγχάνονται μέσω της μεθόδου getUserMedia() [6] του API Media Devices [23].

### 1.6.4 Bluetooth και NFC

Δύο ακόμη κομμάτια hardware με τα οποία μπορούν να αλληλοεπιδράσουν οι Progressive Web Apps, είναι το Bluetooth και το NFC. Μέσω του Web NFC API [24], οι εφαρμογές μπορούν να αλληλοεπιδράσουν με διάφορα NFC tags που έχουν γίνει κάπως δημοφιλή τα τελευταία χρόνια και χρησιμοποιούνται σε διάφορους χώρους του σπιτιού ή στα αμάξια. Επίσης, μέσω του Web Bluetooth API [4], οι εφαρμογές μπορούν να «διαβάσουν» και να «γράψουν» τιμές σε συσκευές Bluetooth Low Energy (BLE), όπως μετρητές καρδιακών παλμών ή «έξυπνες» λάμπες φωτισμού.

### 1.6.5 File System και Web Share

Τέλος, δύο ακόμη δυνατότητες που έχουν ενδιαφέρον είναι αρχικά η πρόσβαση στα αρχεία του κινητού, που γίνεται με τη βοήθεια του File System Access API [1], και δίνει τη δυνατότητα στην εφαρμογή να «διαβάζει», να «γράφει» και να αποθηκεύει αρχεία. Αυτό ουσιαστικά δίνει τη δυνατότητα στις εφαρμογές να αναπτύξουν διάφορα features που να αλληλοεπιδρούν με τα αρχεία του χρήστη, όπως το ανέβασμα μιας φωτογραφίας και η επεξεργασίας της ή ακόμα και δυνατότητες ανεβάσματος και αποθήκευσης αρχείου στο cloud.

Η δεύτερη δυνατότητα, η οποία επιτυγχάνεται με τη βοήθεια του Web Share API [5], είναι ο διαμοιρασμός αρχείων, μηνυμάτων ή URL ανάμεσα σε συσκευές και χρήστες. Αυτό ανοίγει τη δυνατότητα για το διαμοιρασμό αρχείων ανάμεσα σε χρήστες σε διάφορες εφαρμογές που έχουν την δυνατότητα του διαμοιρασμού, όπως μέσω email, messaging εφαρμογών ή κοινωνικών δικτύων.

## 1.7 Ιστορίες επιτυχίας PWA

Οι Progressive Web Apps έχουν δει μια μεγάλη ανάπτυξη από τότε που ξεκίνησαν και όλο και περισσότερες μεγάλες εταιρίες αποφασίζουν να τις υιοθετήσουν, ενώ όσο περνάνε τα χρόνια βγαίνουν όλο και περισσότερα success stories. Μερικές από αυτές που αξίζει να αναφερθούν είναι οι παρακάτω:

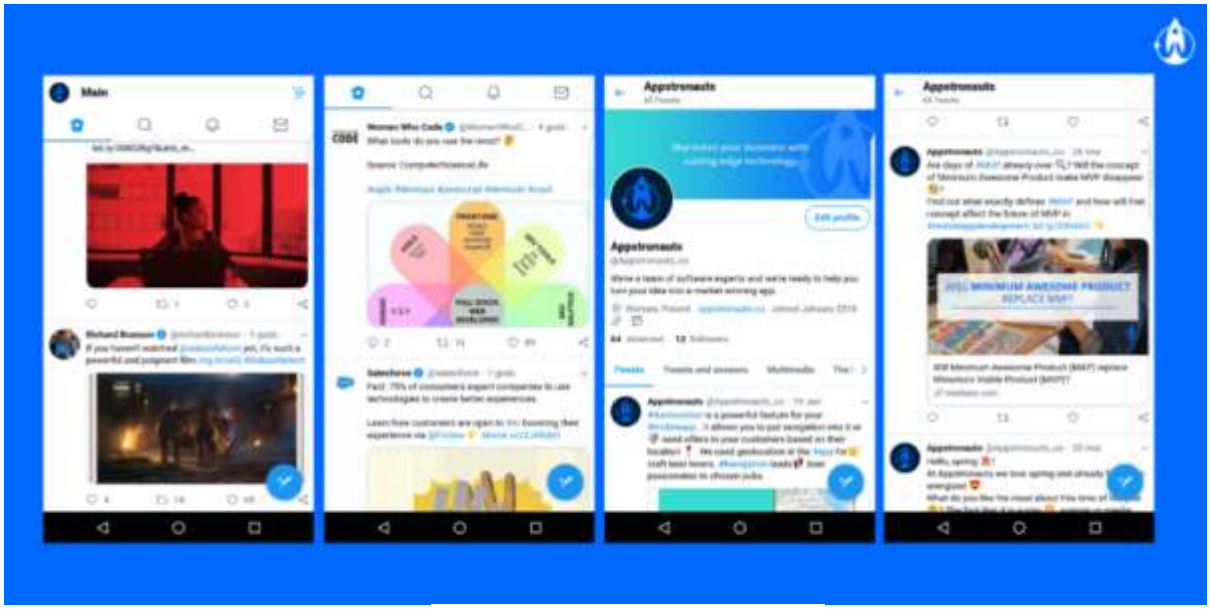
**Twitter**

Το Twitter έβγαλε σε κυκλοφορία την PWA εφαρμογή το 2017 και ίσως μπορεί να χαρακτηριστεί μέχρι και σήμερα η μεγαλύτερη εταιρία/εφαρμογή που στράφηκε προς τις τεχνολογίες PWA. Το 80% των χρηστών του Twitter ήταν από κινητά τηλέφωνα, οπότε ήθελαν να μειώσουν την κατανάλωση δεδομένων, ειδικά για τους χρήστες που δεν είχαν πολύ καλή σύνδεση. Αυτό που κατάφερε το Twitter ήταν [21] [22] [28]:

- **Αύξησαν την ενασχόληση** των χρηστών λόγω των push notifications που στέλνονται ακόμα και όταν ο χρήστης δεν έχει ανοιχτό τον browser
- **Καλύτερη εμπειρία για το χρήστη**, αφού η PWA εφαρμογή είχε ακριβώς την ίδια εμπειρία χρήσης με τη Native App, φόρτωνε πολύ γρήγορα την πρώτη φορά σε συνδέσεις μέσω 2G/3G, ενώ οι μεταγενέστερες φορτώσεις ήταν πρακτικά στιγμιαίες
- **Πιο ελαφριά και πιο προσβάσιμη** από την Native εφαρμογή, αφού χρησιμοποιεί λιγότερα δεδομένα, βασιζόμενη σε δεδομένα που έχουν γίνει cached, ενώ και οι εικόνες βελτιστοποιούνται για να μειώσουν την κατανάλωση δεδομένων
- **Αύξησε του χρήστες**, αφού η εφαρμογή μπορεί να χρησιμοποιηθεί από χρήστες με κακές συνδέσεις, αλλά και λιγότερο δυνατά κινητά.

Μερικά από τα σημαντικά νούμερα της επιτυχίας ήταν ότι η PWA εφαρμογή είχε μέγεθος 600KB σε αντίθεση με την Android εφαρμογή που είχε 23.5MB και την εφαρμογή για iOS που ήταν 116.5MB, μια πάρα πολύ σημαντική μείωση. Το Twitter επίσης παρατήρησε αύξηση της τάξης του 65% στις σελίδες που επισκέπτεται ο χρήστης σε μια συνεδρία του, 75% αύξηση στα tweets, 25% μείωση στο «ποσοστό αναπήδηση», δηλαδή των χρηστών που δεν επισκέπτονται

ξανά την εφαρμογή και μείωση 30% στο χρόνο φόρτωσης. Στην Εικόνα 11, παίρνουμε μια γεύση από την PWA του Twitter.

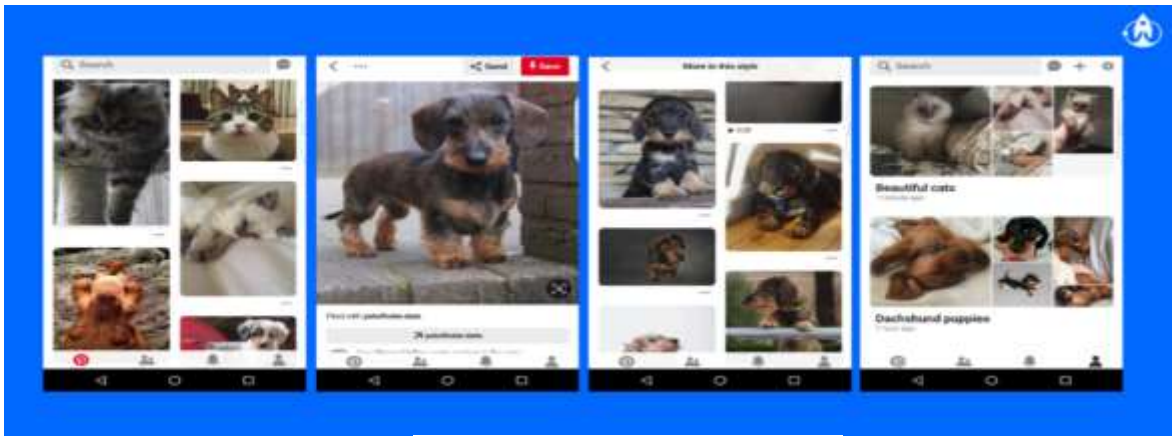


**Εικόνα 11:** Η PWA του Twitter

**Pinterest**

Το Pinterest αντιμετώπιζε και αυτό προβλήματα παρόμοια με το Twitter, αλλά πιο συγκεκριμένα με την εμπειρία χρήσης και ενασχόλησης των χρηστών από κινητά από τη web εφαρμογή. Το κακό UX και αργή φόρτωση είχε ως αποτέλεσμα οι χρήστες να μην συνδέονταν στο λογαριασμό τους, να μην εγγράφονταν και να μην κατέβαζαν τη Native εφαρμογή. Η διαδικασία μετατροπής της εφαρμογής σε PWA τους πήρε 3 μήνες και οδήγησες σε πολλές τεχνολογικές βελτιώσεις [21] [22].

Το Pinterest είδε τα ίδια αποτελέσματα με το Twitter, η φόρτωση της εφαρμογής έπεσε από τα 23 δευτερόλεπτα στα 5.6, το μέγεθος έγινε 160KB σε σύγκριση με 17MB της Android εφαρμογής και τα 56MB της IOS. Οι εγγραφές αυξήθηκαν κατά 843%, μέσα σε 6 μήνες 800 χιλιάδες χρήστες άνοιγαν την εφαρμογή από την αρχική τους οθόνη, ενώ είδαν και τρομερή αύξηση στους χρήστες σε αγορές που δεν είχαν παρουσία όπως η Ινδία και η Βραζιλία [21] [22]. Στην Εικόνα 12 βλέπουμε πώς μοιάζει η PWA του Pinterest.



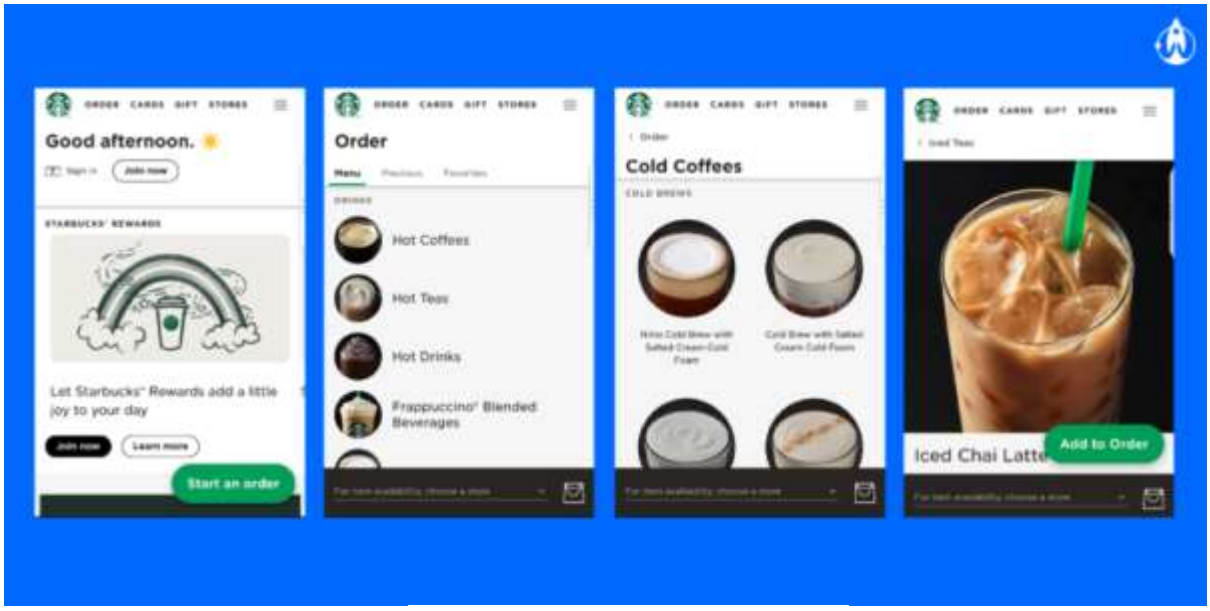
**Εικόνα 12:** Το PWA του Pinterest

**Starbucks**

Άλλη μία ιστορία επιτυχίας είναι τα Starbucks, η μεγαλύτερη αλυσίδα καφετεριών στην Αμερική αλλά και στον κόσμο. Στόχος τους ήταν να «φτάσουν» και τους χρήστες με χειρότερες συνδέσεις στο διαδίκτυο, αλλά και να φτιάξουν μια εύκολη εμπειρία χρήσης και να μπορούν οι χρήστες να κοιτάζουν το μενού, να εξατομικεύουν τις παραγγελίες και τελικά να παραγγέλνουν, παρά την κακή σύνδεσή τους, αλλά και χρήση χωρίς πρόσβαση στο διαδίκτυο [21].

Η ευκολονόητη διαδικασία παραγγελίας και η αξιοπιστία της εφαρμογής οδήγησε στο διπλασιασμό των παραγγελιών από το website. Η εμπειρία χρήσης έγινε καλύτερη και για τους χρήστες κινητών αλλά και για αυτούς που χρησιμοποιούσαν την εφαρμογή από υπολογιστή, λόγω της αξιοπιστίας και της ταχύτητας [21].

Με τη χρήση των push notifications αύξησαν πολύ την ενασχόληση των χρηστών, αφού στέλναν ειδοποιήσεις για ειδικές προσφορές, ενώ οι ενεργοί ημερήσιοι και μηνιαίοι χρήστες διπλασιάστηκαν. Η εφαρμογή τώρα έχει μέγεθος 600KB σε σύγκριση με τα 20MB της Android εφαρμογής και τα 146MB της iOS [21] [28]. Στην Εικόνα 13 βλέπουμε το PWA των Starbucks.




**Εικόνα 13:** Το PWA των Starbucks

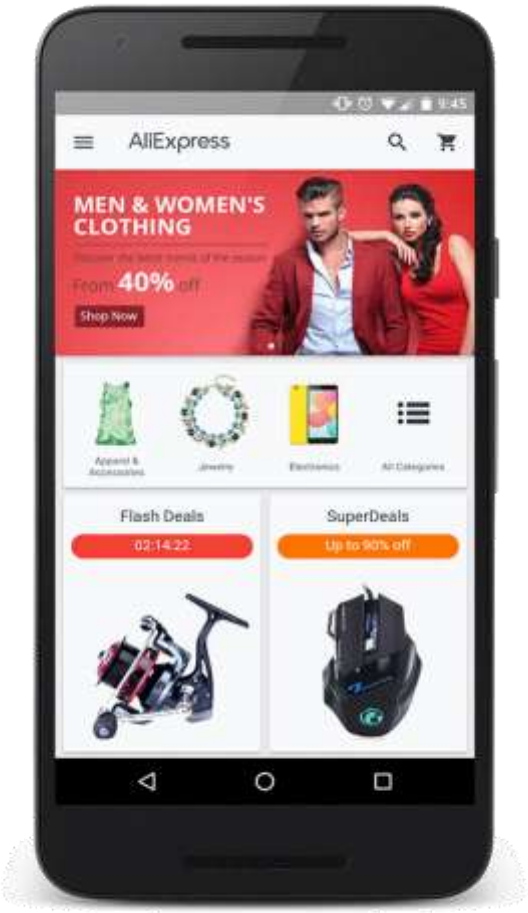
**AliExpress**

Ένα τελευταίο παράδειγμα επιτυχίας των PWA είναι και στον ταχέα αναπτυσσόμενο χώρο του ecommerce και του μεγαλύτερου site του χώρου, το Alibaba. Τα στοιχεία τους έδειχναν ότι ο περισσότερος κόσμος τους έβρισκε από κινητές συσκευές και για αυτό είναι απολύτως λογικό να θέλουν να έχουν σημαντική παρουσία στο χώρο των κινητών [22].

Με τη νέα τους PWA εφαρμογή είχαν στόχο να εντυπωσιάσουν τους νέους χρήστες, αλλά και τους χρήστες που επισκέπτονται συχνά το site, με μια εφαρμογή πολύ γρήγορη, πιο αξιόπιστη

και με καλύτερη εμπειρία χρήστης από αυτή που είχαν συνηθίσει. Αυτό είχε ως αποτέλεσμα να δουν αύξηση 70% στη μετατροπή (conversion) σε όλους τους browser, 14% αύξηση στους ενεργούς χρήστες στο iOS και 30% στο Android και τετραπλασιασμό στην αλληλεπίδραση του μηνύματος που βγαίνει στον browser για την προσθήκη της εφαρμογής στην αρχική οθόνη [22].

Ο διευθυντής της ομάδας κινητών τηλεφώνων, Zou Yu, είπε [28] ότι μια εφαρμογή που είναι φτιαγμένη πρώτα για κινητές συσκευές και για τους χρήστες της, που αξιοποιεί όλα τα θετικά των Native εφαρμογών, μαζί με την προσβασιμότητα και την ευελιξία χρήσης του διαδικτύου μέσω του κινητού, είναι το κλειδί για να διατηρήσουν το πλεονέκτημα και ο δρόμος για καλύτερες μετατροπές. Στην Εικόνα 14, βλέπουμε την αρχική σελίδα του AliExpress από την PWA εφαρμογή.




**Εικόνα 14:** Το PWA του AlieExpress

# ΚΕΦΑΛΑΙΟ 2

# ΤΕΧΝΙΚΗ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ

## 2.1 Τεχνολογίες και γλώσσες προγραμματισμού που χρησιμοποιήθηκαν

Για την υλοποίηση του frontend κομματιού της εφαρμογής επέλεξα να χρησιμοποιήσω AngularJS. Η επιλογή αυτή έγινε για διάφορους λόγους. Ο κυριότερος ήταν γιατί θεώρησα ότι το project ήταν αρκετά μεγάλο και πολύπλοκο ώστε να αξίζει η μετάβαση από την υλοποίηση με απλή JavaScript, καθώς θα μου έκανε τη ζωή πιο εύκολη και θα μπορούσα να χρησιμοποιήσω ήδη έτοιμες βασικές λειτουργίες υλοποίησης μια εφαρμογής, χωρίς να χρειαστεί να γράψω τον κώδικα από την αρχή. Την περίοδο καταπόνησης της πτυχιακής, τα βασικά και δημοφιλή frontend frameworks ήταν 3-4, μεταξύ αυτών και η AngularJS, αλλά επιλέχθηκε η AngularJS γιατί ήταν αρκετά χρόνια στον χώρο, η ανάπτυξή της είχε τελειώσει, ήταν αρκετά stable, ενώ υπάρχει μεγάλο community στο διαδίκτυο που μπορούσε να βοηθήσει έναν αρχάριο με tutorials ή ερωτήσεις.

Αξίζει να αναφερθεί ότι χρησιμοποιήθηκε και το CSS framework, Bootstrap, στο frontend, που παρόλο δεν θα το θεωρούσα απαραίτητο για την υλοποίηση της πτυχιακής, βοήθησε αρκετά να μειωθεί ο χρόνος ενασχόλησης με το styling της εφαρμογής αλλά και το responsiveness σε διάφορες συσκευές, αφού όλα αυτά καλύπτονται από το Bootstrap.

Για το backend, επιλέχθηκε η Node.js, η οποία ήταν μια τεχνολογία που είχε αρχίσει να γίνεται ιδιαίτερα δημοφιλές στον χώρο του προγραμματισμού με πολλές εταιρίες να την επιλέγουν για την υλοποίηση του backend τους. Η Node.JS προσφέρει αρκετά πλεονεκτήματα, τα οποία θα αναφερθούν και αργότερα, τα οποία την έκαναν πολύ δελεαστική επιλογή, ενώ θα ήταν και ένα πολύ χρήσιμο skill να γνωρίζω για την μετέπειτα καριέρα μου.

Ως λύση για την αποθήκευση των διάφορων δεδομένων της εφαρμογής, χρησιμοποιήθηκε η γνωστή σχεσιακή βάση δεδομένων PostgreSQL και λόγω της προηγούμενης ενασχόλησής μου με σχεσιακές βάσεις δεδομένων, οπότε δεν θα μου ήταν κάτι καινούργιο να σχεδιάσω και υλοποιήσω, αλλά και επειδή τα δεδομένα που θα χρειάζονταν αποθήκευση, με μια πρώτη ματιά, φαίνεται να ταίριαζαν σε μια υλοποίηση σχεσιακής βάσης δεδομένων, σε αντίθεση με τις μη σχεσιακές βάσεις, όπως η MongoDB, που χρησιμοποιείται πολύ μαζί τη Node.JS.

Τέλος, η εφαρμογή ανέβηκε στο Heroku, μια cloud πλατφόρμα φιλοξένησης εφαρμογών, που διαθέτει και δωρεάν πλάνο για πιο μικρές εφαρμογές χωρίς μεγάλη κίνηση.

Ακολουθούν μερικές πληροφορίες για τις τεχνολογίες που χρησιμοποιήθηκαν.

## AngularJS


**Εικόνα 15:** Το logo της AngularJS

Η AngularJS (το logo της βλέπουμε στην Εικόνα 15) είναι ένα framework ανοιχτού κώδικα βασισμένο στη JavaScript, που κυκλοφόρησε το 2010 και τώρα πια διατηρείται κυρίως από τη Google, αλλά και από μεμονωμένους προγραμματιστές ή άλλες μικρότερες εταιρείες. Δημιουργήθηκε για να αντιμετωπίσει τις δυσκολίες ανάπτυξης των Single Page Applications και στόχος της είναι να απλοποιήσει την ανάπτυξη και το τεστάρισμα τέτοιων εφαρμογών [16].

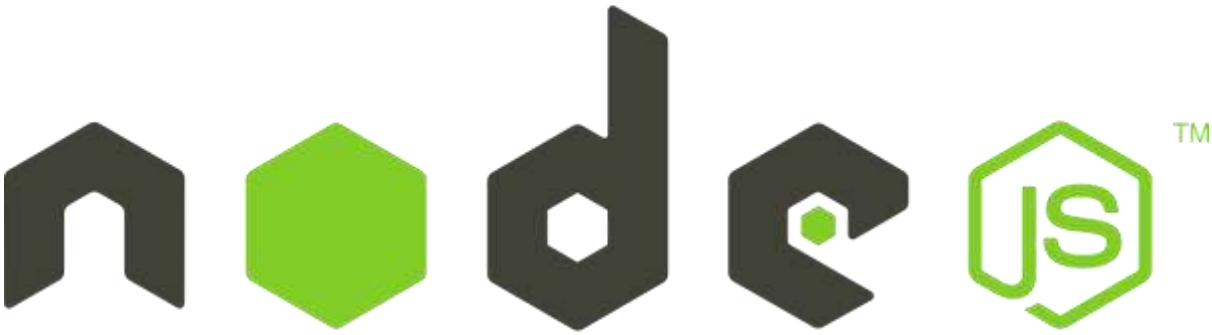
Η AngularJS σχεδιάστηκε με στόχους:

- Να διαχωρίσει τη χειραγώγηση του DOM από τη λογική της εφαρμογής
- Να διαχωρίσει την πλευρά του client και του server της εφαρμογής, ώστε η ανάπτυξη να μπορεί να συνεχίσει παράλληλα και τα δύο κομμάτια να μπορούν να χρησιμοποιηθούν ξανά
- Να προσφέρει οργάνωση καθ' όλη τη διάρκεια του project, από το σχεδιασμό της διεπαφής χρήστη, το γράψιμο της επιχειρησιακής λογικής και το τεστάρισμα

**Γιατί AngularJS;**

Όταν ξεκίνησε η εκπόνηση της εφαρμογής και ήμουν στο δίλημμα επιλογής ενός frontend framework, οι επιλογές ήταν 3-4 και παρόλο που η AngularJS δεν ήταν πια η πιο δημοφιλής επιλογή, ούτε η πιο καινούργια και μοντέρνα, ήταν μια καλή επιλογή γιατί ήταν αρκετά χρόνια ωριμασμένη στο χώρο, με πολλές εταιρείες παγκοσμίως να την έχουν χρησιμοποιήσει για την κατασκευή του frontend τους, ενώ διέθετε και ένα μεγάλο community από προγραμματιστές από πίσω, όπου μπορούσες να αντλήσεις πληροφορίες. Επίσης, ήταν μια καλή επιλογή για κάποιον αρχάριο στα frontend frameworks, αφού κάποιος με μόνη γνώση HTML, CSS και JavaScript, μπορούσε σχετικά εύκολα να ξεκινήσει ένα project σε AngularJS χωρίς ιδιαίτερο πρόβλημα.

**Node.js**



**Εικόνα 16:** Το logo της node.JS

Η Node.js (το logo της βλέπουμε στην Εικόνα 16) είναι μια πλατφόρμα ανάπτυξης λογισμικού, κυρίως για server, χτισμένη σε περιβάλλον JavaScript και τρέχει στη μηχανή V8. Κυκλοφόρησε το 2009, αφού ο δημιουργός της είχε ασκήσει πολλές φορές κριτική στον πιο δημοφιλή, τότε, web server, τον Apache HTTP Server, όσον αφορά το πώς μπορεί να χειριστεί τις πολλές (πάνω από 10.000) παράλληλες συνδέσεις. Στόχος είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών, ενώ σε αντίθεση με τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου [17].

Η Node.js έχει δει ραγδαία αύξηση στη χρήση της τα τελευταία χρόνια, με όλο και περισσότερες εταιρείες να τη χρησιμοποιούν για την υλοποίηση των backend υποδομών τους, κυρίως για APIs. Η δημοτικότητα της παρατηρείται κυρίως σε πιο καινούργιες εταιρείες τεχνολογίες, τις startups, αφού είναι ίσως η πιο γρήγορη λύση για την υλοποίηση ενός καλού API. Παρόλα αυτά, ακόμα και εταιρείες κολοσσοί όπως η Microsoft, το LinkedIn και το Netflix, μεταξύ άλλων, έχουν βρει τρόπο να την ενσωματώσουν στα συστήματά και της λειτουργίες του, ενώ ίσως το πιο σημαντικό πλεονέκτημα της Node.JS είναι ότι γράφεις παντού JavaScript, δηλαδή δεν είναι πάντα απαραίτητο να έχεις γνώσεις διαφορετικών γλωσσών προγραμματισμού για το frontend και το backend.

Εγώ την επέλεξα για την πτυχιακή μου γιατί πρόσφερε έναν πιο γρήγορο τρόπο να στήσω το απαιτούμενο API. χωρίς πολύ boilerplate και ανούσιες πολυπλοκότητες που έχουν άλλα περιβάλλοντα. Επίσης μου άρεσε πολύ η ιδέα να χρησιμοποιήσω παντού JavaScript, ενώ με την ταχεία άνοδο που είχε η Node.JS στον επαγγελματικό χώρο, ήταν ένα πολύ δυνατό χαρτί για την μετέπειτα επαγγελματική πορεία μου.

**Npm**

**Εικόνα 17:** Το logo του npm

Ένα εργαλείο που πάει πακέτο με την υλοποίηση εφαρμογών σε Node.JS, είναι το npm (το logo του βλέπουμε στην Εικόνα 17) ή ολογράφως Node Package Manager και είναι διαχειριστής πακέτων για το περιβάλλον της Node.JS. Το npm παρέχει μια online βάση δεδομένων όπου οι χρήστες μπορούν να μοιράζονται ή να κατεβάζουν δωρεάν αλλά και επί πληρωμή πακέτα ή βιβλιοθήκες (libraries) για την πιο εύκολη ανάπτυξη λογισμικού σε Node.JS. Μέχρι στιγμής υπάρχουν πάνω από 1.3 εκατομμύρια πακέτα, ενώ πρόσφατα αποκτήθηκε και έγινε θυγατρική του GitHub [29].

**PostgreSQL**


**Εικόνα 18:** Το logo της PostgreSQL

Η βάση δεδομένων που χρησιμοποίησα είναι η PostgreSQL (το logo της βλέπουμε στην Εικόνα 18) και είναι μια από τις πιο δημοφιλής Σχεσιακές Βάσεις Δεδομένων. Η επιλογή της δεν έγινε γιατί έχει κάποια σοβαρά πλεονεκτήματα σε σχέση με άλλες γνωστές Σχεσιακές Βάσεις Δεδομένων, αλλά γιατί ήταν μια βάση που δεν είχα δουλέψει μέχρι τώρα και ήθελα να δω κάτι χρησιμοποιήσω κάτι διαφορετικό από την Oracle ή τη MySQL [19].

**Heroku**


**Εικόνα 19:** Το logo του Heroku

Τέλος, για να έχω μια πιο ολοκληρωμένη λύση, αποφάσισα να ανεβάσω την εφαρμογή στο Heroku (το logo του βλέπουμε στην Εικόνα 19). Το Heroku είναι μια αρκετά δημοφιλής cloud πλατφόρμα, όπου μπορείς να ανεβάσεις τις εφαρμογές σου, για να είναι διαθέσιμες στο διαδίκτυο, κάτω από ένα subdomain του Heroku, αν δεν έχεις δικό σου domain. Θεωρείται μία από τις πρώτες cloud πλατφόρμες, αφού άρχισε να αναπτύσσεται το 2007, υποστηρίζοντας αρχικά μόνο εφαρμογές σε Ruby, αν και τώρα πια υποστηρίζει όλες τις δημοφιλής γλώσσες προγραμματισμού [18].

Το Heroku είναι ιδιαίτερα δημοφιλή για προσωπικά project, μιας και έχει δωρεάν πλάνο, ενώ είναι και πολύ δημοφιλή σε startups που έχουν μικρότερο αριθμό υπαλλήλων και δεν έχουν την πολυτέλεια να ξοδεύουν πολύ χρόνο σε πιο εξειδικευμένες πλατφόρμες, μιας και το Heroku είναι πολύ εύκολο στη χρήση του, κάτι το οποίο διαπίστωσα πολύ γρήγορα, αφού και εγώ ήμουν πολύ αρχάριος στον χώρο και δεν αντιμετώπισα κανένα θέμα με την πλατφόρμα. Τέλος, ένα πολύ θετικό που έχει είναι ότι μπορείς πολύ εύκολα να κάνεις scale την εφαρμογή σου, πατώντας μονάχα ένα κουμπί, και δίνοντας περισσότερα dynos στην εφαρμογή σου. Τα dynos είναι απομονωμένα containers που φιλοξενούν τον κώδικα της εφαρμογής με τα dependencies του, μαζί με τους απαραίτητους υπολογιστικούς πόρους. Αυτό το είδος scaling ονομάζεται και horizontal scaling και είναι το κλειδί για μεγάλο scaling στο cloud.

## 2.2 Σχεδιασμός Βάσης Δεδομένων

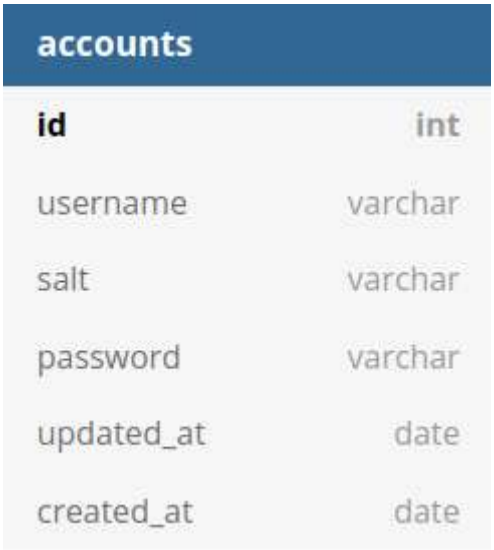
Όπως ανέφερα και παραπάνω, διάλεξα μια σχεσιακή βάση δεδομένων, την PostgreSQL, καθώς πίστευα ότι καλύπτει απόλυτα τις ανάγκες μου. Τα τραπέζια που δημιούργησα είναι τα παρακάτω:

- accounts
- students
- courses
- requests
- grades
- announcements
- enrollment

**Accounts**

Το τραπέζι accounts είναι ένα πολύ βασικό και απλό τραπέζι με σκοπό την αποθήκευση των πληροφοριών σύνδεσης των χρηστών της εφαρμογής.
Οι στήλες του accounts είναι:

- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε χρήστη
- username, το οποίο πρόκειται να το email του χρήστη
- password, εκεί αποθηκεύεται το hashed password του χρήστη
- salt, το salt για την κωδικοποίηση και αποκωδικοποίηση του κωδικού
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής




**Σχήμα 3:** Το table των accounts

**Students**

Το τραπέζι students, το οποίο είναι και το μεγαλύτερο, εμπεριέχει και όλες τις πληροφορίες για τους φοιτητές και χρησιμοποιείται σε διάφορες περιπτώσεις, όπως η αρχική σελίδα του φοιτητολογίου που δείχνει τις βασικές πληροφορίες του φοιτητή. Επίσης, χρησιμοποιείται για την ταυτοποίηση των αιτημάτων για έγγραφα που κάνουν οι φοιτητές. Οι στήλες που έχει είναι οι παρακάτω:

- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε φοιτητή
- email, που είναι το προσωπικό email του φοιτητή
- first_name, που είναι το όνομα του φοιτητή
- last_name, που είναι το επίθετο του φοιτητή
- birth_date, που είναι η ημερομηνία γέννησης του φοιτητή
- admission_date, που είναι η ημερομηνία εγγραφής του φοιτητή στο πανεπιστήμιο
- english_full_name, που είναι το όνομα του φοιτητή σε λατινικούς χαρακτήρες
- address, που είναι η διεύθυνση κατοικίας του φοιτητή
- city, που είναι η πόλη κατοικίας του φοιτητή
- postal_code, που είναι ο ταχυδρομικός κώδικας της κατοικίας του φοιτητή
- district, που είναι ο νομός κατοικίας του φοιτητή
- country, που είναι η χώρα κατοικίας του φοιτητή
- phone, που είναι το τηλέφωνο ή κινητό του φοιτητή
- department, που είναι το τμήμα φοίτησης του φοιτητή
- current_semester, που είναι το εξάμηνο που βρίσκεται ο φοιτητής
- gpa, που είναι ο μέσος όρος του φοιτητή
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 4 μπορούμε να δούμε και την αναπαράσταση του ***students*** table.

| students | |
|---|---|
| **id** | **int** |
| email | varchar |
| first_name | varchar |
| last_name | varchar |
| birth_date | date |
| admission_date | date |
| english_full_name | varchar |
| address | varchar |
| city | varchar |
| postal_code | int |
| district | varchar |
| country | varchar |
| phone | int |
| department | varchar |
| current_semester | int |
| gpa | float |
| updated_at | date |
| created_at | date |

**Σχήμα 4:** Το table των students

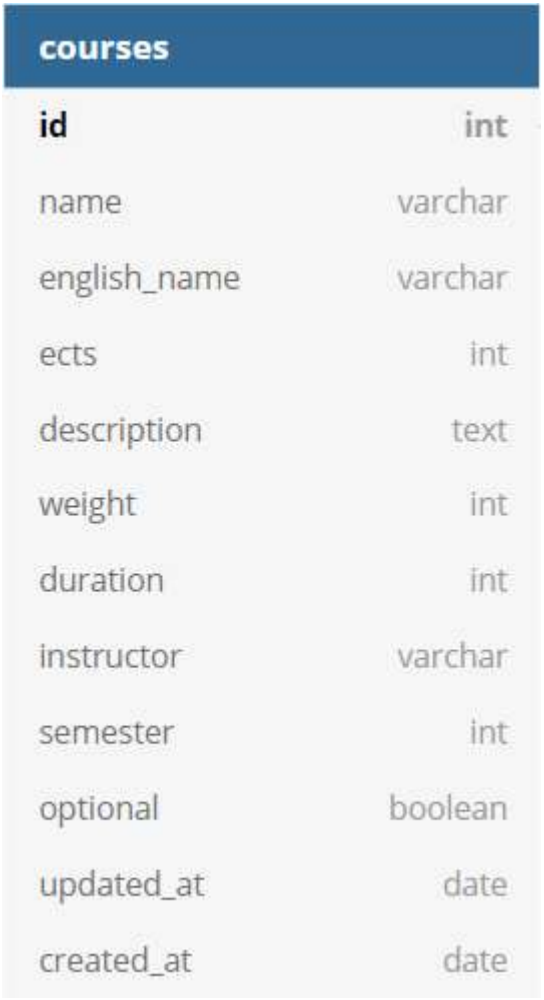
**Courses**

Το τραπέζι courses χρησιμοποιείται για την αναπαράσταση των μαθημάτων της σχολής και περιέχει όλες τις απαραίτητες πληροφορίες για κάθε μάθημα. Μεταξύ άλλων, από αυτό το τραπέζι διαβάζει το σύστημα για να δείξει τα μαθήματα που μπορεί να δηλώσει ο κάθε φοιτητής κάθε εξάμηνο, αλλά και από αυτό το τραπέζι τραβάει το σύστημα τις πληροφορίες για τα μαθήματα που έχει περάσει ο φοιτητής. Οι στήλες που έχει είναι οι παρακάτω:

- id, το οποίο είναι το μοναδικό id που δίνει η σχολή σε κάθε μάθημα
- name, το οποίο είναι το όνομα του μαθήματος
- english_name, το οποίο είναι το όνομα του μαθήματος στα αγγλικά
- ects, που είναι οι μονάδες ECTS του μαθήματος
- description, που είναι η περιγραφή του μαθήματος

- weight, που είναι ο συντελεστής βαρύτητας του μαθήματος
- duration, που είναι οι ώρες διδασκαλίας του μαθήματος
- instructor, που είναι ο υπεύθυνος καθηγητής του μαθήματος
- semester, που είναι το εξάμηνο που αντιστοιχεί το μάθημα
- optional, που είναι ένα boolean που αναπαριστά αν το μάθημα είναι επιλογής ή όχι
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 5 μπορούμε να δούμε και την αναπαράσταση του ***courses*** table.




**Σχήμα 5:** Το table των courses

**Requests**

Το τραπέζι requests αναπαριστά τις δηλώσεις που κάνουν οι φοιτητές μέσω του φοιτητολογίου στη γραμματεία. Εμπεριέχει όλες τις πληροφορίες της δήλωσης και χρησιμοποιείται τόσο κατά τη δήλωση των φοιτητών, όπου αποθηκεύεται μια καινούρια γραμμή, αλλά και από την πλευρά της γραμματείας, όπου το σύστημα απαριθμεί τις δηλώσεις που δεν έχουν ολοκληρωθεί. Οι στήλες που έχει είναι οι παρακάτω:

- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε δήλωση
- type, που είναι ο τύπος της δήλωσης
- student_id, που είναι το id του φοιτητή που έκανε τη δήλωση
- comments, που είναι ένα πεδίο για τα σχόλια που μπορεί να έχει αφήσει ο φοιτητής μαζί με τη δήλωση
- completed, που αντικατοπτρίζει την κατάσταση που βρίσκεται η δήλωση και αν έχει ολοκληρωθεί ή όχι
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 6 μπορούμε να δούμε και την αναπαράσταση του ***requests*** table.


**Σχήμα 6:** Το table των requests

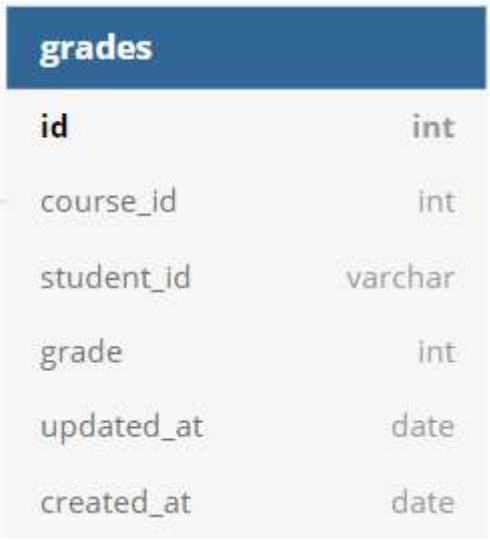
**Grades**

Το τραπέζι grades αποθηκεύει τους βαθμούς που παίρνουν οι μαθητές σε κάποιο μάθημα. Χρησιμοποιείται κατά την εισαγωγή των βαθμών από τη γραμματεία, όπου εκεί γίνεται η αποθήκευση στη βάση. Επίσης, η εφαρμογή διαβάζει από αυτό το τραπέζι στη σελίδα όπου οι χρήστες μπορούν να δουν τις βαθμολογίες τους στα μαθήματα.
Οι στήλες που έχει είναι οι παρακάτω:

- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε δήλωση

- course_id, που είναι το id του μαθήματος
- grade, που είναι η βαθμολογία για το μάθημα
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 7 μπορούμε να δούμε και την αναπαράσταση του ***grades*** table.




**Σχήμα 7:** Το table των grades

**Announcements**

Το τραπέζι announcements αποθηκεύει τις ανακοινώσεις που βγάζει η γραμματεία και είναι το μόνο τραπέζι που δεν έχει καμία σχέση ξένου κλειδιού με τα άλλα τραπέζια της βάσης. Χρησιμοποιείται κατά την υποβολή μιας ανακοίνωσης από τη γραμματεία, όπου και αποθηκεύεται από τη βάση, αλλά η εφαρμογή διαβάζει από το τραπέζι στη σελίδα που δείχνει τις ανακοινώσεις από τη γραμματεία.
Οι στήλες που έχει είναι οι παρακάτω:

- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε δήλωση
- description, που είναι το κείμενο της ανακοίνωσης
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 8 μπορούμε να δούμε και την αναπαράσταση του ***announcements*** table.

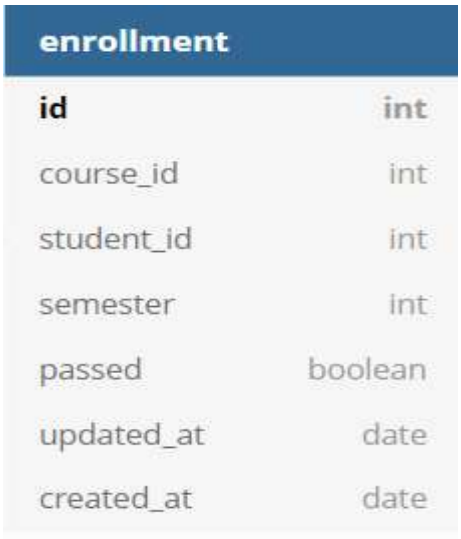

**Σχήμα 8:** Το table των announcements

**Enrollment**

Το τραπέζι enrollment αποθηκεύει τις δηλώσεις μαθημάτων των φοιτητών κάθε εξάμηνο. Εκτός από τη χρήση του κατά τη δήλωση των μαθημάτων, χρησιμοποιείται και από τη σελίδα που μπορεί η γραμματεία να δει όλες τις δηλώσεις μαθημάτων.
Οι στήλες που έχει είναι οι παρακάτω:

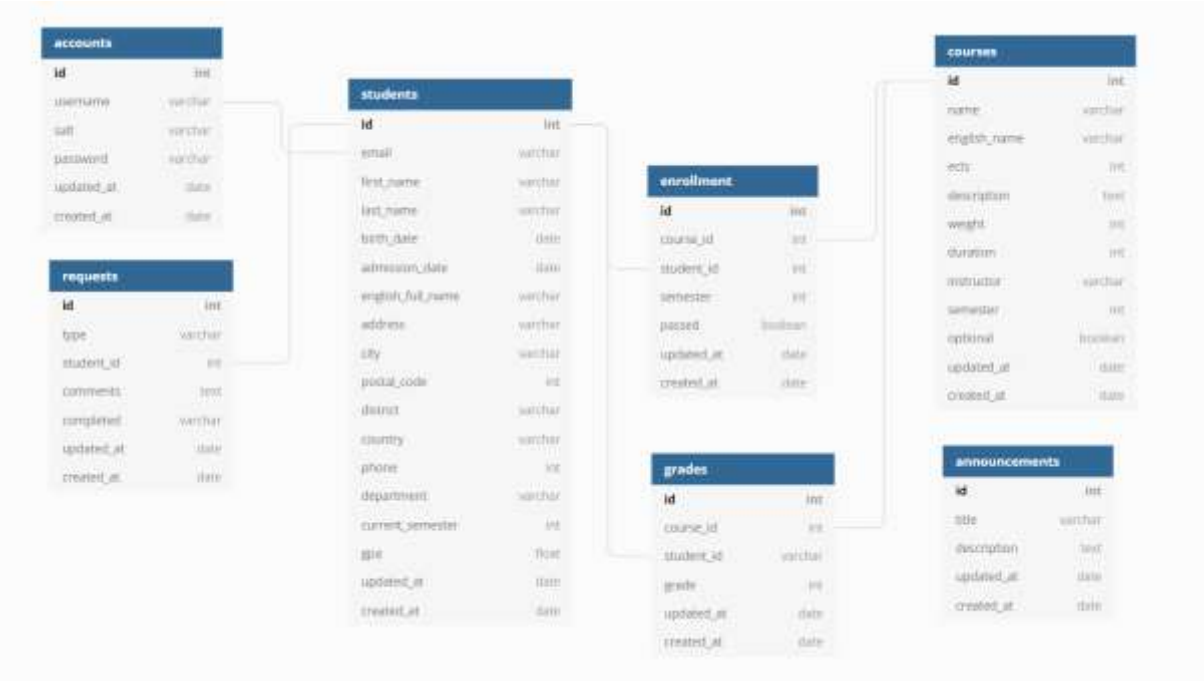
- id, ένας τυχαίος και μοναδικός ακέραιος αριθμός για κάθε δήλωση
- course_id, που είναι το id του μαθήματος
- student_id, που είναι το id του φοιτητή που έκανε τη δήλωση
- semester, που είναι το τωρινό εξάμηνο του φοιτητή όταν κάνει τη δήλωση
- passed, μια boolean τιμή για το εάν έχει περάσει το μάθημα ο φοιτητής
- updated_at, είναι η ημερομηνία της τελευταίας επεξεργασίας στη γραμμή
- created_at, είναι η ημερομηνία καταχώρησης της γραμμής

Στο Σχήμα 9 μπορούμε να δούμε και την αναπαράσταση του ***enrollment*** table.




**Σχήμα 9:** Το table του enrollment

Το τελικό σχήμα της βάσης αναπαρίσταται στο Σχήμα 10:




**Σχήμα 10:** Το τελικό σχήμα της Βάσης Δεδομένων

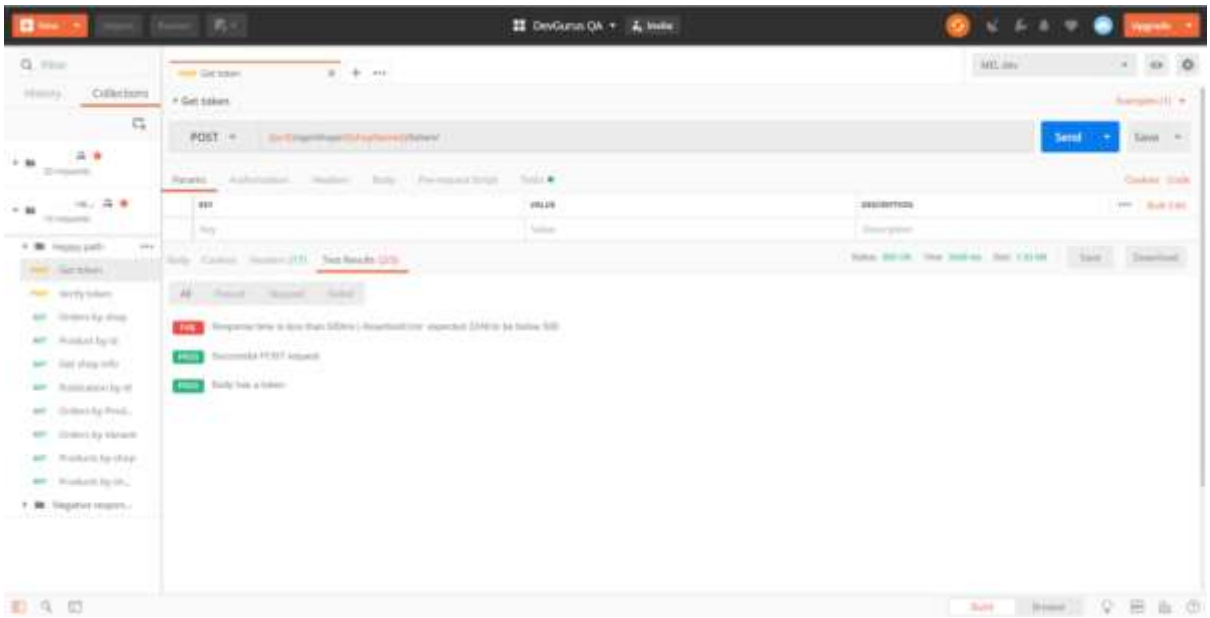
## 2.3 Σχεδίαση REST API

Για τις ανάγκες του backend της εφαρμογής υλοποιήθηκε ένα REST API σε node.JS. Στα πλαίσια της ανάλυσης και επεξήγησης του API, θα χωρίσω τα διαφορετικά endpoints ανάλογα με τις βασικές λειτουργίες της εφαρμογής και της βάσης με την οποία αλληλοεπιδρούν.

**Postman**

Ένα εργαλείο που χρησιμοποιήθηκε αρκετά κατά την ανάπτυξη του API ήταν το Postman. Το Postman είναι ένα πολύ χρήσιμο εργαλείο για τους backend developers που φτιάχνουν endpoints, γιατί τους δίνει τη δυνατότητα να στέλνουν HTTP requests μέσα από το εργαλείο, στα endpoints που έχουν φτιάξει και τρέχουν σε έναν τοπικό σέρβερ, και να δοκιμάζουν αν το endpoint συμπεριφέρεται όπως είναι αναμενόμενο. Φυσικά, η λειτουργία του δεν σε περιορίζει μόνο να “χτυπήσεις” μόνο έναν τοπικό σέρβερ, αφού μπορείς να “χτυπήσεις” και κάποιο 3rd party API και να δεις την απάντηση του σέρβερ.

Αυτή η λειτουργία είναι άκρως εξυπηρετική, αφού για αυτόν που θέλει να τεστάρει ένα endpoint, δεν χρειάζεται να έχει ή να πηγαίνει στο γραφικό περιβάλλον της εφαρμογής και να κάνει τις απαραίτητες ενέργειες για να γίνει το request στον σέρβερ. Αυτό είναι ιδιαίτερα εξυπηρετικό σε περιπτώσεις που θες να δοκιμάσεις κάποιο POST request μιας φόρμας για παράδειγμα, όπου χωρίς κάποιο τέτοιο εργαλείο θα χρειαζόταν να συμπληρώνεις τη φόρμα με τις πληροφορίες κάθε φορά που θες να δοκιμάσεις το endpoint, σε αντίθεση με εργαλεία όπως το Postman, που απλά γράφεις μια φορά το body του request και στη συνέχεια απλώς πατάς “SEND”. Στην Εικόνα 20, βλέπουμε ένα παράδειγμα request στο Postman



**Εικόνα 20:** Παράδειγμα POST request στο Postman

**Γενικές πληροφορίες**

Όλα τα endpoint, όπως είναι φυσικό, απαντούν σε μορφή JSON, ενώ μορφή JSON θα πρέπει να έχει και το body των request. Επιπλέον έχουν φτιαχτεί έτσι ώστε αν συμβεί κάποιο error, η απάντηση να είναι της μορφής:

```
{
   success: false,
   error: 'There was an error.'
}
```

Η περιγραφή του error που γυρνούσε ως απάντηση στον client είχε αναγνωριστεί στο backend και παρουσιαζόταν σε τέτοια μορφή ώστε να μπορούσε να εμφανιστεί σε κάποιο μικρό παράθυρο ειδοποιήσεων ή σε κάποιο error text area, ώστε ο χρήστης να ενημερώνεται για το τι πήγε στραβά.

Αντίθετα, για τα POST και PATCH requests, όταν όλα έχουν πάει καλά η απάντηση θα είναι:

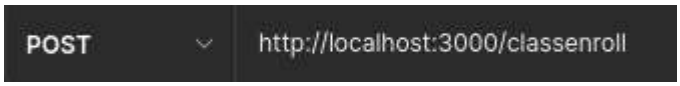
```
{
  success: true
}
```

ενώ για τα GET requests θα επιστρέφεται η ζητούμενη απάντηση σε μορφή JSON και πάλι.

Ακολουθούν περιγραφές για τα endpoints της εφαρμογής.

## Εγγραφή σε μάθημα (class enrollment)

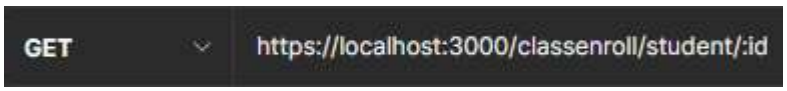
Τα συγκεκριμένα endpoints χρησιμοποιούνται και από τον φοιτητή και από τη γραμματεία.




Το παραπάνω endpoint χρησιμοποιείται όταν ο φοιτητής κάνει τη δήλωση των μαθημάτων του, και μετά την επιλογή κάνει submit τη φόρμα. Στο endpoint αυτό στέλνουμε την πληροφορία μέσα σε ένα Array από Objects, με τα μαθήματα και κάποιες άλλες πληροφορίες και στη συνέχεια να αποθηκεύει τη δήλωση στη βάση. Ένα παράδειγμα ενός τέτοιου request είναι το παρακάτω:

```
{
  data: [
    {
      studentID: 123456,
      courseID: 69,
      currentSemester: 4
    },
    {
      studentID: 123456,
      courseID: 70,
      currentSemester: 4
    }
  ]
}
```

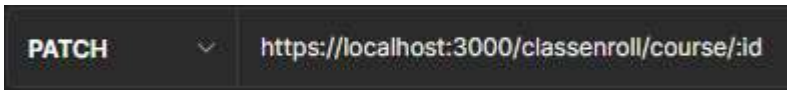
Βλέπουμε ότι από το request έρχονται 3 σημαντικές πληροφορίες, το id του μαθητή που είναι απαραίτητη πληροφορία φυσικά, το id του μαθήματος στο οποίο κάνει εγγραφή ο μαθητής, καθώς και το τωρινό εξάμηνο του φοιτητή ώστε να υπάρχει στη βάση ιστορικό για κάθε εγγραφή σε κάποιο μάθημα από κάποιο φοιτητή. Κάθε φορά που γίνεται μια νέα εγγραφή σε μάθημα, το API περνάει και στην εγγραφή στη βάση το πεδίο “passed” ως false, μιας και σε μια νέα εγγραφή ενός μαθήματος από κάποιο φοιτητή θεωρείται λογικό ότι δεν έχει περάσει το μάθημα.




Το παραπάνω request χρησιμοποιείται για να τραβήξει τα μαθήματα στα οποία ο φοιτητής έχει εγγραφεί καθ' όλη τη διάρκεια φοίτησής του, ενώ προσθέτοντας μια παράμετρο semester με τιμή ένα εξάμηνο, θα γυρίσει τα μαθήματα εγγραφής του συγκεκριμένου εξαμήνου. Προσθέτοντας την παράμετρο unique ως true, το API θα μας επιστρέψει τις τελευταίες και μοναδικές εγγραφές για κάθε μάθημα, κάτι το οποίο χρησιμοποιείται στις σελίδες που ο φοιτητής μπορεί να δει τα επιτυχώς ή μη μαθήματα που έχει δώσει.


Παρόμοιο request με το παραπάνω, μόνο με τη διαφορά ότι τώρα ψάχνουμε με βάση το id του course και σαν απάντηση θα έχουμε όλες τις εγγραφές που έχουν γίνει για το συγκεκριμένο μάθημα, ενώ και πάλι, προσθέτοντας το semester ως παράμετρο με μία τιμή, γυρνάει τα αποτελέσματα για ένα συγκεκριμένο εξάμηνο. Το συγκεκριμένο endpoint χρησιμοποιείται μόνο από μια σελίδα από τη γραμματεία. Ακολουθεί η απάντηση του API για το πρώτο GET request, ενώ η απάντηση για το 2ο είναι παρόμοια, μόνο που αντί για courseID επιστρέφει studentID μέσα σε κάθε object.

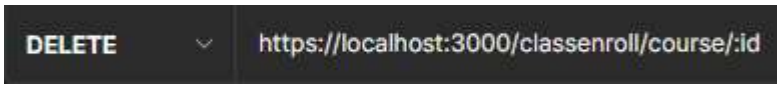

```
{
  data: [
    {
      name: "Λογική Σχεδίαση",
      courseID: 69,
      semester: 2,
      passed: true
    },
    {
      name: "Κρυπτογραφία",
      courseID: 70,
      semester: 8,
      passed: false
    }
  ]
}
```


Το συγκεκριμένο endpoint χρησιμοποιείται αποκλειστικά από τη γραμματεία κατά τη διάρκεια βαθμολόγησης των μαθημάτων, με τη μικρή διαφορά ότι το συγκεκριμένο endpoint δεν καλείται από το frontend, αλλά μονάχα από το backend, από το endpoint που χρησιμοποιείται για την καταγραφή των βαθμών. Αν ένα μάθημα έχει βαθμολογηθεί με πάνω από τη βάση, το

API χτυπάει αυτό το endpoint, το οποίο βρίσκει την τελευταία εγγραφή για το συγκεκριμένο μάθημα από τον φοιτητή και του ανανεώνει το πεδίο passed ως true.

```
{
  data: [
    {
      studentID: 12345,
      passed: true,
    },
    {
      studentID: 1234567,
      passed: true
    }
  ]
}
```




Το παραπάνω endpoint χρησιμοποιείται αποκλειστικά από τη γραμματεία για την αφαίρεση της τελευταίας εγγραφής κάποιου μαθήματος για κάποιον φοιτητή. Όπως και στα προηγούμενα endpoints, έτσι και αυτό το endpoint μπορεί να δεχτεί ένα array από διαφορετικούς φοιτητές και να σβήσει την τελευταία εγγραφή μαθήματος χωρίς να γίνουν πολλαπλά χτυπήματα στο endpoint, αφού και η εφαρμογή αφήνει τη γραμματεία να προσθέσει πολλά id μαθητών για την αφαίρεση μαθημάτων.

```
{
  data: [
    {
      studentID: 12345
    },
    {
      studentID: 123456
    }
  ]
}
```

## Μαθήματα (courses)

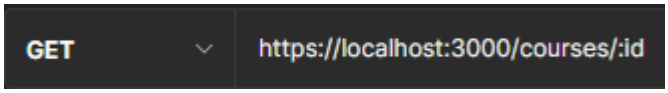
Τα συγκεκριμένα endpoints χρησιμοποιούνται από τη γραμματεία για τη διαχείριση των μαθημάτων της σχολής, δηλαδή την προσθήκη ή αφαίρεση μαθημάτων και τυχόν αλλαγές σε κάποιο μάθημα.


Το παραπάνω endpoint χρησιμοποιείται για την προσθήκη ενός ή παραπάνου μαθήματος στην εφαρμογή. Το endpoint καλείται μονάχα από μία σελίδα μέσα από τον λογαριασμό της γραμματείας. Παρακάτω βλέπουμε το body του request.

```
{
  data: [
    {
      name: 'Προγραμματισμός Ι',
      englishName: 'Programming I',
      ects: 5,
      description: 'Course description here',
      weight: 3,
      duration: 2,
      instructor: 'Konstantinos Tserpes',
      semester: 1,
      optional: false
    },
    {
      name: 'Προγραμματισμός ΙΙ',
      englishName: 'Programming II',
      ects: 5,
      description: 'Course description here',
      weight: 3,
      duration: 2,
      instructor: 'Konstantinos Tserpes',
      semester: 2,
      optional: false
    }
  ]
}
```

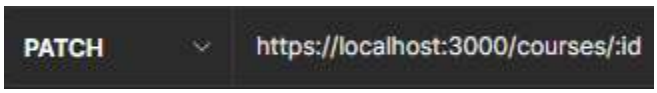

Το παραπάνω endpoint γυρίζει όλα τα μαθήματα και τις πληροφορίες τους και χρησιμοποιείται σε μία σελίδα όπου η γραμματεία μπορεί να δει όλα τα μαθήματα, καθώς και τις πληροφορίες κάθε μαθήματος με ένα κλικ πάνω στο μάθημα. Το endpoint γυρίζει ένα array από objects. Αν περαστεί η παράμετρος minimal ως true, τότε επιστρέφονται μόνο οι βασικές πληροφορίες για κάθε μάθημα, όπως βλέπουμε στην απάντηση παρακάτω.

```
{
  data: [
    {
      name: 'Προγραμματισμός Ι',
      id: 123456,
      semester: 1
    },
    {
      name: 'Προγραμματισμός ΙΙ',
      id: 123454567',
      semester: 2
    }
  ]
}
```




Το παραπάνω endpoint είναι παρόμοιο με το προηγούμενο, με τη διαφορά ότι αυτό εδώ γυρίζει μόνο ένα μάθημα και τις πληροφορίες του, για αυτό και στο url χρειάζεται και το id του μαθήματος. Χρησιμοποιείται από κάποιες σελίδες της γραμματείας, όταν θέλουν να δουν τις πληροφορίες για ένα συγκεκριμένο μάθημα.

```
{
  name: 'Προγραμματισμός Ι',
  id: 123456,
  englishName: 'Programming I',
  ects: 5,
  description: 'Course description here',
  weight: 3,
  duration: 2,
  instructor: 'Konstantinos Tserpes',
  semester: 1,
  optional: false
}
```



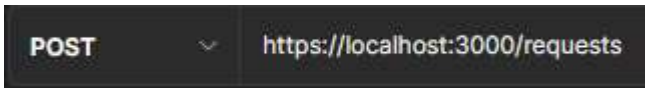

Το παραπάνω endpoint δίνει τη δυνατότητα στη γραμματεία να κάνει αλλαγές στις πληροφορίες κάποιου μαθήματος. Αν για παράδειγμα θέλαμε να αλλάξουμε την περιγραφή του μαθήματος, το body του request θα ήταν το παρακάτω.

```
{
  description: 'New description'
}
```


Το παραπάνω endpoint είναι πολύ απλό, χρησιμοποιείται για τη διαγραφή ενός συγκεκριμένου μαθήματος από τη βάση δεδομένων της εφαρμογής.

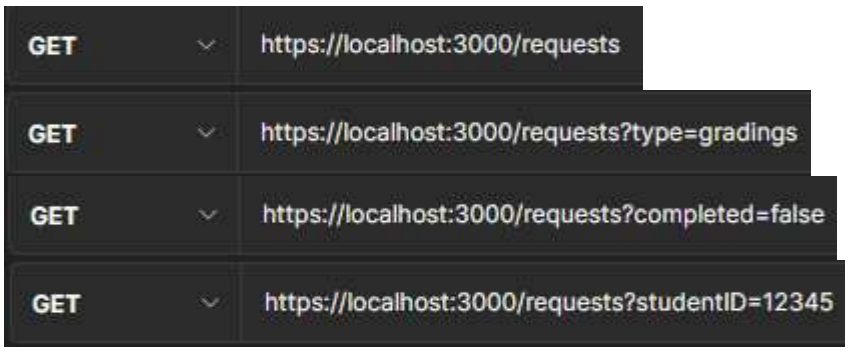
## Αιτήματα (requests)

Η συγκεκριμένη συλλογή endpoints χειρίζεται τη λειτουργικότητα που προσφέρει η εφαρμογή για τις διάφορες αιτήσεις που μπορούν να κάνουν οι φοιτητές στη γραμματεία. Τα endpoint χρησιμοποιούνται και από τους φοιτητές και από τη γραμματεία, για διαφορετικές χρήσεις.




Το παραπάνω endpoint χρησιμοποιείται από τους φοιτητές για την εισαγωγή μιας καινούργιας αίτησης στο σύστημα. Δεν υπάρχει κάποια ιδιαίτερη πολυπλοκότητα στα endpoints των αιτημάτων, όπως μπορούμε να δούμε και στο body του request, στέλνεται ο τύπος (type) του αιτήματος, οι οποίοι είναι συγκεκριμένοι και επιλέγονται από το UI, το id του φοιτητή, η φόρμα με κάποιο προαιρετικό σχόλιο και η πληροφορία για το αν έχει ολοκληρωθεί (completed) το αίτημα από τη γραμματεία, που στη συγκεκριμένη περίπτωση θα είναι false.

```
{
  data: [
    {
      type: 'Αναλυτική Βαθμολογία',
      studentID: 12345,
      comments: 'Κάποιο σχόλιο',
      completed: false
    },
    {
      type: 'Αναλυτική Βαθμολογία στα αγγλικά',
      studentID: 12345,
      comments: 'Κάποιο σχόλιο',
      completed: false
    }
  ]
}
```



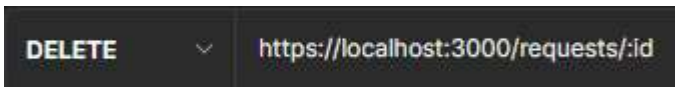

Το παραπάνω endpoint είναι το βασικό GET της συλλογής των αιτημάτων, το οποίο επιστρέφει όλα τα αιτήματα που έχουν γίνει. Μπορεί αυτό να είναι το βασικό endpoint, όμως κανένα μέρος της εφαρμογής δεν χρειάζεται να ξέρει όλα τα αιτήματα που έχουν γίνει ποτέ, οπότε κυρίως χρησιμοποιούνται παραλλαγές, προσθέτοντας κάποια παράμετρο. Οι 3 παράμετροι που μπορούν να χρησιμοποιηθούν είτε ξεχωριστά, είτε και μαζί είναι το type, στο studentID και το completed.

Αν προστεθεί το type, το οποίο είναι κάποιοι προκαθορισμένοι τύποι αιτημάτων, τότε επιστρέφονται όλα τα αιτήματα που έχουν αυτόν τον τύπο. Αν προστεθεί η παράμετρος completed θα γυρίσει τα αιτήματα που είτε έχουν ολοκληρωθεί είτε όχι, ανάλογα με την τιμή της παραμέτρου (true/false). Η βασική σελίδα της γραμματείας όσον αφορά τα αιτήματα, χρησιμοποιεί αυτό το endpoint με completed=false για να βλέπει όλα τα μη ολοκληρωμένα αιτήματα των φοιτητών. Τέλος, μια σελίδα από τη μεριά των φοιτητών χρησιμοποιεί αυτό το endpoint με την παράμετρο studentID να ισούται το ID του φοιτητή, για να εμφανίσει όλες τις αιτήσεις που έχουν γίνει από τον φοιτητή. Ακολουθεί η απάντηση του endpoint σε ένα από αυτά τα requests.

```
{
  type: 'Αναλυτική Βαθμολογία',
  studentID: 12345,
  comments: 'Κάποιο σχόλιο',
  completed: false,
  createdAt: '2020-12-12',
  updatedAt: '2020-12-13'
}
```


Το παραπάνω endpoint χρησιμοποιείται από τη γραμματεία για να ορίσει ένα αίτημα ως ολοκληρωμένο ή όχι.

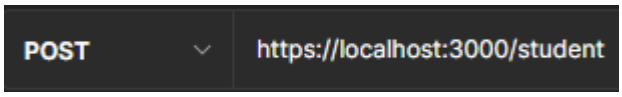
```
{
  completed: true
}
```




Το παραπάνω endpoint χρησιμοποιείται κι αυτό αποκλειστικά και μόνο από τη γραμματεία για τη διαγραφή κάποιου συγκεκριμένου αιτήματος μέσω του ID του.

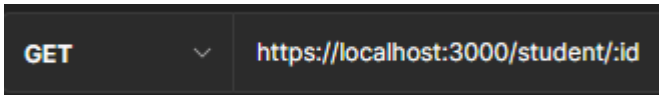
## Μαθητής (student)

Η συγκεκριμένη συλλογή από endpoints υπάρχει για την αλληλεπίδρασης της εφαρμογής με τη βάση δεδομένων που αποθηκεύει τις πληροφορίες για τους φοιτητές και τα προσωπικά τους στοιχεία.

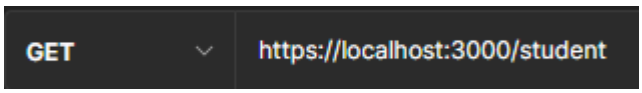



Το παραπάνω endpoint χρησιμοποιείται αποκλειστικά από τη γραμματεία για τη δημιουργία μιας νέας εγγραφής ενός φοιτητή στη βάση του συστήματος. Αυτό θεωρητικά γίνεται στις αρχές κάθε φοιτητικού έτους, όπου γράφονται οι καινούριοι φοιτητές. Παρακάτω είναι το body του request που γίνεται στο endpoint:

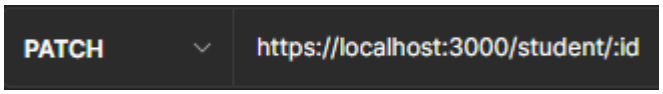
```
{
  email: 'myemail@gmail.com',
  firstName: 'Δημήτρης',
  lastName: 'Κατωμέρης',
  birstDate: '14/02/1996',
  admissionDate: '01/10/2014',
  englishFullName: 'Dimitris Katomeris',
  address: 'Γράμμου 69',
  city: 'Μοσχάτο',
  postalCode: 18345,
  district: 'Αττική',
  country: 'Ελλάδα',
  phone: 6980657689,
  department: 'Πληροφορικής'
}
```




Το παραπάνω endpoint είναι αυτό που χρησιμοποιείται και περισσότερο από την εφαρμογή, αφού η εφαρμογή χτυπάει το συγκεκριμένο endpoint στην αρχική σελίδα της εφαρμογής, για να τραβήξει και να δείξει τις πληροφορίες του φοιτητή που είναι συνδεδεμένος, μέση του id του. Η απάντηση του endpoint στο παραπάνω GET request είναι αρκετά παρόμοια με το body του POST request, με τη διαφορά ότι επιστρέφει και δύο ακόμα πληροφορίες, το τωρινό εξάμηνο (currentSemester) και το μέσο όρο (gpa).

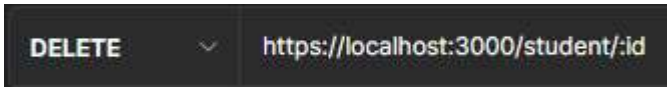



Παρομοίως, αν δεν χρησιμοποιηθεί το id τότε επιστρέφονται όλοι οι φοιτητές. Αυτό το endpoint χρησιμοποιείται από μία σελίδα από τη μεριά της γραμματείας, όπου μπορούν να δουν όλους τους εγγεγραμμένους φοιτητές.




Το παραπάνω endpoint χρησιμοποιείται για την αλλαγή στοιχείων στους φοιτητές από τη γραμματεία. Εκτός από τις αρχικές πληροφορίες που δίνονται όταν γίνεται η εγγραφή ενός φοιτητή, ο πίνακας των φοιτητών κρατάει και την πληροφορία για τωρινό εξάμηνο φοίτησης του φοιτητή και τον μέσο όρο του. Οπότε, όταν είναι να γίνει ενημέρωση αυτών των δύο στοιχείων, η εφαρμογή χτυπάει, μεταξύ άλλων, και αυτό το endpoint. Ένα request μπορεί να έχει το παρακάτω body:

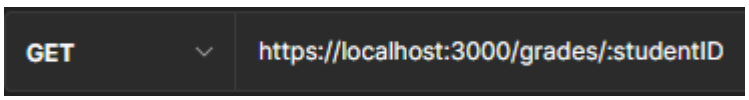
```
{
  gpa: 7.8,
  currentSemester: 5
}
```




Το παραπάνω endpoint χρησιμοποιείται για τη διαγραφή ενός φοιτητή από το σύστημα μέσω του id του.

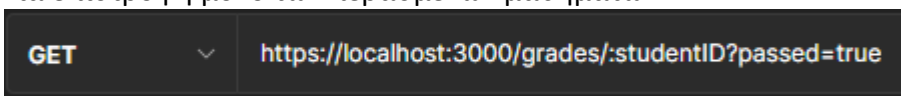
## Βαθμοί (grades)

Η συγκεκριμένη συλλογή από endpoints υπάρχει για την αλληλεπίδραση της εφαρμογής με το table της βάσης που αποθηκεύονται οι βαθμοί των φοιτητών. Αυτά τα endpoints χρησιμοποιούνται και από το interface των φοιτητών, αλλά και από τη γραμματεία. Από τη μεριά των φοιτητών χρησιμοποιούνται τα GET endpoints, ώστε οι φοιτητές να έχουν μια σελίδα να βλέπουν τους βαθμούς τους, ενώ από τη μεριά της γραμματείας είναι διαθέσιμα τα υπόλοιπα endpoints (POST, PUT, DELETE) για να μπορεί να καταχωρεί, επεξεργάζεται ή να σβήνει βαθμούς.




Το παραπάνω endpoint είναι το βασικό GET endpoint που χρησιμοποιείται από τη σελίδα που οι φοιτητές μπορούν να δουν τις βαθμολογίες για όλα τα μαθήματα που έχουν δώσει. Το endpoint όταν καλείται χωρίς κάποια άλλη παράμετρο, επιστρέφει όλες τις βαθμολογίες του φοιτητή χωρίς κανένα φιλτράρισμα. Μπορεί, βέβαια, μέσω κάποιων παραμέτρων να γίνει και κάποιο φιλτράρισμα για μην επιστρέφονται όλες οι βαθμολογίες.

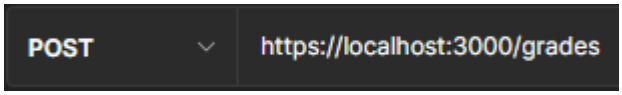
Για επιστροφή μόνο των περασμένων μαθημάτων:




Για επιστροφή όλων των βαθμολογιών σε ένα χρονολογικό έτος:

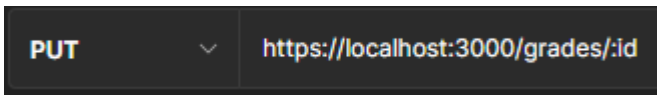

Οι παραπάνω παράμετροι μπορούν να χρησιμοποιηθούν και μαζί. Η απάντηση που έρχεται από το backend είναι η παρακάτω:

```
{
  data: [
    {
      courseName: 'Προγραμματισμος Ι',
      grade: 9,
      creationDate: '2010/01/01',
      passed: true
    },
    {
      courseName: 'Προγραμματισμος ΙΙ',
      grade: 7,
      creationDate: '2010/01/01',
      passed: true
    },
    {
      courseName: 'Λογική Σχεδίαση',
      grade: 8,
      creationDate: '2010/01/01',
      passed: true
    }
  ]
}
```



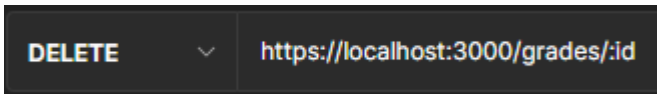

Το παραπάνω endpoint χρησιμοποιείται από τη γραμματεία για την εισαγωγή νέων βαθμολογιών στην εφαρμογή. Έτσι, χωρίς καμία παράμετρο, στο body του request στέλνεται η πληροφορία για μία βαθμολογία, αλλά στην εφαρμογή η γραμματεία έχει τη δυνατότητα να εισάγει πολλούς βαθμούς ταυτόχρονα για ένα συγκεκριμένο μάθημα, οπότε τότε μπαίνει η παράμετρος courseID (το id του μαθήματος) στο URL του request και το body περιέχει ένα array.

```
{
  data: [
    {
      studentID: 21438
      grade: 9
    },
    {
      studentID: 12345,
      grade: 7
    },
    {
      studentID: 141414,
      grade: 8
    }
  ]
}
```


Το παραπάνω endpoint χρησιμοποιείται από τη γραμματεία αν χρειαστεί να γίνει κάποια τροποποίηση σε κάποια βαθμολογία.

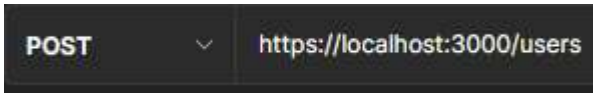

```
{
  grade: 7,
  studentID: 21438
}
```


Το παραπάνω endpoint χρησιμοποιείται από τη γραμματεία για τη διαγραφή μιας βαθμολογίας.

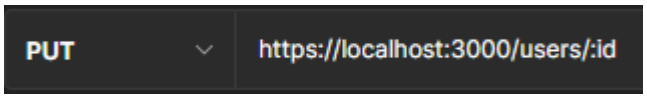
## Χρήστες (users)

Η παραπάνω συλλογή endpoints αλληλοεπιδρά με τη βάση δεδομένων που αποθηκεύονται οι χρήστες της εφαρμογής. Τα endpoints χρησιμοποιούνται αποκλειστικά από τη γραμματεία για τη δημιουργία καινούργιων χρηστών, την τροποποίηση υπάρχον χρηστών αλλά και τη διαγραφή τους.



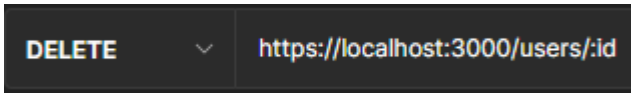
Το παραπάνω endpoint χρησιμοποιείται για τη δημιουργία ενός καινούργιου χρήστη. Παρακάτω φαίνεται το body του request:

```
{
  username: 'it212437@hua.gr',
  password: 'randompwd123'
}
```



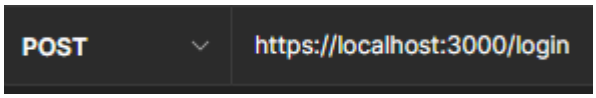

Το παραπάνω endpoint χρησιμοποιείται για την αλλαγή του username ή του password ενός χρήστη. Παρακάτω είναι το body του request για την αλλαγή του password ενός χρήστη.

```
{
  password: 'randompwd1234'
}
```




Το παραπάνω endpoint χρησιμοποιείται για τη διαγραφή ενός χρήστη.

### Σύνδεση στην εφαρμογή (login)




Το συγκεκριμένο endpoint χρησιμοποιείται για να κάνει log in τους χρήστες στην εφαρμογή. Αν το username και το password που πληκτρολογεί ο χρήστης ταυτοποιηθούν από το backend, τότε ο χρήστης μεταφέρεται στην αρχική σελίδα της εφαρμογής.

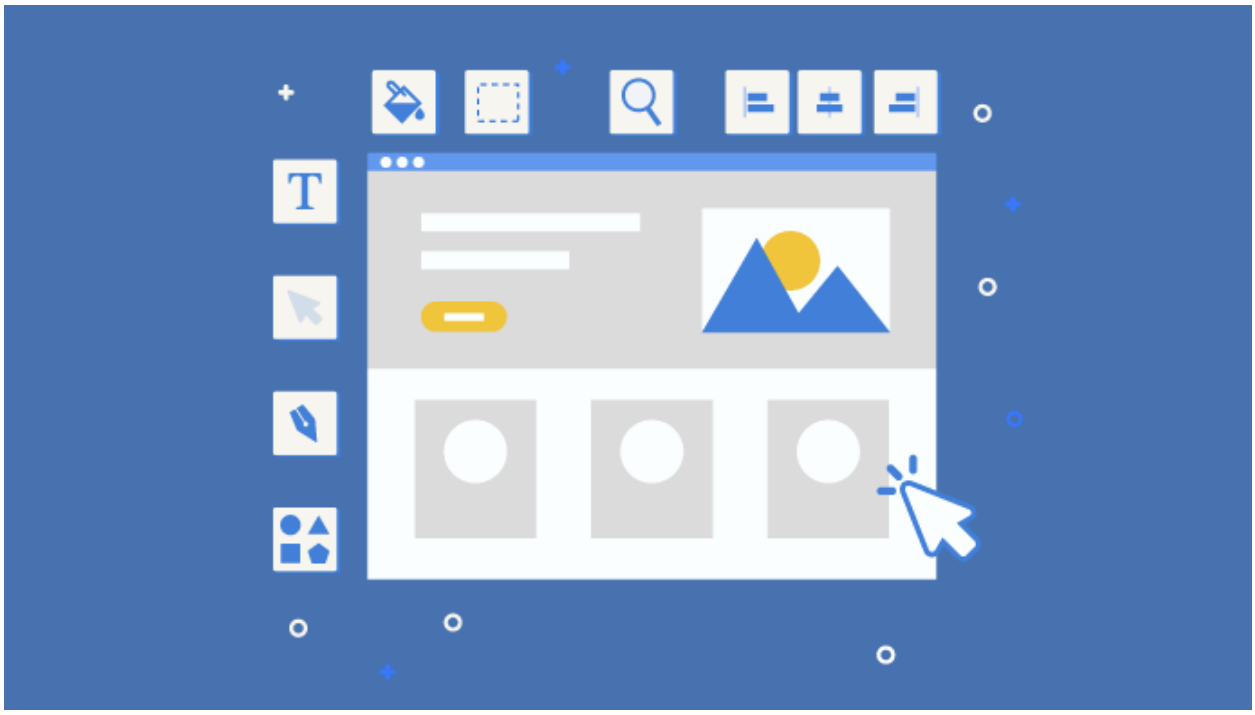
### Ασφάλεια σύνδεσης χρήστη στην εφαρμογή

Για κάθε εφαρμογή που υποστηρίζει χρήστες και login, είναι πολύ σημαντικό να ακολουθούνται ορισμένες αρχές ασφάλειας, ώστε σε περίπτωση κυβερνοεπίθεσης και όχι μόνο στους server της εφαρμογής, οι δράστες να μην μπορούν να υποκλέψουν τους λογαριασμούς των χρηστών και να συνδεθούν μέσω αυτών. Ένα ελάχιστο μέτρο ασφαλείας, και αυτό που ακολουθήθηκε στη συγκεκριμένη εφαρμογή είναι να μην αποθηκεύεται ο κωδικός του χρήστη σε plain text.

Έτσι στην εφαρμογή, κατά τη δημιουργία ενός νέου χρήστη περνά μέσα από μια κρυπτογραφική συνάρτηση κατακερματισμού και το αποτέλεσμα της συνάρτησης, ένα μεγάλο αλφαριθμητικό αποθηκεύεται ως password στη βάση δεδομένων, μαζί με με ένα άλλο τυχαίο αλφαριθμητικό που ονομάζεται salt, το οποίο δίνεται ως παράμετρος μαζί με τον κωδικό στη συνάρτηση. Στη συγκεκριμένη εφαρμογή χρησιμοποιείται η συνάρτηση SHA512.

Κατά τη σύνδεση του χρήστη στην εφαρμογή, ο κωδικός περνάει πάλι από την κρυπτογραφική συνάρτηση κατακερματισμού μαζί με το αποθηκευμένο salt και το αποτέλεσμα του ελέγχεται έναντι στο αποθηκευμένο hashed κωδικό. Αν τα δύο αλφαριθμητικά είναι ίδια, τότε δίνεται άδεια στον χρήστη να συνδεθεί στην εφαρμογή.

# ΚΕΦΑΛΑΙΟ 3
# ΤΟ USER INTERFACE ΤΗΣ ΕΦΑΡΜΟΓΗΣ



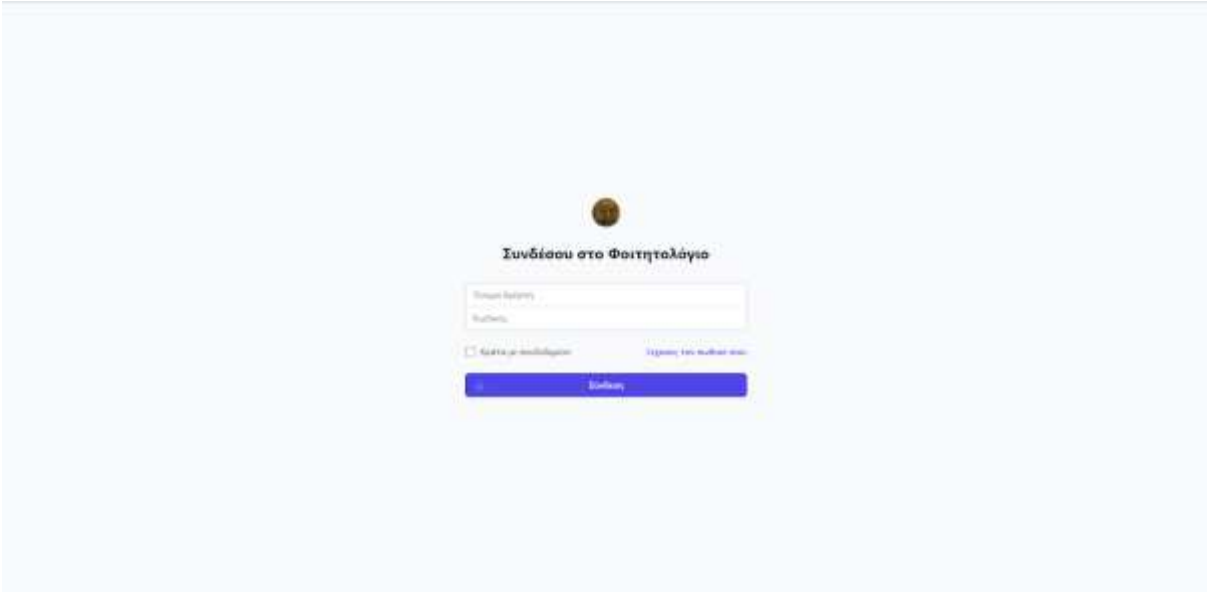
## 3.1 Το Interface και οι λειτουργίες της Εφαρμογής

Το UI της εφαρμογής φτιάχτηκε ως mobile first, αφού ως PWA θα πρέπει να παρέχει την καλύτερη δυνατή εμπειρία χρήσης σε χρήστες κινητών τηλεφώνων. Το UX της εφαρμογής είναι ταυτόχρονα απλό και χρηστικό αλλά και μοντέρνο. Αυτά είναι μερικά από τα στοιχεία που «λείπουν» από το υπάρχον Φοιτητολόγιο που προσφέρει το πανεπιστήμιο.

Όλες οι δυνατότητες που προσφέρει το υπάρχον Φοιτητολόγιο υλοποιήθηκαν και σε αυτή την εφαρμογή, ενώ προστέθηκαν και μερικές ακόμα δυνατότητες για να γίνει πιο πλήρες. Όπως έχει αναφερθεί ήδη, η εφαρμογή υποστηρίζει 2 διαφορετικά interfaces με διαφορετικές δυνατότητες, μία για τους φοιτητές και μία για τη γραμματεία. Παρακάτω περιγράφονται οι δυνατότητες των 2 interfaces.

### 3.1.1 Σύνδεση στην εφαρμογή

Η σύνδεση στην εφαρμογή γίνεται μέσω μιας απλής σελίδας, όπως βλέπουμε και στην Εικόνα 21, όπου οι χρήστες βάζουν το username και τον κωδικό τους για να συνδεθούν. Αν το username ή ο κωδικός είναι λάθος, θα εμφανιστεί ένα βοηθητικό μήνυμα.




**Εικόνα 21:** Η login σελίδα της εφαρμογής

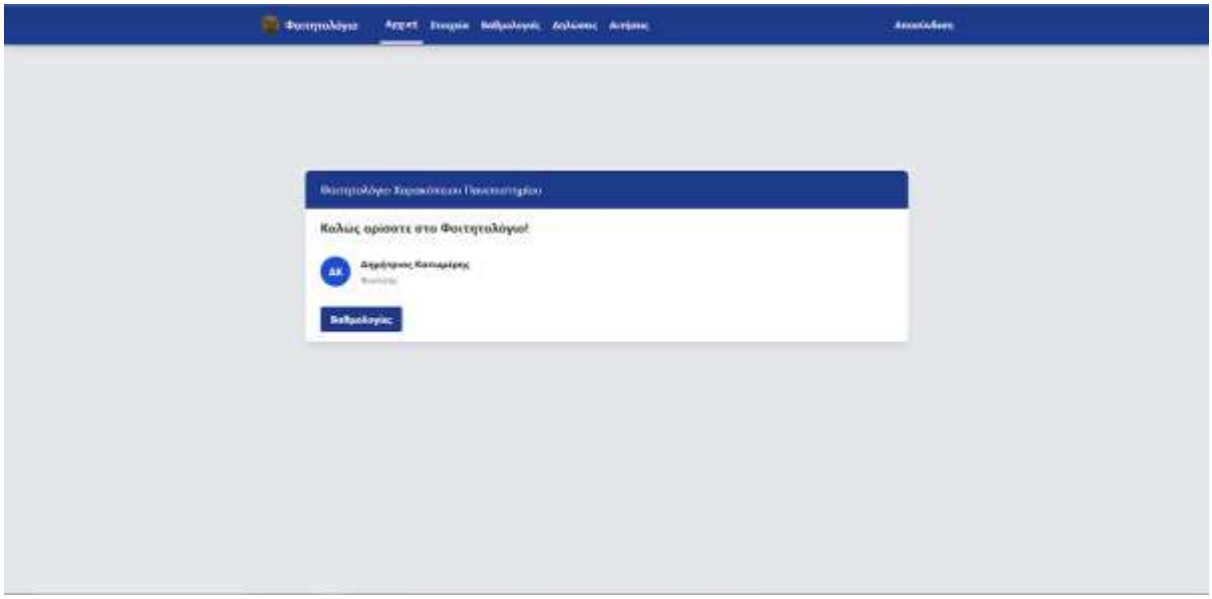
## 3.2 Interface φοιτητών

### 3.2.1 Γραμμή Πλοήγησης

Η γραμμή πλοήγησης, όπως φαίνεται και στην Εικόνα 22, είναι παρόμοια και στα 2 interfaces, με μόνες διαφορές στις επιλογές που έχει. Όπως βλέπουμε παρακάτω, στο Interface των φοιτητών, η γραμμή πλοήγησης έχει τις επιλογές «Αρχική», «Στοιχεία», «Βαθμολογίες», «Δηλώσεις» και «Αιτήσεις», ενώ πιο δεξιά βρίσκεται το κουμπί για αποσύνδεση από την εφαρμογή.


**Εικόνα 22:** Η γραμμή πλοήγησης

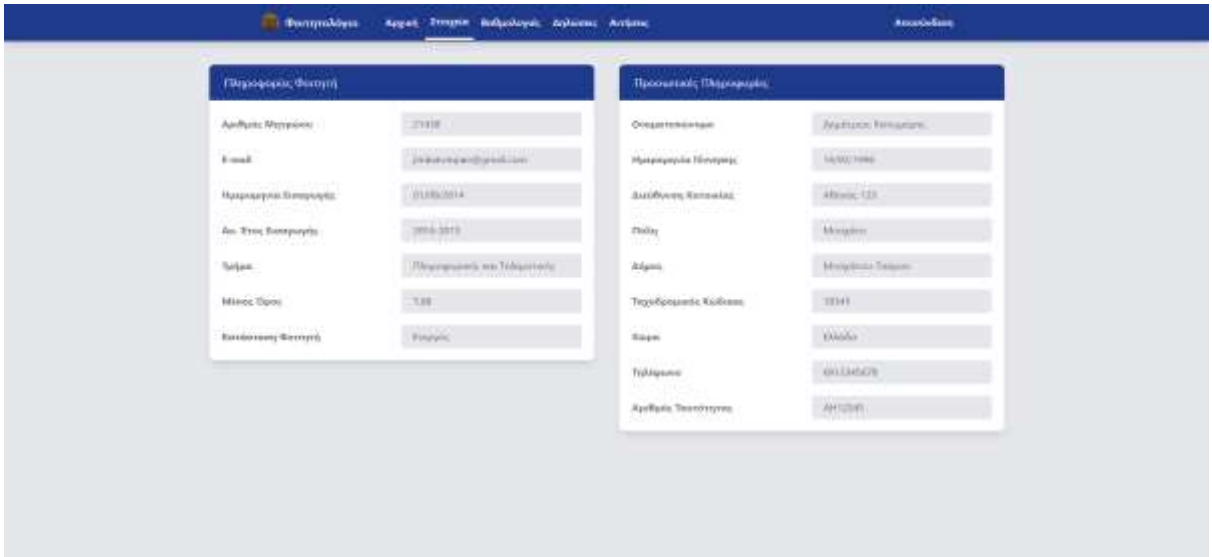
## 3.2.2 Αρχική σελίδα




**Εικόνα 23:** Η αρχική σελίδα στο interface των φοιτητών

Η αρχική σελίδα (Εικόνα 23) είναι η σελίδα την οποία αντικρίζει ο χρήστης κάθε φορά που συνδέεται στο Φοιτητολόγιο, ενώ μπορεί ανά πάσα στιγμή να την επισκεφτεί πατώντας στη γραμμή πλοήγησης την επιλογή «Αρχική». Σε αυτή τη σελίδα, η εφαρμογή καλωσορίζει τον φοιτητή, ενώ του δίνει και τη δυνατότητα να πάει κατευθείαν στην πιο δημοφιλή κατηγορία του φοιτητολογίου, τις Βαθμολογίες.

## 3.2.3 Στοιχεία



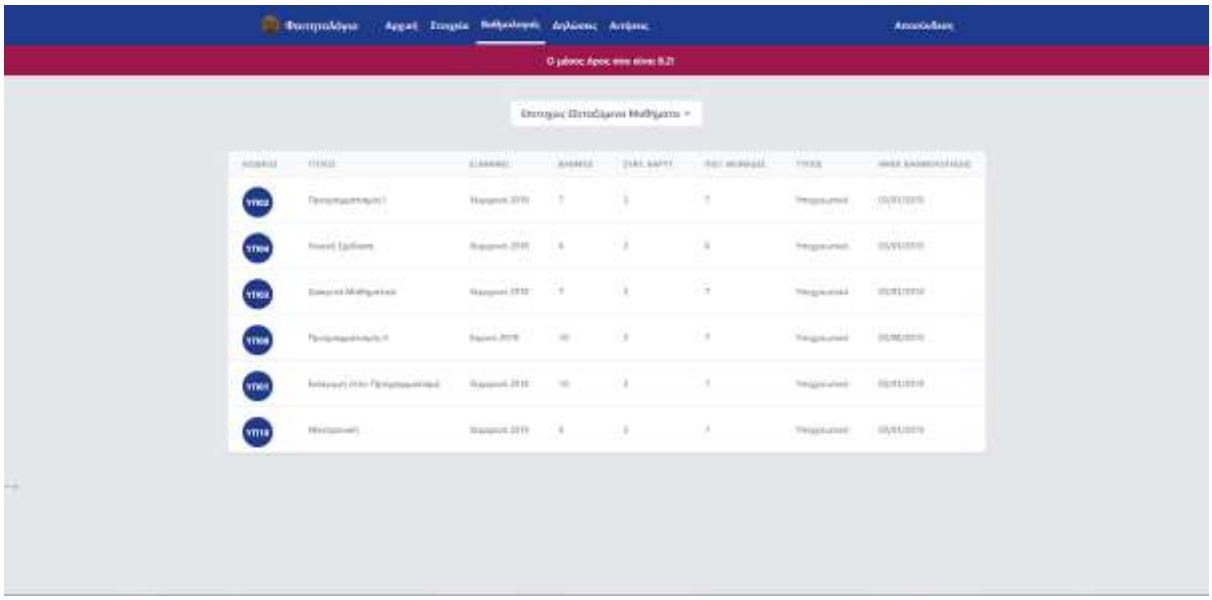
**Εικόνα 24:** Η σελίδα των στοιχείων στο interface των φοιτητών

Η σελίδα με τα στοιχεία (Εικόνα 24) περιέχει 2 καρτέλες με διάφορα στοιχεία και πληροφορίες για τον φοιτητή, τόσο προσωπικά στοιχεία, αλλά και τα στοιχεία σχετικά με τη φοίτηση στη σχολή. Τα στοιχεία αυτά δεν μπορούν να αλλάξουν από τους φοιτητές.

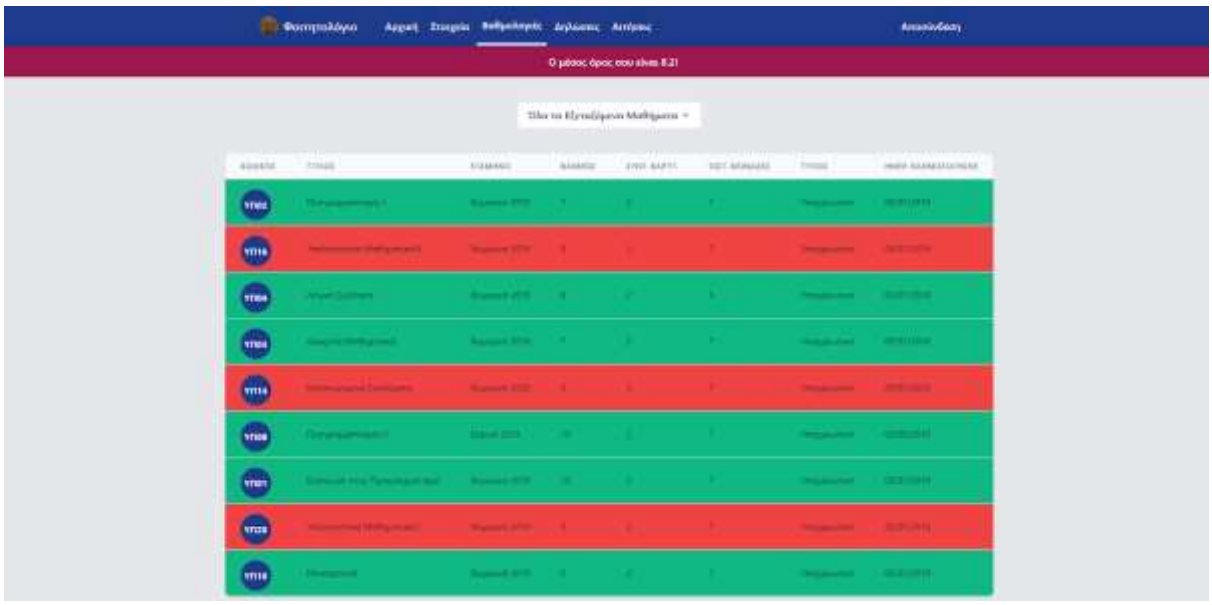
## 3.2.4 Βαθμολογίες

Στη σελίδα των Βαθμολογιών υπάρχουν 2 επιλογές για τους φοιτητές. Η προεπιλεγμένη κάθε φορά είναι ο πίνακας με τα «Επιτυχώς Εξεταζόμενα Μαθήματα», όπου ο μαθητής μπορεί να δει όλα τα μαθήματα που έχει περάσει και με τι βαθμό, μαζί με κάποιες άλλες πληροφορίες για τα μαθήματα.

Η άλλη επιλογή είναι να δει «Όλα τα Εξεταζόμενα Μαθήματα», όπου σε αυτό τον πίνακα μπορεί να δει όλες τις βαθμολογίες για μαθήματα που έχει δώσει, είτε είναι περασμένα είτε όχι, καθώς και φυσικά όλες τις πληροφορίες που χρειάζεται για κάθε βαθμολόγηση. Και στις 2 επιλογές, ο φοιτητής μπορεί να δει τον μέσο όρο του κάτω από τη γραμμή πλοήγησης. Στην Εικόνα 25 φαίνεται η προεπιλεγμένη σελίδα με τα επιτυχώς εξεταζόμενα μαθήματα, ενώ στην Εικόνα 26 βλέπουμε τη σελίδα με όλα τα εξεταζόμενα μαθήματα.



**Εικόνα 25:** Τα επιτυχή μαθήματα στη σελίδα των βαθμολογιών στο interface των φοιτητών

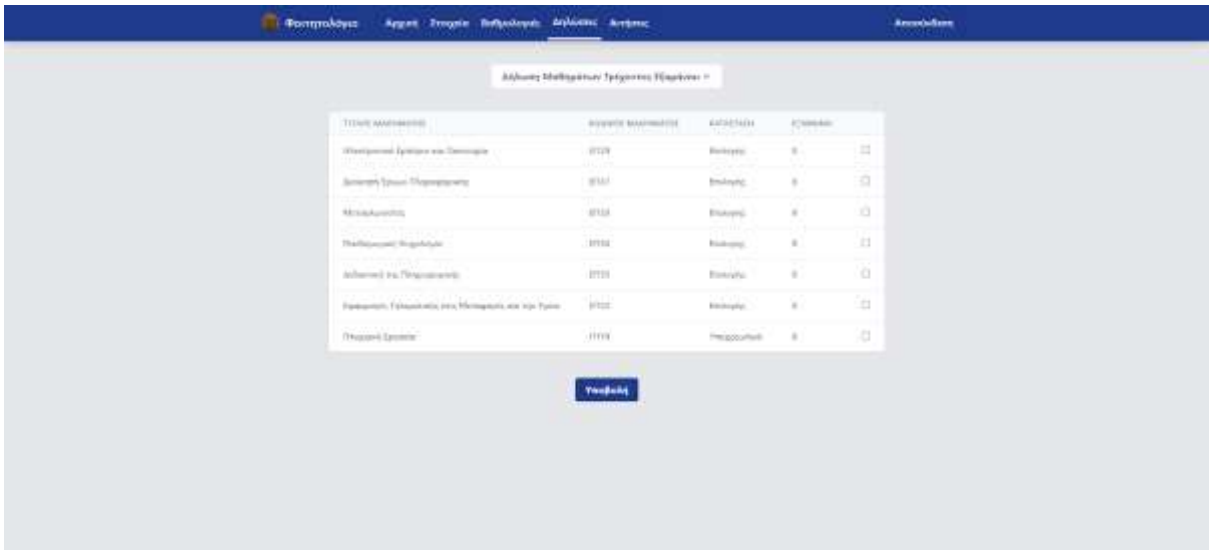


**Εικόνα 26:** Όλα τα εξεταζόμενα μαθήματα στη σελίδα των βαθμολογιών στο interface των φοιτητών

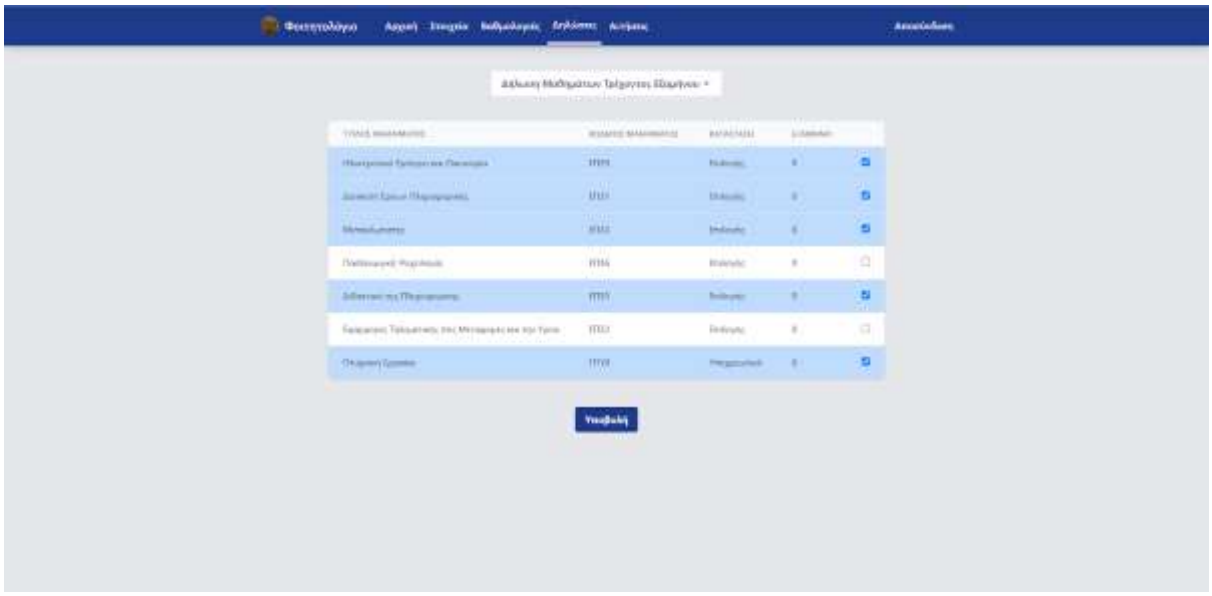
## 3.2.5 Δηλώσεις

Η σελίδα των Δηλώσεων, όπως και αυτή των Βαθμολογιών, έχει 2 επιλογές. Η προεπιλεγμένη είναι αυτή της δήλωσης μαθημάτων (Εικόνα 27 και Εικόνα 28) για το τρέχον εξάμηνο. Εκεί οι φοιτητές μπορούν να δουν σε έναν πίνακα όλα τα μαθήματα που μπορούν να δηλώσουν, να επιλέξουν αυτά που θέλουν και να κάνουν υποβολή. Όταν κλείνει η διορία για τη δήλωση μαθημάτων, οι φοιτητές βλέπουν ένα ενημερωτικό μήνυμα ότι η διορία έχει λήξει. Πριν γίνει η υποβολή των αιτήσεων γίνεται επαλήθευση από την εφαρμογή για την εγκυρότητα της δήλωσης και αν δεν είναι έγκυρη (πχ. έχουν δηλωθεί παραπάνω μαθήματα επιλογής από ό,τι επιτρέπεται) τότε βγαίνει ενημερωτικό μήνυμα για να ξαναγίνει σωστά η δήλωση.

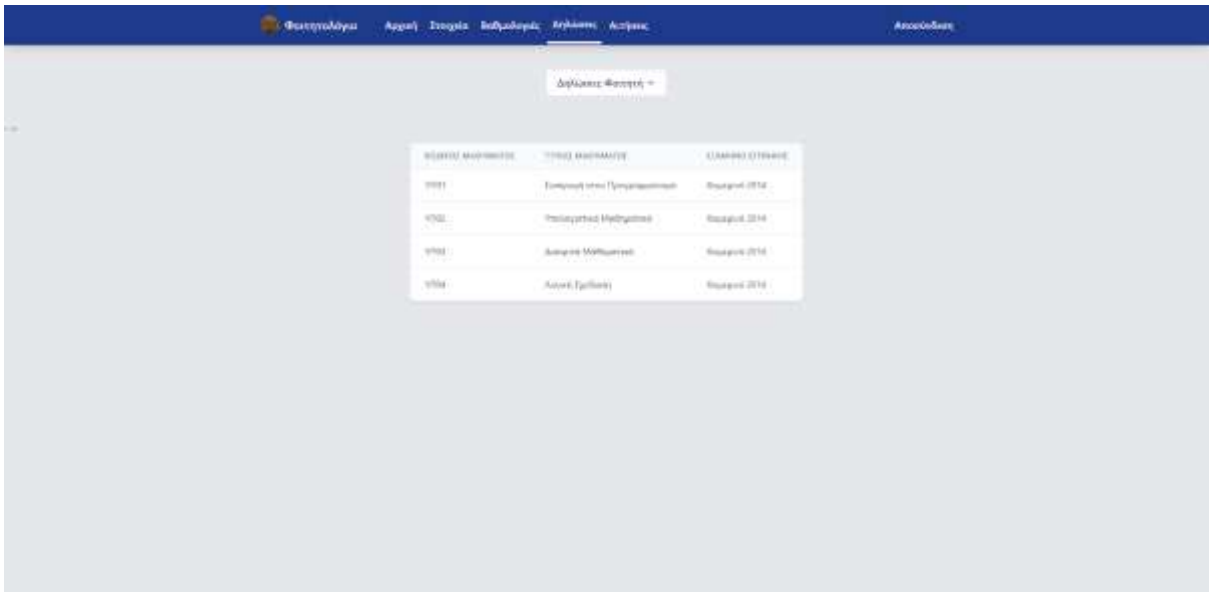
Η άλλη επιλογή είναι να δουν όλες τις δηλώσεις που έχουν κάνει κατά τη διάρκεια φοίτησης τους (Εικόνα 29) και είναι καθαρά για ενημερωτικούς σκοπούς.



**Εικόνα 27:** Η σελίδα δήλωσης μαθημάτων στο interface των φοιτητών



**Εικόνα 28:** Η δήλωση μαθημάτων στο interface των φοιτητών

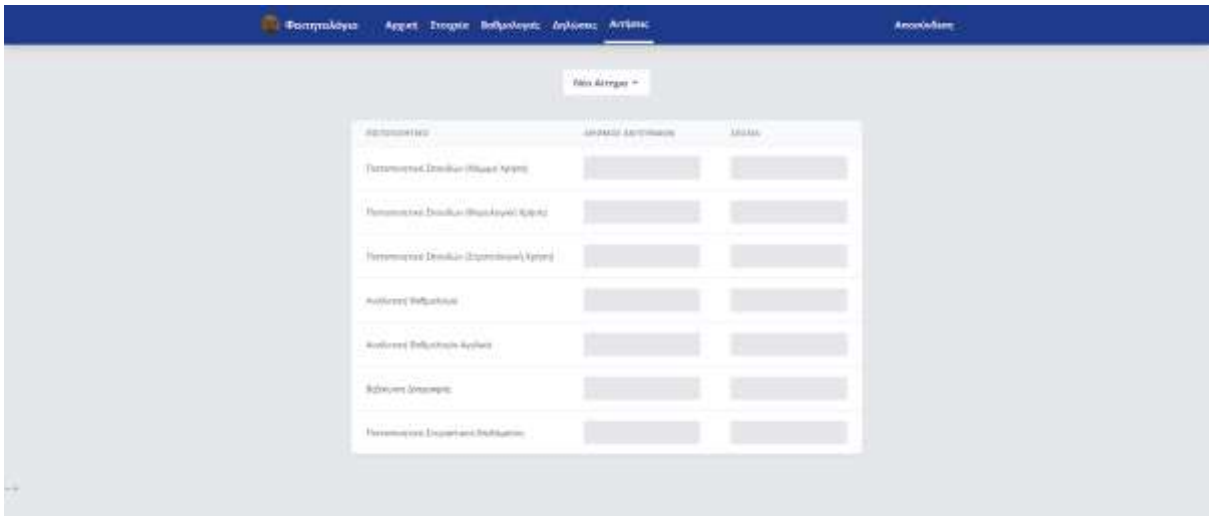


**Εικόνα 29:** Η σελίδα με όλες τις δηλώσεις μαθημάτων στο interface των φοιτητών

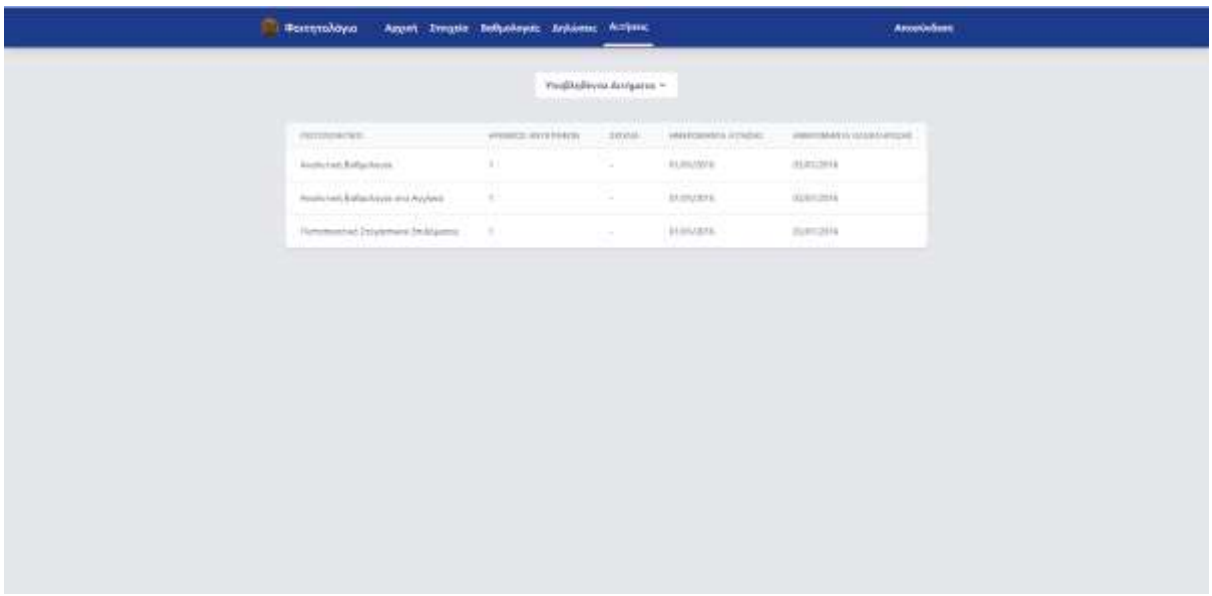
## 3.2.6 Αιτήσεις

Η σελίδα των Αιτήσεων, όπως και οι 2 προηγούμενες, προσφέρει δύο επιλογές στους φοιτητές. Η πρώτη και προεπιλεγμένη επιλογή (Εικόνα 30) είναι η φόρμα των διαφόρων αιτήσεων που μπορούν να κάνουν οι φοιτητές προς τη γραμματεία. Μέσα από το φόρμα έχουν τη δυνατότητα να κάνουν αίτηση για διάφορα έγγραφα όπως πιστοποιητικά σπουδών, αναλυτικές βαθμολογίες κτλ., έχοντας τη δυνατότητα να ζητήσουν ακριβώς πόσα αντίγραφα χρειάζονται ή να προσθέσουν κάποιο σχόλιο για τη γραμματεία. Για να μη γίνεται κατάχρηση του συστήματος, η φόρμα θέτει όρια για το πόσα αντίγραφα μπορούν να εκδοθούν.

Η δεύτερη επιλογή (Εικόνα 31) είναι ένας ενημερωτικός πίνακας με τα διάφορα αιτήματα που έχουν κάνει οι φοιτητές κατά τη διάρκεια των σπουδών τους, με πληροφορίες για το πότε δημιουργήθηκε το αίτημα αλλά και πότε ολοκληρώθηκε η διαδικασία από τη γραμματεία.



**Εικόνα 30:** Η σελίδα υποβολής αιτήσεων στο interface των φοιτητών



**Εικόνα 31:** Η σελίδα με όλες τις αιτήσεις στο interface των φοιτητών

## 3.3 Interface γραμματείας

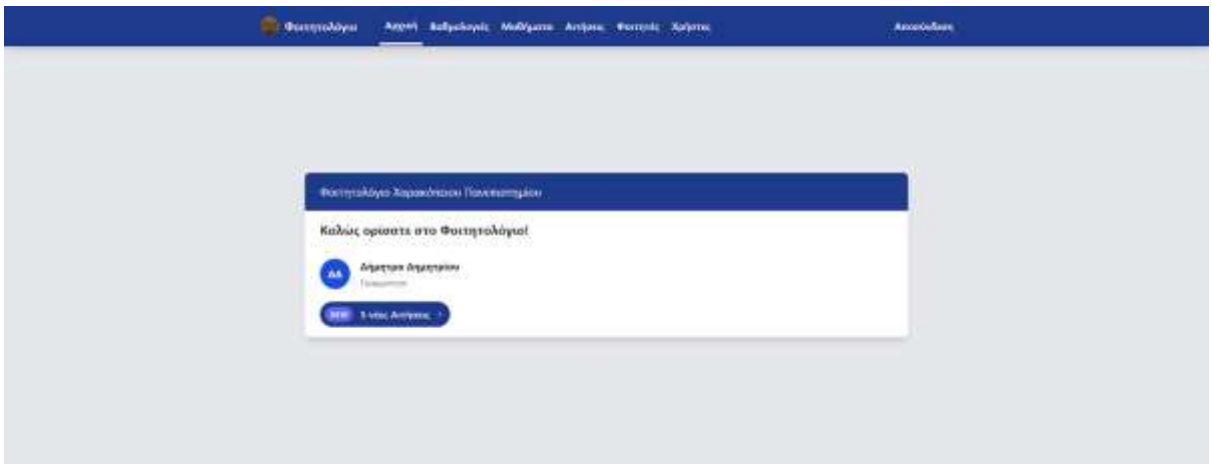
## 3.3.1 Γραμμή πλοήγησης

Η γραμμή πλοήγησης της γραμματείας (Εικόνα 32) είναι λίγο διαφορετική από αυτής των φοιτητών. Εκτός από την «Αρχική Σελίδα», τις «Βαθμολογίες» και τις «Αιτήσεις», οι υπόλοιπες επιλογές είναι διαφορετικές, ώστε να μπορούν να εξυπηρετούνται οι διαφορετικές λειτουργίες που χρειάζεται η γραμματεία.


**Εικόνα 32:** Η γραμμή πλοήγησης στο interface της γραμματείας

## 3.3.2 Αρχική σελίδα

Η αρχική σελίδα της γραμματείας (Εικόνα 33) είναι αρκετά παρόμοια με αυτή των φοιτητών, με τη μόνη διαφορά ότι αντί να υπάρχει το κουμπί που σε πάει κατευθείαν στις βαθμολογίες, υπάρχει μια ειδοποίηση με τις εκκρεμείς αιτήσεις που έχουν κάνει οι φοιτητές, ενώ αν πατήσουν πάνω στην ειδοποίηση μεταφέρονται στη σελίδα των «Αιτήσεων». Κάθε φορά που ένας λογαριασμός από τη γραμματεία πηγαίνει στην αρχική σελίδα, η ειδοποίηση θα είναι πάντα ανανεωμένη με τις αιτήσεις που περιμένουν ολοκλήρωση.

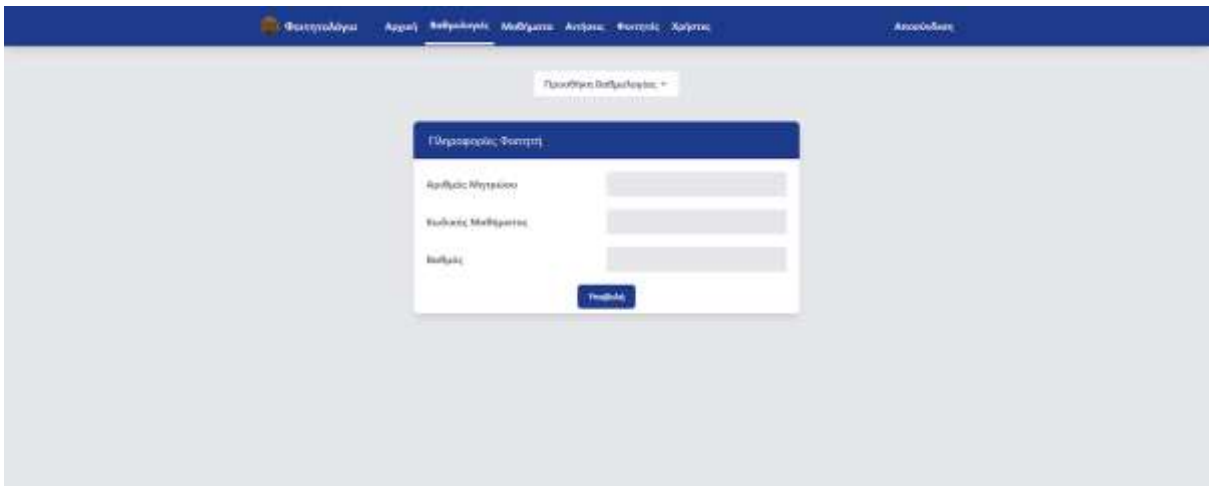



**Εικόνα 33:** Η αρχική σελίδα στο interface της γραμματείας

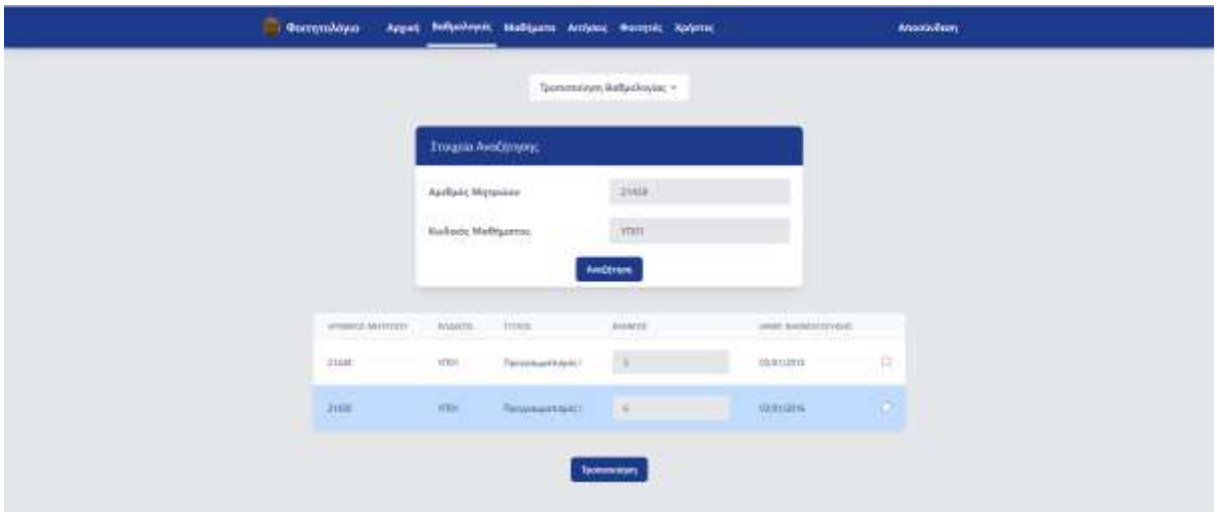
## 3.3.3 Βαθμολογίες

Η σελίδα των «Βαθμολογιών», όπως και οι περισσότερες σελίδες στο Interface της γραμματείας, έχουν παρόμοια λειτουργικότητες, και αυτές είναι η Προσθήκη μιας νέας βαθμολογίας (Εικόνα 34), η τροποποίηση μιας υπάρχουσας (Εικόνα 35) και η διαγραφή μιας υπάρχουσας (Εικόνα 36). Η προεπιλεγμένη επιλογή όταν εισέρχονται στη σελίδα των Βαθμολογιών είναι η Προσθήκη μιας νέας βαθμολογίας.

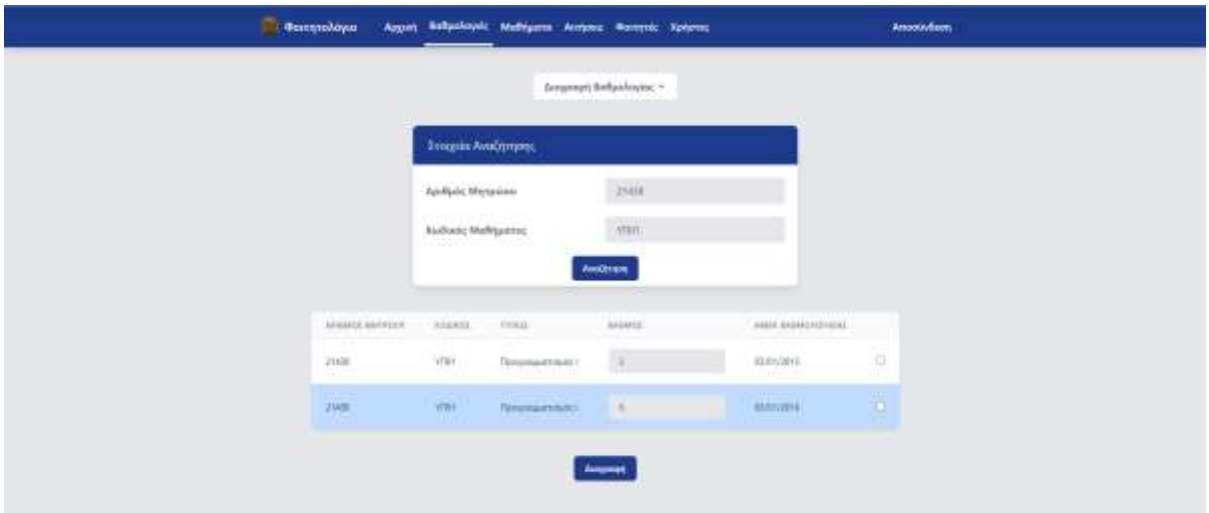
Για την προσθήκη μιας βαθμολογίας απαιτείται η συμπλήρωση του αριθμού μητρώου του φοιτητή, ο κωδικός του μαθήματος και ο βαθμός. Για την τροποποίηση μιας βαθμολογίας απαιτείται ο αριθμός μητρώου και ο κωδικός του μαθήματος, και γίνεται αναζήτηση στη βάση δεδομένων, γιατί πιθανόν να υπάρχουν πολλές βαθμολογίες για έναν συγκεκριμένο φοιτητή πάνω σε ένα συγκεκριμένο μάθημα. Όταν η αναζήτηση τελειώσει, θα γυρίσει τα αποτελέσματα στο frontend και η γραμματεία θα μπορεί να επιλέξει ποια ακριβώς βαθμολογία θέλει να τροποποιήσει. Αντίστοιχα και για τη διαγραφή, από το backend θα επιστρέψουν όλες οι βαθμολογίες που αντιστοιχούν σε αυτόν τον αριθμό μητρώου και κωδικό μαθήματος, και η γραμματεία θα μπορεί να διαγράψει αυτόν που θέλει.



**Εικόνα 34:** Προσθήκη βαθμολογίας στο interface της γραμματείας



**Εικόνα 35:** Τροποποίηση βαθμολογίας στο interface της γραμματείας

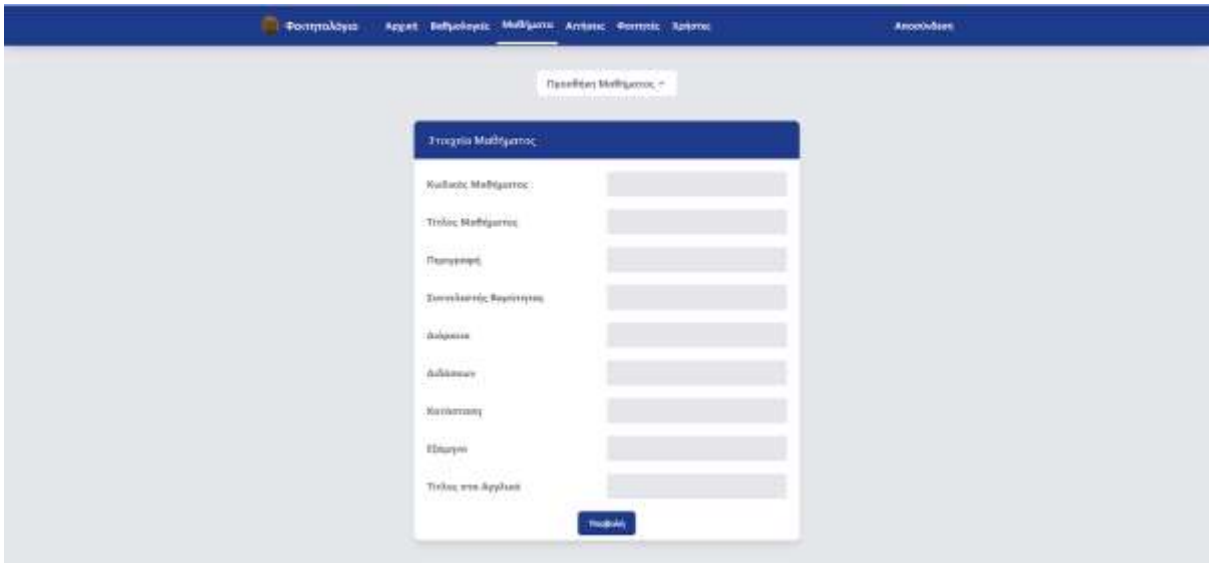


**Εικόνα 36:** Διαγραφή βαθμολογίας στο interface της γραμματείας

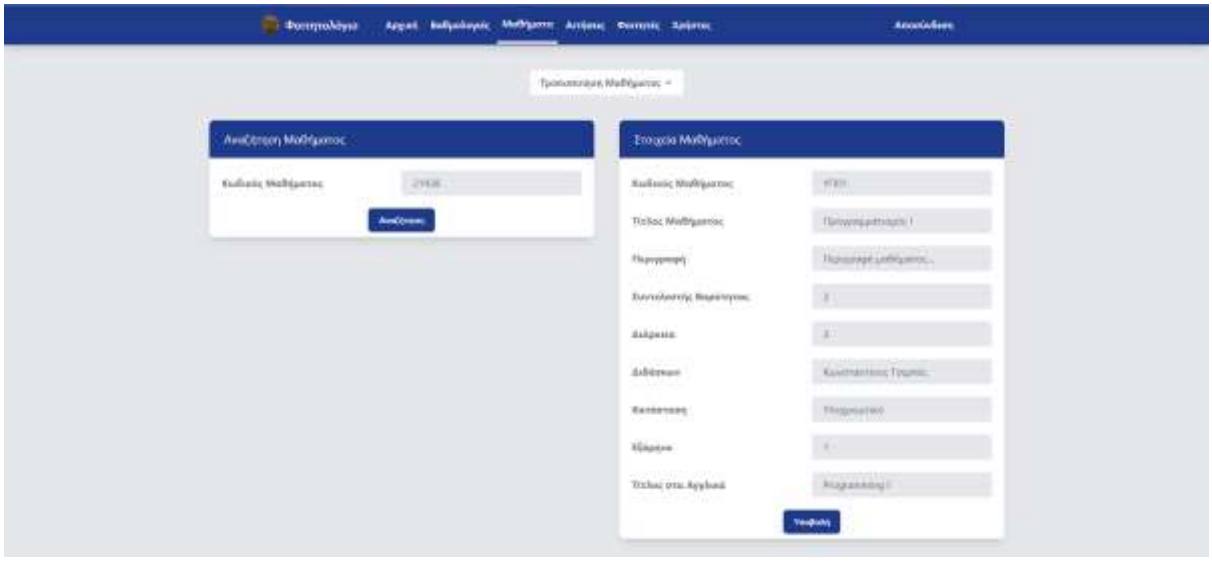
### 3.3.4 Μαθήματα

Η σελίδα των Μαθημάτων είναι παρόμοια με αυτή των Βαθμών, και δίνει τη δυνατότητα στη γραμματεία να προσθέσει (Εικόνα 37), να τροποποιήσει (Εικόνα 38), και να διαγράψει (Εικόνα 39), τα μαθήματα που διδάσκονται στη σχολή. Η προεπιλεγμένη επιλογή είναι αυτή της προσθήκης ενός μαθήματος. Για την προσθήκη ενός μαθήματος πρέπει να συμπληρωθούν όλα τα στοιχεία που απαιτούνται όπως Κωδικός Μαθήματος, Τίτλος, ECTS, Περιγραφή κτλ.

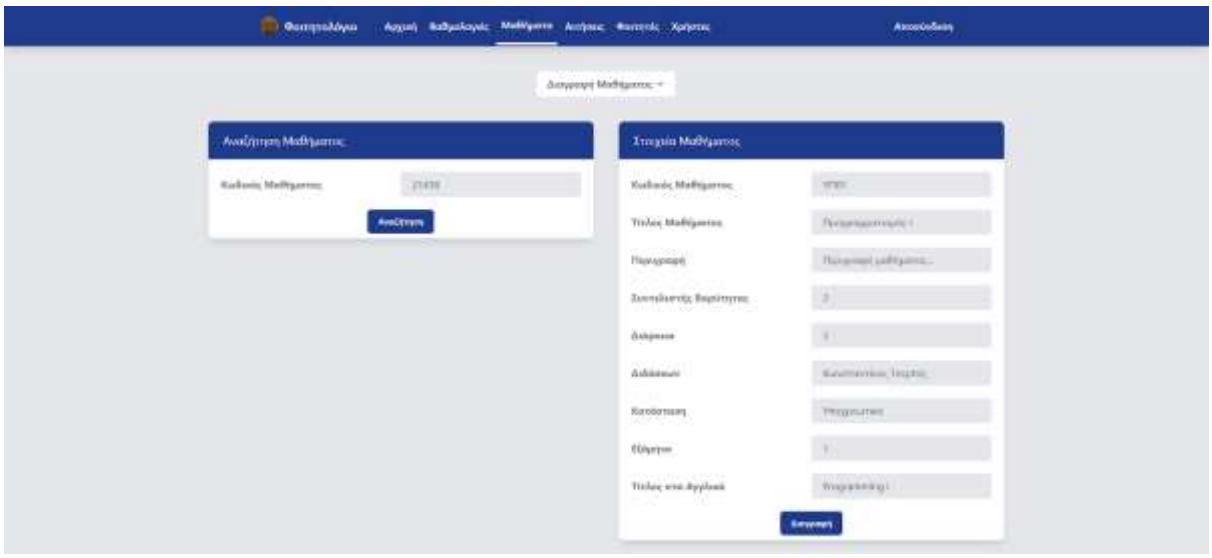
Για την τροποποίηση ενός μαθήματος η γραμματεία αναζητά το μάθημα με τον κωδικό του και στη συνέχεια εμφανίζονται όλα τα στοιχεία του μαθήματος με τη δυνατότητα αλλαγής τους και υποβολής εκ νέου των νέων στοιχείων. Για τη διαγραφή ενός μαθήματος, πάλι η γραμματεία αναζητεί το μάθημα μέσω του κωδικού του, αν το μάθημα βρεθεί τότε η γραμματεία μπορεί να το επιλέξει για διαγραφή.



**Εικόνα 37:** Προσθήκη νέου μαθήματος στο interface της γραμματείας



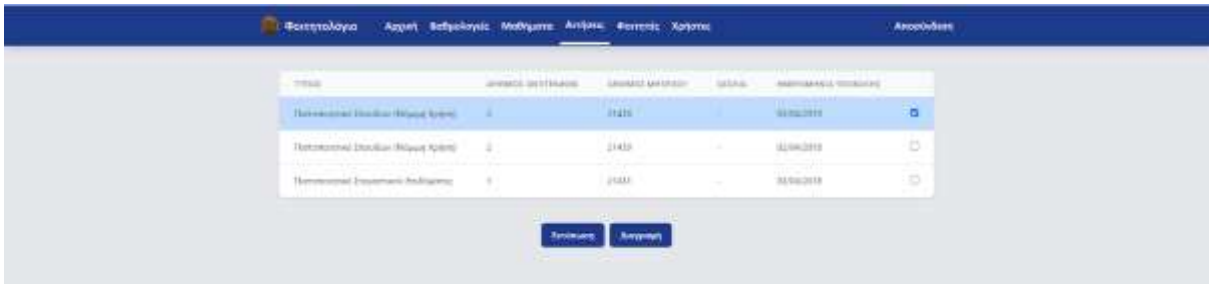
**Εικόνα 38:** Τροποποίηση μαθήματος στο interface της γραμματείας



**Εικόνα 39:** Διαγραφή μαθήματος στο interface της γραμματείας

## 3.3.5 Αιτήσεις

Η σελίδα των Αιτήσεων περιέχει έναν πίνακα με όλα τα εκκρεμεί αιτήματα πιστοποιητικών των φοιτητών (Εικόνα 40). Η γραμματεία μπορεί να επιλέξει ένα αίτημα από τον πίνακα και πατήσει «Εκτύπωση» το οποίο ανοίγει το παράθυρο εκτύπωσης του browser, και αν γίνει επιτυχώς η εκτύπωση τότε η γραμματεία μπορεί να επιλέξει ξανά το αίτημα και να πατήσει «Διαγραφή». Αυτή η ενέργεια θα αλλάξει την κατάσταση της αίτησης στη βάση δεδομένων σε «ολοκληρωμένο» και θα το αφαιρέσει από τον πίνακα με τις ανολοκλήρωτες αιτήσεις.

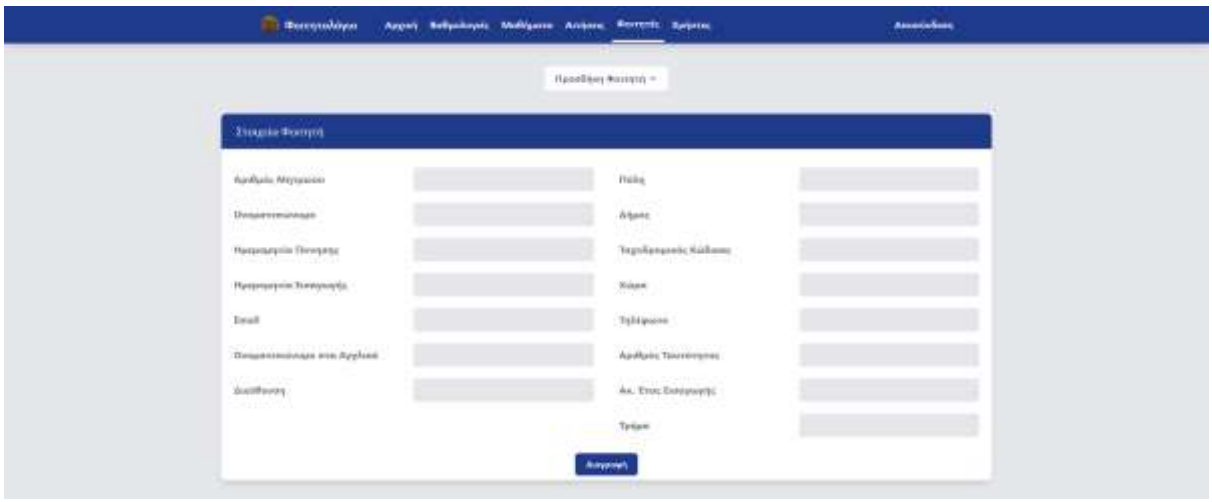


**Εικόνα 40:** Σελίδα αιτήσεων στο interface της γραμματείας

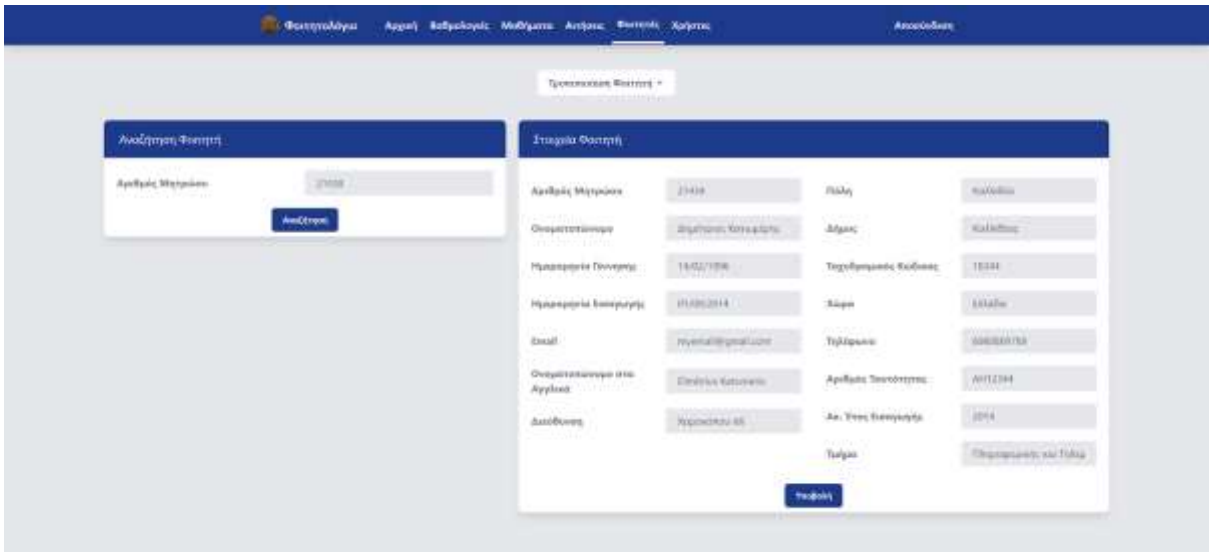
## 3.3.6 Φοιτητές

Στη συγκεκριμένη σελίδα, η γραμματεία έχει τη δυνατότητα να προσθέτει, τροποποιεί αλλά και να διαγράφει μαθητές από τη βάση δεδομένων του συστήματος. Η προεπιλεγμένη επιλογή όταν εισέρχεται στη σελίδα είναι η Προσθήκη ενός νέου φοιτητή (Εικόνα 41). Εκεί συμπληρώνονται τα διάφορα απαραίτητα στοιχεία και γίνεται υποβολή.

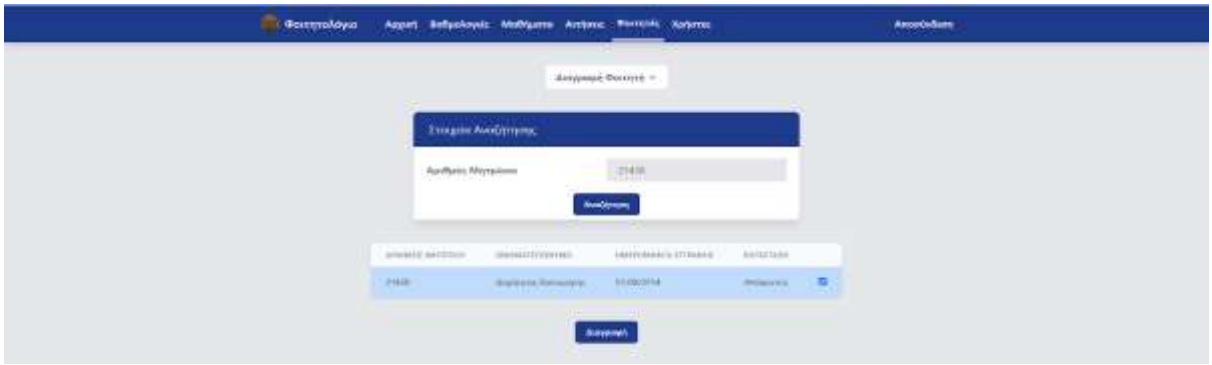
Στην τροποποίηση (Εικόνα 42), η γραμματεία αναζητεί με τον Αριθμό Μητρώου των φοιτητή για τον οποίο θέλει να κάνει αλλαγές, και αν αυτός υπάρχει, επιστρέφεται μια φόρμα με τα τωρινά στοιχεία του φοιτητή και η γραμματεία μπορεί να κάνει αλλαγές και να πατήσει «Υποβολή». Παρόμοια και για τη Διαγραφή (Εικόνα 43), η γραμματεία αναζητεί τον φοιτητή μέσω του Αριθμού Μητρώου του, αν αυτός υπάρχει εμφανίζεται ένα μικρός πίνακας με τις απαραίτητες πληροφορίες και η γραμματεία μπορεί να τον επιλέξει για διαγραφή.



**Εικόνα 41:** Προσθήκη μαθητή στο interface της γραμματείας



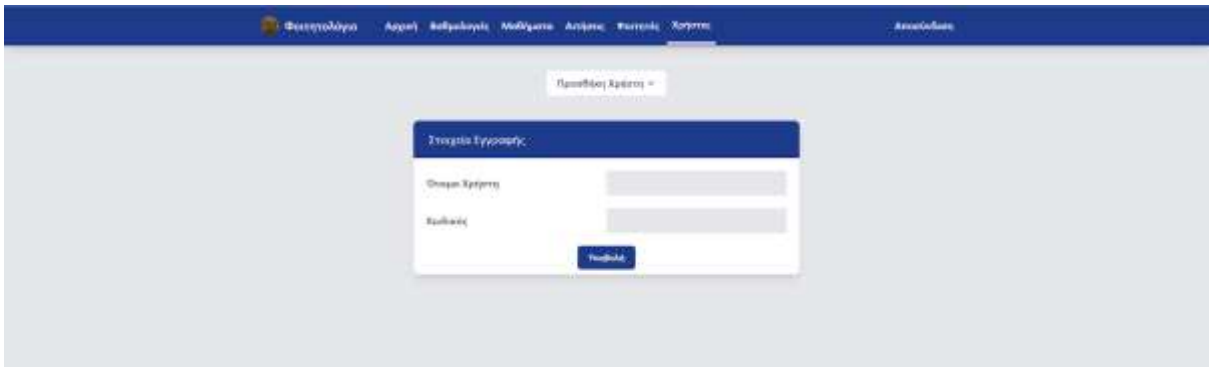
**Εικόνα 42:** Τροποποίηση μαθητή στο interface της γραμματείας



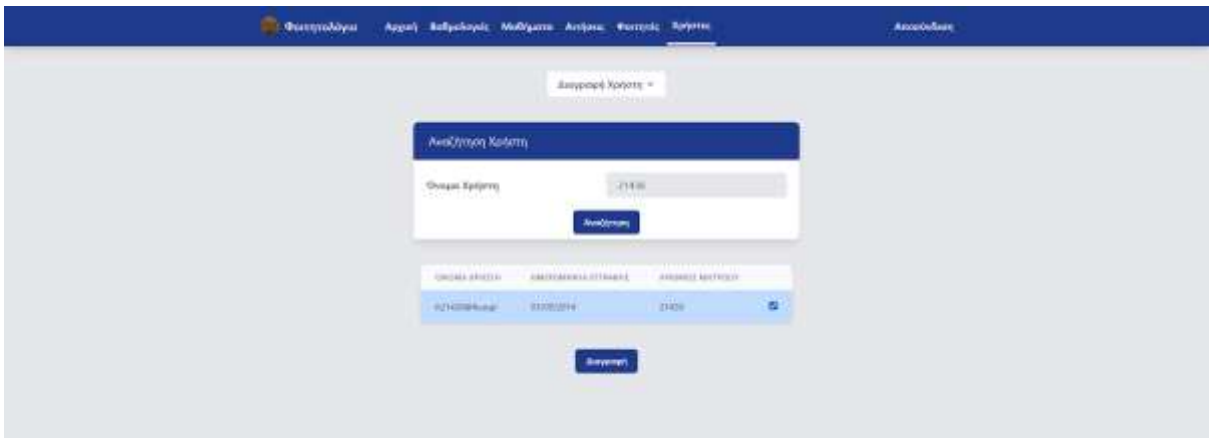
**Εικόνα 43:** Διαγραφή μαθητή στο interface της γραμματείας

### 3.3.7 Χρήστες

Η τελευταία σελίδα στο interface της γραμματείας είναι αυτή της διαχείρισης των χρηστών στην εφαρμογή. Εκεί η γραμματεία έχει τη δυνατότητα να προσθέτει νέους λογαριασμούς (Εικόνα 44) στην εφαρμογή αλλά και να διαγράφει χρήστες. Στην προσθήκη νέου χρήστη, το μόνο που χρειάζεται είναι η συμπλήρωση του ονόματος του χρήστη και του κωδικού. Στη διαγραφή (Εικόνα 45), η γραμματεία αναζητεί με βάση το όνομα χρήστη και αν ο χρήστης βρεθεί, μπορεί να τον επιλέξει για διαγραφή.

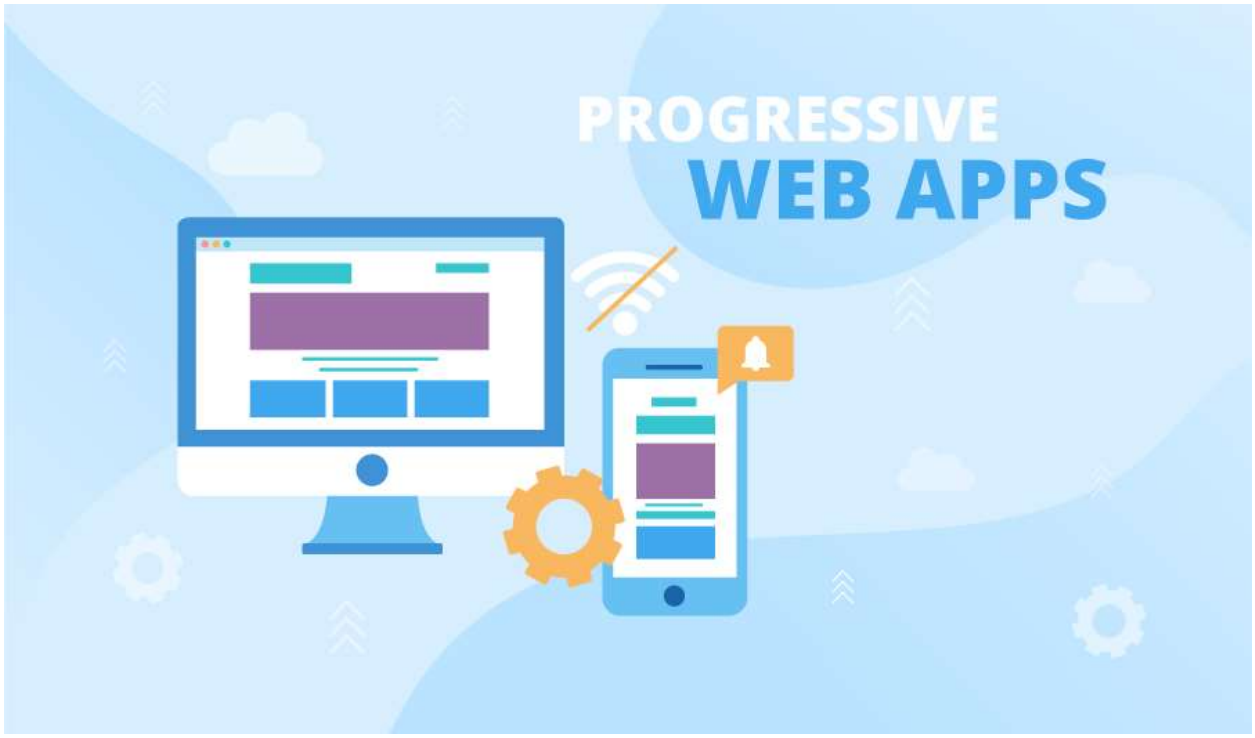


**Εικόνα 44:** Προσθήκη χρήστη στο interface της γραμματείας



**Εικόνα 45:** Διαγραφή χρήστη στο interface της γραμματείας

# ΚΕΦΑΛΑΙΟ 4
# ΤΟ ΦΟΙΤΗΤΟΛΟΓΙΟ ΩΣ PROGRESSIVE WEB APP

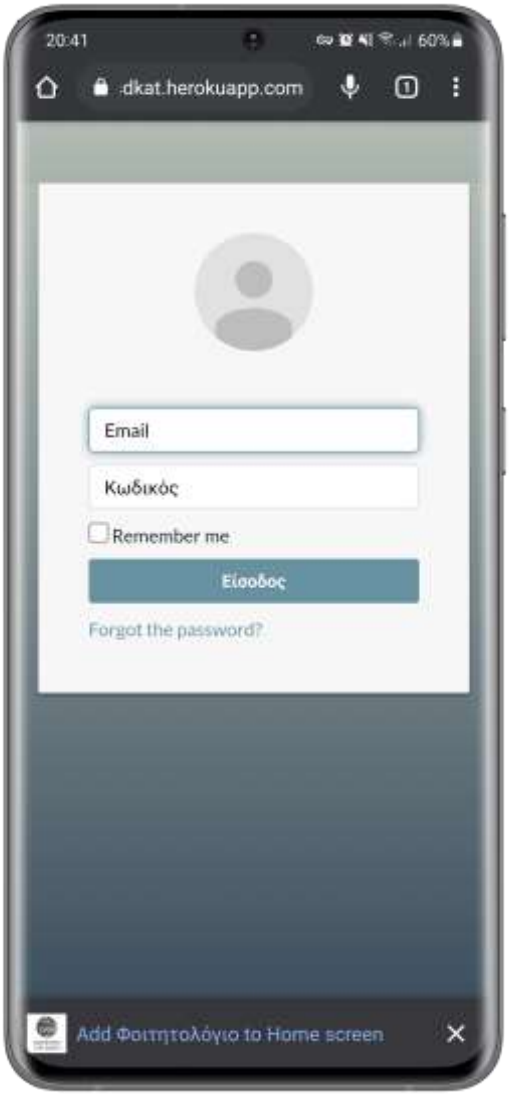


## 4.1 Βασικά χαρακτηριστικά των PWA στο Φοιτητολόγιο

Όπως έχει αναφερθεί και πιο πάνω, οι Progressive Web Apps έχουν κάποια χαρακτηριστικά που τις κάνουν να ξεχωρίζουν από τις απλές Web εφαρμογές. Κάποια από αυτά τα χαρακτηριστικά, που είχαν πρακτική αξία για την εφαρμογή ενός φοιτητολογίου έχουν αναπτυχθεί για την παρούσα πτυχιακή και περιγράφονται αναλυτικά παρακάτω.

### 4.1.1 Διαθέσιμο για κατέβασμα από τον browser

Όπως έχει αναφερθεί και πιο πάνω, οι Progressive Web Apps έχουν κάποια χαρακτηριστικά που τις κάνουν να ξεχωρίζουν από τις απλές Web εφαρμογές. Ένα πολύ σημαντικό feature, που μάλιστα είναι και μία από τις προϋποθέσεις να θεωρηθεί μια εφαρμογή PWA, είναι να εμφανίζει ένα μήνυμα στον χρήστη για να εγκαταστήσει την εφαρμογή στην αρχική οθόνη του κινητού του, όπως βλέπουμε και στην Εικόνα 46.




**Εικόνα 46:** Εμφάνιση μηνύματος για εγκατάσταση της εφαρμογής

Στην παραπάνω εικόνα βλέπουμε το banner που εμφάνισε ο browser (Chrome) όταν μπήκα στην εφαρμογή. Για να μπορέσει να βγει αυτό το μήνυμα υπήρχαν κάποιες προϋποθέσεις [8]:

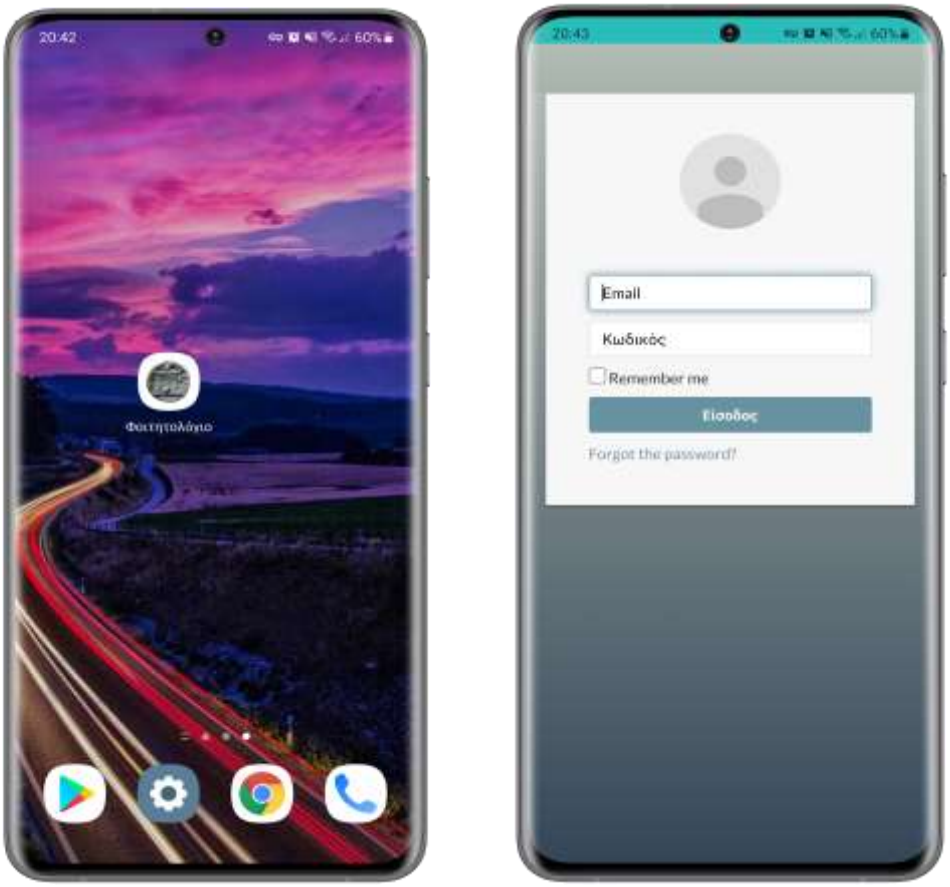
- Η εφαρμογή να «σερβίρεται» μέσω HTTPS, μιας και το διαδίκτυο γίνεται όλο και πιο ασφαλές και πολλές καινούριες τεχνολογίες και χαρακτηριστικά θα δουλεύουν μόνο σε ένα ασφαλές περιβάλλον
- Η εφαρμογή θα πρέπει να έχει το manifest αρχείο με όλα τα πεδία σωστά συμπληρωμένα και να είναι linked στο head του HTML
- Να υπάρχει το κατάλληλο εικονίδιο που θα φαίνεται στην αρχική σελίδα του κινητού
- Ο Chrome θέλει επιπλέον να είναι εγγεγραμμένος και ένας service worker, ώστε να μπορεί να δουλεύει και χωρίς σύνδεση στο διαδίκτυο

Από την έκδοση 68 και μετά του Chrome, αυτό το μπανεράκι εμφανίζεται χωρίς κάποιον επιπλέον κώδικα στην εφαρμογή, αρκεί να καλύπτονται οι προϋποθέσεις. Αν ο χρήστης πατήσει το «Χ», τότε αυτό το μπανεράκι δεν θα εμφανιστεί ξανά για περίπου 3 μήνες, εκτός αν ο χρήστης σβήσει τα cookies του και καθαρίσει την cache.

Αν ο χρήστης πατήσει πάνω στο μπανεράκι, τότε εμφανίζεται ένα επιπλέον μήνυμα για επιβεβαίωση της εγκατάστασης, όπως βλέπουμε την Εικόνα 47.


**Εικόνας 47:** Εμφάνιση μηνύματος επιβεβαίωσης εγκατάσταση της εφαρμογής

Αν ο χρήστης πατήσει «Install», τότε η εφαρμογή θα εγκατασταθεί στο κινητό και ο χρήστης θα μπορεί να την έχει σε μία από τις αρχικές του οθόνες, σαν μια Native εφαρμογή από το Play Store και φυσικά θα μπορεί να την ανοίγει από εκεί, χωρίς να χρειάζεται να μπαίνει από κάποιο browser. Στην Εικόνα 48 βλέπουμε το εικονίδιο της Εφαρμογής στο κινητό, ενώ στην Εικόνα 49 φαίνεται η σελίδα login, μέσα από την εφαρμογή.




**Εικόνα 48:** Η εφαρμογή του φοιτητολογίου εγκατεστημένη στο κινητό
**Εικόνα 49:** Πρόσβαση στην εγκατεστημένη εφαρμογή

## 4.1.2 Λειτουργία εκτός σύνδεσης

Ένα ακόμα πολύ σημαντικό χαρακτηριστικό που έχουν οι PWA, είναι ότι μπορούν να χρησιμοποιήσουν τους service workers ώστε να κρατήσουν την εφαρμογή λειτουργική, ως ένα σημείο, ακόμα και όταν δεν υπάρχει σύνδεση στο διαδίκτυο, κάτι που οι κοινές εφαρμογές δεν μπορούν να κάνουν, με αποτέλεσμα ο χρήστης να βλέπει το εκάστοτε error μήνυμα του browser του. Αυτό μπορεί να επιτευχθεί είτε τραβώντας πληροφορίες που είχαν αποθηκευτεί σε προηγούμενα sessions χρησιμοποιώντας την IndexedDB ή απλώς δείχνοντας μια απλή σελίδα HTML/CSS στο γενικό στυλ της εφαρμογής με ένα ενημερωτικό μήνυμα.

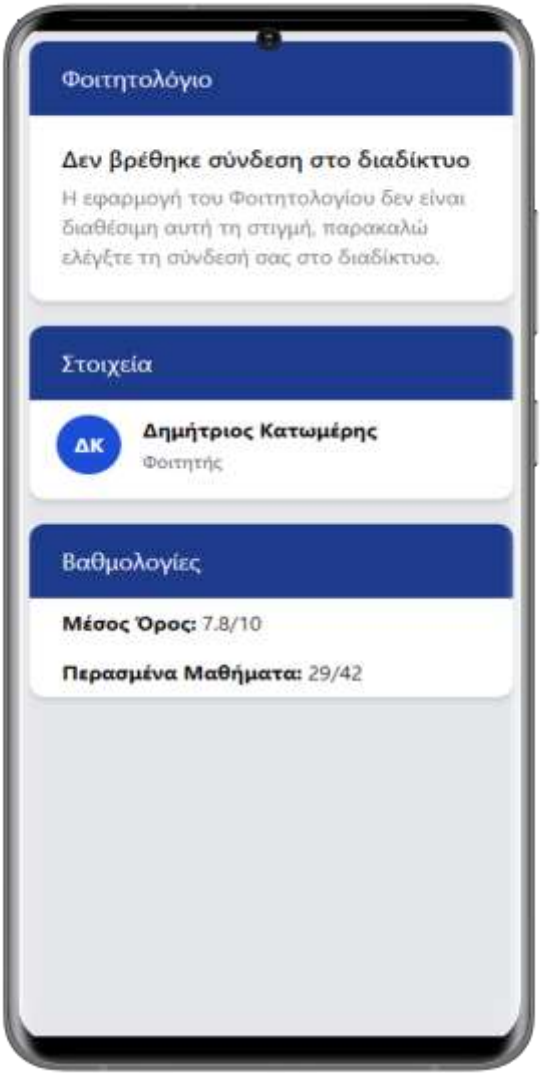
Στην προκειμένη περίπτωση επιλέχθηκε ο πρώτος τρόπος, σε έναν μικρό συνδυασμό με τον δεύτερο. Σε κάθε session ενός χρήστη στην εφαρμογή, αποθηκεύονται πληροφορίες στην IndexedDB τα οποία μετά ο service worker τραβάει και τα δείχνει μέσα σε κάρτες. Οι πληροφορίες που αποθηκεύονται είναι το όνομα και επίθετο του χρήστη, η ιδιότητα του

(φοιτητής ή γραμματεία), ο μέσος όρος του και το πόσα μαθήματα έχει περάσει, αν πρόκειται για χρήστη με ιδιότητα φοιτητή. Στην Εικόνα 50 βλέπουμε την offline σελίδα της εφαρμογής.

Η IndexedDB είναι μια NoSQL βάση και αποθηκεύει την πληροφορία ως κλειδί-τιμή σε ένα object. Η IndexedDB στην εφαρμογή μοιάζει με τον παρακάτω Πίνακα 1:

| # | Key (keypath 'id') | Value |
|---|---|---|
| 0 | 1234 | {name: "Δημήτριος Κατωμέρης", gpa: 7.8, role: "student", passedClasses: 32} |

**Πίνακας 1:** Οι πληροφορίες στην IndexedDB

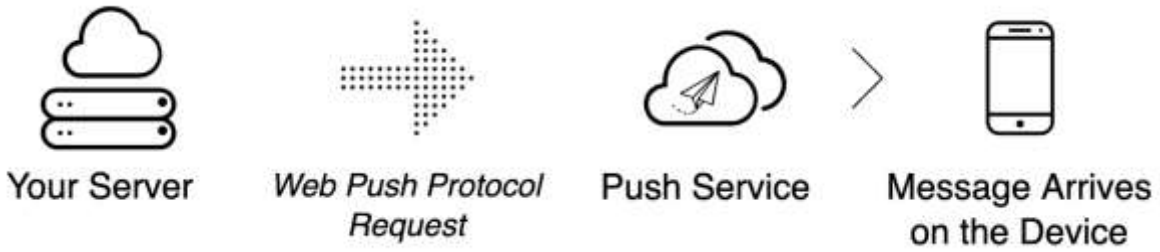



**Εικόνα 50:** Η offline σελίδα της εφαρμογής

### 4.1.3 Push Notifications

Ένα από τα πιο σημαντικά και δυνατά χαρακτηριστικά των Progressive Web App είναι τα Push Notifications, τα οποία είναι ένα πολύ χρήσιμο εργαλείο για πολλές διαφορετικές κατηγορίες εφαρμογών, αφού δίνει τη δυνατότητα στους δημιουργούς της εφαρμογής να ειδοποιήσουν τους χρήστες ανά πάσα στιγμή και αυτό έχει πολύ θετικά αποτελέσματα, Αρχικά, οι ειδοποιήσεις μπορούν να κινήσουν το ενδιαφέρον σε χρήστες που δεν χρησιμοποιούσαν πια την εφαρμογή, να τη χρησιμοποιήσουν ξανά, και όχι μονάχα αυτό, αλλά βοηθάει και να κρατήσουν τους χρήστες στην εφαρμογή. Ένα άλλο θετικό είναι ότι οι εφαρμογές μπορούν να στοχεύσουν σε ποιους ακριβώς χρήστες θέλουν να εμφανιστεί η ειδοποίηση και έτσι μπορούν να έχουν στοχευμένες ειδοποιήσεις για διάφορα γκρουπ χρηστών [9] [10].

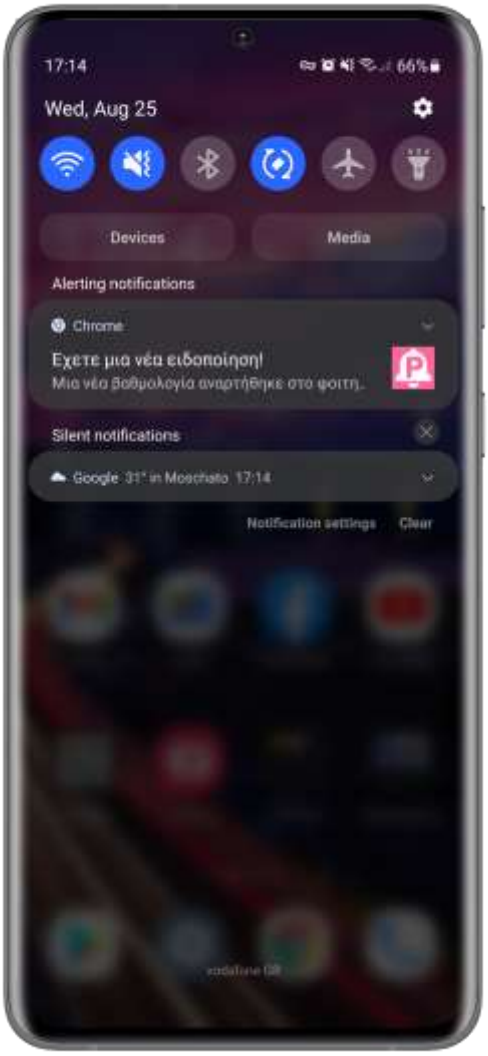
Οι ειδοποιήσεις των PWA εμφανισιακά είναι σαν όλες τις υπόλοιπες ειδοποιήσεις του τηλεφώνου του χρήστη και ο πιο συνηθισμένος τρόπος για να έρθει η ειδοποίηση είναι να σταλεί από τον server της εφαρμογής. Τα Push Notifications έχουν φτιαχτεί πάνω στο Notifications API και το Push API, με το πρώτο να δίνει τη δυνατότητα στην εφαρμογή να δείχνει την ειδοποίηση, ενώ το δεύτερο αφήνει τους service workers να διαχειρίζονται τα μηνύματα που στέλνονται από τον server, ακόμα και όταν η εφαρμογή δεν είναι ενεργή. Στην Εικόνα 51 βλέπουμε μια απεικόνιση του πρωτοκόλλου των Push Notifications [9] [10].




**Εικόνα 51:** Το πρωτόκολλο των Push Notifications

Για να μπορέσει μια εφαρμογή να στείλει Push Notifications, θα πρέπει να ακολουθήσει ορισμένα προαπαιτούμενα βήματα. Αρχικά, θα πρέπει να ζητήσει την άδεια του χρήστη, με τον χρήστη να πρέπει να αποδεχθεί. Σε αυτό το σημείο, αν ο χρήστης αποδεχτεί, τότε η εφαρμογή τον εγγράφει στις ειδοποιήσεις και θα μπορεί μετά να τους στέλνει [9] [10].

Στη δική μας περίπτωση, για το Φοιτητολόγιο, οι ειδοποιήσεις ναι μεν δεν ήταν κάτι πάρα πολύ σημαντικό, γιατί δεν υπάρχουν πολλά use cases, αλλά σίγουρα ήταν μια πολύ καλή προσθήκη σε σχέση με το τωρινό Φοιτητολόγιο, για το οποίο ήταν και στόχος να βελτιωθεί. Οι ειδοποιήσεις που στέλνονται στην εφαρμογή αφορούν μονάχα τους φοιτητές και είναι κάθε φορά που αναρτάται μια καινούρια βαθμολογία στο σύστημα από τη γραμματεία, όπως φαίνεται και στην Εικόνα 52, και όταν ένα αίτημα έχει ολοκληρωθεί από τη γραμματεία και είναι έτοιμο για παραλαβή.



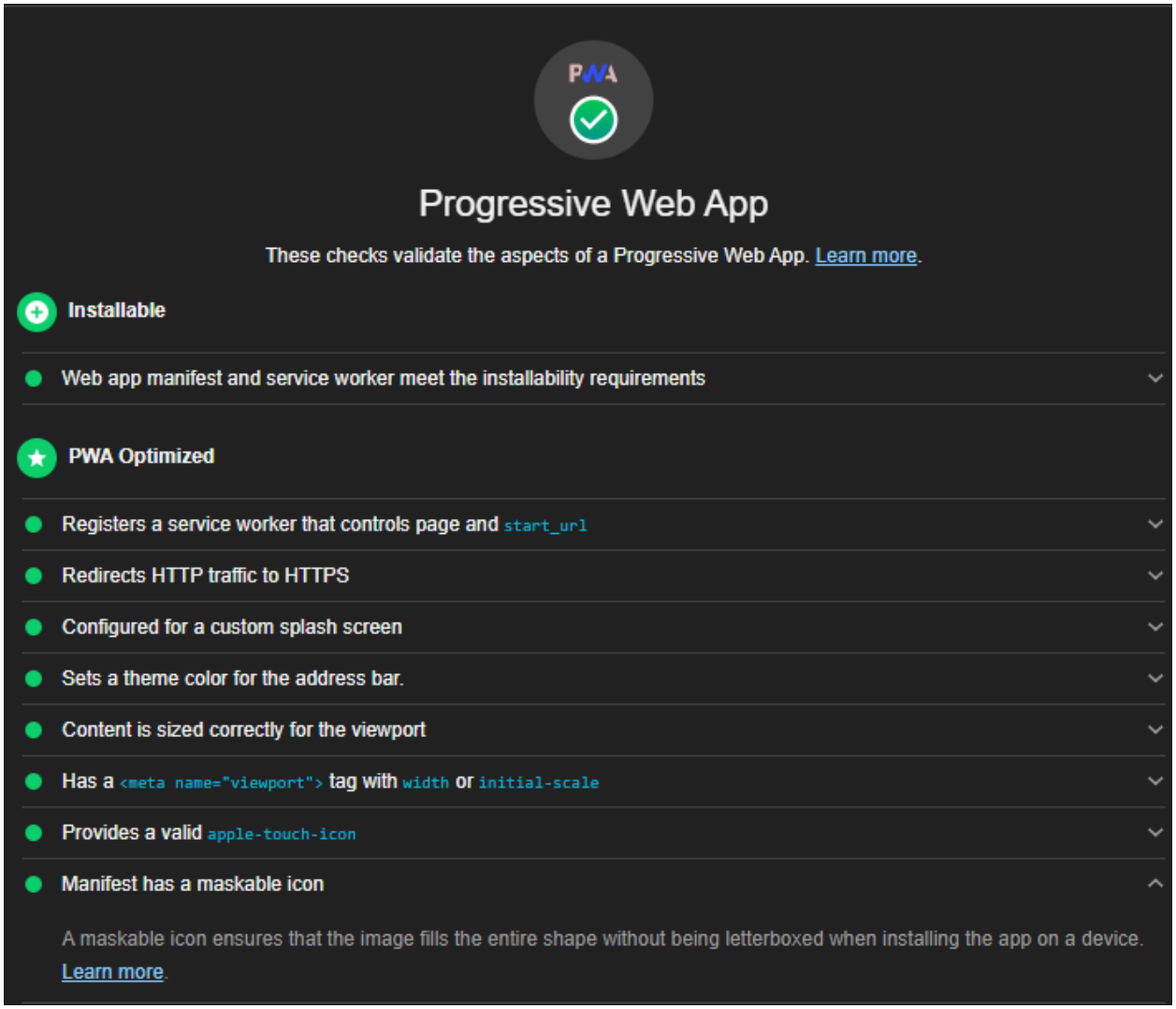

**Εικόνα 52:** Push Notification για μια νέα βαθμολογία

## 4.2 Κάλυψη προϋποθέσεων PWA στο Φοιτητολόγιο

Για να μπορεί να θεωρηθεί μια web εφαρμογή ως Progressive Web App θα πρέπει να πληροί κάποιες προϋποθέσεις. Σύμφωνα με το Lighthouse, η βασική προϋπόθεση είναι η εφαρμογή να μπορεί να εγκαταστείται. Αυτό επιτυγχάνεται κυρίως μέσω των service workers και του αρχείου manifest. Εκτός από αυτή την απαιτούμενη προϋπόθεση, υπάρχουν και κάποιες ακόμα που κάνουν την εφαρμογή πιο βελτιστοποιημένη [14]:

- Γίνεται εγγραφή του service worker
- Κάνει redirect την κίνηση από HTTP σε HTTPS
- Έχει ρυθμισμένη μια “splash screen”
- Έχει σεταρισμένο ένα χρώμα για την μπάρα αναζήτησης
- Το περιεχόμενο είναι στο κατάλληλο μέγεθος για τη συσκευή χρήσης
- Υπάρχει το tag <meta name=”viewport”> με width ή initial-scale
- Παρέχει ένα έγκυρο apple-touch-icon
- Το αρχείο manifest παρέχει ένα “maskable icon”

Αυτή η λίστα τσεκάρεται από το εργαλείο Lighthouse στα dev tools του Chrome και όπως μπορούμε να δούμε από την Εικόνα 53, η εφαρμογή του Φοιτητολογίου πληροί όλες τις προϋποθέσεις.




**Εικόνα 53:** Αποτελέσματα Lighthouse για τις προϋποθέσεις PWA

Εκτός από τους παραπάνω ελέγχους που κάνει αυτόματα το Lighthouse, το εργαλείο προτείνει ακόμα 3 πράγματα τα οποία θα πρέπει να ελεγχτούν χειροκίνητα. Αυτά είναι να μπορεί η εφαρμογή να τρέξει σε πολλούς διαφορετικούς browsers, κάθε σελίδα να έχει το δικό της URL και οι μεταβάσεις από σελίδα σε σελίδα να μην φαίνονται σαν να μπλοκάρουν το δίκτυο. Μετά από χειροκίνητο έλεγχο, το Φοιτητολόγιο φαίνεται ότι πληροί και αυτές τις απαιτήσεις.

Πέρα από τους ελέγχους του Lighthouse, υπάρχουν και κάποια ακόμα χαρακτηριστικά, μερικά παρόμοια με αυτά του Lighthouse, που καλό θα ήταν να ικανοποιούνται για το καλύτερο αποτέλεσμα. Μπορούν να χωρισθούν σε 2 κατηγορίες: τα βασικά χαρακτηριστικά ή αλλιώς τον «κορμό» [11] και σε αυτά που καλό θα ήταν να υπήρχαν και αναφέρονται παρακάτω.

## Βασικά χαρακτηριστικά

### Η εφαρμογή ξεκινάει γρήγορα και παραμένει γρήγορη

Η απόδοση της εφαρμογής είναι ίσως το πιο σημαντικό κομμάτι σε μία εφαρμογή, αφού τα αποδοτικά websites «κρατάνε» τους χρήστες τους για πολύ περισσότερο χρόνο. Οπότε η ταχύτητα της εφαρμογής είναι πολύ σημαντική για τους χρήστες και όσο πιο αργή είναι τόσο μεγαλύτερες πιθανότητες υπάρχουν ο χρήστης να φύγει από την εφαρμογή. Υπάρχουν αρκετοί τρόποι που μπορούν να βοηθήσουν στη βελτιστοποίηση της ταχύτητας της εφαρμογής, κατά την ανάπτυξη της. Οι φωτογραφίες και τα βίντεο παίρνουν χρόνο για να φορτώσουν, οπότε μερικά πράγματα που μπορούν να γίνουν είναι οι φωτογραφίες να έχουν το κατάλληλο φορμάτ, να έχουν σωστή συμπίεση, να είναι responsive και με τις σωστές διαστάσεις, ενώ πολύ σημαντικό είναι και η χρήση CDN και το “lazy-loading” των φωτογραφιών και των βίντεο.

Μερικά ακόμα πράγματα που μπορούν να γίνουν είναι η διαγραφή κώδικα που δεν χρησιμοποιείται και η CSS να γίνει minify. Ένα ακόμα στοιχείο που δημιουργεί προβλήματα στην ταχύτητα των εφαρμογών είναι τα 3rd party resources, όπως JavaScript βιβλιοθήκες, οπότε καλό είναι να αναγνωριστούν οι βιβλιοθήκες με κακή απόδοση και να βελτιστοποιηθούν.

### Η εφαρμογή μπορεί να δουλεύει σε κάθε browser

Μια ακόμα σημαντική προϋπόθεση είναι να μπορεί η εφαρμογή να δουλεύει το ίδιο καλά σε όλους τους browser, γιατί προπάντων παραμένει μια web εφαρμογή, παρόλο που είναι και PWA. Οπότε πριν η εφαρμογή εγκατασταθεί στην εφαρμογή, είναι σημαντικό να είναι προσβάσιμη και λειτουργική από όλους τους browser. Για αυτό είναι σημαντικό κατά την υλοποίηση της εφαρμογής να υπάρχει σαν στόχος η εφαρμογή να μπορεί να χρησιμοποιηθεί από όσο το δυνατό περισσότερους χρήστες, από διάφορους browser και συσκευές.

### Η εφαρμογή θα πρέπει να είναι responsive για όλα τα μεγέθη οθονών

Με τόσα διαφορετικά μεγέθη οθονών που κυκλοφορούν είναι πολύ σημαντικό οι εφαρμογές να φαίνονται το ίδιο σε όλες τις συσκευές και να προσφέρουν την ίδια λειτουργικότητα. Για να επιτευχθεί κάτι τέτοιο υπάρχουν διάφορες τεχνικές, με μία από αυτές, που αποκτά όλο και περισσότερη δημοτικότητα τα τελευταία χρόνια, είναι η υλοποίηση εφαρμογών mobile first, δηλαδή να φτιάχνεται το UI της εφαρμογής πρώτα για μεγέθη οθονών κινητών συσκευών, και έπειτα να γίνονται οι όποιες προσαρμογές για μεγαλύτερα μεγέθη.

### Η εφαρμογή πρέπει να έχει custom offline σελίδα

Οι χρήστες, γενικά, περιμένουν από τις εφαρμογές που έχουν εγκαταστήσει να δουλεύουν ανεξαρτήτως της σύνδεσής του χρήστη στο διαδίκτυο. Οι Native εφαρμογές ποτέ δεν δείχνουν μια κενή σελίδα όταν δεν υπάρχει σύνδεση, και το ίδιο θα πρέπει να ισχύει και για τις PWA.

Όταν υπάρχουν custom σελίδες και για σελίδες που δεν έχει «κασαριστεί» κάποια πληροφορία, αλλά και όταν ο χρήστης πάει να χρησιμοποιήσει κάποιο feature που απαραίτητα χρειάζεται πρόσβαση στο διαδίκτυο, βοηθάει να φανεί η εφαρμογή σαν να είναι «μέρος» της συσκευής που τρέχει, όπως μια Native εφαρμογή.

### Η εφαρμογή θα πρέπει να μπορεί να εγκατασταθεί

Τέλος, η τελευταία και ίσως πιο σημαντική προϋπόθεση και αυτό που κάνει τις PWA να διαφέρουν περισσότερο από τις απλές web εφαρμογές, είναι η εφαρμογή να μπορεί να εγκατασταθεί στη συσκευή του χρήστη μέσω ενός οποιουδήποτε browser. Τα θετικά αυτού του χαρακτηριστικού αλλά και πώς μπορεί να επιτευχθεί προγραμματιστικά έχουν αναφερθεί με πολλές λεπτομέρειες σε προηγούμενες ενότητες.

## Χαρακτηριστικά που καλό θα ήταν να υπήρχαν

### Προσφέρει κάποια λειτουργικότητα ακόμα και offline

Όπως έχει αναφερθεί και πιο αναλυτικά σε προηγούμενες ενότητες, ένα πολύ ιδιαίτερο και χρήσιμο χαρακτηριστικό των Progressive Web App είναι ότι διατηρούν κάποια λειτουργικότητα ακόμα και όταν ο χρήστης δεν έχει πρόσβαση στο διαδίκτυο. Όπως είδαμε και παραπάνω, το Φοιτητολόγιο παρέχει τις βασικότερες πληροφορίες που μπαίνει να δει ο φοιτητής στην εφαρμογή, δηλαδή τη συνολική του βαθμολογία, ακόμα και όταν δεν έχει πρόσβαση στο διαδίκτυο. Φυσικά αυτές οι πληροφορίες είναι σύμφωνα με την τελευταία φορά που ο χρήστης μπήκε στην εφαρμογή όταν είχε Ίντερνετ.

### Μπορεί να ανακαλυφθεί από μηχανές αναζήτησης

Οι PWA θα πρέπει να μπορούν να βρεθούν πολύ εύκολα μέσω αναζήτησης από τις διάφορες μηχανές αναζήτησης, καθώς αυτό είναι και ένα πολύ μεγάλο θετικό του διαδικτύου. Για να επιτευχθεί αυτό είναι σημαντικό κάθε διαφορετικό URL να έχει ένα μοναδικό, περιγραφικό τίτλο και να υπάρχει και η μετα-περιγραφή. Μετά, τα Google Search Console και τα Search Engine Optimization audits του Lighthouse μπορούν να βοηθήσουν να διορθωθούν τυχών θέματα εύρεσης της εφαρμογής.

### Η εφαρμογή θα πρέπει να δουλεύει με διάφορους τύπους input

Οι PWA είναι καλό να μπορούν να χρησιμοποιηθούν το ίδιο καλά τόσο με ποντίκι και πληκτρολόγιο, αλλά και με γραφίδα ή αφή. Ο χρήστης θα πρέπει να μην μπορεί να βλέπει αλλαγές αν χρησιμοποιεί διαφορετικούς τύπος input, και αν αυτό είναι δυνατόν, η εφαρμογή να έχει χαρακτηριστικά που να βελτιώνουν τη χρήση με συγκεκριμένα input, όπως για παράδειγμα η ανανέωση της σελίδας με το «τράβηγμα» προς τα κάτω

### Να δίνονται πληροφορίες όταν ζητείται κάποια συναίνεση

Για να μπορεί η εφαρμογή να χρησιμοποιήσει δυνατά APIs όπως οι ειδοποιήσεις ή το GPS, τότε είναι υποχρεωτικό να ζητείται από τον χρήστη να συναινέσει. Αυτά τα «παραθυράκια» που εμφανίζονται είναι αρκετά ενοχλητικά για τον χρήστη, όποτε αν εμφανίζονται από την εφαρμογή χωρίς καμία πληροφορία για ποιον λόγο χρειάζεται η συγκεκριμένη συναίνεση ή αν εμφανίζονται σε άκυρες στιγμές, όπως όταν φορτώνει μια σελίδα, τότε είναι πολύ πιθανόν ο χρήστη να τις απορρίψει και κάποιες σημαντικές λειτουργίες της εφαρμογής να μην μπορούν να αξιοποιηθούν. Οπότε είναι σημαντικό να δίνεται η πληροφορία στο χρήστη πριν εμφανιστεί το παραθυράκι της συναίνεσης.

# ΚΕΦΑΛΑΙΟ 5
# ΣΥΜΠΕΡΑΣΜΑΤΑ

## 5.1 Γενικές Σκέψεις

Σε γενικές είμαι πάρα πολύ ευχαριστημένος με το τελικό αποτέλεσμα της πτυχιακής. Ο στόχος ήταν να φτιαχτεί μία εφαρμογή mobile first για το φοιτητολόγιο, κάτι που επιτευχθεί αφού όλη η εμπειρία χρήσης δημιουργήθηκε με πρώτο στόχο τη χρήση από κινητές συσκευές, οπότε όλες οι σελίδες της εφαρμογής είναι responsive για κινητά, ενώ η δυνατότητα εγκατάστασης της εφαρμογής σε οποιοδήποτε κινητό, ανεξάρτητα του λειτουργικού του, δίνει μια παραπάνω έννοια στο “mobile first”. Ο επόμενος στόχος ήταν να φτιαχτεί ένα Φοιτητολόγιο αρκετά πιο μοντέρνο από το τωρινό, και το αποτέλεσμα και σε αυτό ήταν άκρως ικανοποιητικό, αφού το Φοιτητολόγιο της παρούσας πτυχιακής φαίνεται σαν μια εφαρμογή φτιαγμένη το 2021, με πιο ωραία χρώματα, τα διάφορα elements της εφαρμογής όπως φόρμες, tables ή ακόμα και κουμπιά να είναι πιο εμφανίσιμα και αντάξια από μια εφαρμογή αυτής τη δεκαετίας.

Ένα άλλο κομμάτι που έμεινα αρκετά ευχαριστημένος είναι η επιλογή των τεχνολογιών, γλωσσών προγραμματισμού και frameworks που επέλεξα. Η επιλογή να γίνει το backend σε Node.JS δεν θα μπορούσε να είναι καλύτερη, αφού δίνεται η δυνατότητα να φτιαχτεί ένα API σε πολύ σύντομο χρονικό διάστημα, και με κώδικα πολύ κοντά σε production ready, σε αντίθεση με άλλες γλώσσες προγραμματισμού όπως Java ή .NET, όπου για να στήσεις ένα API χρειάζεται πολύ μεγαλύτερο boilerplate κώδικα κάτι που αύξανε σημαντικά το χρόνο υλοποίησης του. Ήταν επίσης πολύ σημαντικό ότι υπάρχει μεγάλη κοινότητα γύρω από το Node.JS και άφθονες πηγές υλικού για να βοηθήσουν τους προγραμματιστές χωρίς εμπειρία να φτιάξουν το πρώτο τους backend σε Node.JS. Οπότε, μετά από ένα πολύ μικρό learning curve, κατάφερα να φτιάξω με επιτυχία τα πρώτα endpoints της εφαρμογής, και μετά από αυτό ήταν αρκετά εύκολο να φτιάξω και τα υπόλοιπα endpoints και να ολοκληρώσω το REST API. Περνώντας στη Βάση Δεδομένων που χρησιμοποιήθηκε, παρόλο που η ανάπτυξη εφαρμογών με το backend να είναι σε Node.JS, συνήθως συνοδεύεται από μια NoSQL βάση όπως η MongoDB, εδώ το use case της εφαρμογής φάνηκε να ταιριάζει περισσότερο σε μια σχεσιακή Βάση Δεδομένων, και το αποτέλεσμα φάνηκε να συμφωνεί με την αρχική αυτή επιλογή.

Όσον αφορά το frontend κομμάτι της εφαρμογής, η χρήση ενός CSS Framework όπως του Bootstrap, βοήθησε σημαντικά στη μείωση του χρόνου styling των διαφόρων element της εφαρμογής με τις πολλές έτοιμες στιλιστικά κλάσεις που παρέχει το Bootstrap, ενώ και ένας από τους βασικούς στόχους της πτυχιακής, να είναι η εφαρμογή mobile first και responsive σε όλες τις διάφορες συσκευές, έγινε πολύ πιο εύκολος, αφού και σε αυτό βοηθάει πολύ το Bootstrap. Η AngularJS σαν επιλογή για frontend JavaScript framework ήταν ναι μεν σωστή, γιατί ένα framework θεωρείται de facto σε όλες τις σημερινές εφαρμογές, αλλά σίγουρα ήρθε και με τις «παραξενιές» αλλά και δυσκολίες ενός framework που έχει πολλά χρόνια ζωής και μάλιστα κοντεύει να γίνει και unsupported από τους developers. Παρόλα αυτά, το θετικό ενός framework με αρκετά χρόνια ζωής ήταν οι αμέτρητες πηγές υλικού που υπάρχουν στο διαδίκτυο για να βοηθήσουν τους developers και φυσικά η μεγάλη κοινότητα προγραμματιστών σε AngularJS που είναι εκεί να βοηθήσει. Σημαντικό είναι να αναφερθεί ότι η χρήση ενός framework λύνει τα χέρια στους developers για διάφορες ενέργειες που θα ήταν πολύ πιο δύσκολο να γίνουν με απλή JavaScript.

Στο πιο σημαντικό κομμάτι, ίσως, αυτής της πτυχιακής, το κομμάτι της υλοποίησης της εφαρμογής ως Progressive Web App, νιώθω αρκετά ευχαριστημένος με το αποτέλεσμα, αφού κατάφερα να υλοποιήσω σχεδόν όλα τα χαρακτηριστικά που ξεχωρίζουν τις απλές εφαρμογές από μια PWA, ενώ όλα αυτά τα features ταίριαξαν πολύ καλά στην εφαρμογή. Αυτά τα χαρακτηριστικά θεωρώ ότι βοηθούν αρκετά της εμπειρία χρήσης του φοιτητή, είναι αρκετά μοντέρνα και ίσως και εντυπωσιακά για μια εφαρμογή που εξυπηρετεί λειτουργικότητες ενός πανεπιστημίου, ενώ ακόμα και η απόδοση και ταχύτητα της εφαρμογής ήταν αρκετά καλύτερα με τα PWA χαρακτηριστικά.

Τέλος, νιώθω αρκετά ικανοποιημένος με το γεγονός ότι έφτιαξα 2 διαφορετικά interfaces για την εφαρμογή, τόσο για τους φοιτητές αλλά και για τη γραμματεία, με διαφορετικές λειτουργίες για την κάθε μία. Αυτό πρόσθεσε χρόνο στην υλοποίηση της εφαρμογής, αλλά νιώθω ότι είναι αρκετά πιο complete η εφαρμογή.

## 5.2 Σύγκριση με το τωρινό Φοιτητολόγιο

Συγκρίνοντας το παρών Φοιτητολόγιο με την εφαρμογή που υλοποιήθηκε για την παρούσα πτυχιακή, οι διαφορές είναι ορατές από το πρώτο λεπτό. Ένα στοιχείο που δεν είναι καλό στο τωρινό Φοιτητολόγιο είναι το User Experience (UX). Συγκρίνοντας τις διάφορες λειτουργικότητες μεταξύ των δύο εφαρμογών είναι πολύ ξεκάθαρο το πόσο πιο απλές και κατανοητές είναι οι λειτουργικότητες στην εφαρμογή της παρούσας πτυχιακής, με τις πιο δημοφιλείς ενέργειες του φοιτητή να γίνονται πολύ πιο γρήγορα και εύκολα και να εξοικονομείται χρόνος. Για παράδειγμα το menu είναι αρκετά πιο απλό και ξεκάθαρο, προσφέροντας όμως τις ίδιες λειτουργικότητες ενώ και οι δηλώσεις μαθημάτων είναι πιο απλές.

Φυσικά, τεράστιες διαφορές υπάρχουν και στην εμπειρία χρήσης από το κινητό μεταξύ των δύο εφαρμογών. Στο τωρινό φοιτητολόγιο ακόμα και η σελίδα σύνδεσης δεν είναι προσαρμοσμένη για κινητά, είναι απλώς η έκδοση που υπάρχει για τον υπολογιστή σε μια πιο μικρή έκδοση. Μπαίνοντας στην εφαρμογή, πάλι βλέπουμε ότι τίποτα δεν είναι προσαρμοσμένος για κινητές συσκευές και πρέπει να κάνει ζουμ για να μπορέσει να επιλέξεις κάτι από το μενού, ενώ και οι υπόλοιπες λειτουργίες, όπως να επιλέξεις μαθήματα ή ακόμα και να δεις τις βαθμολογίες σου, είναι αρκετά πιο δύσκολες. Αυτός ήταν ένας πολύ σημαντικός λόγος που αποφάσισα να κάνω τη συγκεκριμένη πτυχιακή και το αποτέλεσμα βγήκε πολύ καλό και η εμπειρία χρήσης πια σε κινητές συσκευές είναι πολύ καλύτερη.

Το Φοιτητολόγιο που υλοποιήθηκε για την πτυχιακή έχει κρατήσει όλες τις βασικές λειτουργικότητες που χρησιμοποιούν οι φοιτητές, δηλαδή να μπορούν να βλέπουν τα στοιχεία τους, να βλέπουν τις βαθμολογίες του, να δηλώνουν τα μαθήματα του εξαμήνου, να κάνουν αιτήσεις για βεβαιώσεις στη γραμματεία, ενώ μπορούν να βλέπουν και τις παλιές αιτήσεις και δηλώσεις τους. Φυσικά, πέρα αυτά, προστέθηκαν και δυνατότητες που δεν παρέχονται από το υπάρχον Φοιτητολόγιο και πρόκειται για τις δυνατότητες που προσφέρουν οι PWA, όπως για παράδειγμα τα push notifications, η ύπαρξη offline λειτουργικότητας και η δυνατότητα εγκατάστασης της εφαρμογής στο κινητό.

## 5.3 Μελλοντικές Βελτιώσεις

Παρόλο που είμαι αρκετά ικανοποιημένος με το αποτέλεσμα της πτυχιακής, σίγουρα υπάρχουν κάποια πράγματα που θα μπορούσα να κάνω καλύτερα. Για αρχή, αν συνέχιζα το development της εφαρμογής θα άλλαζα τελείως τον τρόπο που γίνεται το login και θα πήγαινα σε user authentication με τη βοήθεια του Passport. Το passport είναι μια πάρα πολύ δημοφιλής βιβλιοθήκη στην JavaScript που βοηθάει στην επαλήθευση του χρήστη κατά τη σύνδεση στην εφαρμογή. Προσφέρει διάφορα έτοιμα modules για διαφορετικούς τρόπους σύνδεσης στην εφαρμογή, από απλή σύνδεση με κωδικό και password μέχρι και σύνδεση με λογαριασμούς social media. Επιπλέον, στο ίδιο μήκος κύματος με το user authentication, θα πρόσθετα και τη χρήση JWT για την εξουσιοδότηση των request που γίνονται από το frontend στον server της εφαρμογής από τους χρήστες. Προσθέτοντας το JWT κάθε χρήστη στα request, το backend μπορεί να αναγνωρίσει αν αυτό το αίτημα πρέπει να εξυπηρετηθεί ή έρχεται από έναν μη εξουσιοδοτημένο χρήστη.

Ένα ακόμα πράγμα που θα έφτιαχνα, είναι να δομούσα καλύτερα τον τρόπο που το backend επικοινωνεί με την Βάση Δεδομένων. Θα έφτιαχνα έναν wrapper για όλα τα queries που θα γινόντουσαν στην βάση και θα χρησιμοποιούσα κάποια βιβλιοθήκη που θα μου επέτρεπε να κάνω queries πιο «όμορφα», αλλάζοντας τον τρόπου που γίνεται τώρα, ο οποίος είναι κατευθείαν γραμμένα queries στη βάση για κάθε επικοινωνία με τη βάση. Έτσι για παράδειγμα αντί κάθε φορά που θέλω κάποια συγκεκριμένα πεδία από ένα τραπέζι να γράφω μια γραμμή κώδικα του στυλ “db.users.get(name, surname, email, id)” σε αντίθεση με “db.query(‘SELECT name, surname, email, id from USERS’)”. Φυσικά, όσο πιο περίπλοκα είναι τα queries, τόσο πιο πολύ χρήσιμη είναι μια τέτοια βιβλιοθήκη.

Κάτι ακόμα που θα προσπαθούσα να βελτιώσω είναι το API της εφαρμογής. Το structure των endpoint βελτιώθηκε κατά τη διάρκεια της πτυχιακής ώστε να μοιάζει όσο περισσότερο γίνεται σε ένα API κανονικής production εφαρμογής, αλλά παρόλα αυτά σίγουρα χρειάζεται λίγη βελτίωση ακόμα. Κάτι ακόμα που θα έβαζα είναι διάφορα middleware στα routes του API, για διάφορους ελέγχους που πρέπει να γίνουν πριν καταλήξει να τρέξει ο βασικός κώδικας του endpoint. Για παράδειγμα, ένα middleware θα μπορούσε να τσεκάρει το JWT του χρήστη για να δει αν έχει πρόσβαση στο API, ένα άλλο middleware θα μπορούσε να τσεκάρει το body των request και να μην αφήνει το request να προχωρήσει αν δεν υπάρχει το κατάλληλο body. Ο κώδικας επίσης των endpoint χρειάζεται αρκετή βελτίωση, γιατί παρόλο που είναι λειτουργικός, σίγουρα δεν είναι κώδικας που είναι έτοιμος για production.

Όσον αφορά features, παρόλο που κάλυψα όλες τις λειτουργικότητες που προσφέρει το τωρινό Φοιτητολόγιο, σίγουρα υπάρχουν και μερικά ακόμα πράγματα που θα ήταν χρήσιμα. Για αρχή, κάτι που δεν χρησιμοποιείται από την τωρινή εφαρμογή, είναι οι Ανακοινώσεις. Οπότε, ένα feature που θα πρόσθετα, θα ήταν ένα ακόμα tab στην γραμμή περιήγησης για τις Ανακοινώσεις. Στο interface της γραμματείας, θα υπάρχει μια μικρή φόρμα με το κείμενο της ανακοίνωσης και το κουμπί για την υποβολή της, ενώ στο interface των φοιτητών, οι φοιτητές θα μπορούν να δουν όλες τις ανακοινώσεις, ενώ ένα επιπρόσθετο χαρακτηριστικό θα ήταν να έρχονται και push notifications κάθε φορά που υπάρχει μια καινούρια Ανακοίνωση. Ένα ακόμα feature που θα πρόσθετα, που υπάρχει κάπως στη τωρινή εφαρμογή είναι η εξέλιξη του μέσου όρου των φοιτητών όσο περνούσαν τα εξάμηνα με ένα διάγραμμα που θα δείχνει την εξέλιξη.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] Developer.mozilla.org. (2021). *File System Access API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/File_System_Access_API (τελευταία πρόσβαση: 19/08/2021).

[2] Developer.mozilla.org. (2021). *Using the Payment Request API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/Payment_Request_API/Using_the_Payment_Request_API (τελευταία πρόσβαση: 19/08/2021).

[3] Developer.mozilla.org. (2021). *Web Authentication API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/Web_Authentication_API (τελευταία πρόσβαση: 19/08/2021).

[4] Developer.mozilla.org. (2021). *Web Bluetooth API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API (τελευταία πρόσβαση: 19/08/2021).

[5] Developer.mozilla.org. (2021). *Web Share API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/Web_Share_API (τελευταία πρόσβαση: 19/08/2021).

[6] Developer.mozilla.org. (2021). *MediaDevices.getUserMedia()* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia (τελευταία πρόσβαση: 19/08/2021).

[7] WhatPWACanDoToday. (2021). Διαθέσιμο στο: https://whatpwacando.today

[8] Developer.mozilla.org. (2021). *Progressive web apps (PWAs) → Add to Home Screen* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen (τελευταία πρόσβαση: 22/08/2021).

[9] Developer.mozilla.org. (2021). *How to make PWAs re-engageable using Notifications and Push* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Re-engageable_Notifications_Push (τελευταία πρόσβαση: 22/08/2021).

[10] Google. (2021). *Introduction to Push Notifications* [Online]. Διαθέσιμο στο: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications (τελευταία πρόσβαση: 22/08/2021)

[11] Web.dev. (2021). *What makes a good Progressive Web App* [Online]. Διαθέσιμο στο:
https://web.dev/pwa-checklist
(τελευταία πρόσβαση: 28/8/2021)

[12] Google. (2021). *Get Started | Workbox | Google Developers* [Online]. Διαθέσιμο στο:
https://developers.google.com/web/tools/workbox/guides/get-started
(τελευταία πρόσβαση: 9/9/2021)

[13] Google. (2021). *Workbox | Google Developers* [Online]. Διαθέσιμο στο:
https://developers.google.com/web/tools/workbox
(τελευταία πρόσβαση: 9/9/2021)

[14] Google. (2021). *Lighthouse | Tools for Developers | Google Developers* [Online]. Διαθέσιμο στο:
https://developers.google.com/web/tools/lighthouse
(τελευταία πρόσβαση: 14/1/2021)

[15] Wikipedia. (2021). *Progressive web application – Wikipedia* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/Progressive_web_application
(τελευταία πρόσβαση: 14/1/2021)

[16] Wikipedia. (2021). *AngularJS – Wikipedia* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/AngularJS
(τελευταία πρόσβαση: 14/1/2021)

[17] Wikipedia. (2021). *Node.js – Wikipedia* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/Node.js
(τελευταία πρόσβαση: 14/1/2021)

[18] Wikipedia. (2021). *Heroku – Wikipedia* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/Heroku
(τελευταία πρόσβαση: 14/1/2021)

[19] Wikipedia. (2021). *PostgreSQL – Wikipedia* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/PostgreSQL
(τελευταία πρόσβαση: 14/1/2021)

[20] Wikipedia. (2021). *Web application* [Online]. Διαθέσιμο στο:
https://en.wikipedia.org/wiki/Web_application
(τελευταία πρόσβαση: 14/1/2021)

[21] Tatsiana Tsiukhai. (2020). *9 Successful PWA Examples That May Inspire You*, 01 August 2020 [Online]. Διαθέσιμο στο: https://www.dizzain.com/blog/insights/pwa-examples/
(τελευταία πρόσβαση: 10/1/2021)

[22] David Oragui. (2018). *11 Examples of Progressive Web Apps*, 10 April 2018 [Online]. Διαθέσιμο στο: https://themanifest.com/development/11-examples-progressive-web-apps
(τελευταία πρόσβαση: 10/1/2021)

[23] Developer.mozilla.org. (2021). *Media Devices* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Add_to_home_screen
(τελευταία πρόσβαση: 10/09/2021).

[24] Developer.mozilla.org. (2021). *Web NFC API* [Online]. Διαθέσιμο στο: https://developer.mozilla.org/en-US/docs/Web/API/Web_NFC_API
(τελευταία πρόσβαση: 10/09/2021).

[25] Altexsoft. (2018). *Progressive Web Apps: Core Features, Architecture, Pros and Cons* [Online]. Διαθέσιμο στο: https://www.altexsoft.com/blog/engineering/progressive-web-apps/
(τελευταία πρόσβαση: 10/09/2021).

[26] Kevin Farrugia. (2016). *A Begginer's Guide To Progressive Web Apps* [Online]. Διαθέσιμο στο: https://www.smashingmagazine.com/2016/08/a-beginners-guide-to-progressive-web-apps/
(τελευταία πρόσβαση: 10/09/2021).

[27] Swapnil Ramdas Chavan. (2020). *Service Worker Lifecycle* [Online]. Διαθέσιμο στο: https://medium.com/walmartglobaltech/service-worker-lifecycle-20333ccd570a
(τελευταία πρόσβαση: 10/09/2021).

[28] Asper Brothers. (2019). *Best PWA examples. Success stories of Progressive Web Apps* [online]. Διαθέσιμο στο: https://asperbrothers.com/blog/best-pwa-examples/
(τελευταία πρόσβαση: 10/09/2021).

[29] Wikipedia. (2021). *Npm (software)* [Online]. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Npm_(software)
(τελευταία πρόσβαση: 10/09/2021).