ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη web εφαρμογής κοινωνικού δικτύου με τεχνολογίες progressive web apps και Firebase** **Social media application development using progressive web apps tech stack and Firebase**
Ονοματεπώνυμο Φοιτητή	**Τσίλικας Γεώργιος**
Πατρώνυμο	**Αριστείδης**
Αριθμός Μητρώου	**ΜΠΠΛ / 15075**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Σεπτέμβριος 2022**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής	Μαρία Βίρβου Καθηγήτρια	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου κ. Αλέπη για τη στήριξη και την ευκαιρία που μου πρόσφερε να υλοποιήσω τη παρούσα μεταπτυχιακή διατριβή.

Θα ήθελα να κάνω ειδική αναφορά στην κ. Βίρβου που κατά τη διάρκεια του μεταπτυχιακού προγράμματος σπουδών ήταν πάντα δίπλα στο φοιτητή και κατέβαλε κάθε δυνατή προσπάθεια, προκειμένου να δώσει λύσεις στα προβλήματα που κάθε φορά παρουσιάζονταν.

Περίληψη

Σκοπός της πτυχιακής διατριβής είναι να δείξουμε τις δυνατότητες που μπορεί να έχει μία web εφαρμογή κάνοντας απλά χρήση ενός σύγχρονου browser. Η πλειοψηφία των χρηστών κινητών συσκευών είναι εξοικειωμένοι με τη χρήση native εφαρμογών. Υπάρχει η δυνατότητα, όμως, να αναπτυχθούν καθαρά web εφαρμογές με τις ίδιες δυνατότητες και χαρακτηριστικά και να προσφέρουν παρόμοια εμπειρία χρήσης με τις native εφαρμογές που γνωρίζουμε. Αυτές οι εφαρμογές ονομάζονται progressive web apps.

Για να γίνει παρουσίαση αυτών των δυνατοτήτων δημιουργήθηκε η εφαρμογή κοινωνικού δικτύου με την ονομασία Xoxa. Στη γλώσσα της φυλής των Zulu, Xoxa σημαίνει να μιλάς ή να συζητάς και ο καλύτερος τρόπος για να το κάνεις αυτό στη σύγχρονη εποχή είναι με το να μοιραστείς μία selfie με τους φίλους σου.

Abstract

This thesis aims to present the capabilities a web app can have just by using any modern browser. Most mobile users are familiar with using native apps on their phones. But there is the possibility to develop pure web applications with the same capabilities and characteristics and offer the same user experience as native apps. These apps are called progressive web apps.

To present these capabilities a social media web app was developed called Xoxa. In the Zulu tribe language, Xoxa means to chat or to converse and there isn’t any better way to do this than sharing a selfie with your friends.

Πίνακας περιεχομένων

1. Εισαγωγή

1.1. Τι είναι οι progressive web apps

Οι progressive web apps είναι σαν τις συνηθισμένες web εφαρμογές που γνωρίζουμε με αυξημένες δυνατότητες και χαρακτηριστικά που συνήθως βρίσκουμε μόνο σε native εφαρμογές. Αυτό σημαίνει ότι μπορούμε να κάνουμε τα εξής:

- Εγκατάσταση της εφαρμογής και εμφάνιση εικονιδίου (shortcut) στην επιφάνεια εργασίας για άμεση πρόσβαση σε αυτήν είτε σε mobile είτε σε desktop περιβάλλον.
- Βελτίωση της ταχύτητας και των επιδόσεων της εφαρμογής μέσω τεχνικών αποθήκευσης δεδομένων (cache, precache, αποθήκευση φωτογραφιών και font τοπικά και όχι σε server).
- Δυνατότητα χρήσης της εφαρμογής ακόμα και χωρίς σύνδεση στο internet (offline δυνατότητες).
- Background Sync, δηλαδή ο χρήστης μπορεί να κάνει χρήση της εφαρμογής (π.χ. να τραβήξει μία φωτογραφία) και να ανέβει εκ των υστέρων στο server, όταν η εφαρμογή αποκτήσει πρόσβαση σε αυτόν.
- Δυνατότητα αποστολής push notifications ακόμα και όταν ο χρήστης δεν έχει ανοικτή την εφαρμογή εκείνη τη στιγμή.
- Δυνατότητα χρήσης native λειτουργιών της συσκευής, όπως κάμερα και Geolocation.
- Δυνατότητα χρήσης της εφαρμογής ακόμα και αν ο browser που χρησιμοποιούμε δεν υποστηρίζει όλες τις λειτουργίες (π.χ. internet explorer).

Η εφαρμογή που αναπτύχθηκε για να δείξει αυτές τις δυνατότητες προσομοιώνει σύγχρονες εφαρμογές κοινωνικών δικτύων και είναι πιο κοντά στο Instagram. Συγκεκριμένα, ο χρήστης θα μπορεί να τραβάει φωτογραφία με τη χρήση της κάμερας της συσκευής του ή να ανεβάζει μία φωτογραφία που είχε τραβήξει εκ των προτέρων, να εισάγει ένα όνομα και μήνυμα, να δηλώνει την τοποθεσία του με χρήση του geolocation του browser και, τέλος, να την ανεβάζει σε έναν κοινόχρηστο τοίχο στον οποίο έχουν πρόσβαση όλοι οι χρήστες της εφαρμογής. Επίσης, όταν γίνεται νέο post θα πηγαίνει push notification σε όλους τους χρήστες που έχουν κάνει subscribe σε αυτήν την υπηρεσία. Η εφαρμογή θα είναι λειτουργική ακόμα και όταν ο χρήστης δεν έχει internet τη στιγμή που κάνει χρήση της εφαρμογής και προσπαθεί να ανεβάσει το post. Τέλος, μπορεί να γίνει εγκατάσταση της εφαρμογής στην επιφάνεια εργασίας για άμεση πρόσβαση και χρήση.

Η ανάπτυξη έγινε με Vue.js 2 στο front-end και για back-end έγινε μία serverless προσέγγιση. Για να το πετύχουμε αυτό έγινε χρήση των υπηρεσιών Firebase της Google. Κατά τη διάρκεια δημιουργίας της εφαρμογής επιλέχθηκε η έκδοση 2 του Vue.js, γιατί ήταν πιο σταθερό και είχε καλύτερη συμβατότητα με τα dependencies και libraries των οποίων έγινε χρήση. Το τελευταίο διάστημα, όμως, έχουν γίνει μεγάλα βήματα στην υποστήριξη του Vue.js 3 και η συμβατότητά του έχει βελτιωθεί σε τέτοιο βαθμό, ώστε να είναι η κύρια και προτεινόμενη έκδοση του framework. Μία καλή άσκηση θα ήταν στο μέλλον να γίνει refactor η εφαρμογή και να γραφεί σε Vue.js 3, ώστε να δούμε τα οφέλη της νέας έκδοσης.

Και τα τρία μεγάλα και γνωστά framework, Angular, React και Vue.js υποστηρίζουν τη δημιουργία progressive web apps και μπορούμε να έχουμε και με τα τρία τα ίδια αποτελέσματα. Δημιουργία progressive web apps μπορεί να γίνει ακόμα και με χρήση Vanilla JavaScript, οπότε το προτιμότερο είναι να γίνει χρήση του framework με το οποίο ο προγραμματιστής έχει τη μεγαλύτερη οικειότητα.

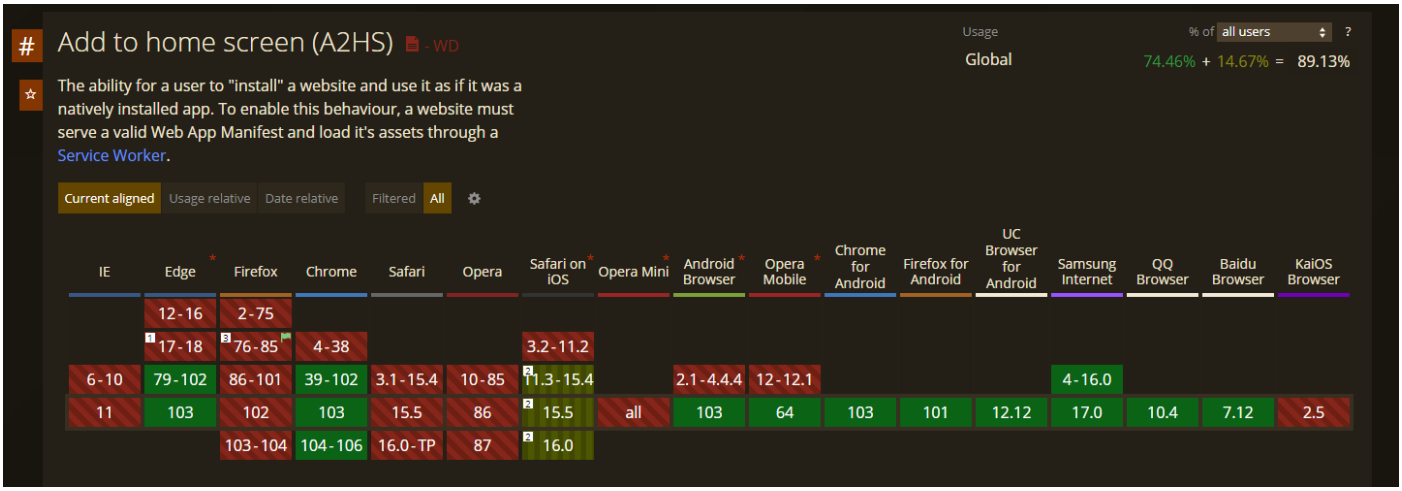
Όσον αφορά τα Firebase Services, έγινε χρήση της Cloud Firestore Database που είναι και ο τελευταίος τύπος βάσης δεδομένων που έχει δημιουργήσει η Google. Θεωρείται μία NoSQL βάση, γιατί η αποθήκευση των δεδομένων γίνεται με μορφή συλλογών αρχείων. Προσφέρει ακόμα και τη Realtime Database που είναι μία παλαιότερη υπηρεσία βάσης δεδομένων που έχει δημιουργήσει η Google αλλά προτείνεται η χρήση της Firestore σαν πιο καινούρια υπηρεσία με περισσότερες δυνατότητες. Επίσης, γίνεται χρήση του Firebase Storage για αποθήκευση των αρχείων (φωτογραφιών) και Firebase Hosting για να ανεβάσουμε την εφαρμογή και να έχουμε πρόσβαση σε αυτήν. Τέλος, γίνεται και χρήση Firebase Functions, ώστε να παραμείνει η εφαρμογή serverless και να προσθέσουμε κάποιες λειτουργίες που μας είναι απαραίτητες και θα

δούμε στη συνέχεια. Η εφαρμογή που δημιουργήθηκε είναι προσβάσιμη και μπορούμε να τη βρούμε στο παρακάτω URL:

https://xoxa-2677f.web.app/

1.2. Συμβατότητα με Browsers

Παρά το γεγονός ότι οι progressive web apps υπάρχουν εδώ και αρκετά χρόνια, αυτές δεν έχουν ακόμα ευρεία υποστήριξη. Ουσιαστικά ο μοναδικός browser που υποστηρίζει τα περισσότερα χαρακτηριστικά και δυνατότητες των PWAs είναι ο Chrome της Google. Για παράδειγμα, η υποστήριξη για τη δυνατότητα εγκατάστασης του app στην επιφάνεια εργασίας είναι η παρακάτω:



1-1 Υποστήριξη PWA

Παρατηρούμε ότι τα πράγματα δεν είναι πολύ θετικά. Ο Safari της Apple παρά το γεγονός ότι θεωρείται ένας σύγχρονος browser δεν υποστηρίζει τις περισσότερες δυνατότητες των PWAs. Το ίδιο μονοπάτι ακολουθεί και ο Firefox παρά το γεγονός ότι θεωρείται ανεξάρτητος browser. Ο σημαντικότερος λόγος που συμβαίνει αυτό είναι ότι με τη διάδοση των progressive web apps είναι εύκολο να οδηγηθούν οι χρήστες εκτός των app stores και του οικοσυστήματος που έχει χτίσει η κάθε εταιρεία. Αυτό θα έχει ως αποτέλεσμα την απώλεια τεράστιων χρηματικών ποσών τόσο για τις εταιρείες όσο και για τους δημιουργούς εφαρμογών.

Ένα άλλο πρόβλημα είναι ότι τα specifications δεν έχουν ωριμάσει, με αποτέλεσμα να υπάρχει κίνδυνος να γίνουν αλλαγές που μπορούν να καταστήσουν την εφαρμογή μας μη λειτουργική. Για παράδειγμα, πριν λίγο καιρό η Google ανακοίνωσε ότι θα άλλαζε τις απαιτήσεις για τις offline εφαρμογές, όπως μπορούμε να δούμε στο παρακάτω σύνδεσμο:

https://developer.chrome.com/blog/improved-pwa-offline-detection/

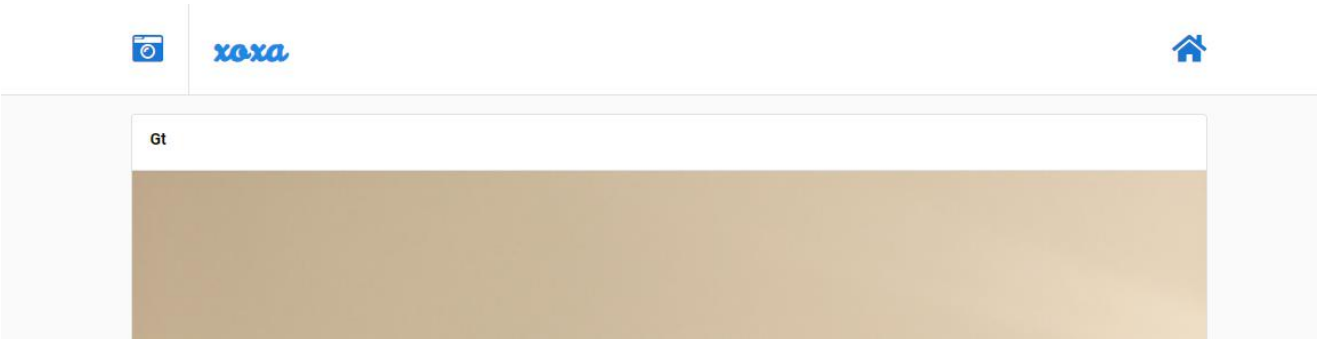
2. Εμφάνιση Και Αρχιτεκτονική Εφαρμογής

2.1. Πλοήγηση/Μενού

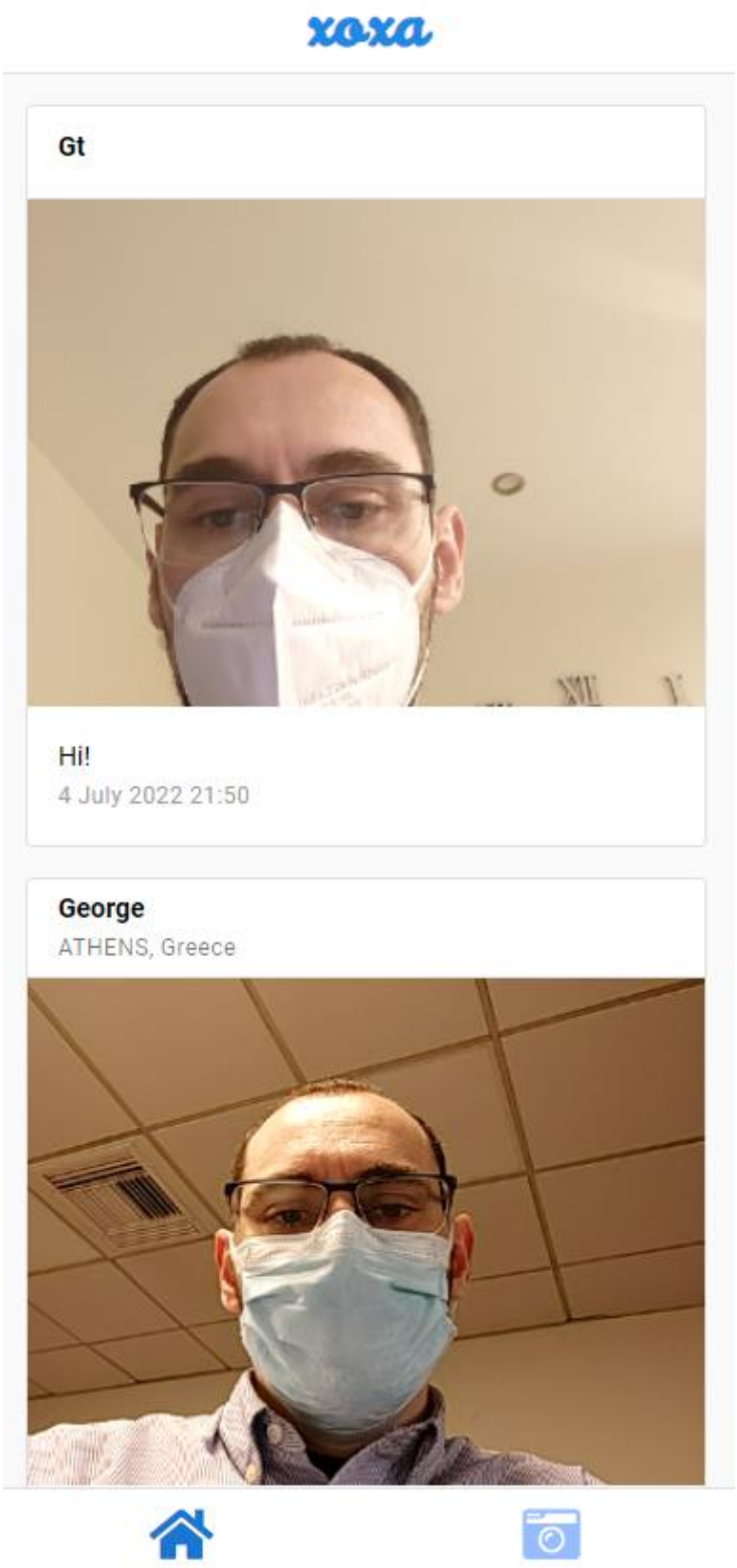
Η εφαρμογή έχει responsive χαρακτηριστικά και το layout διαφοροποιείται ανάλογα με τη συσκευή και το μέγεθός της οθόνης, π.χ. σταθερός υπολογιστής με κινητό τηλέφωνο. Σε μεγάλου μεγέθους οθόνες τα εικονίδια και η μπάρα πλοήγησης βρίσκονται στο πάνω μέρος όπου αριστερά προς δεξιά έχουμε:

- **Camera Icon:** Το εικονίδιο με τη συντόμευση για τη σελίδα της κάμερας, όπου μπορούμε να τραβήξουμε/ανεβάσουμε φωτογραφίες.
- **Header:** Φαίνεται η ονομασία της εφαρμογής.
- **Home Icon:** Το εικονίδιο που μας μεταφέρει στη κεντρική σελίδα της εφαρμογής, όπου βλέπουμε τον τοίχο με τα post.

Σε μικρότερου μεγέθους οθόνες η σχεδίαση διαφοροποιείται. Εκεί έχουμε στην κορυφή το Header με την ονομασία της εφαρμογής και στο κάτω μέρος της οθόνης στα αριστερά το Home icon και δεξιά το Camera icon. Επίσης, ανάλογα με τη σελίδα που βρισκόμαστε, το αντίστοιχο εικονίδιο έχει διαφορετικό χρώμα και το εικονίδιο της επιλεγμένης σελίδας φαίνεται πιο έντονα.



2-1 Desktop Header



2-2 Navigation menu σε κινητό

Η υλοποίηση και ο σχεδιασμός έχει γίνει με τη mobile-first προσέγγιση (mobile-first approach, mobile-first design strategy) που σημαίνει ότι προτεραιότητα είναι η εμφάνιση στα κινητά, γιατί από εκεί ο χρήστης θα μπορέσει να έχει καλύτερη υποστήριξη των δυνατοτήτων της εφαρμογής και καλύτερη εμπειρία χρήσης. Είναι μία τάση των τελευταίων ετών στη σχεδίαση των web εφαρμογών, γιατί τα στατιστικά δείχνουν ότι η πλειοψηφία των χρηστών προτιμάει τις κινητές του συσκευές για να πλοηγηθεί στο διαδίκτυο. Επίσης, επειδή η πλειοψηφία των χρηστών έχει πάντα μία κινητή συσκευή μαζί του, εφαρμογές που ακολουθούν την mobile first στρατηγική έχουν μεγαλύτερη πιθανότητα να πετύχουν κάποια αλληλεπίδραση από το χρήστη και να αποκτήσουν ένα μεγαλύτερο κοινό χρηστών που μπορεί να οδηγήσει σε μεγαλύτερα κέρδη.

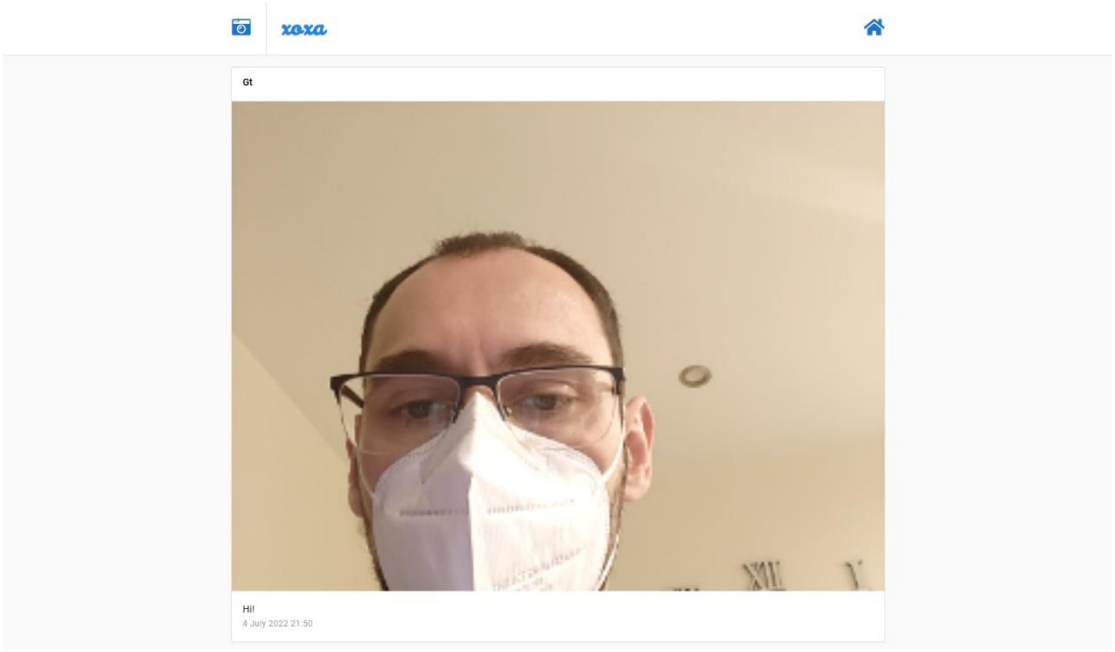
Η εφαρμογή αποτελείται από δύο βασικές οθόνες/σελίδες, τη κεντρική σελίδα (home page) και τη σελίδα της κάμερας για να μπορέσουμε να ανεβάσουμε post (camera page).

2.2. Κεντρική Σελίδα (Home Page)

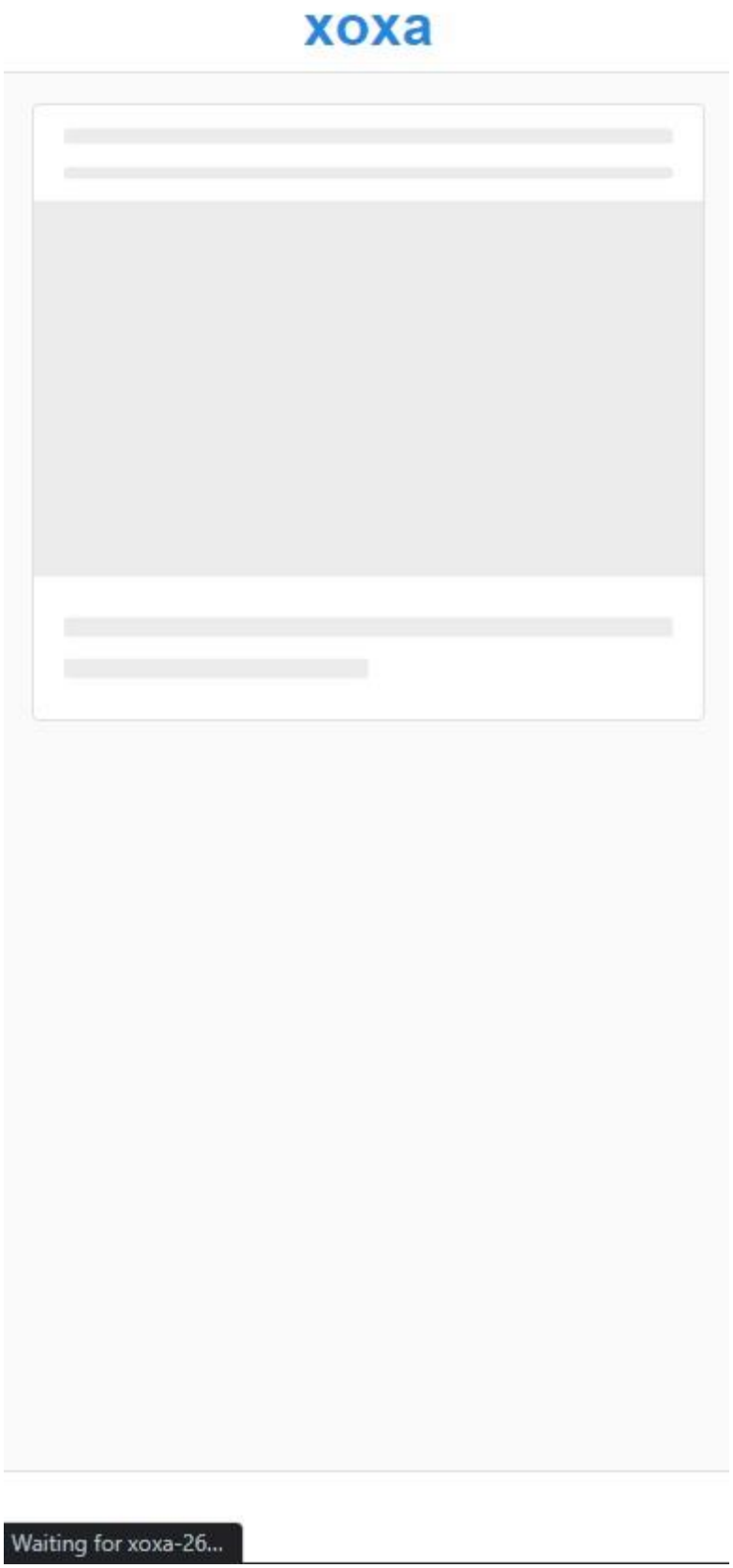
Η κεντρική σελίδα, γνωστή ως home page, είναι η σελίδα στην οποία εμφανίζονται όλες οι φωτογραφίες τις οποίες έχουν ανεβάσει οι χρήστες. Θυμίζει το γνωστό μας τοίχο που βλέπουμε σε δημοφιλή social media. Οι φωτογραφίες έχουν τη μορφή κάρτας και περιλαμβάνουν τα εξής:

- **Nickname:** το όνομα ή το παρατσούκλι που επέλεξε ο χρήστης στην κορυφή της κάρτας.
- **Location:** η πόλη και η χώρα που τραβήχτηκε η φωτογραφία.
- **Photograph:** την ίδια τη φωτογραφία που καταλαμβάνει το μεγαλύτερο μέρος της κάρτας
- **Message:** το μήνυμα που επέλεξε ο χρήστης να εισάγει κατά τη δημιουργία του post.
- **Datetime:** την ημερομηνία και την ώρα που ανέβηκε το post στην εφαρμογή.

Τα post που έχουν κάνει οι χρήστες εμφανίζονται ταξινομημένα με το νεότερο να είναι στην κορυφή και, στη συνέχεια, να εμφανίζονται τα παλαιοτέρα post. Επίσης, στο πρώτο load μέχρι να μεταφορτώσει τα post η εφαρμογή από το server για καλύτερη εμπειρία χρήσης (User Experience – UX) εμφανίζουμε ένα “skeleton component”, όπως ονομάζεται, για να καταλαβαίνει ο χρήστης ότι γίνεται μία διεργασία στο background της εφαρμογής και να γεμίζει προσωρινά τη σελίδα, ώστε να μην φαίνεται άδεια μέχρι να έρθουν τα αποτελέσματα.



2-3 Κεντρική σελίδα σε desktop



2-4 Skeleton component

2.3. Σελίδα Κάμερας (Camera Page)

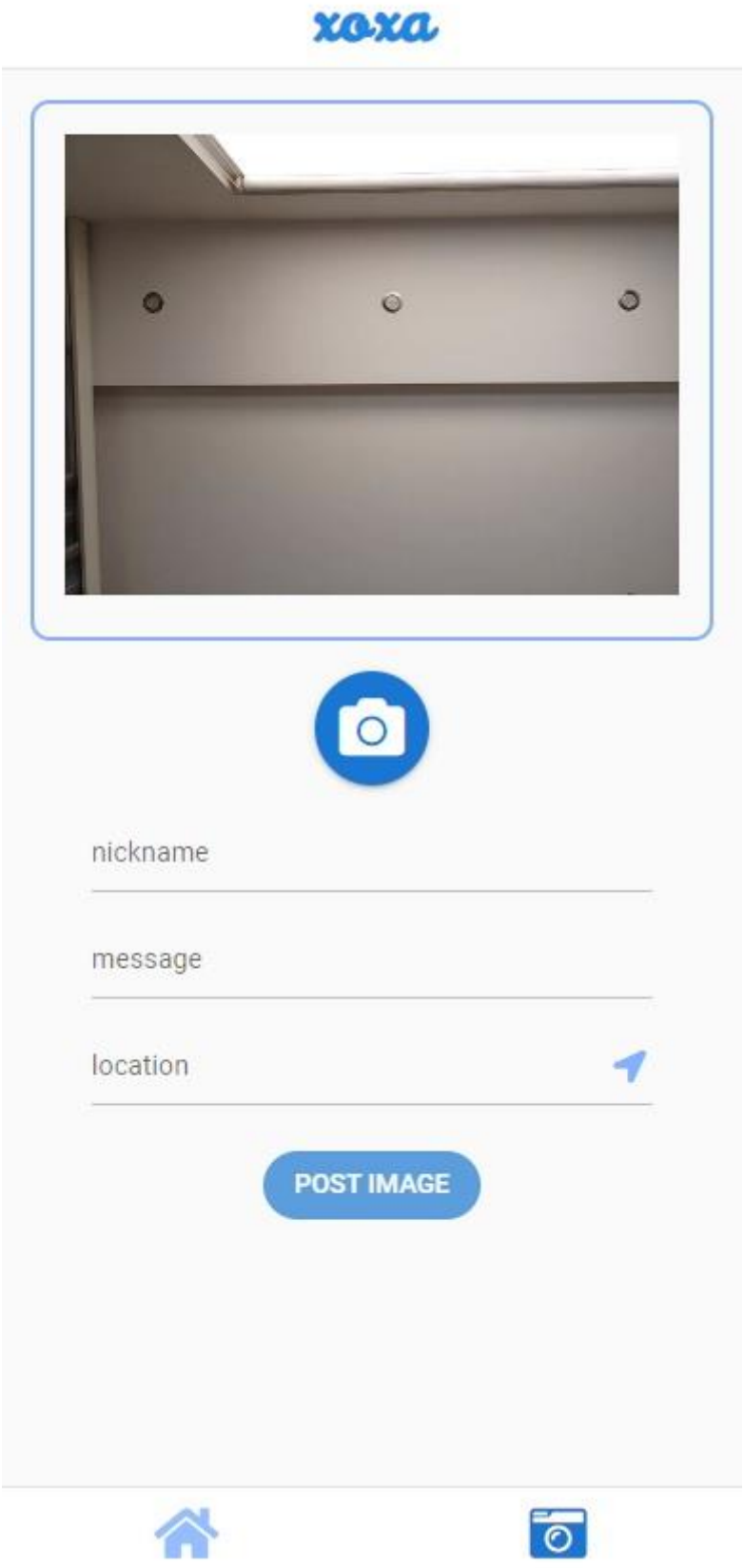
Η δεύτερη οθόνη/σελίδα είναι η σελίδα της κάμερας την οποία χρησιμοποιεί ο χρήστης για να τραβήξει και να ανεβάσει τη φωτογραφία που μόλις τράβηξε. Αυτή η σελίδα περιλαμβάνει τα εξής πεδία και κουμπιά:

- **Camera Frame:** το πλαίσιο της κάμερας, όπου έχουμε live video feed, για να μπορέσει ο χρήστης να τραβήξει φωτογραφία.
- **Camera button:** το κουμπί το οποίο πρέπει να πατήσει ο χρήστης για να τραβήξει φωτογραφία.
- **Upload File:** αυτό το πεδίο εμφανίζεται ακριβώς κάτω από το πλαίσιο κάμερας στην περίπτωση που δεν έχει τη δυνατότητα να γίνει χρήση κάμερας για να τραβηχτεί φωτογραφία.
- **Nickname πεδίο:** το όνομα ή παρατσούκλι που επιθυμεί να εισάγει ο χρήστης κατά το ανέβασμα της φωτογραφίας.
- **Message πεδίο:** το μήνυμα του χρήστη που επιθυμεί να ανεβάσει μαζί με τη φωτογραφία του.
- **Location πεδίο:** ο χρήστης μπορεί να εισάγει την τοποθεσία του κάνοντας χρήση του geolocation API.
- **Post Image button:** το κουμπί που πρέπει να πατήσει ο χρήστης για να ανέβει η φωτογραφία του μαζί με τις υπόλοιπες πληροφορίες.

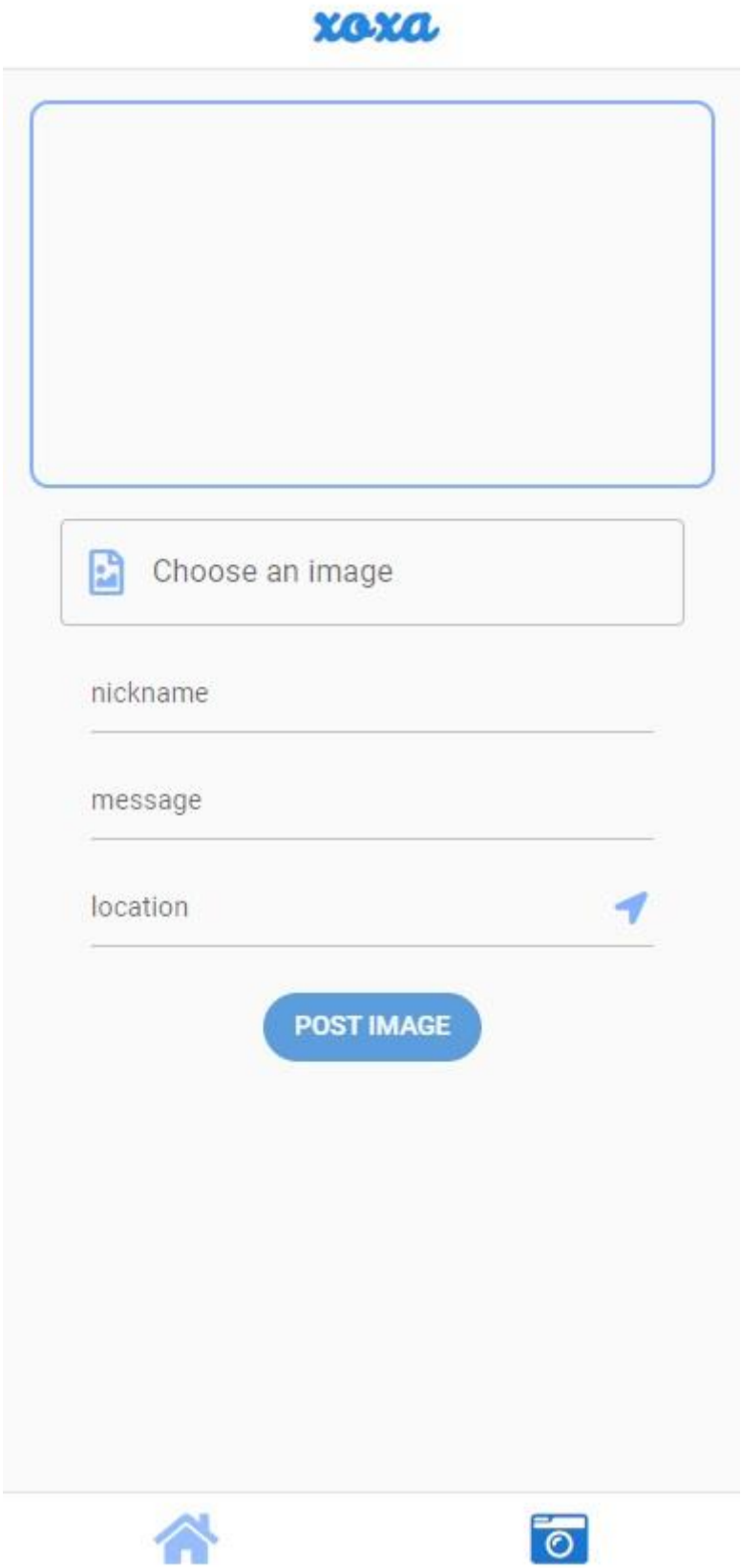
Αν ο χρήστης δε δώσει τη συγκατάθεση του, όταν του ζητηθεί, για να κάνει χρήση της κάμερας ο browser ή σε παλαιότερους browsers όπου δεν υποστηρίζεται η χρήση κάμερας, η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη να ανεβάσει ένα αρχείο φωτογραφίας που έχει αποθηκεύσει τοπικά, ώστε να μπορεί να συνεχίσει τη χρήση της. Το ίδιο συμβαίνει και στη περίπτωση που δεν εντοπιστεί κάποια λειτουργική κάμερα στη συσκευή από την οποία έχει πρόσβαση ο χρήστης στην εφαρμογή.

Ένα άλλο χαρακτηριστικό είναι η δυνατότητα προεπισκόπησης. Δηλαδή, όταν ο χρήστης τραβήξει ή ανεβάσει μία φωτογραφία, αυτή θα εμφανιστεί μέσα στο πλαίσιο της κάμερας αντί για το live video feed, ώστε ο χρήστης να την αξιολογήσει και να αποφασίσει αν θέλει να ανεβάσει ή όχι τη φωτογραφία που μόλις τράβηξε.

Τέλος, για να μπορέσει ο χρήστης να ανεβάσει τη φωτογραφία του, υποχρεωτικά είναι τα πεδία nickname και message. Το location δεν είναι υποχρεωτικό πεδίο, γιατί ο χρήστης μπορεί να μη θέλει να δώσει την τοποθεσία του ή να υπάρχει κάποιο πρόβλημα με το geolocation API.



2-5 Η σελίδα της κάμερας



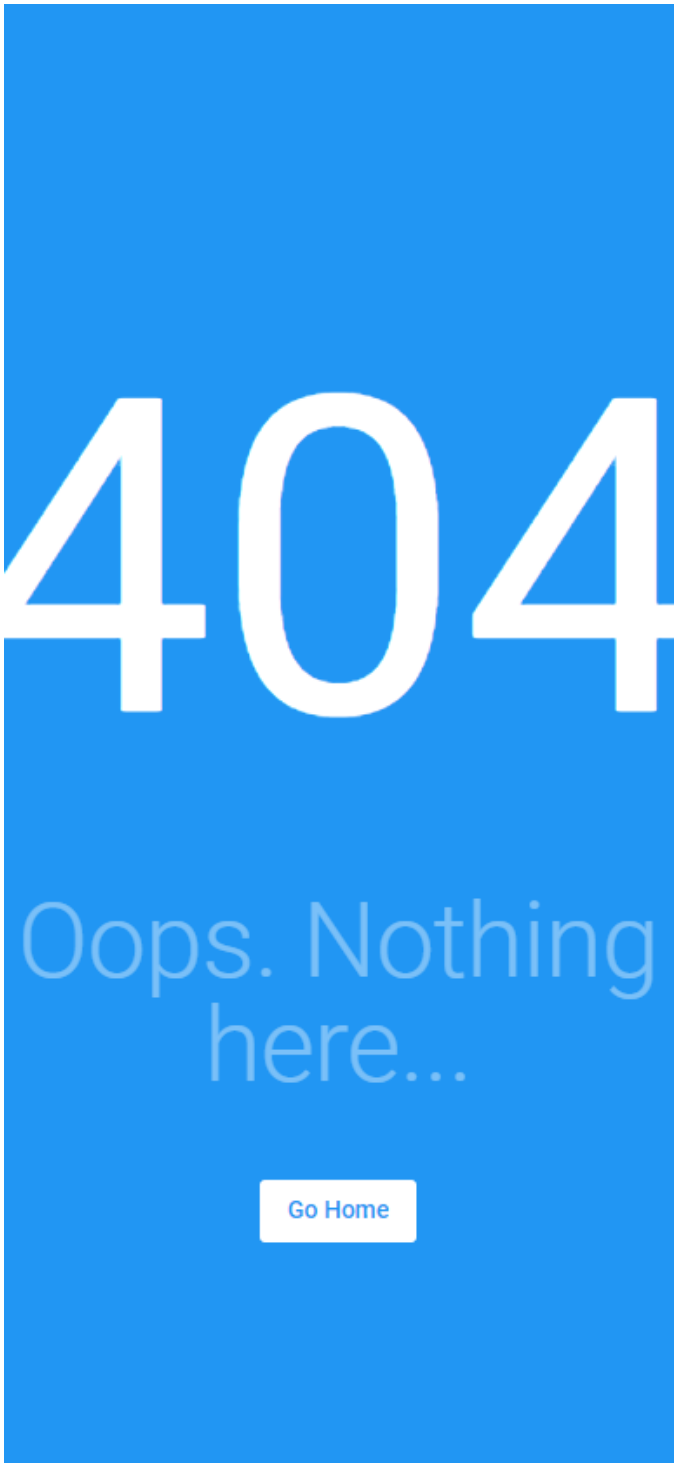
2-6 Ανέβασμα αρχείου - No camera support

2.3.1. Geolocation

Για να βρούμε την τοποθεσία στην οποία τραβήχτηκε η φωτογραφία, χρησιμοποιούμε το geolocation API που υποστηρίζει ο browser. Επιπροσθέτως, επειδή το geolocation API επιστρέφει τις συντεταγμένες που δεν είναι πολύ φιλικές για το χρήστη, κάνουμε χρήση ενός δεύτερου API για να βρούμε βάσει αυτών των συντεταγμένων την ονομασία της περιοχής. Για να το πετύχουμε αυτό κάνουμε χρήση του δωρεάν geocode.xyz API στο οποίο μπορείς να τροφοδοτήσεις τις συντεταγμένες που μας έχει δώσει το geolocation API του browser και, στη συνέχεια, να μας δώσει την ονομασία της τοποθεσίας βάσει των συντεταγμένων αυτών. Για να βελτιώσουμε την εμπειρία χρήσης του χρήστη όταν πατάει το κουμπί για να βρει η εφαρμογή την τοποθεσία, το εικονίδιο αλλάζει, ώστε να φαίνεται ότι προσπαθεί να την φορτώσει και ότι γίνεται κλήση στο API. Με αυτόν τον τρόπο αποφεύγουμε να γίνει νέα κλήση στο API, πριν ολοκληρωθεί η πρώτη και ταυτόχρονα ο χρήστης καταλαβαίνει ότι η εφαρμογή προσπαθεί να κάνει επικοινωνία. Αν ο browser του χρήστη δεν υποστηρίζει το geolocation API, τότε δεν εμφανίζεται το εικονίδιο και θα πρέπει να εισάγει την τοποθεσία του χειροκίνητα.

2.4. Σελίδα Σφάλματος (Error Screen)

Σε περίπτωση που ο χρήστης οδηγηθεί σε μία διαφορετική σελίδα από τις home page και camera page, π.χ. πληκτρολογήσει λάθος URL, τότε θα του εμφανιστεί μία σελίδα σφάλματος που θα τον ενημερώνει ότι δε βρίσκεται στη σωστή σελίδα. Για να βοηθήσουμε το χρήστη, υπάρχει κουμπί που τον κάνει ανακατεύθυνση στο home page, για να μπορέσει να συνεχίσει τη χρήση της εφαρμογής.



2-7 Σελίδα Σφάλματος

2.5. Firebase

Για τη λειτουργία της εφαρμογή γίνεται χρήση της πλατφόρμας Firebase που ανήκει στη Google. Η πλατφόρμα αυτή προσφέρει πολλές διαφορετικές υπηρεσίες δωρεάν (με κάποιους περιορισμούς) αλλά και επί πληρωμή με αυξημένες δυνατότητες. Ουσιαστικά είναι μία all-in one σουίτα υπηρεσιών που σκοπό έχουν να αντικαταστήσουν το παραδοσιακό back-end infrastructure που έχουν οι εφαρμογές. Ο στόχος είναι να μπορούν οι δημιουργοί εφαρμογών να υλοποιήσουν την ιδέα τους γρήγορα χωρίς να ξοδέψουν μεγάλο χρονικό διάστημα και μεγάλα χρηματικά ποσά στην υλοποίηση της υποδομής. Ταυτόχρονα επωφελούνται από τις τεχνολογίες και την ασφάλεια που μπορεί να προσφέρει μία εταιρεία του μεγέθους και κύρους της Google.

Επίσης, ένα άλλο σημαντικό χαρακτηριστικό είναι ότι η χρέωση των υπηρεσιών είναι ανάλογη της χρήσης που γίνεται. Δηλαδή, όταν έχουμε λίγους χρήστες που οι ανάγκες σε πόρους (resources) είναι χαμηλές, μπορούμε να παραμείνουμε ακόμα και στα δωρεάν επίπεδα που προσφέρει η υπηρεσία. Όσο αυξάνονται οι χρήστες της εφαρμογής, οι ανάγκες σε πόρους αυξάνονται αυτόματα, γνωστό και ως scaling, με την αντίστοιχη αναλογική χρέωση. Αυτό είναι ένα σημαντικό πλεονέκτημα, γιατί αφαιρεί από τους δημιουργούς της εφαρμογής το άγχος της παρακολούθησης των αναγκαίων πόρων για την αποτελεσματική λειτουργία της εφαρμογής. Ως αποτέλεσμα δε χρειάζεται να κάνουν και πρόβλεψη των αναγκών και να αγοράζουν εκ των προτέρων υλικό-υποδομή, η οποία μπορεί για αρκετό χρονικό διάστημα να μην είναι απαραίτητη, κάτι το οποίο από μόνο του προσθέτει κόστος. Στην παρακάτω εικόνα μπορούμε να δούμε τις χρεώσεις της πλατφόρμας για τις υπηρεσίες που χρησιμοποιήσαμε στην εφαρμογή.

Products	No-cost Spark Plan Generous limits to get started	Pay as you go Blaze Plan Calculate pricing for apps at scale ✓ No-cost usage from Spark plan included*
Cloud Firestore		
Stored data	1 GiB total	No-cost up to 1 GiB total Then $0.108 per additional GiB
Network egress	10 GiB/month	No-cost up to 10 GiB/month Then Google Cloud pricing
Document writes	20K writes/day	No-cost up to 20K writes/day Then Google Cloud pricing
Document reads	50K reads/day	No-cost up to 50K reads/day Then Google Cloud pricing
Document deletes	20K deletes/day	No-cost up to 20K deletes/day Then Google Cloud pricing
Cloud Functions		
Invocations	*Not applicable*	No-cost up to 2M/month Then $0.40/million
GB-seconds		No-cost up to 400K/month Then Google Cloud pricing
CPU-seconds		No-cost up to 200K/month Then Google Cloud pricing
Outbound networking		No-cost up to 5GB/month Then $0.12/GB
Cloud Build minutes		No-cost up to 120min/day Then $0.003/min
Container storage		Usage has costs $0.026/GB
		*Pricing varies based on location
Hosting		
Storage	10 GB	$0.026/GB
Data transfer	360 MB/day	$0.15/GB
Custom domain & SSL	✓	✓
Multiple sites per project	✓	✓
Cloud Storage		
GB stored	5 GB	$0.026/GB
GB downloaded	1 GB/day	$0.12/GB
Upload operations	20K/day	$0.05/10k
Download operations	50K/day	$0.004/10k
Multiple buckets per project	✗	✓

2-8 Τιμολόγηση Firebase

Για τις ανάγκες της εφαρμογής θα κάνουμε χρήση του δωρεάν επιπέδου υπηρεσιών. Οι υπηρεσίες που χρησιμοποιήθηκαν είναι οι εξής:

- **Firebase Firestore:** η οποία είναι μία μη σχεσιακή βάση δεδομένων document type.
- **Firebase Cloud Storage:** η οποία μας δίνει τη δυνατότητα να αποθηκεύουμε αρχεία (φωτογραφίες στην περίπτωση της εφαρμογής μας) στους cloud server της Google.
- **Firebase Hosting:** η οποία μας δίνει τη δυνατότητα να ανεβάσουμε online την εφαρμογή μας στους server της Google, ώστε να έχουμε πρόσβαση σε αυτή με οποιοδήποτε browser απλά με μία επίσκεψη στο URL της.
- **Firebase Cloud Functions:** η οποία μας δίνει τη δυνατότητα να φτιάξουμε το δικό μας API με τα endpoints του σε serverless περιβάλλον, χωρίς να χρειάζεται να κάνουμε όλες τις ρυθμίσεις που είναι απαραίτητες όταν φτιάχνουμε ένα server, π.χ. με express.js και αντίστοιχα ότι είναι αναγκαίο σε hardware.

2.6. Application Programming Interface (API)

Ως Application Programming Interface ή όπως είναι ευρέως γνωστά με το ακρωνύμιο API, ορίζουμε ένα είδος λογισμικού το οποίο λειτουργεί σαν ενδιάμεσος και σκοπό έχει να επιτρέπει την επικοινωνία και ανταλλαγή πληροφορίας μεταξύ δύο προγραμμάτων/συστημάτων κάνοντας χρήση ενός πρωτοκόλλου. Υπάρχουν πολλά και διαφορετικά APIs με διαφορετικές δυνατότητες (weather APIs, crypto APIs, κτλ.), τεχνολογίες και σχεδιασμό (Rest APIs, SOAP APIs, κτλ.) που κατασκευάζονται για να καλύψουν αυτήν την ανάγκη επικοινωνίας.

Για τη λειτουργία της εφαρμογής δημιουργήθηκε ένα API το οποίο έχει τα εξής τρία endpoint:

- **posts Endpoint:** το οποίο όταν το καλούμε μας επιστρέφει όλα τα post που έχουν ανεβάσει οι χρήστες, ώστε να τα εμφανίσουμε στον τοίχο μας.
- **createPost Endpoint:** το οποίο όταν το καλούμε μας επιτρέπει τη δημιουργία και αποθήκευση του post μας στα Firebase Services που κάνουνε χρήση.
- **createSubscription Endpoint:** το οποίο το καλούμε για να ενεργοποιήσουμε το subscription service, ώστε να μπορούν οι χρήστες να εγγραφούν ως συνδρομητές στην εφαρμογή μας και να λαμβάνουν ειδοποιήσεις όταν ανεβαίνει νέο post στην εφαρμογή.

Για τη δημιουργία του API έγινε χρήση του service Firebase Cloud Functions και κάθε endpoint αντιστοιχεί και σε ένα διαφορετικό function που δημιουργήσαμε. Στην εικόνα που ακολουθεί βλέπουμε τον κώδικα που γράφτηκε για το posts endpoint, όπου τραβάει τα posts μας από τη βάση μας και τα ταξινομεί από το νεότερο στο παλαιότερο.

```
// Retrieve posts from Firebase Firestore endpoint
exports.posts = functions.https.onRequest((request, response) => {
  response.set('Access-Control-Allow-Origin', '*')

  let posts = []

  db.collection('posts').orderBy('date', 'desc').get().then(snapshot => {
    snapshot.forEach(doc => {
        posts.push(doc.data())
    })

    response.send(posts)
  })
})
```

2-9 Ανάκτηση posts endpoint

3. Progressive Web Apps (PWAs)

3.1. Web App Manifest

Κάθε PWA εφαρμογή χρειάζεται να έχει ένα manifest αρχείο το οποίο παρέχει κάποιες απαραίτητες πληροφορίες στον browser για την εφαρμογή που τρέχει. Ανάλογα με την εφαρμογή μπορεί να χρειαστεί να εισάγουμε διάφορα properties σε αυτό το αρχείο. Αυτό το αρχείο χρειάζεται για να αποκτήσει η εφαρμογή μας τη δυνατότητα εγκατάστασης. Για την παρούσα εφαρμογή χρησιμοποιήσαμε τα εξής:

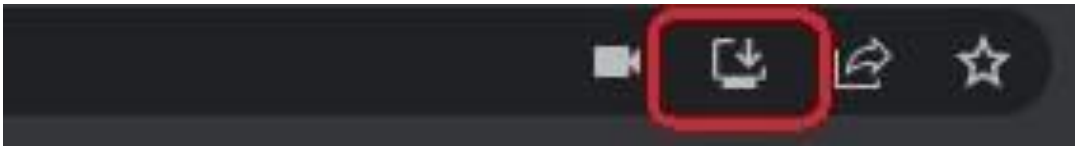
- **name:** Περιέχει το όνομα της εφαρμογής που μπορεί να εμφανιστεί στο χρήστη, π.χ. όταν την κάνει εγκατάσταση.
- **Short_name:** Περιέχει το όνομα της εφαρμογής που εμφανίζεται σε περίπτωση που δεν υπάρχει αρκετός χώρος για να εμφανιστεί το name.
- **Description:** Περιέχει μία σύντομη περιγραφή στην οποία ο δημιουργός μπορεί να αναφέρει πληροφορίες για την εφαρμογή.
- **Display:** Το προτιμητέο display mode από το δημιουργό για την εφαρμογή, δηλαδή πως θα εμφανίζει την εφαρμογή ο browser (Fullscreen κτλ.).
- **Orientation:** Ο προκαθορισμένος προσανατολισμός της εφαρμογής.
- **Background_color:** Το background χρώμα που θα χρησιμοποιεί η εφαρμογή μέχρι να φορτώσει τα CSS αρχεία και να δει τις ρυθμίσεις χρώματος από το αρχεία αυτά.
- **Theme_color:** Το βασικό θέμα χρώματος που έχει η εφαρμογή.
- **Icons:** Είναι ένα array of objects που περιέχει το path για τα εικονίδια που μπορεί να χρησιμοποιήσει η εφαρμογή ανάλογα με τις ανάγκες, π.χ. διαφορετικές αναλύσεις οθόνης.

```
manifest: {
  name: `Xoxa`,
  short_name: `Xoxa`,
  description: `A social media application for my thesis`,
  display: 'standalone',
  orientation: 'portrait',
  background_color: '#ffffff',
  theme_color: '#1e88e5',
  icons: [
    {
      src: 'icons/icon-128x128.png',
      sizes: '128x128',
      type: 'image/png'
    },
    {
      src: 'icons/icon-192x192.png',
      sizes: '192x192',
      type: 'image/png'
    },
    {
      src: 'icons/icon-256x256.png',
      sizes: '256x256',
      type: 'image/png'
    },
    {
      src: 'icons/icon-384x384.png',
      sizes: '384x384',
      type: 'image/png'
    },
    {
      src: 'icons/icon-512x512.png',
      sizes: '512x512',
      type: 'image/png'
    }
  ]
}
```

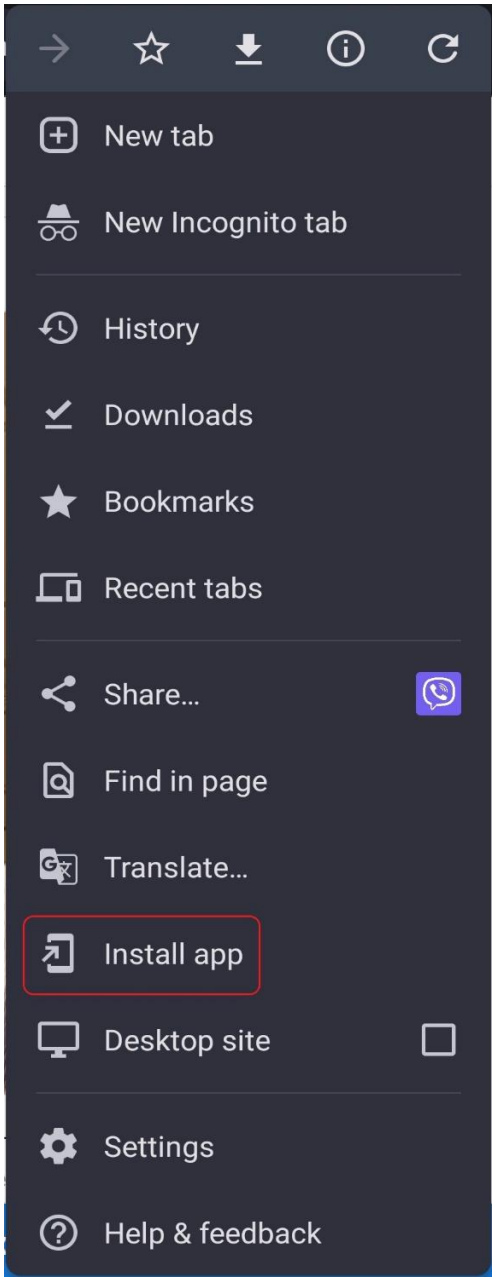
3-1 Αρχείο manifest

3.2. Home Screen Installation

Όταν ο browser δει ότι η εφαρμογή μας είναι ένα progressive web app θα μας δώσει τη δυνατότητα να την κάνουμε εγκατάσταση. Σε desktop browser θα εμφανιστεί ένα ακόμα εικονίδιο στη μπάρα διευθύνσεων, ενώ σε κινητό θα πρέπει να πατήσουμε τις 3 τελείες για να εμφανιστεί το μενού με τις επιλογές.



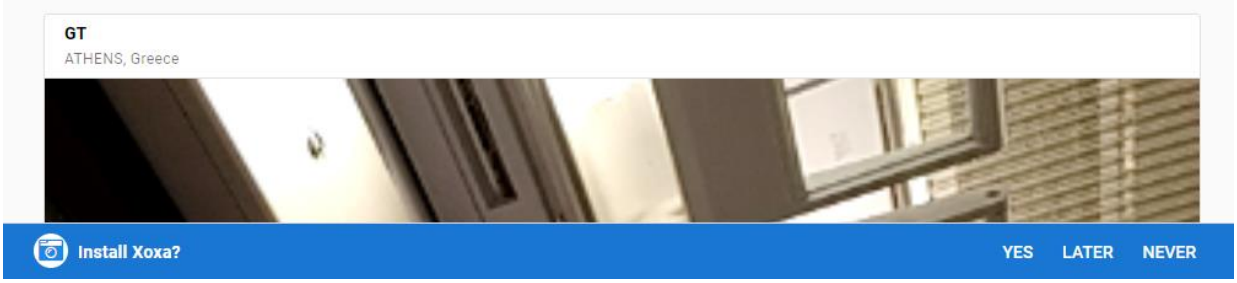
3-2 Εικονίδιο για εγκατάσταση σε desktop



3-3 Συντόμευση για εγκατάσταση σε κινητό

Στην εφαρμογή που δημιουργήσαμε υπάρχει και μία ακόμα δυνατότητα εγκατάστασης. Δημιουργήθηκε ένα banner στο κάτω μέρος της οθόνης το οποίο ενημερώνει το χρήστη ότι μπορεί να κάνει εγκατάσταση την εφαρμογή εμφανίζοντας την ερώτηση “Install Xoxa?” και του δίνει τρεις επιλογές (actions):

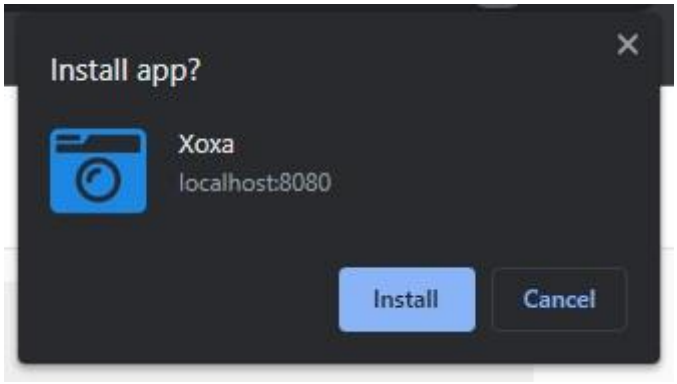
- **Yes:** επιλογή για να κάνει εγκατάσταση την εφαρμογή.
- **Later:** επιλογή για να κρύψει προσωρινά το banner αλλά να το εμφανίσει ξανά αργότερα σε επόμενη χρήση της εφαρμογής.
- **Never:** επιλογή η οποία δεν κάνει εγκατάσταση την εφαρμογή και δεν εμφανίζει ξανά το banner εγκατάστασης στο χρήστη.



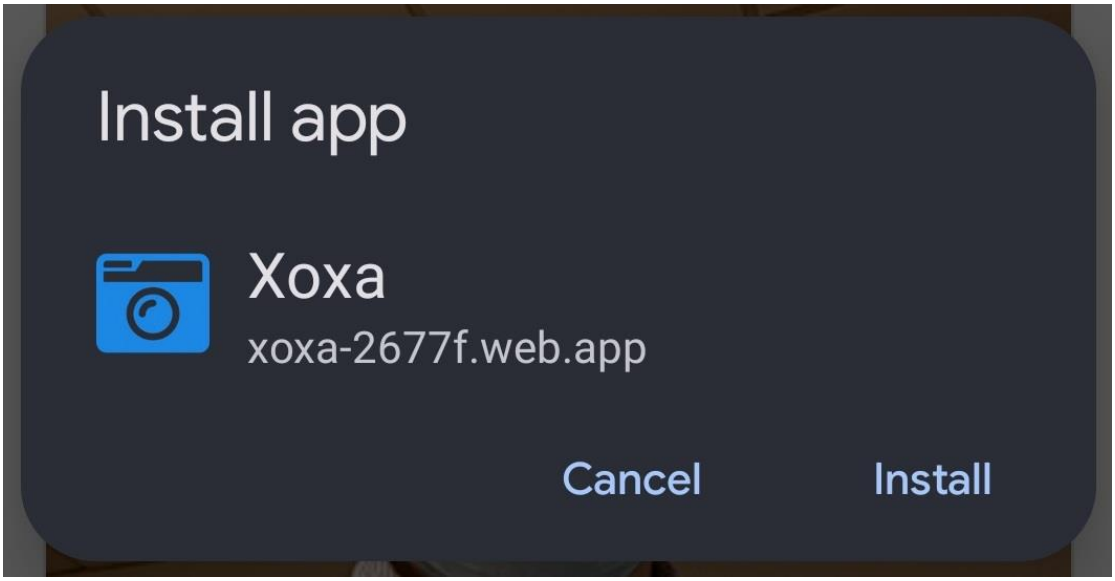
3-4 Banner εγκατάστασης σε desktop

3-5 Banner εγκατάστασης σε κινητό

Σε κάθε περίπτωση όταν ο χρήστης επιλέξει εγκατάσταση της εφαρμογής θα του εμφανιστεί μία ειδοποίηση αν θέλει να κάνει εγκατάσταση σε μορφή alert.



3-6 Alert εγκατάστασης σε desktop

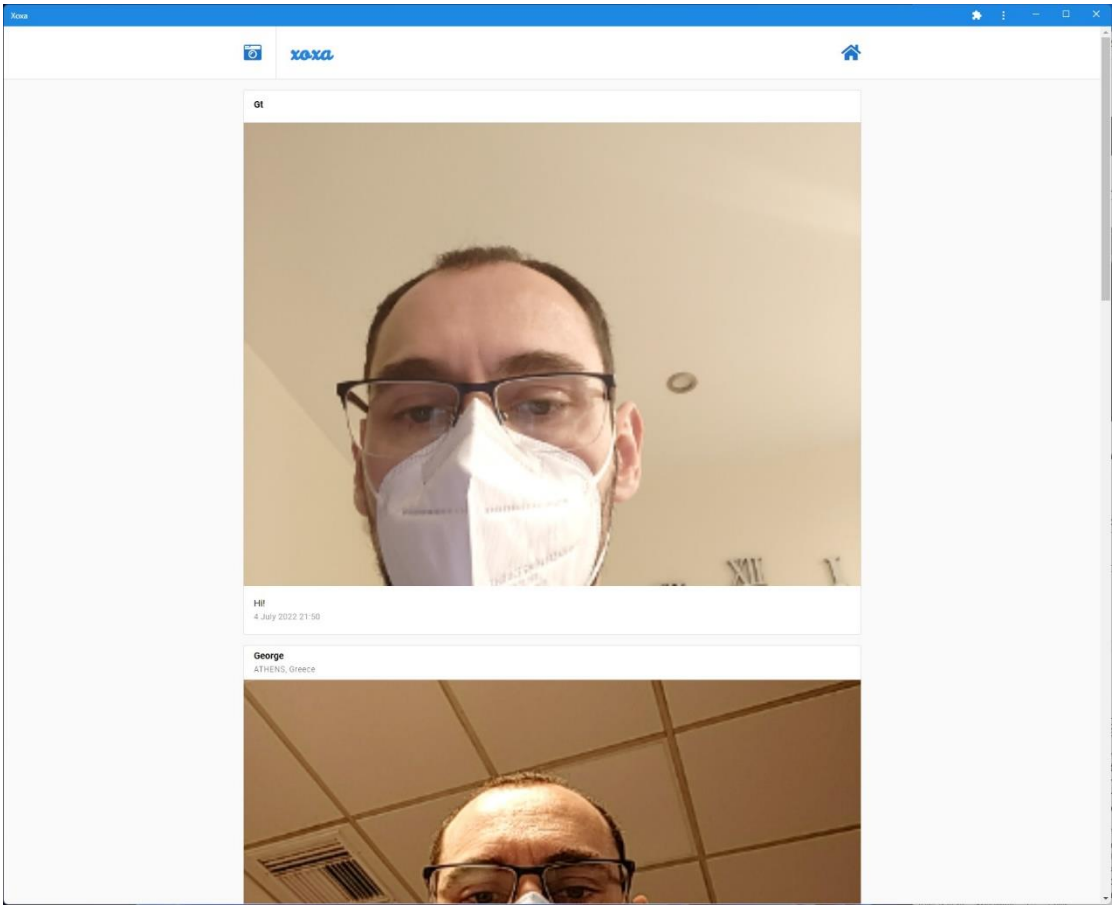


3-7 Alert εγκατάστασης σε κινητό

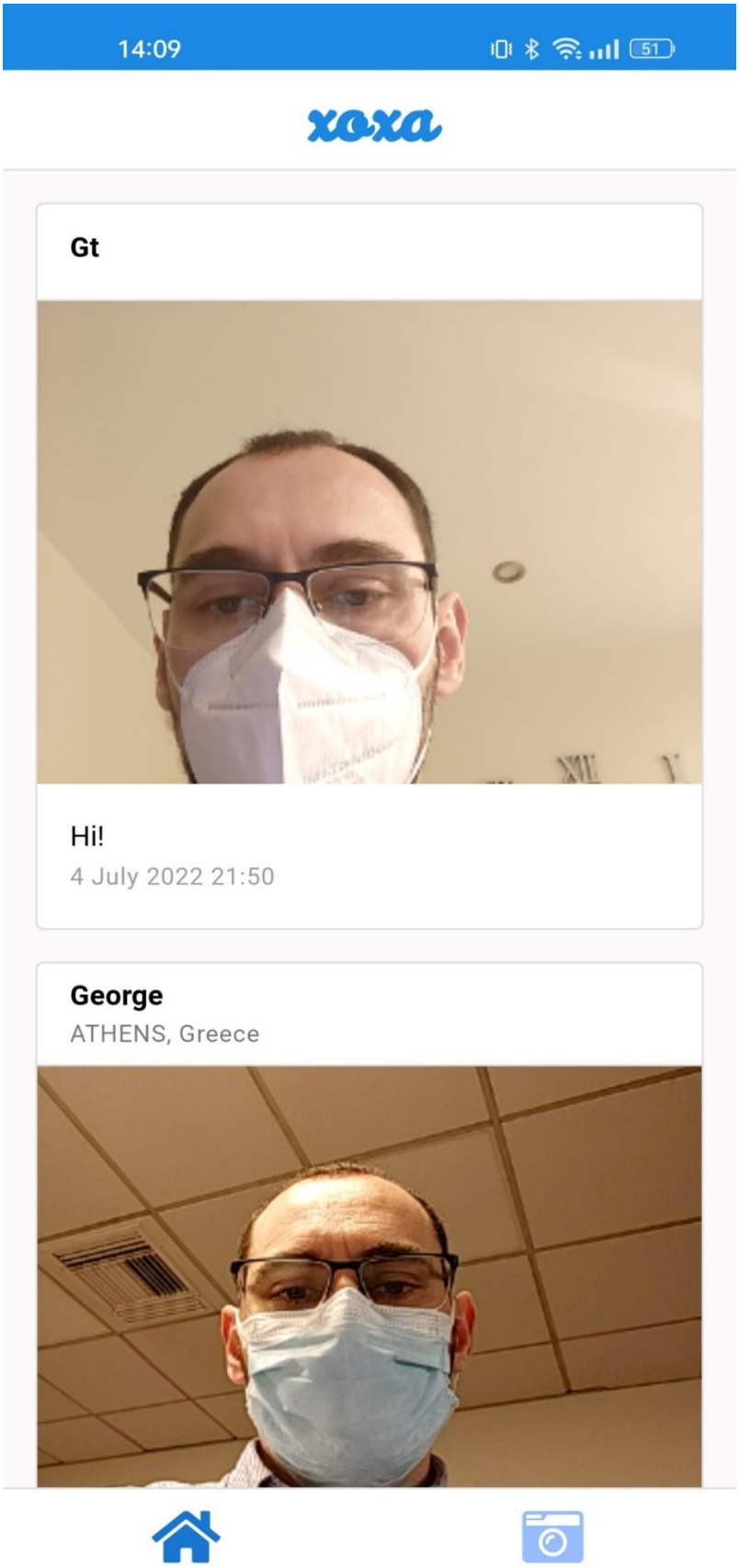
Τέλος, όταν ολοκληρωθεί η εγκατάσταση, εμφανίζεται το ακόλουθο εικονίδιο στην επιφάνεια εργασίας από το οποίο μπορεί ο χρήστης να εκκινήσει την εφαρμογή.

3-8 Εικονίδιο εκκίνησης εφαρμογής κινητό

Μετά την εγκατάσταση αν ο χρήστης πατήσει το εικονίδιο για να εκκινήσει την εφαρμογή, το περιβάλλον που θα δει δε θα είναι του browser αλλά θα θυμίζει εφαρμογή, όπως μπορούμε να δούμε στις εικόνες που ακολουθούν:



3-9 Η εφαρμογή σε desktop



3-10 Η εφαρμογή σε κινητό

3.3. Service Workers

Οι service workers είναι τα πιο σημαντικά εργαλεία που έχουμε στη δημιουργία progressive web apps. Ουσιαστικά είναι αρχεία JavaScript τα οποία λειτουργούν ως proxy servers και τοποθετούνται αναμεσά στην επικοινωνία που έχουν οι web εφαρμογές με το browser και το δίκτυο. Είναι αυτοί που επιτρέπουν στις progressive web apps να έχουν όλες αυτές τις δυνατότητες και χαρακτηριστικά, όπως για παράδειγμα να λειτουργούν offline.

Αυτό το καταφέρνουν, γιατί τρέχουν στο background σε διαφορετικό thread από το main thread της JavaScript που χρησιμοποιείται για να τρέξει η εφαρμογή. Αυτό σημαίνει ότι οι service workers είναι non-blocking και ασύγχρονοι αλλά δεν έχουν πρόσβαση στο DOM της εφαρμογής. Επίσης, για λόγους ασφαλείας οι service workers μπορούν να τρέξουν μόνο πάνω από HTTPS σύνδεση.

Όσο τρέχουν στο background μπορούν να ακούσουν κάποια events τα οποία μας επιτρέπουν να δώσουμε σε web εφαρμογή χαρακτηριστικά native. Αυτά είναι:

- **Fetch events:** Όλες οι κλήσεις που γίνονται, π.χ. σε κάποιο API, μπορούν να περάσουν και να καταγραφούν από το service worker.
- **Push notification events:** Μας επιτρέπει να εμφανίζουμε στο χρήστη ειδοποιήσεις ακόμα και όταν έχει την εφαρμογή κλειστή.
- **Notification Interactions events:** Όταν ο χρήστης έχει κάποια αλληλεπίδραση με κάποια ειδοποίηση που έλαβε, αυτή καταγράφεται και μας δίνεται η δυνατότητα να ανοίξουμε την εφαρμογή, αν αυτή είναι κλειστή.
- **Background Sync events:** Μας επιτρέπει να αποθηκεύουμε κάποια δεδομένα για μετέπειτα χρήση, π.χ. θέλει ο χρήστης να ανεβάσει post αλλά δεν έχει internet αυτήν τη στιγμή, το post δε θα χαθεί αλλά θα ανέβει μόλις η συσκευή βρει σύνδεση internet.
- **Service Worker Lifecycle events:** Σχετίζονται με τους ίδιους τους service worker, π.χ. πότε είναι έτοιμοι ή τρέχουν.

Για να γίνει πιο εύκολη και αποτελεσματική η διαχείριση και λειτουργία των service workers, η Google δημιούργησε ένα εργαλείο που ονομάζεται Workbox, το οποίο και χρησιμοποιούμε στην παρούσα εφαρμογή. Στόχος του Workbox είναι με τη χρήση κάποιον module και έτοιμων functions να απλοποιήσει τη χρήση όλων αυτών των event που αναφέρθηκαν προηγουμένως.

3.4. Precaching

Ένα χαρακτηριστικό των service workers είναι να αποθηκεύουν τοπικά στη μνήμη cache ένα σύνολο αρχείων όσο γίνεται η εγκατάσταση του service worker. Αυτό το χαρακτηριστικό ονομάζεται “precaching”, γιατί αποθηκεύουμε περιεχόμενο πριν τη χρήση του service worker.

Το μεγάλο πλεονέκτημα του precaching είναι ότι δίνει στους προγραμματιστές τη δυνατότητα να έχουν έλεγχο πάνω στη cache μνήμη και να αποφασίζουν για πόσο καιρό θα αποθηκεύει αρχεία ο browser πριν τα αναζητήσει στο δίκτυο. Αυτό μας επιτρέπει να έχουμε web εφαρμογές οι οποίες μπορούν να προσφέρουν λειτουργικότητα ακόμα και όταν δεν έχουν κάποια σύνδεση στο internet.

Το precaching είναι ένα χαρακτηριστικό το οποίο το προσφέρει το workbox της Google. Όταν φορτώνει για πρώτη φορά η web εφαρμογή, βλέπει ποια αρχεία πρέπει να κατεβάσει, φιλτράρει όσα επαναλαμβάνονται, τα κατεβάζει και τα αποθηκεύει τοπικά. Αυτό συμβαίνει κατά την εγκατάσταση του service worker.

3.5. Στρατηγικές Αποθήκευσης (Caching Strategies)

Με την εμφάνιση και χρήση των service workers εμφανίστηκε η ανάγκη για τους δημιουργούς progressive web app εφαρμογών να αποφασίσουν πως θα κάνουν χρήση της cache και για ποια αρχεία. Η χρήση της μνήμης cache διαχωρίζεται συνήθως στις παρακάτω πέντε διαφορετικές στρατηγικές:

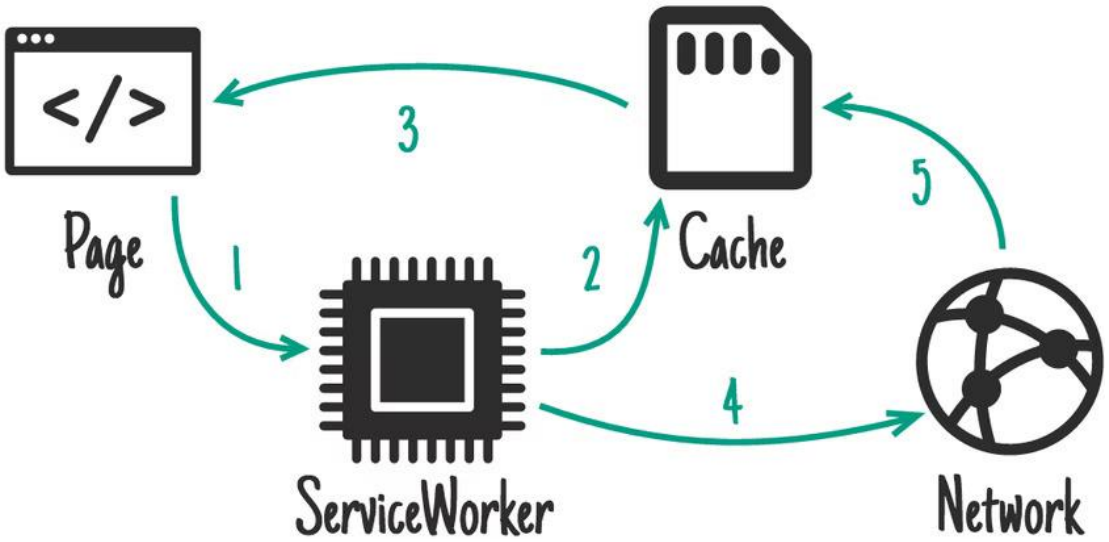
- **Stale-While-Revalidate**
- **Cache First (Cache Falling Back to Network)**
- **Network First (Network Falling Back to Cache)**
- **Network Only**
- **Cache Only**

3.5.1. Stale-While-Revalidate

Με αυτήν τη στρατηγική cache, ο service worker θα εμφανίσει στο χρήστη πρώτα τα αρχεία που έχει αποθηκεύσει στη cache μνήμη. Μετά θα πάει να κάνει το request και θα φέρει την τελευταία έκδοση των αρχείων, αν έχουν γίνει αλλαγές, για να τις αποθηκεύσει στη cache. Τα ανανεωμένα αρχεία, αν υπάρχουν, θα εμφανιστούν στο χρήστη την επόμενη φορά.

Αυτή η στρατηγική είναι πιο χρήσιμη για πληροφορίες και αρχεία που δεν είναι απολύτως κρίσιμα, ώστε να θέλουμε να έχουμε πάντα την τελευταία έκδοση διαθέσιμη.

Όπως βλέπουμε και στην παρακάτω εικόνα, όταν φορτώσει η εφαρμογή, ο service worker, για να δώσει απάντηση όσο πιο γρήγορα γίνεται, θα εμφανίσει τα αρχεία που βρήκε στη μνήμη cache, αν είναι διαθέσιμα. Αν δεν είναι διαθέσιμα, τότε θα κάνει μία κλήση στο δίκτυο για να βρει τα απαραίτητα αρχεία και θα ανανεώσει την cache. Η κλήση στο δίκτυο με αυτήν τη στρατηγική γίνεται κάθε φορά ακόμα και αν ο service worker βρει τα αρχεία που αναζητούσε στη cache, ώστε να κάνει επικαιροποίηση και ανανέωση των αρχείων που έχουν αποθηκευτεί.



3-11 Stale-While-Revalidate strategy

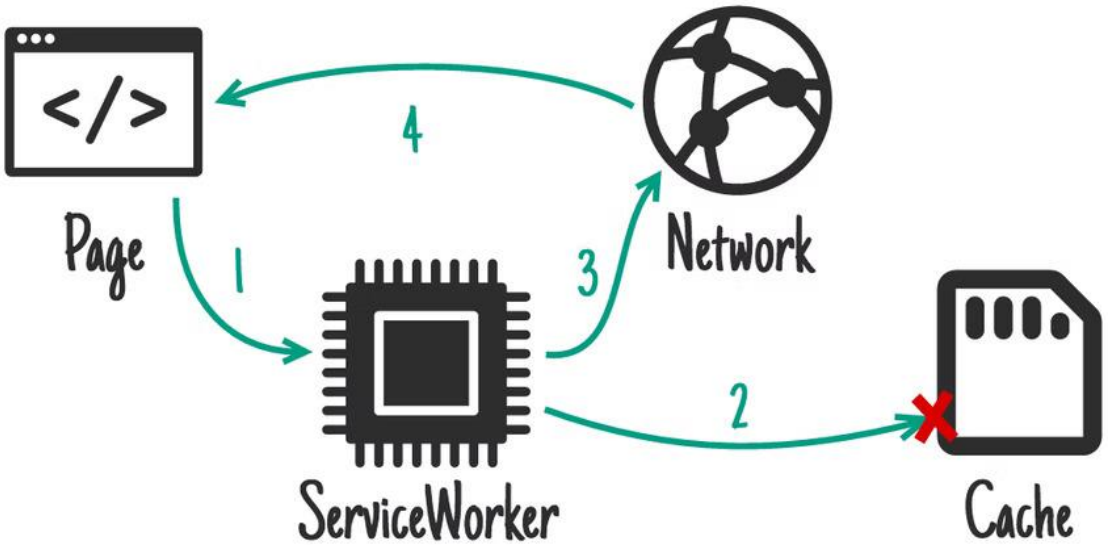
Στην εφαρμογή κάνουμε χρήση αυτής της στρατηγικής σε όλες τις κλήσεις που ξεκινάνε από “http”, δηλαδή στην πλειοψηφία των κλήσεων, όπως μπορούμε να δούμε και στον παρακάτω κώδικα. Έτσι, ο χρήστης ακόμα και όταν δεν έχει internet, θα μπορέσει να ανοίξει την εφαρμογή και να πάρει κάποιες πληροφορίες.

3-12 http requests Cache Strategy

3.5.2. Cache First (Cache Falling Back to Network)

Με αυτήν τη στρατηγική cache ο service worker θα αναζητήσει πρώτα τα απαραίτητα αρχεία στη μνήμη cache και δε θα τα αναζητήσει στο δίκτυο. Μόνο σε περίπτωση που δε βρεθούν τα αρχεία που αναζητούμε στη μνήμη cache, θα γίνει μία κλήση στο δίκτυο, ώστε να πάρουμε τα αρχεία και να τα αποθηκεύσουμε στη cache για να γίνει από εκεί χρήση την επόμενη φορά.

Αυτή η στρατηγική είναι χρήσιμη για αρχεία που δεν περιμένουμε να γίνουν update ή για αρχεία που δεν είναι κρίσιμο να έχουμε την τελευταία έκδοση. Στην εικόνα που ακολουθεί βλέπουμε πως δουλεύει αυτή η στρατηγική.



3-13 Cache First Strategy

Στην εφαρμογή μας κάνουμε χρήση αυτής της στρατηγικής στα αρχεία font της Google που κάνουμε χρήση μιας και δεν περιμένουμε να υπάρχουν αλλαγές σε αυτά. Ακολουθεί ο σχετικός κώδικάς.

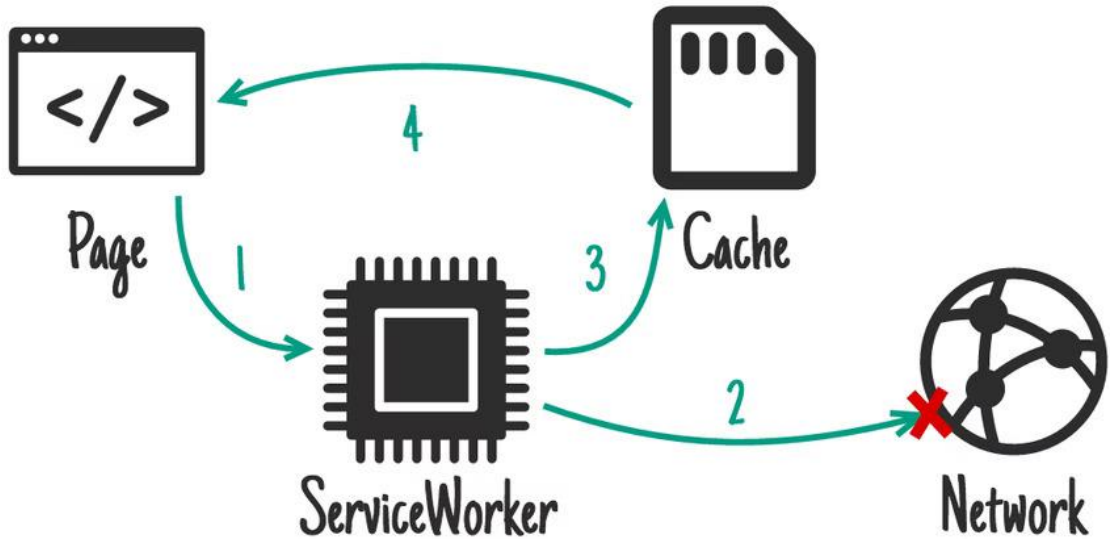
```
// Cache First (Cache Falling Back to Network)
registerRoute(
    ({url}) => url.host.startsWith('fonts.g'),
    new CacheFirst({
        cacheName: 'google-fonts',
        plugins: [
            new ExpirationPlugin({
                maxEntries: 30,
            }),
            new CacheableResponsePlugin({
                statuses: [0, 200],
            })
        ],
    })
)
```

3-14 Fonts Cache Strategy

3.5.3. Network First (Network Falling Back to Cache)

Με αυτήν τη στρατηγική cache ο service worker θα αναζητήσει τα αρχεία πρώτα στο δίκτυο και αν η κλήση γίνει με επιτυχία θα αποθηκεύσει την απάντηση στην cache. Σε περίπτωση που η κλήση στο δίκτυο δε γίνει με επιτυχία, τότε θα αναζητήσει τα απαραίτητα αρχεία στη μνήμη cache για να τα εμφανίσει στην εφαρμογή.

Η στρατηγική αυτή είναι χρήσιμη για αρχεία στα οποία θέλουμε πάντα να έχουμε την τελευταία έκδοση αλλά να μπορούμε να δείξουμε κάτι στο χρήστη σε περίπτωση που η κλήση στο δίκτυο δε γίνει με επιτυχία.



3-15 Network First Strategy

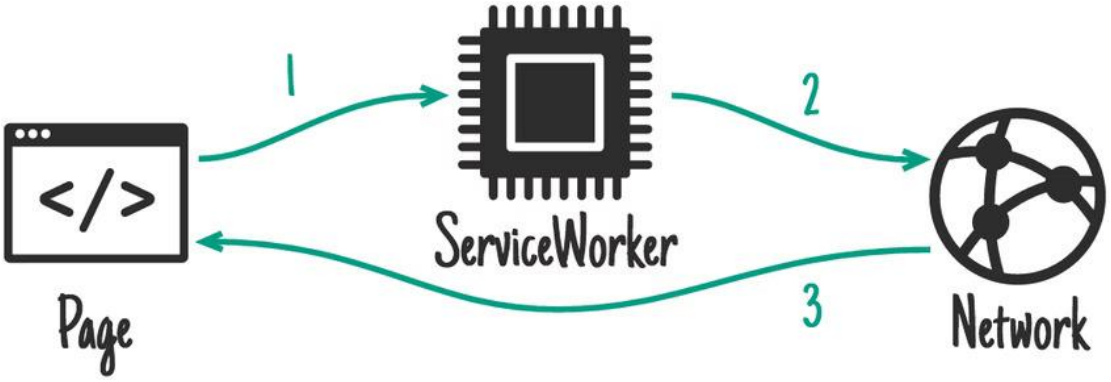
Στην εφαρμογή μας κάνουμε χρήση αυτής της στρατηγικής στα posts, έτσι ώστε ο χρήστης να έχει πάντα διαθέσιμα τα τελευταία post που έχουν ανέβει στην εφαρμογή. Σε περίπτωση που η εφαρμογή αποτύχει να φέρει τα τελευταία post από το δίκτυο, θα δείξει τα post που έχει αποθηκεύσει στη μνήμη cache.

```
// Network First (Network Falling Back to Cache)
registerRoute(
    ({url}) => url.pathname.startsWith('/posts'),
    new NetworkFirst()
)
```

3-16 Posts Cache Strategy

3.5.4. Network Only

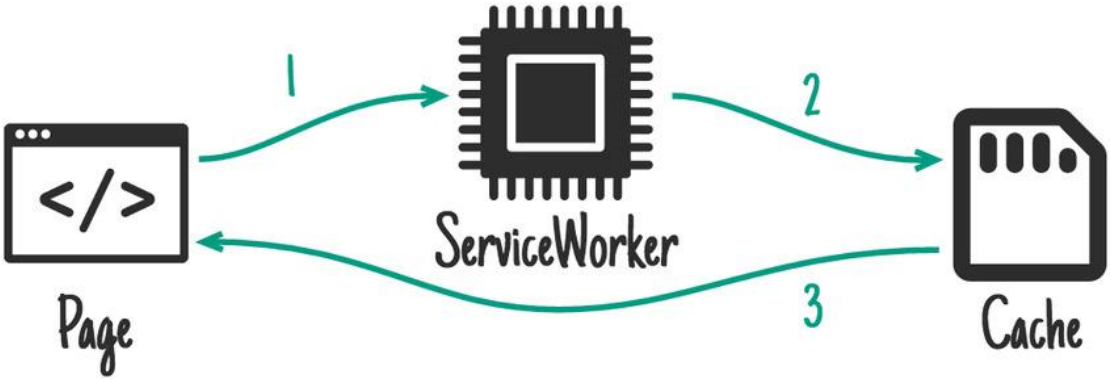
Με αυτήν τη στρατηγική ο service worker κάνει πάντα κλήση στο δίκτυο και δεν χρησιμοποιεί ποτέ τη μνήμη cache για να διαβάσει αρχεία. Είναι χρήσιμη μόνο σε περιπτώσεις που χρειάζεται να γίνεται πάντα κλήση στο δίκτυο και δε θέλουμε να παρεμβάλλεται η cache. Στην εφαρμογή μας δε γίνεται χρήση αυτής της στρατηγικής.



3-17 Network Only Strategy

3.5.5. Cache Only

Σε αυτήν τη στρατηγική ο service worker κάνει πάντα αναζήτηση των αρχείων από τη μνήμη cache και δεν κάνει ποτέ κλήση στο δίκτυο. Χρήση αυτής της στρατηγικής γίνεται σπάνια και δεν κάνουμε χρήση της στην εφαρμογή.



3-18 Cache Only Strategy

3.6. Background Sync

Ένα από τα σημαντικότερα χαρακτηριστικά των progressive web apps είναι το background sync. Σε περίπτωση που η εφαρμογή στείλει δεδομένα στο server και η αποστολή αποτύχει, π.χ. γιατί δεν έχουμε σύνδεση στο δίκτυο ή γιατί ο server δεν είναι διαθέσιμος, ιδανικά θα θέλαμε να στείλουμε τα δεδομένα αργότερα όταν αυτό θα είναι δυνατόν. Με το background sync μπορούμε να κάνουμε ακριβώς αυτό.

Κάνοντας χρήση του BackgroundSync API όταν ο service worker ανιχνεύσει ότι η κλήση στο server απέτυχε, καταγράφει αυτό το event και επιτρέπει να γίνει αυτόματα η κλήση αργότερα όταν το δίκτυο είναι ξανά διαθέσιμο. Ο συγχρονισμός των δεδομένων που έχουν αποθηκευτεί τοπικά με το server γίνεται ακόμα και αν ο χρήστης έχει φύγει από την εφαρμογή.

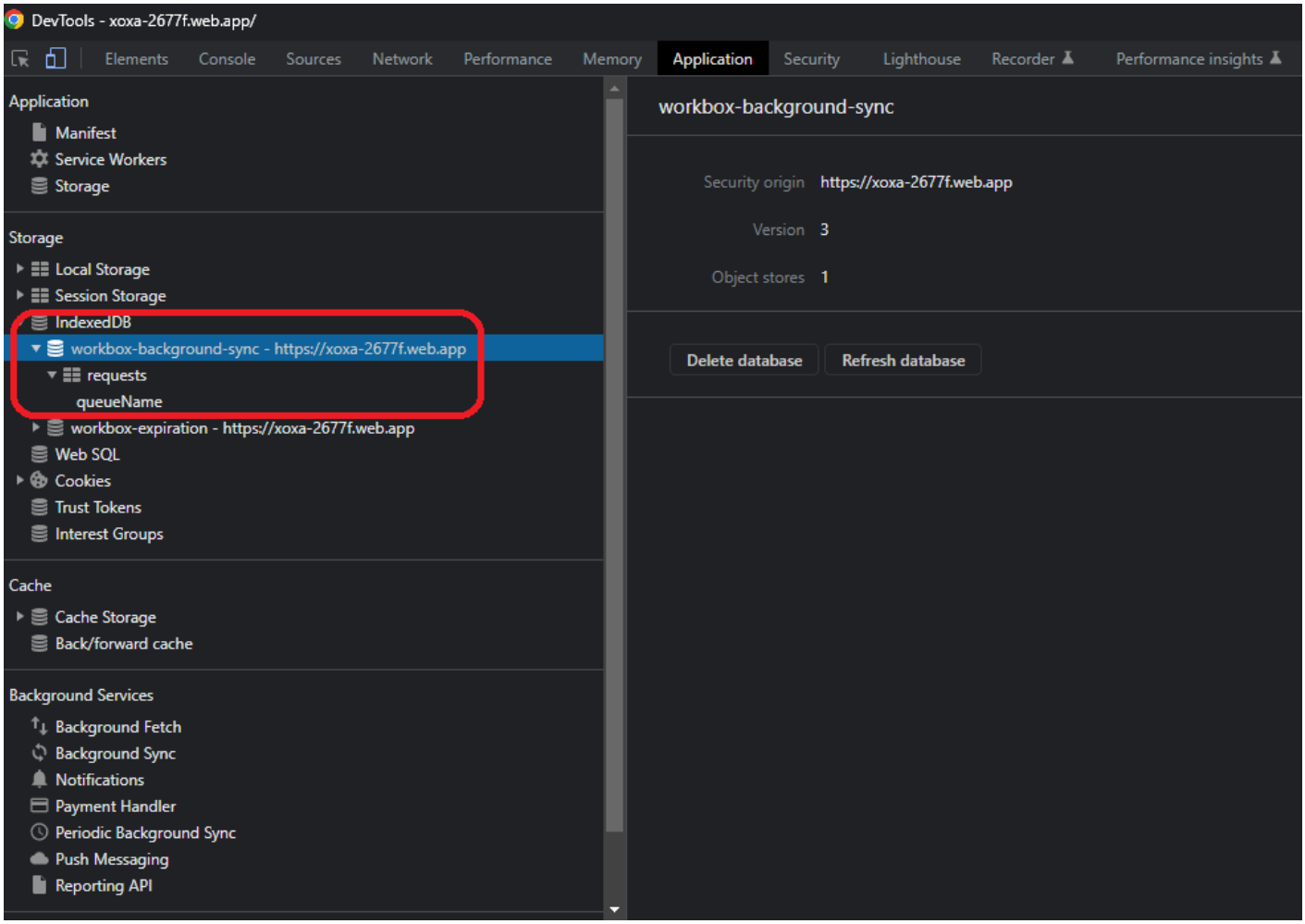
Το μεγαλύτερο πρόβλημα με το Background Sync, όπως βλέπουμε και στην εικόνα που ακολουθεί, είναι ότι δεν έχει μεγάλη υποστήριξη.

3-19 Υποστήριξη Background Sync

Παρατηρούμε ότι την καλύτερη υποστήριξη την έχουμε σε Chrome, Opera και σε κάποιους mobile browsers. Παρά την έλλειψη υποστήριξης μπορούμε να χρησιμοποιήσουμε την εφαρμογή και χωρίς αυτό το χαρακτηριστικό. Μπορεί να γίνει έλεγχος με μία απλή γραμμή κώδικα για να δούμε αν ο browser μας υποστηρίζει αυτό το χαρακτηριστικό.

3-20 Έλεγχος υποστήριξης Background Sync

Ο τρόπος που δουλεύει το background sync είναι να ακούει τα σχετικά events και σε περίπτωση που είμαστε offline βάζει τα δεδομένα μας σε μία “ουρά”, όπως ονομάζεται, π.χ. όταν έχουμε να ανεβάσουμε παραπάνω από ένα post offline, και να τα αποθηκεύει τοπικά με μία βάση που μας προσφέρει ο browser και ονομάζεται IndexedDB.



3-21 IndexedDB database

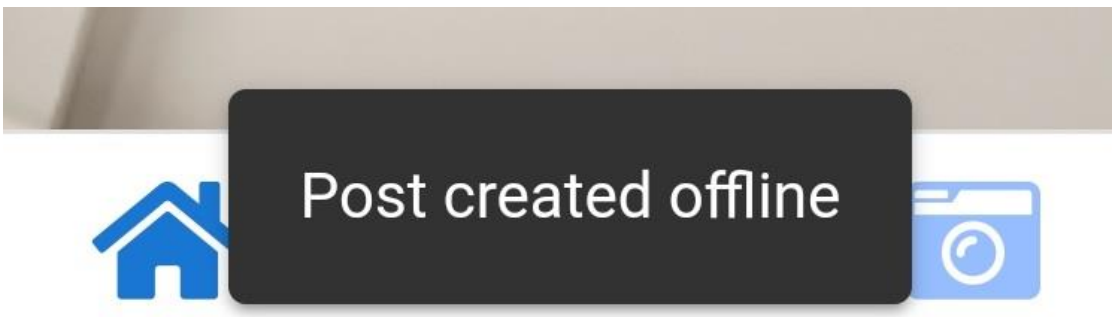
Ακολουθεί και ο σχετικός κώδικας που μας επιτρέπει να το κάνουμε αυτό.

```
// Queue for creating posts
let createPostQueue = null

if (backgroundSyncSupported) {
    createPostQueue = new Queue('createPostQueue', {
        onSync: async ({queue}) => {
          let entry;
          while (entry = await queue.shiftRequest()) {
            try {
              await fetch(entry.request);
              console.log('Replay successful for request', entry.request);
              const channel = new BroadcastChannel('sw-messages');
              channel.postMessage({msg: 'offline-post-uploaded'});
            } catch (error) {
              console.error('Replay failed for request', entry.request, error);

              // Put the entry back in the queue and re-throw the error:
              await queue.unshiftRequest(entry);
              throw error;
            }
          }
          console.log('Replay complete!');
        }
    })
}
```

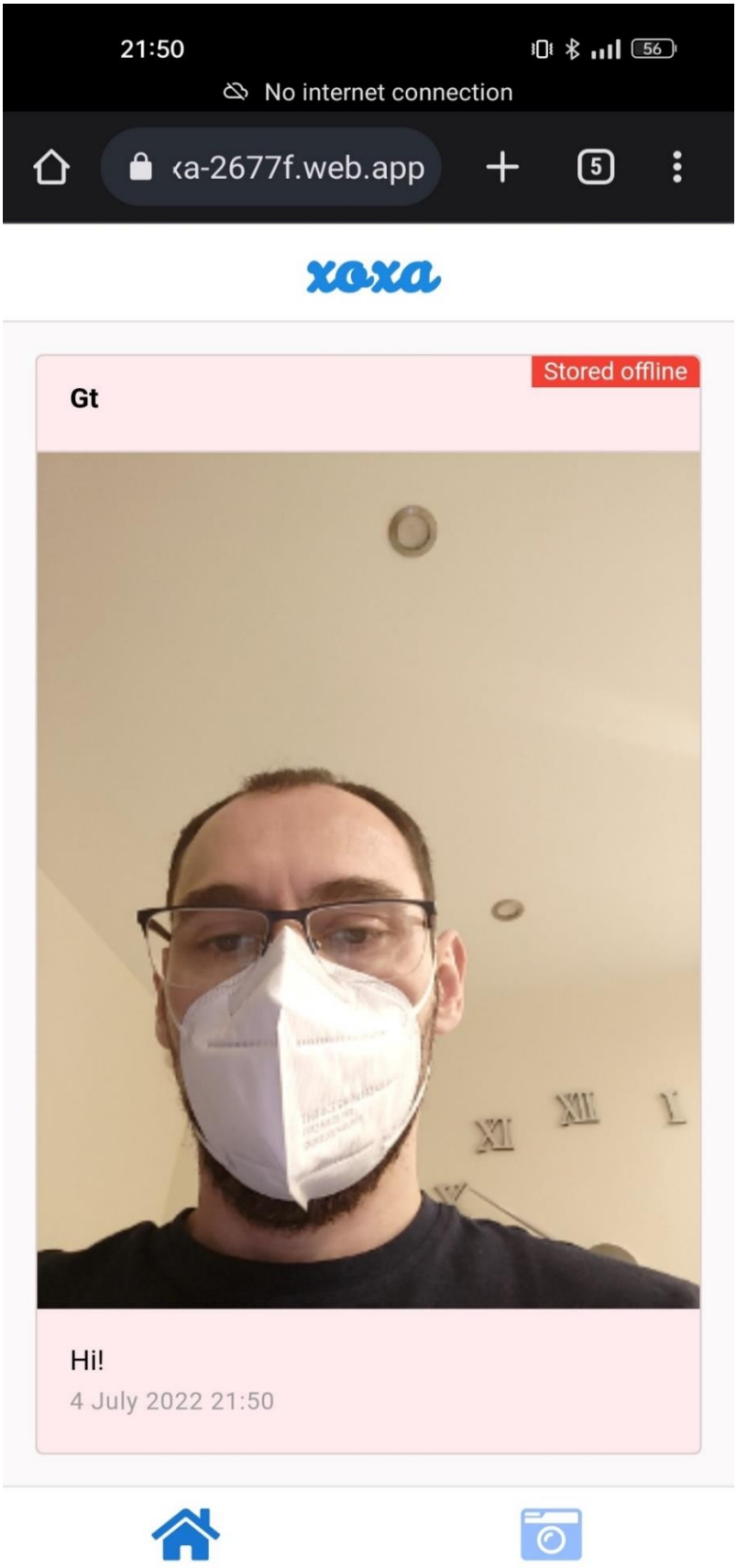
3-22 Υλοποίηση Background Sync

Στην εφαρμογή που δημιουργήσαμε υπάρχει η δυνατότητα ο χρήστης να κάνει χρήση αυτού του χαρακτηριστικού. Μπορεί να τραβήξει φωτογραφία και όταν προσπαθήσει να την ανεβάσει, σε περίπτωση που δεν έχει δίκτυο, δε θα πάρει κάποιο σφάλμα. Αντίθετα, θα δει το post να έχει ανέβει στο τοίχο του με τη σήμανση ότι είναι ακόμα offline. Όταν αργότερα η συσκευή αποκτήσει σύνδεση με το server, τότε θα φύγει η offline σήμανση και θα δουν το post και οι υπόλοιποι χρήστες.

Ακολουθούν οι εικόνες με την ειδοποίηση του χρήστη ότι το post του δημιουργήθηκε offline και η ειδική σήμανση που παίρνει το post όταν είναι offline.



3-23 Offline post alert



3-24 Offline post

3.7. Push Notifications

Το push notifications αν και είναι και αυτά ειδοποιήσεις έχουν διαφορές από τα απλά notifications.

Τα απλά notifications έχουν τα εξής χαρακτηριστικά και απαιτήσεις:

- Χρειάζεται να πάρουμε permission από το χρήστη για να μπορέσουμε να τα δείξουμε.
- Μπορούν να εμφανιστούν οποιαδήποτε στιγμή αλλά μόνο όταν ο χρήστης χρησιμοποιεί την εφαρμογή.
- Μπορούν να ενεργοποιηθούν απλά γράφοντας JavaScript στον κώδικα της εφαρμογής μας.
- Ελάχιστες απαιτήσεις, δε χρειάζεται συνδρομή από τη μεριά του χρήστη για να εμφανιστεί, δε χρειάζεται backend server ή κάποιος push server.

Τα push notifications διαφοροποιούνται και έχουν τα εξής χαρακτηριστικά και απαιτήσεις:

- Χρειαζόμαστε permission από το χρήση για να τα εμφανίσουμε.
- Στέλνονται σε όλους τους χρήστες που έχουν εγγραφεί στις ειδοποιήσεις μας την ίδια στιγμή.
- Εμφανίζονται οποιαδήποτε στιγμή, ακόμα και όταν ο χρήστης δε χρησιμοποιεί την εφαρμογή εκείνη τη στιγμή.
- Έχουν αρκετές απαιτήσεις όπως:
  - Πρέπει ο κάθε χρήστης να έχει συνδρομή σε αυτές για να εμφανιστούν.
  - Χρειαζόμαστε μία βάση για να αποθηκεύσουμε τις συνδρομές.
  - Χρειαζόμαστε backend server για να στείλουμε τις ειδοποιήσεις.
  - Χρειαζόμαστε service worker για να παρακολουθεί τις ειδοποιήσεις και να τις εμφανίζει.

Επίσης, τα push notifications είναι διαθέσιμα μόνο στη συσκευή και στο browser με τον οποίο κάναμε εγγραφή. Δηλαδή, αν σε μία συσκευή κάναμε εγγραφή κάνοντας χρήση Chrome, οι ειδοποιήσεις θα είναι διαθέσιμες μόνο σε αυτό το browser και όχι π.χ. και στον Firefox, γιατί κάνουν χρήση διαφορετικού push server.

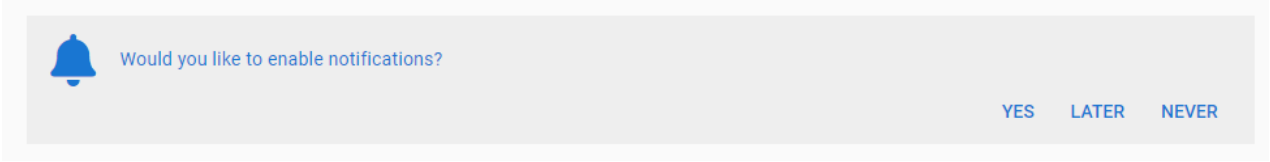
Επίσης, για λόγους ασφαλείας η επικοινωνία για τα push notifications πρέπει να είναι ασφαλής και να χρησιμοποιείται κάποιο είδος κρυπτογράφησης. Αλλιώς αν κάποιος αποκτήσει πρόσβαση στο URL του subscription του χρήστη θα μπορεί να στέλνει μη επιθυμητά notifications.

Βλέπουμε και εδώ ότι δεν έχουν όλοι οι browser υποστήριξη γι' αυτό το χαρακτηριστικό.

3-25 Υποστήριξη Push Notifications API

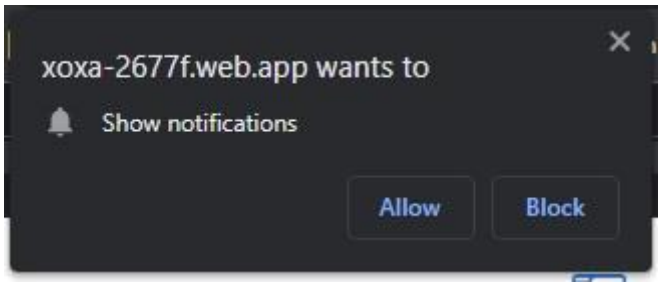
Στη παρούσα εφαρμογή ο χρήστης μπορεί να κάνει εγγραφή στα push notifications της εφαρμογής πατώντας πάνω στο banner που εμφανίζεται στο πάνω μέρος της εφαρμογής. Σε αυτό το banner έχει τρεις επιλογές:

- **Yes:** αν επιθυμεί να λαμβάνει ειδοποιήσεις.
- **Later:** αν θέλει να αποκρύψει προσωρινά το banner και να αποφασίσει αργότερα αν θέλει να λαμβάνει ειδοποιήσεις.
- **Never:** αν δεν επιθυμεί να λαμβάνει ειδοποιήσεις και δε θέλει να εμφανιστεί το banner ξανά.



3-26 Notifications Banner

Σε desktop browser αν πατήσουμε YES, στη συνέχεια, θα πρέπει να πατήσουμε allow και στο alert που θα εμφανιστεί για να μπορέσουμε να εγγραφούμε στις ειδοποιήσεις.



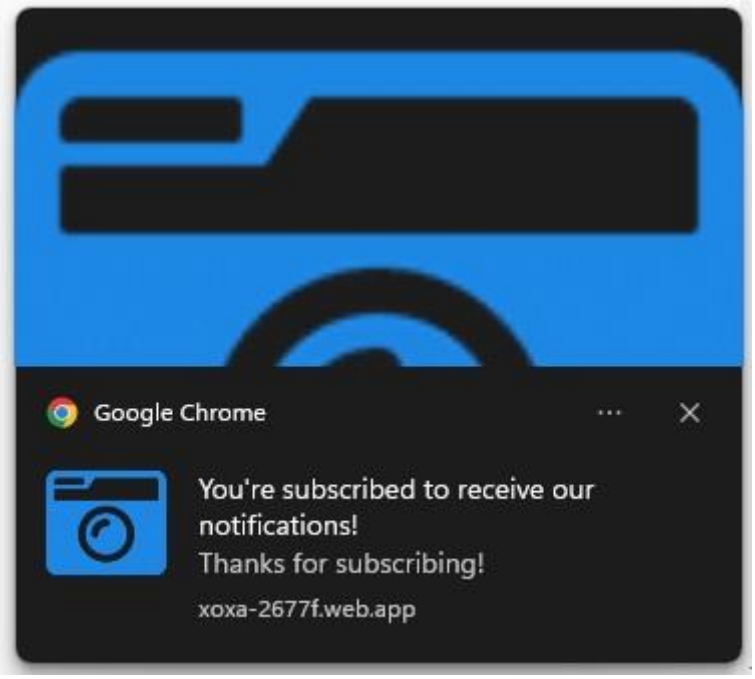
3-27 Notifications alert desktop

Για να αποθηκεύσουμε τα στοιχεία του χρήστη σχετικά με το subscription, χρησιμοποιούμε το Cloud Firestore. Για τις ανάγκες του backend, από τη στιγμή που η προσέγγιση μας είναι serverless, κάνουμε χρήση των Cloud Functions. Το function που κάνουμε χρήση είναι το παρακάτω:

```
// Create user subscription endpoint
exports.createSubscription = functions.https.onRequest((request, response) => {
  response.set('Access-Control-Allow-Origin', '*')

  db.collection('subscriptions').add(request.query).then(docRef => {
    response.send({
      message: 'Subscription added!',
      postData: request.query
    })
  })
})
```

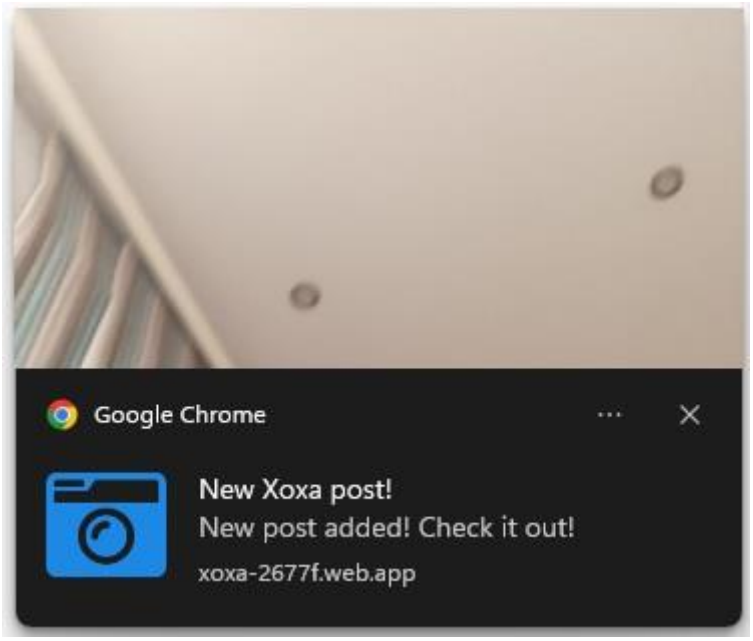
3-28 Subscriptions endpoint

Σε desktop browser αν εγγραφούμε συνδρομητές με επιτυχία εμφανίζεται η παρακάτω ειδοποίηση:



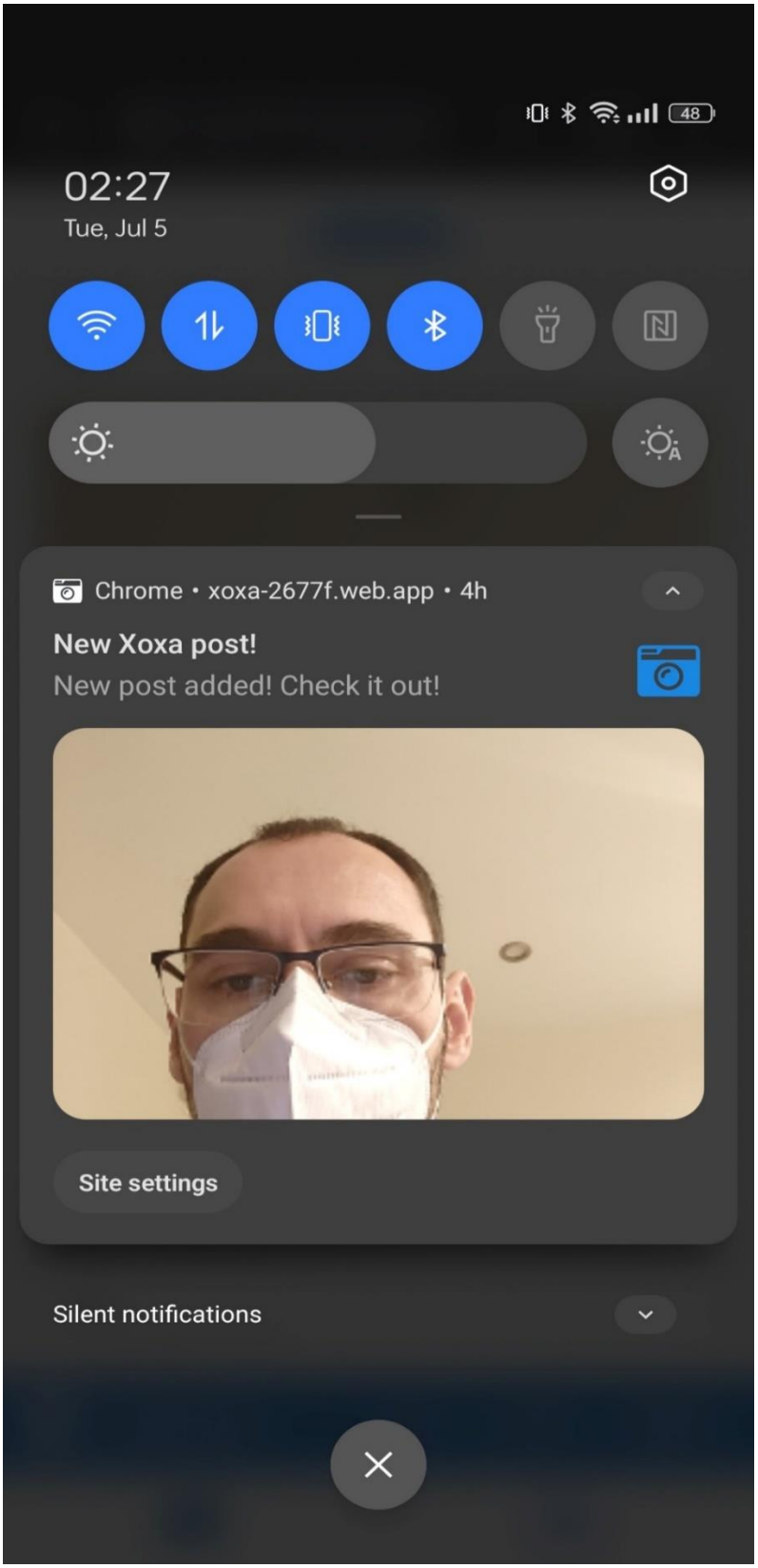
3-29 Εγγραφή συνδρομής με επιτυχία

Στη συνέχεια, αν ανέβει νέο post και έχουμε εγγραφεί συνδρομητές, θα λάβουμε ειδοποίηση γι' αυτό το post. Σε desktop browser θα έχει την ακόλουθη μορφή:



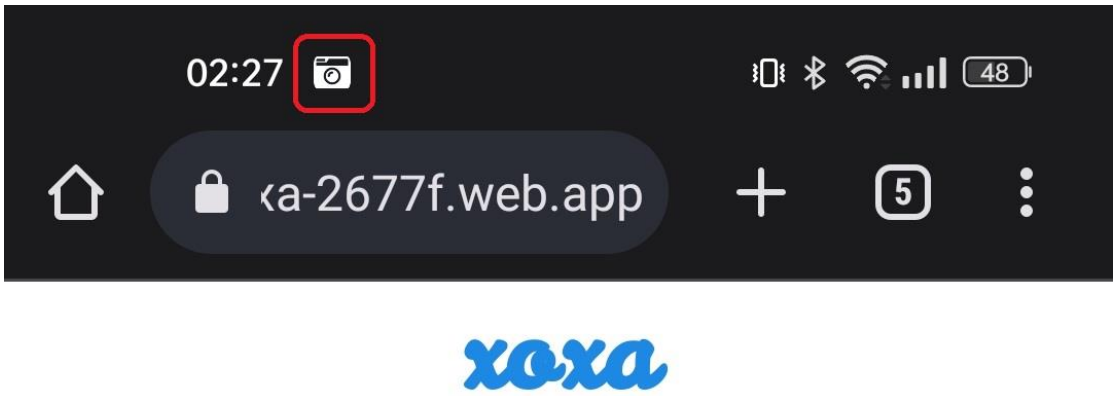
3-30 Notification νέου post desktop

Ενώ σε mobile συσκευή θα έχει αυτή τη μορφή:



3-31 Notification νέου post mobile

Επίσης, πριν ανοίξει το notification drawer στο κινητό του, ο χρήστης θα μπορεί να δει ένα εικονίδιο, ώστε να ξέρει ότι έχει ειδοποίηση από την εφαρμογή στο notification bar:



3-32 Notification Icon mobile

Ανάλογα με την έκδοση του λογισμικού και το θέμα της συσκευής μπορεί να υπάρχουν μικρές διαφορές στην εμφάνιση.

4. Μελλοντικές Σκέψεις - Βελτιώσεις

Όπως αναφέρθηκε, σκοπός της παρούσας εφαρμογής είναι να δείξει τις δυνατότητες που έχουν τα progressive web apps και όχι να γίνει μία πλήρης social media εφαρμογή. Αυτό σημαίνει ότι υπάρχει περιθώριο βελτίωσης αυτής και προσθήκης και άλλων λειτουργιών.

Μερικές βελτιώσεις και αναβαθμίσεις που θα μπορούσαν να γίνουν στην εφαρμογή είναι οι ακόλουθες:

- Προσθήκη login για πρόσθετη ασφάλεια και για να έχουμε τη δυνατότητα να παίρνουμε κάποιες πληροφορίες για το χρήστη από το account του.
- Refactoring της εφαρμογής και ανάπτυξη σε Vue.js 3 για να δούμε τα οφέλη από την τελευταία και ανανεωμένη έκδοση του framework.
- Προσθήκη avatars στα post που κάνει ο κάθε χρήστης.
- Προσπάθεια εύρεσης workarounds στα προβλήματα ασυμβατότητας που έχουν τα progressive web apps με browsers εκτός του Chrome.
- Αλλαγή του hosting domain name της εφαρμογής με κάποιο της επιλογής μας.

5. Βιβλιογραφικές Αναφορές - Πηγές

“What are Progressive Web Apps? (18/05/2016). Blog Ionic Framework.
https://blog.ionicframework.com/what-is-a-progressive-web-app/
“The offline cookbook” (09/12/2014). Jakearchibald.com.
https://jakearchibald.com/2014/offline-cookbook/
“Workbox”. Workbox documentation.
https://developer.chrome.com/docs/workbox/
“A quick but complete guide to IndexedDB and storing data in browsers” (01/06/2019). Freecodecamp.
https://www.freecodecamp.org/news/a-quick-but-complete-guide-to-indexeddb-25f030425501
“Vue patterns” Learn Vuejs.
https://learn-vuejs.github.io/vue-patterns/
“Offline Post Requests With Workbox” (15/02/2019). Medium.
https://medium.com/@mario.brendel1990/offline-post-requests-with-workbox-vuejs-4df0e9f93da9
“Build a Progressive Web App in VueJs” (21/02/2019). Codequs.
https://codequs.com/p/BJ-S_y3rE/build-a-progressive-web-app-in-vuejs?utm_campaign=Vue.js%20Feed&utm_medium=email&utm_source=Revue%20newsletter
“How to provide your own in-app install experience” (14/02/2020). Web.dev.
https://web.dev/customize-install/
“Notification”. MDN web docs.
https://developer.mozilla.org/en-US/docs/Web/API/Notification/Notification
“Progressive Web Apps” (06/01/2020). Web.dev.
https://web.dev/progressive-web-apps/
“Everything You Wanted To Know About Native, Hybrid and Web Apps; but Were Afraid To Ask” (21/05/2014). Telerik.
https://www.telerik.com/blogs/everything-you-wanted-to-know-about-native-hybrid-and-web-apps-but-were-afraid-to-ask
“Get started: write, test, and deploy your first functions”. Firebase Documentation.
https://firebase.google.com/docs/functions/get-started
“Firebase Pricing”. Firebase.
https://firebase.google.com/pricing
“What is an API?”. Red Hat.
https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces
“Web app manifests”. MDN web docs.
https://developer.mozilla.org/en-US/docs/Web/Manifest
“Use Progressive Web Apps”. Google Chrome Help Center.
https://support.google.com/chrome/answer/9658361?hl=en&co=GENIE.Platform%3DDesktop
“Service workers”. Web.dev.
https://web.dev/learn/pwa/service-workers/
“Service worker API”. MDN web docs.
https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
“Service worker overview” (24/09/2021). Google Chrome Developers.
https://developer.chrome.com/docs/workbox/service-worker-overview/
“What is Workbox?” (24/09/2021). Google Chrome Developers.

https://developer.chrome.com/docs/workbox/what-is-workbox/
“Improving Progressive Web App offline support detection” (08/02/2021). Google Chrome Developers.
https://developer.chrome.com/blog/improved-pwa-offline-detection/
“Vue.Js 2 Guide”. Vue.js Documentation.
https://v2.vuejs.org/v2/guide/
“Learn the fundamentals”. Firebase Documentation.
https://firebase.google.com/docs