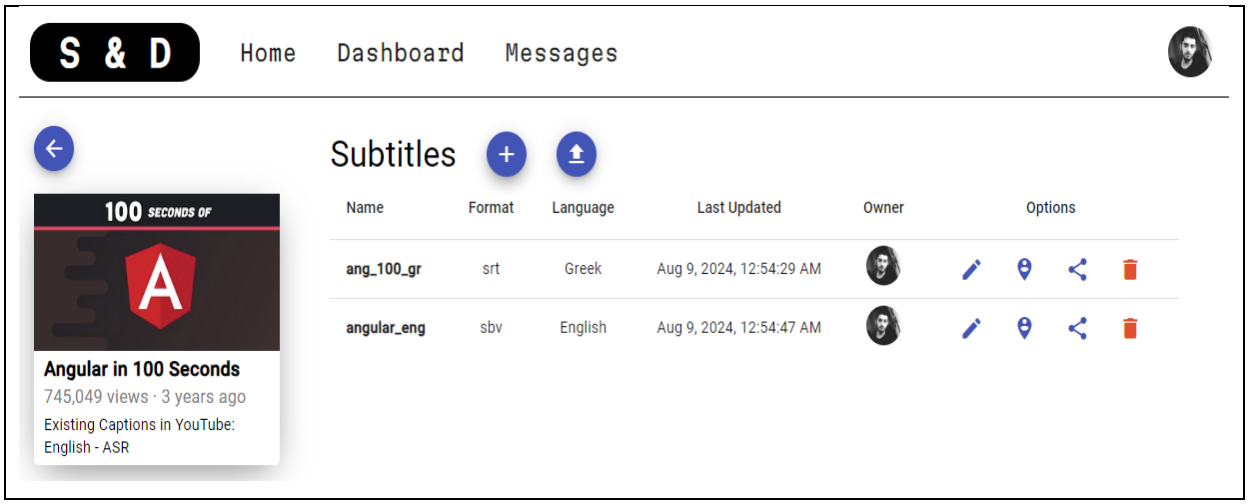
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Web Εφαρμογής για Υποβοηθούμενο Υποτιτλισμό και Μεταγλώττιση YouTube Βίντεο»



Του φοιτητή:
Χατζή Σταμάτιου
Αρ. Μητρώου: 164777

Επιβλέπων
Σαλαμπάσης Μιχαήλ
Βαθμίδα: Καθηγητής

Ημερομηνία 09/09/2024

Τίτλος Δ.Ε. Ανάπτυξη Web Εφαρμογής για Υποβοηθούμενο Υποτιτλισμό και Μεταγλώττιση YouTube Βίντεο

Κωδικός Δ.Ε. 24136

Ονοματεπώνυμο φοιτητή Χατζής Σταμάτιος

Ονοματεπώνυμο εισηγητή Σαλαμπάσης Μιχαήλ

Ημερομηνία ανάληψης Δ.Ε. 03/03/2024

Ημερομηνία περάτωσης Δ.Ε. 09/09/2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Η διπλωματική αυτή εργασία αφιερώνεται σε
όλους όσους στάθηκαν στα δύσκολα στην διάρκεια
της φοίτησής μου

Πρόλογος

Ο λόγος που επιλέχθηκε αυτή η διπλωματική εργασία ήταν κυρίως για να έρθω σε επαφή με καινούργιες τεχνολογίες που χρειάζονται για την ανάπτυξη web εφαρμογών ώστε να τις κατανοήσω και να μπορέσω να γίνω βέλτιστος σε αυτό το κομμάτι. Με τεχνολογίες όπως η Angular που χρησιμεύει για την ανάπτυξη web εφαρμογών, αλλά και την Firebase της Google που βοηθάει και απλοποιεί την ανάπτυξη τέτοιου είδους εφαρμογών σε web αλλά και σε άλλες πλατφόρμες όπως το Android, με βοήθησε ώστε να κατανοήσω και φυσικά να εκπαιδευτώ στο κομμάτι της ανάπτυξης μιας εφαρμογής αλλά και να μπορέσω να διευρύνω τους τρόπους με τους οποίους θα μπορούσα να την σχεδιάσω.

Περίληψη

Ο σκοπός της διπλωματικής μου εργασίας ήταν η περαιτέρω ανάπτυξη και βελτίωση της υπάρχουσας εφαρμογής που χρησιμεύει για την ανάπτυξη και επεξεργασία υποτίτλων για YouTube βίντεο. Στόχος μου ήταν να προσθέσω νέες δυνατότητες αλλά και να κάνω τις απαραίτητες διορθώσεις στην εφαρμογή ώστε να απλοποιηθεί η διαδικασία στους χρήστες που θέλουν να δημιουργήσουν υπότιτλους για βίντεο που υπάρχουν στην πλατφόρμα του YouTube. Στην εφαρμογή πλέον οι χρήστες θα μπορούν να αποθηκεύουν και να επεξεργάζονται τους υπότιτλους με μεγαλύτερη άνεση και χωρίς προβλήματα, καθώς πλέον έχει βελτιωθεί το σύστημα που διαχειρίζεται τους υπότιτλους, και σε συνεργασία με τις εφαρμογές Firestore Database και Storage της Firebase, γίνεται σωστά η αποθήκευση των δεδομένων και όλων των αρχείων των υποτίτλων. Επιπρόσθετα έχει δημιουργηθεί σελίδα διαχείρισης του προφίλ του χρήστη όπου ο χρήστης θα μπορεί να προσθέσει τις δικές του πληροφορίες, όπως οι γλώσσες που γνωρίζει. Επίσης, οι χρήστες θα μπορούν να κάνουν εξαγωγή των υποτίτλων σε όποια μορφή αρχείων υποτίτλων επιθυμούν αυτοί, ή, να εξάγουν όλα τα αρχεία υποτίτλων που έχουν υλοποιήσει για ένα βίντεο, σε ένα συμπιεσμένο αρχείο μορφής .zip. Τέλος έγιναν οι απαραίτητες αλλαγές στην συνολική εμφάνιση της εφαρμογής ώστε να είναι πιο φιλική και διαδραστική προς τον χρήστη που την χρησιμοποιεί.

«Web Application Development for Assisted Subtitling and Dubbing of YouTube Video»

Chatzis Stamatios

Abstract

The purpose of my thesis was to further develop and improve the existing application that serves to develop and edit subtitles for YouTube videos. My goal was to add new features but also make the necessary fixes to the application to simplify the process for users who want to create subtitles for videos that exist on the YouTube platform. In the application, users will now be able to store and edit subtitles more comfortably and without problems, as the system that manages subtitles has now been improved, and in collaboration with the Firestore Database and Storage applications of Firebase, the subtitle files and all data are stored successfully. In addition, a page for managing the user's profile has been created where the user will be able to add his own information, such as the languages he knows. Also, users will be able to export the subtitles to any subtitle file format they want, or, export all the subtitle files they have implemented for a video to a compressed .zip file. Finally, the necessary changes were made to the overall appearance of the application to make it more user-friendly and interactive.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου και όλους τους φίλους μου που μου συμπαραστάθηκαν κατά την ανάπτυξη της πτυχιακής εργασίας καθώς χωρίς αυτούς δεν θα το είχα καταφέρει. Επίσης θα ήθελα να ευχαριστήσω και τον καθηγητή, κ. Μιχαήλ Σαλαμπάση, που μου έδωσε την ευκαιρία και συνεργαστήκαμε ώστε να υλοποιήσω αυτήν την εργασία.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
UI	User Interface
DI	Dependency Injection
SnD	Sub & Dub
A.I.	Artificial Intelligence

Κεφάλαιο 1ο: Εισαγωγή

1.1 Ο ρόλος του βίντεο στις μέρες μας

Στην εποχή που ζούμε, ένα αναπόσπαστο κομμάτι της ζωής μας είναι η παρακολούθηση οπτικοακουστικού υλικού, ή, όπως αλλιώς το λέμε, βίντεο (από την αγγλική λέξη *video*). Πλέον, αν όχι όλοι μας, οι περισσότεροι έχουμε από μια συσκευή τουλάχιστον που μπορεί να καταγράψει βίντεο, όπως τα **smartphone** (έξυπνα κινητά).

Το βίντεο πλέον έχει γίνει ένα πανίσχυρο εργαλείο ενημέρωσης και ψυχαγωγίας, καθώς η ανθρωπότητα από μόνο χειροπιαστό υλικό όπως οι εφημερίδες και τα κόμικς, και ακουστικό υλικό όπως το ράδιο και η μουσική, έχει πλέον στα χέρια της το οπτικοακουστικό υλικό, που είναι πιο ζωντανό, διαδραστικό και μπορεί να ανεβάσει το επίπεδο της ενημέρωσης και την ψυχαγωγίας.

Ξεκίνησε φυσικά από την τηλεόραση που είχε μια απλή ασπρόμαυρη εικόνα χωρίς ήχο, έχει μετατραπεί στο να έχει ζωντανά χρώματα και με ήχο. Βέβαια μέχρι πρότινος η τηλεόραση ήταν αυτή που είχε το «προνόμιο» του να παρακολουθεί κάποιος κάτι. Έπειτα ανακαλύφθηκαν τα smartphone, στα οποία όχι μόνο μπορείς να παρακολουθήσεις κάτι, αλλά μπορείς και να καταγράψεις τα πάντα χάρη στην κάμερα και μικρόφωνο που διαθέτουν.

Επομένως ότι θέλουμε, όποτε το θέλουμε, χάρη σε αυτές τις συσκευές, μπορούμε να το καταγράψουμε εκείνη την στιγμή και να το κοινοποιήσουμε σε κάποιον γνωστό μας, είτε να το ανεβάσουμε σε κάποια πλατφόρμα διαχείρισης βίντεο, όπως το **YouTube**, έτσι ώστε να το παρακολουθήσουν όποιοι θέλουν.

1.2 Η σημασία των υποτίτλων

Φυσικά αυτό μπορεί να δημιουργήσει προβλήματα για ανθρώπους που είτε δεν γνωρίζουν την γλώσσα του βίντεο, είτε και ακόμα να μην μπορούν να ακούσουν τον ήχο του βίντεο.

Κάπου εκεί έρχονται οι **υπότιτλοι**, που στην ουσία μπορούν να λύσουν τέτοιου είδους προβλήματα ακοής. Οι υπότιτλοι είναι κείμενα που αντιπροσωπεύουν τον ήχο του εκάστοτε βίντεο. Είναι μικροί σε μέγεθος καθώς οι διάρκεια εμφάνισής τους δεν ξεπερνάει πολλές φορές τα 2 δευτερόλεπτα και ο θεατής θα πρέπει να προλαβαίνει να τα αναγνώσει.

Εμφανίζονται συνήθως στο κάτω μέρος της οθόνης, εκτός αν φυσικά υπάρχει κάτι που δεν επιτρέπει την εμφάνισή τους εκεί, όπως άλλα γράμματα η κάποιο σημαντικό μέρος του βίντεο που πρέπει να προβληθεί από τους θεατές.

Επομένως μπορούμε να καταλάβουμε πως οι υπότιτλοι είναι ένα σημαντικό μέρος του βίντεο. Έτσι θα πρέπει να υπάρχει και μια εφαρμογή που να μπορεί να τους δημιουργεί, επεξεργάζεται και να τους διαμορφώνει. Σε αυτό σημείο έρχεται η εφαρμογή για την οποία θα μιλήσουμε παρακάτω, την Sub & Dub, που επί της ουσίας πραγματεύεται αυτό ακριβώς το αντικείμενο.

1.3 YouTube

Σχήμα 1.1: Το λογότυπου του YouTube

Πριν όμως μιλήσουμε για την εφαρμογή της Π.Ε., θα πρέπει πρώτα να αναλύσουμε και την πλατφόρμα με την οποία η εφαρμογή έχει και άμεση σχέση μαζί της, το **YouTube,** καθώς τα βίντεο για τα οποία ο χρήστης δημιουργεί υπότιτλους προέρχονται από αυτό**.**

Το YouTube είναι μια διαδικτυακή πλατφόρμα κοινοποίησης βίντεο. Είναι προσβάσιμη για όλους σε όλο τον πλανήτη και από εκεί οι χρήστες της μπορούν να «ανεβάζουν» βίντεο. Δημιουργήθηκε το 2005 από τους Αμερικανούς Steve Chen, Chad Hurley και Jawed Karim και εξαγοράστηκε από την Google το 2006 για το ποσό των 1.65 δισεκατομμυρίων δολαρίων.

Για να μπορέσει ένας χρήστης να ανεβάσει βίντεο θα πρέπει πρώτα φυσικά να κάνει εγγραφή στο YouTube με τον λογαριασμό του από την Google και να δημιουργήσει το δικό του κανάλι. Αφού γίνει αυτό ο χρήστης θα μπορεί να ανεβάζει τα δικά του βίντεο και αναλόγως θα επιλέγει αν θέλει να τα κοινοποιήσει παντού και να τα παρακολουθήσουν όσοι και όποιοι θέλουν, ή θα μπορεί να επιλέξει συγκεκριμένη ομάδα χρηστών που θα τα παρακολουθεί.

Έπειτα χρήστες από όλη την κοινότητα θα μπορούν να κάνουν εγγραφή (το γνωστό **Subscribe**) στο κανάλι του και έτσι θα μπορούν να λαμβάνουν ειδοποιήσεις για το όποτε οι αγαπημένοι τους δημιουργοί βίντεο ανεβάζουν στην πλατφόρμα.

Φυσικά χάρη στο σύστημα που παρέχει το YouTube, οι χρήστες θα μπορούν, αν θέλουν, να πατήσουν το κουμπί Μου Αρέσει στο βίντεο για να δείξουν την ικανοποίηση τους ή το κουμπί Δεν μου Αρέσει, για να δείξουν την δυσαρέσκεια τους, και φυσικά να το κοινοποιήσουν σε όλα τα μέσα κοινωνικής δικτύωσης.

Το YouTube, τα τελευταία χρόνια, έχει πάρει τεράστιες διαστάσεις και, μιας και είναι η νούμερο 1 πλατφόρμα διαχείρισης βίντεο, πλέον πολλοί χρήστες το να δημιουργούν και να ανεβάζουν βίντεο στην πλατφόρμα είναι η κύρια τους ασχολία και το έχουν σαν επάγγελμα, με πολλούς από αυτούς να λαμβάνουν τεράστια ποσά από το YouTube και τις διαφημίσεις που μπορούν να έχουν στα βίντεο τους. Φυσικά αυτό έχει και αρνητικές επιπτώσεις στον κόσμο καθώς διαδίδεται και ψευδής πληροφορία από fake news και αυτό ειδικά τα τελευταία 2 χρόνια έχει γίνει ακόμα πιο έντονο εξαιτίας της τεχνητής νοημοσύνης (Artificial Intelligence), που έχει εντείνει αυτό το φαινόμενο.

Το κανάλι με τους περισσότερους subscribers κατά την διάρκεια της εγγραφής της Π.Ε. (Αύγουστος 2024) το κατέχει ο **MrBeast**, που αριθμεί πάνω από **311 εκατομμύρια** εγγεγραμμένους χρήστες, ενώ στην Ελλάδα αυτήν την πρωτιά την κατέχει το κανάλι **GL Show**, που αριθμεί γύρω στους **1,6 εκατομμύρια** subscribers.

1.4 Μικρή περιγραφή της εφαρμογής

Σχήμα 1.2: Το λογότυπο της εφαρμογής

Η εφαρμογή που αναπτύχθηκε, ονομάζεται **Sub & Dub** και ο σκοπός της είναι να βοηθάει τους χρήστες να δημιουργούν υποτίτλους για βίντεο που υπάρχουν στην πλατφόρμα του **YouTube**.

Ο κύριος λόγος που δημιουργήθηκε είναι να για μπορέσει να εξελίξει το **Studio του YouTube** που ο κάθε χρήστης απλά μπορεί να δημιουργήσει για το δικό του βίντεο υποτίτλους μόνο, και να γίνει καλύτερο στο να δημιουργεί ο καθένας υποτίτλους, όχι μόνο για τον εαυτό του, αλλά και για όλη την κοινότητα και τους χρήστες που έχουν κάνει εγγραφή στην εφαρμογή αυτή. Έτσι η εφαρμογή προσθέτει πολλές περισσότερες δυνατότητες από το απλά να δημιουργεί υποτίτλους, όπως η κοινότητα αλλά και η υποστήριξη από την **Τεχνητή Νοημοσύνη**.

Ο κάθε χρήστης μπορεί κάνει εγγραφή στην εφαρμογή μέσω του λογαριασμού του που έχει στο Google και στην αρχή διαμορφώνει το δικό του **προφίλ**. Εκεί μπορεί να διαλέξει μερικά στοιχεία για τον εαυτό του, όπως μητρική γλώσσα ή και ξένες γλώσσες.

Έπειτα μπορεί να προσθέσει βίντεο μέσω των URL του YouTube, και μετά να δημιουργήσει υπότιτλους σε κάθε γλώσσα που επιθυμεί και σε 2 διαφορετικά format υποτίτλων.

Επιπρόσθετα μπορεί να κοινοποιήσει τους υποτίτλους του στην κοινότητα της εφαρμογής ούτως ώστε να τον βοηθήσουν στην δημιουργία υποτίτλων.

Τέλος μπορεί να δεχτεί και προσφορές ώστε άλλοι χρήστες από την κοινότητα, να φτιάξουν τους υποτίτλους που επιθυμεί.

1.5 Σύντομη παρουσίαση των κεφαλαίων της Πτυχιακής

Στο πλαίσιο αυτής της Πτυχιακής και για να επιτευχθεί η υλοποίηση της εφαρμογής χρησιμοποιήθηκαν μερικές από τις πιο νέες τεχνολογίες που υπάρχουν αυτή τη στιγμή. Στα επόμενα κεφάλαια της πτυχιακής θα αναλυθούν πλήρως αυτές τις τεχνολογίες, όπως το πως δημιουργήθηκαν και ποιος είναι ο σκοπός τους.

Όπως είναι ήδη γνωστό το 1ο Κεφάλαιο κάνει μια εισαγωγή στην σημασία των υποτίτλων και τον σκοπό της εφαρμογής Sub & Dub, καθώς και τον σκοπό του YouTube

Για αρχή, το 2ο Κεφάλαιο αναφέρεται στο framework της **Angular**, το οποίο είναι ένα ισχυρό εργαλείο που επιτρέπει στους χρήστες της να δημιουργούν γρήγορα web εφαρμογές και ταυτόχρονα να είναι αξιόπιστες.

Στη συνέχεια, στο 3ο Κεφάλαιο αναλύεται η λειτουργία της πλατφόρμας της **Firebase** που είναι ένα εργαλείο που αναπτύχθηκε από την Google, και ο ρόλος της είναι η παροχή εργαλείων και υπηρεσιών που μας βοηθάει για την ανάπτυξη web εφαρμογών, εφαρμογών Android, κ.α.

Επιπρόσθετα στο 4ο Κεφάλαιο θα μιλήσουμε για το **Git**, το οποίο είναι ένα σύστημα διαχείρισης εκδόσεων το οποίο κρατάει ιστορικό από τις εκδόσεις των αρχείων αλλά και τους δημιουργούς του και αυτούς που το επεξεργάστηκαν.

Έπειτα στο 5ο Κεφάλαιο γίνεται πλήρης επεξήγηση των λειτουργιών της **εφαρμογής Sub & Dub**, όπως το πως δημιουργήθηκε, πως λειτουργεί, πως ένας χρήστης μπορεί να δημιουργήσει λογαριασμό και, φυσικά, το πως μπορεί να αναπτύξει υπότιτλους.

Συνεχίζοντας στο 6ο και τελευταίο Κεφάλαιο περιγράφονται τα **συμπεράσματα** που προέκυψαν κατά την διάρκεια της ανάπτυξης της εφαρμογής αλλά και οι **βελτιώσεις** που μπορούν να γίνουν σε αυτή.

Φυσικά στο τέλος υπάρχει και η **βιβλιογραφία**, όπου υπάρχουν όλες οι πηγές από τις οποίες αντλήθηκαν οι πληροφορίες για την ανάπτυξη αυτής της εργασίας.

Κεφάλαιο 2ο: Angular

Σχήμα 2.1: Το λογότυπο της Angular

2.1 Πρώτη γνωριμία με Angular

Η Angular είναι ένα **web framework** της **JavaScript** η οποία αναπτύχθηκε από μια εξιδεικευμένη ομάδα προγραμματιστών της Google, και είναι open-source, δηλαδή ο κώδικάς της είναι διαθέσιμος για όλους να τον χρησιμοποιήσουν. Βασίζεται στην **TypeScript,** και είναι ένα από τα πιο διαδεδομένα web framework μιας και η αρχιτεκτονική που χρησιμοποιεί είναι πολύ στιβαρή, οι δυνατότητές της είναι πάρα πολλές και είναι πολύ αποδοτική για την ανάπτυξη εφαρμογών στο web.

2.2 Ιστορία της Angular

Το framework της Angular **δημιουργήθηκε από τον Miško Hevery το 2009 [3]**. Η πρώτη έκδοση της, γνωστή και ως **AngularJS,** ήταν πάλι ένα open-source web framework, το οποίο όμως ήταν βασισμένο στην JavaScript.

Καθώς όμως τα προβλήματα ασφάλειας και συμβατότητας με τους browser και την **jQuery** αυξάνονταν, η ομάδα της Google το 2016 δημιούργησε την Angular 2, η οποία χρησιμοποιούσε νεότερες τεχνολογίες όπως την TypeScript. Από τότε βγαίνουν συνεχώς νέες εκδόσεις στην «νέα» Angular με τελευταία να είναι η 18 έκδοση η οποία και δημοσιεύτηκε το 2024.

Το 2022 σταμάτησε και η υποστήριξη για την AngularJS, μιας και η νέα Angular ήταν πλέον πιο διαδεδομένη και πιο σταθερή.

2.3 TypeScript

Για να κατανοηθεί καλύτερα η Angular θα πρέπει πρώτα να γίνει ανάλυση της **TypeScript**, η οποία είναι μια γλώσσα προγραμματισμού η οποία κατασκευάστηκε από την Microsoft το 2012. Η TypeScript στην ουσία αποτελεί ένα υπερσύνολο της **JavaScript**, δηλαδή έχει όλες τις δυνατότης της JavaScript και ένα επιπλέον επίπεδο πάνω σε αυτήν, που είναι το σύστημα γραφής της TypeScript.

Η TypeScript, σε αντίθεση με την JavaScript που είναι μια γλώσσα που **δεν** απομνημονεύει τους τύπους των μεταβλητών της, προσθέτει στην ουσία αυτό το χαρακτηριστικό το οποίο δεν επιτρέπει στον χρήστη να μεταβάλει τον τύπο των μεταβλητών της.

Δηλαδή αν έχουμε μια μεταβλητή **let a = `Hello World`** στην JavaScript μπορεί να μεταβληθεί ο τύπος του και να το μετατρέψουμε σε αριθμό από αλφαριθμητικό με το τρόπο: **a = 2.** Αυτό στην TypeScript

πλέον δεν μπορεί να γίνει καθώς ο τύπος της μεταβλητής παραμένει σταθερός και όταν θα εκτελεσθεί το σημείο του κώδικα **a = 2** τότε θα υπάρξει σφάλμα.

Με αυτόν τον τρόπο η TypeScript μειώνει σημαντικά τα προβλήματα που μπορεί να προκληθούν από την αλλαγή των μεταβλητών μεταξύ string, number, κ.α.

Στον κώδικα της TypeScript, πριν μετατραπεί σε JavaScript, πραγματοποιείται έλεγχος σφαλμάτων των τύπων των μεταβλητών και έτσι αποφεύγονται τα σφάλματα εκτέλεσης κώδικα, τα λεγόμενα Runtime Errors.

2.4 Εγκατάσταση Angular

Για να μπορέσουμε να δημιουργήσουμε μια νέα εφαρμογή σε περιβάλλον της Angular θα πρέπει πρώτα από όλα να εγκαταστήσουμε την ίδια την Angular στον προσωπικό μας υπολογιστή. Φυσικά για εργαστεί ο οποιασδήποτε σε project της Angular θα πρέπει να έχει λάβει υπόψη ότι θα πρέπει να γνωρίζει καλά τις γλώσσες **HTML, JavaScript, TypeScript** και για το styling αρκετή **CSS.**

Για αρχή θα πρέπει να εγκαταστήσει την **Node.js** το οποίο είναι ένα open-source εργαλείο εκτέλεσης JavaScript, το οποίο επιτρέπει στον χρήστη να δημιουργεί servers, web εφαρμογές και να εκτελεί γραμμές εντολών και scripts.

Στη συνέχεια θα πρέπει να εγκαταστήσει το **npm** το οποίο είναι ένα από τα μεγαλύτερα μητρώα λογισμικού, καθώς με το npm μπορεί να γίνει εγκατάσταση πακέτων, να κοινοποιήσει τον κώδικά που έχει δημιουργήσει και να κάνει πολλές άλλες λειτουργίες.

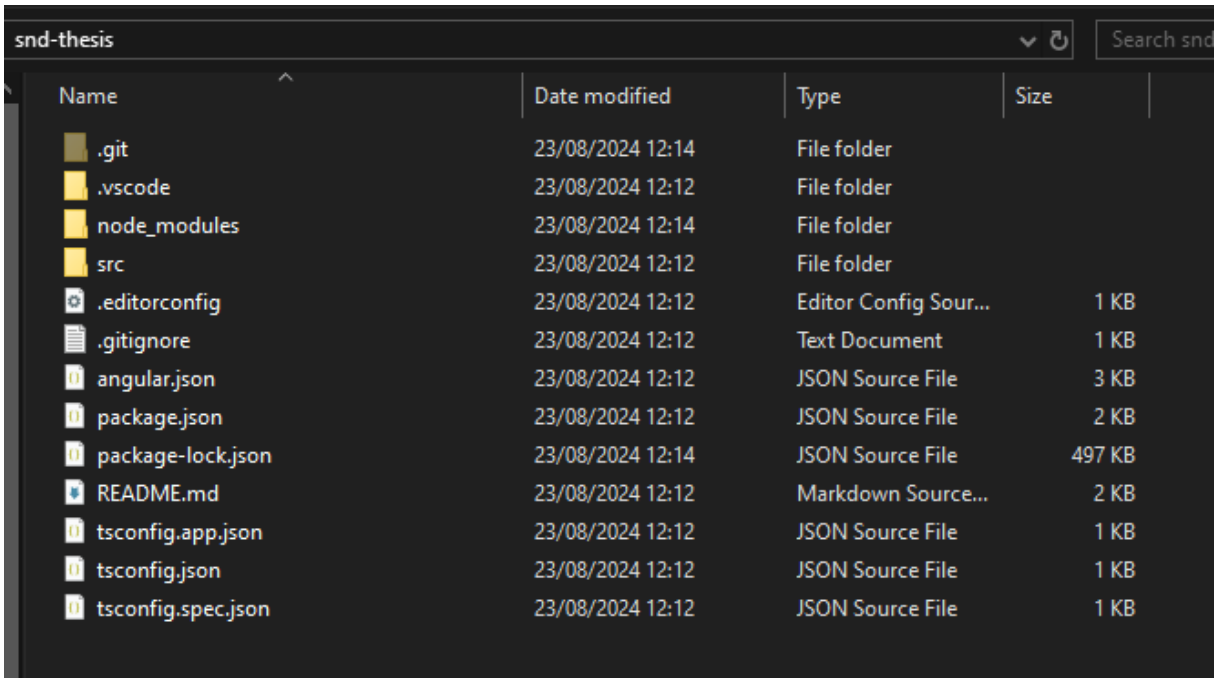
Και αφού έχουμε εγκαταστήσει όλα τα παραπάνω, τότε μπορούμε να εγκαταστήσουμε και την Angular. Ανοίγοντας τον Command Prompt μπορούμε να πληκτρολογήσουμε την παρακάτω γραμμή κώδικα όπως φαίνεται στο Σχήμα 2.2

Σχήμα 2.2: Εγκατάσταση της Angular

Και αφού εγκατασταθεί η Angular στον προσωπικό μας υπολογιστή, πλέον μπορούμε να δημιουργήσουμε ένα νέο Angular project. Πηγαίνοντας στον φάκελο που θέλουμε να δημιουργηθεί το νέο project, πληκτρολογούμε την παρακάτω γραμμή κώδικα όπως φαίνεται στο Σχήμα 2.3 στο Command Prompt:

Σχήμα 2.3: Δημιουργία νέου Angular Project

Σύμφωνα με το παραπάνω θα δημιουργηθεί ένα νέο Angular Project με την ονομασία snd-thesis. Αν μπούμε στον φάκελο θα παρατηρήσουμε ότι έχει δημιουργηθεί ένας φάκελος με το όνομα αυτό και μέσα σε αυτόν θα παρατηρήσουμε ότι έχουν δημιουργηθεί αυτόματα μερικά αρχεία και φάκελοι που απαιτούνται για την εκτέλεση της εφαρμογής.



Σχήμα 2.4: Δομή των φακέλων και αρχείων του Angular Project

Τέλος μπορούμε να χρησιμοποιήσουμε τον **Visual Studio Code** ώστε να ανοίξουμε αυτό το project και να μπορέσουμε να το εκτελέσουμε και να το υλοποιήσουμε. Έπειτα με την χρήση του **Angular CLI,** που θα δούμε παρακάτω, μπορούμε να τρέξουμε εντολές για την νέα εφαρμογή μας.

2.5 Angular CLI

Το **Angular CLI,** είναι ένα σημαντικό εργαλείο, απαραίτητο για την Angular, το οποίο είναι ένα command-line εργαλείο που χρησιμοποιείται για την εκτέλεση, ανάπτυξη, testing, deployment και συντήρηση των εφαρμογών της Angular, απευθείας από ένα command shell.

Ένα από τα βασικά χαρακτηριστικά του CLI είναι ότι μέσω αυτού μπορείς να δημιουργήσεις νέες Angular εφαρμογές με την χρήστη της γραμμής **ng new <app-name>,** όπως έχουμε ήδη αναλύσει στην ενότητα 2.4. Επίσης υπάρχει και το **ng generate** με το οποίο οι προγραμματιστές μπορούν να δημιουργήσουν νέα components, modules, directives, services, κ.α. που τους χρειάζονται στην υλοποίηση της εφαρμογής τους.

Φυσικά, μια ακόμα χρήσιμη λειτουργία του CLI είναι για να τρέχει εντολές για την εκτέλεση και ανάπτυξη της εφαρμογής. Με την εντολή **ng serve,** η εφαρμογή μας ξεκινάει να εκτελείται τοπικά σε έναν server, και μπορούμε να την παρακολουθήσουμε ζωντανά. Εξ ‘ορισμού η εφαρμογή «ξεκινάει» τοπικά στην διεύθυνση `http://localhost:4200/`, από όπου γίνεται και αυτόματο refresh όταν πραγματοποιούμε μια αλλαγή στην εφαρμογή μας.

Μια ακόμα λειτουργία του CLI είναι να δημιουργούμε ένα production build της εφαρμογής μας, ώστε να μπορέσει να γίνει ανάπτυξη σε περιβάλλον παραγωγής. Αυτό μπορεί να γίνει με την εντολή **ng build –prod.**

Τέλος, το CLI χρησιμεύει και για λόγους testing της εφαρμογής μας καθώς μέσω αυτού, μπορούμε να τρέξουμε εντολές ώστε η εφαρμογή να δοκιμαστεί και αν εντοπιστούν τυχών λάθη ή σφάλματα

επιχειρησιακής λογικής. Η εντολή για το testing είναι το **ng test** όπου και η Angular τρέχει τα unit test χρησιμοποιώντας το Jasmine που λειτουργεί ως πλαίσιο για unit test και το Karma που λειτουργεί ως test runner.

2.6 Αρχιτεκτονική της Angular

Και αφού αναπτύχθηκε η γλώσσα της TypeScript, ο τρόπος λειτουργίας της και ο τρόπος εγκατάστασης της Angular, τώρα θα γίνει η ανάλυση της αρχιτεκτονικής της. Η Angular, όπως και πολλά άλλα web frameworks σαν την ReactJS, χρησιμοποιεί την component-based αρχιτεκτονική δηλαδή βασίζεται στα components ή κομμάτια κώδικα που το καθένα αποτελεί και ένα τμήμα της εφαρμογής. Παρακάτω θα αναλυθούν περαιτέρω οι τρόποι με τους οποίους δουλεύει η αρχιτεκτονική της.

2.6.1 Components

Τα Components είναι κομμάτια κώδικα τα οποία το καθένα περιγράφουν ένα σημείο ή τμήμα της web εφαρμογής. Αυτά είναι σχεδιασμένα έτσι ώστε να μπορεί να ξαναχρησιμοποιηθούν και να εκτελούν μια συγκεκριμένη διαδικασία.

Για παράδειγμα μπορούμε να έχουμε ένα component στο οποίο έχει δημιουργηθεί ένας πίνακας ο οποίος θα εμφανίζει κάποια συγκεκριμένα δεδομένα. Έτσι με αυτόν το τρόπο θα μπορούμε να «καλέσουμε» αυτό το component που έχει υλοποιημένο τον πίνακα σε οποιοδήποτε σημείο της εφαρμογής.

Τα components μπορούν να δομηθούν σε μια ιεραρχία από γονείς και παιδιά, δηλαδή ένας «πατέρας» component μπορεί να έχει δικά του «παιδιά» components τα οποία μπορούν και επικοινωνούν μεταξύ τους

2.6.1.1 Δομή των Components

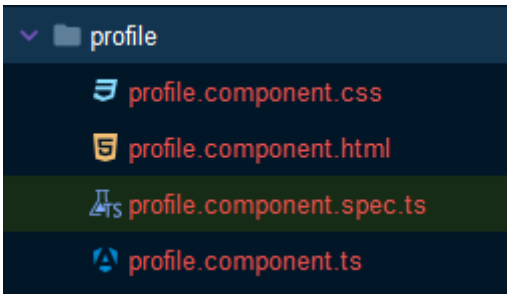
Όπως προαναφέρθηκε και πιο πάνω, η Angular αποτελείται από components τα οποία ερμηνεύουν το καθένα ένα τμήμα του κώδικα της εφαρμογής. Για να δημιουργηθεί ένα νέο component θα πρέπει να εκτελεστεί η παρακάτω εντολή στο **Angular CLI** (2.5), ή μέσω του command prompt όπως εμφανίζεται στο Σχήμα 2.5:

```
\SnD Project\snd-thesis\src\app> ng g c profile
```

Σχήμα 2.5: Δημιουργία νέου Component

Και αφού εκτελέσουμε την παραπάνω εντολή, τότε η Angular θα μας δημιουργήσει αυτόματα ένα component που στην ουσία είναι ένας φάκελος και αποτελείται από 4 αρχεία.

- Το πρώτο αρχείο είναι ένα **html** template αρχείο το οποίο εκεί περιέχονται τα DOM elements από την HTML όπως τα div, p, κ.α. και καθορίζεται το τι θα εμφανίσει το component
- Το δεύτερο είναι ένα **TypeScript** αρχείο το οποίο καθορίζει την συμπεριφορά και λειτουργία του component
- Το τρίτο είναι ένα αρχείο **CSS** το οποίο καθορίζει το στυλ και την μορφή του component.
- Τέλος το τέταρτο είναι ένα αρχείο spec.ts τύπου TypeScript πάλι, που χρησιμεύει για λόγους **testing**.



Σχήμα 2.6: Δομή αρχείων του Component

Για να μπορεί η Angular να αναγνωρίσει ότι είναι component, πριν την έναρξη της κλάσης του component δηλώνεται ένα Decorator "**@Component**" το οποίο στην ουσία δηλώνει την κλάση ως component και μπορεί να την «διακοσμήσει» με ιδιότητες όπως:

- **selector**, το οποίο είναι για να δηλωθεί το όνομα του component το οποίο έτσι θα μπορεί να χρησιμοποιηθεί και σε άλλα components και templates,
- το **html template**, που συνδέεται μέσω data binding για την μεταφορά δεδομένων για εμφάνιση,
- και τέλος τα **styles** (CSS).

Στο Σχήμα 2.7 παρακάτω φαίνεται ο τρόπος με τον οποίο δηλώνεται ένα component.

```
import { Component } from '@angular/core';

@Component({  Show usages
  selector: 'app-profile',
  standalone: true,
  imports: [],
  templateUrl: './profile.component.html',
  styleUrl: './profile.component.css'
})
export class ProfileComponent {

}
```

Σχήμα 2.7: Μορφή αρχείου Component

2.6.1.2 Κύκλος ζωής ενός Component

Το κάθε Component έχει και ένα κύκλο ζωής (γνωστό και ως Component Lifecycle), που είναι μια σειρά από στάδια εκτέλεσης κώδικα σε συγκεκριμένες στιγμές του component από την «γέννηση» του έως και τον «θάνατο» του. Αυτό για να επιτευχθεί θα πρέπει ο εκάστοτε προγραμματιστής, που υλοποιεί το component, να εκτελέσει ένα κομμάτι κώδικα που το καθένα αναλογεί σε μια φάση της ζωής του component. Πιο αναλυτικά τα στάδια ζωής είναι:

- **ngOnChanges():** Καλείται όταν πραγματοποιούνται αλλαγές στις τιμές των Input του component. Αυτό γίνεται λίγο πριν το rendering του component ή κατά την αλλαγή των τιμών.

- **ngOnInit():** Καλείται μόνο μια φορά πριν «φορτωθούν» τα δεδομένα από το input στο component.
- **ngDoCheck():** Καλείται σε κάθε αλλαγή που η Angular δεν μπορεί να εντοπίσει από μόνη της.
- **ngAfterContentInit():** Καλείται όταν είναι να προβληθεί εξωτερικό περιεχόμενο στην προβολή του component.
- **ngAfterContentChecked():** Καλείται όταν είναι να ελεγχθεί το περιεχόμενο που πρόκειται να προβληθεί στο component.
- **ngAfterViewInit():** Καλείται για να γίνει αρχικοποίηση της προβολής του component και των «παιδιών» του.
- **ngAfterViewChecked():** Καλείται για να γίνει έλεγχος της προβολής του component και των «παιδιών» του.
- **ngOnDestroy():** Και τέλος καλείται πριν γίνει καθαρισμός και καταστροφή του component.

Στο παρακάτω Σχήμα 2.8 μπορεί να φανεί η χρήση ενός κύκλου ζωής, και συγκεκριμένα του ngOnInit που είναι και ένα από τα πιο συχνά χρησιμοποιούμενα lifecycle hooks στην Angular. Εκεί μπορεί να φανεί και η χρήση του που είναι κυρίως για να «φορτώνει» τα δεδομένα ενός component κατά την αρχή της ζωής του.

```
ngOnInit() {  Show usages   StamChatzis +1
  this.user$.subscribe({
    next: data => {
      this.loadUserDetails(data);
      this.uid = data.uid;
      this.photoUrl = data.photoURL;
      this.email = data.email;
      this.loadUserRatings(data.uid)
      this.proService.getAllLanguages(data.uid).subscribe({
        next: data => {
          this.loadForeignLanguages(data)
        }
      });
    }
  });

  this.proService.getCountries().subscribe({
    next: data => {this.loadCountries(data);}
  });

  this.proService.getSkillLevels().subscribe({
    next: data => {this.loadSkillLevels(data)}
  });

  this.getAvailableLanguages();
}
```

Σχήμα 2.8: Παράδειγμα χρήσης του κύκλου ngOnInit().

2.6.1.3 Επικοινωνία μεταξύ Components

Φυσικά για να υπάρξει μια σωστή και ολοκληρωμένη εφαρμογή στην Angular, θα πρέπει να υπάρχει και η σωστή επικοινωνία μεταξύ των components ώστε να μπορούν να «ανταλλάσσουν» μεταξύ τους δεδομένα. Για να επιτευχθεί αυτή η σωστή επικοινωνία μεταξύ τους, υπάρχουν μερικοί τρόποι.

Ένας από τους τρόπους επικοινωνίας, είναι με την χρήση των **Input** και **Output properties.** Αυτά χρησιμοποιούνται κυρίως για να μπορεί να γίνει σωστή επικοινωνία μεταξύ των «παιδιών» και «γονιών» components. Βάζοντας τον decorator **@Input** τα παιδιά μπορούν να λάβουν δεδομένα από τους γονείς, ενώ με τον decorator **@Output,** τα παιδιά μπορούν να στέλνουν στους γονείς τους δεδομένα.

Επίσης ένας ακόμα τρόπος να επικοινωνούν τα components είναι μέσω της διαδικασίας των **Event Emitters**, που αυτά κυρίως χρησιμοποιούνται σε συνδυασμό με τον διακοσμητή **@Output** που επιτρέπει την μεταφορά δεδομένων από το παιδί component στο γονιό του. Στην ουσία το `EventEmitter` όπως το λέει και η λέξη *emit,* εκδίδει το συμβάν που περιέχει τα δεδομένα που είναι να σταλούν στον «πατέρα» component. Ένα από τα βασικά χαρακτηριστικά του Event Emitter είναι ότι επιτρέπει στα components να επικοινωνούν μεταξύ τους χωρίς να χρειάζεται να γνωρίζουν τις λεπτομέρειες του «πατέρα» component του. Έτσι με αυτόν τον τρόπο η λογική του «παιδιού» component μπορεί να λειτουργήσει πιο ανεξάρτητη.

Τέλος, ένας ακόμα τρόπος που μπορούν να επικοινωνούν τα components μεταξύ τους είναι τα **Services**. Με τα Services (υπηρεσίες), τα components μπορούν να επικοινωνούν μεταξύ τους χωρίς να έχουν την σχέση που αναφέραμε πιο πάνω «γονέα» και «παιδιού»

2.6.2 Templates

Ένα από τα βασικά κομμάτια της αρχιτεκτονικής της Angular είναι τα **Templates** ή αλλιώς Πρότυπα, που χρησιμεύουν για να καθορίσουν το U.I. ενός component. Εκεί καθορίζεται η δομή, η διάταξη και το περιεχόμενο το οποίο θα προβληθεί στον χρήστη. Αυτά είναι γραμμένα σε γλώσσα **HTML** και ταυτόχρονα συνδυάζονται και με την ειδική σύνταξη της Angular. Παρακάτω θα δούμε μερικά από τα πιο σημαντικά χαρακτηριστικά των Templates.

2.6.2.1 Text Interpolation

Το Text Interpolation ή αλλιώς *παρεμβολή κειμένου* είναι μια δυνατότητα της Angular στα HTML Templates που επιτρέπει την ενσωμάτωση expressions στο HTML κείμενο. Από προεπιλογή, το Interpolation χρησιμοποιεί τα διπλά άγκιστρα **{{** και **}}** ως οριοθέτες, και ανάμεσα σε αυτούς αναγράφεται το expression.

Για παράδειγμα έχουμε μια μεταβλητή displayName στο TypeScript αρχείο του component μας όπως αυτό φαίνεται στο Σχήμα 2.9 που δηλώνεται στον κύκλο ζωής του component **ngOnInit()** ως **'Maria'**.

```
export class ProfileComponent implements OnInit {
  displayName!: string;

  constructor() {}  no usages

  ngOnInit() {  no usages
    this.displayName = 'Maria';
  }

}
```

```
<div class="profile-display-name">
  <p>{{displayName}}</p>
</div>
```

Σχήμα 2.9: Ορισμός μεταβλητής και λειτουργία του Text Interpolation στο Template

Έπειτα, μπορούμε να δούμε το πως ακριβώς λειτουργεί το Text Interpolation στο Template του συγκεκριμένου component. Με αυτόν τον τρόπο η Angular καταλαβαίνει ότι ανάμεσα στα διπλά άγκιστρα ανάμεσα στο DOM Element **p** θα πρέπει να αντικαταστήσει το **displayName** με ό,τι αυτό είναι ορισμένο. Στην προκειμένη περίπτωση θα αντικατασταθεί με την λέξη **Maria,** και θα προβληθεί στον χρήστη.

2.6.2.2 Template Statements

Μια ακόμη χρήσιμη λειτουργία της Angular στα Templates είναι τα **Statements**, τα οποία είναι μέθοδοι ή ιδιότητες τα οποία χρησιμεύουν για να εκτελέσουν ενέργειες στην HTML. Με τα Template Statements, η εφαρμογή μπορεί να εκτελέσει μεθόδους ή και κομμάτια κώδικα που έχουν υλοποιηθεί στην TypeScript, μέσω του HTML Template. Τα Statements χρησιμοποιούν γλώσσα που μοιάζει με την JavaScript και η ανάθεση τους γίνεται μέσα σε εισαγωγικά μετά το σύμβολο ίσων (=). Ενώ μπορούν να δηλωθούν πολλαπλά κομμάτια κώδικα διαχωρίζοντάς τα με το ελληνικό ερωτηματικό (**;**).

Ο τρόπος σύνταξής τους είναι ο εξής όπως στο παράδειγμα:

```
<button type="button" (click)="onSave()">Save</button>
```

Όπου μέσα στις παρενθέσεις είναι η ενέργεια που περιμένει να συμβεί ώστε να εκτελεστεί το Statement. Στην προκειμένη περίπτωση αυτή περιμένει να συμβεί το event του click ώστε μετά να συμβεί το Statement που βρίσκεται μετά το = μέσα στα εισαγωγικά. Δηλαδή, όταν ο χρήστης πατήσει το συγκεκριμένο button, θα εκτελεστεί η μέθοδος **onSave()** που έχει υλοποιηθεί στην κλάση TypeScript του component μας.

2.6.2.3 Data Binding

Ακόμα μια σημαντική δυνατότητα των Templates είναι το binding ή αλλιώς η *σύνδεση*, που στην ουσία είναι η ισχυρή συσχέτιση μεταξύ του UI που δημιουργήθηκε από το Template με τα DOM Elements, και των δεδομένων της εφαρμογής, εξασφαλίζοντας και τον συγχρονισμό μεταξύ τους. Φυσικά υπάρχουν διαφορετικοί τρόποι binding που προσφέρει η Angular.

- **Property Binding:** Ένας από τους τρόπους binding, είναι αυτός του Property Binding ή η *Σύνεση Ιδιοτήτων*. Με τον όρο αυτόν, εννοούμε την δυνατότητα της Angular να μπορεί να ρυθμίσει την ιδιότητα ενός HTML DOM Element δυναμικά. Λειτουργεί με το να εισάγουμε στο element τις τετράγωνες αγκύλες **[ ]** και ανάμεσα εισάγουμε το property του DOM Element που θέλουμε να αναθέσουμε δυναμικά τιμή. Για παράδειγμα έχουμε την παρακάτω γραμμή κώδικα:

```
<img alt="item" [src]="itemImageUrl">
```

  Σε αυτό παρατηρούμε ότι ανάμεσα στις τετράγωνες αγκύλες έχουμε την ιδιότητα **src** του **img** element, που είναι η πηγή της εικόνας. Έτσι μέσα στα εισαγωγικά υπάρχει η μεταβλητή **itemImageUrl** που θα «φορτώσει» δυναμικά την ιδιότητα src της εικόνας με ό,τι έχει οριστεί σε αυτή.

- **Attribute Binding:** Επίσης έχουμε και το Attribute binding ή αλλιώς *Σύνδεση Χαρακτηριστικών* και με αυτό η Angular επιτρέπει το να ορίζουμε τιμές για χαρακτηριστικά απευθείας. Με αυτόν τον τρόπο επιτυγχάνεται πιο εύκολη προσβασιμότητα αλλά και βελτιώνεται το style της εφαρμογής, καθώς μπορεί να διαχειριστεί με αυτόν τον τρόπο πολλαπλά CSS αρχεία. Αυτό το binding έχει να κάνει κυρίως με την ανάθεση **ARIA attributes**, τα οποία είναι ειδικά χαρακτηριστικά τα οποία βελτιώνουν την προσβασιμότητα των ιστοσελίδων και κυρίως για άτομα με ειδικές ανάγκες.
- **Event Binding:** Ένα ακόμα σημαντικό binding της Angular είναι το Event Binding ή αλλιώς η *Σύνδεση Γεγονότων.* Με αυτόν τον τρόπο η Angular μας επιτρέπει να την εκτέλεση μεθόδων η κομματιών κώδικα με την εκτέλεση ενεργειών από τους χρήστες. Για να επιτευχθεί αυτό, στα DOM Elements, μέσα σε παρενθέσεις **( )** εισάγουμε το event, δηλαδή την ενέργεια που θέλουμε να συμβεί ώστε να τρέξει η μέθοδος που έχουμε εισάγει στα εισαγωγικά. Όπως είδαμε και στο παραπάνω παράδειγμα:

```
<button type="button" (click)="onSave()">Save</button>
```

  για να εκτελεστεί η μέθοδος **onSave()** που υπάρχει εντός των εισαγωγικών, θα πρέπει ο χρήστης να κάνει το event του click που είναι μέσα στις παρενθέσεις, δηλαδή να κάνει κλικ πάνω στο button.

- **Two-way Binding:** Τέλος έχουμε ένα ακόμα σημαντικό binding, αυτό του Two-way Binding ή αλλιώς της *Αμφίδρομης Σύνδεσης.* Συνδυάζοντας τα παραπάνω δύο binding, αυτά του Property και Event, η Angular με αυτόν τον τρόπο μας επιτρέπει να βελτιώσουμε την κοινοποίηση των δεδομένων στο component, ώστε να μπορούμε και να περιμένουμε για events και ταυτόχρονα να ενημερώνουμε τιμές ταυτόχρονα στον «γονιό» component και στο «παιδί» component. Αυτό γίνεται με την χρήση των τετράγωνων αγκυλών και παρενθέσεων μαζί σε ένα DOM Element, δηλαδή: **[( )].** Για παράδειγμα:

```
<input [(ngModel)]="name">
<p>Your name is: {{ name }}</p>
```

  στην παραπάνω γραμμή κώδικα, έχουμε το **[(ngModel)]** το οποίο είναι χαρακτηριστικό της Angular για την αμφίδρομη σύνδεση. Έτσι με αυτόν τον τρόπο η μεταβλητή **name** ενημερώνεται δυναμικά όταν ο χρήστης αλλάξει την τιμή του **input.** Έτσι θα ενημερωθεί αυτόματα και το element **p** που έχει σε text interpolation την μεταβλητή name.

2.6.3 Directives

Και αφού αναλύσαμε την χρήση και τα χαρακτηριστικά των Templates, τώρα θα αναπτύξουμε τα **Directives,** ή αλλιώς *οδηγίες,* που είναι κλάσεις που προσθέτουν μια επιπλέον συμπεριφορά στα στοιχεία της Angular εφαρμογής μας.

Υπάρχουν διάφορα είδη από directives, και ένα από τα πιο συχνά είδη είναι αυτό του **Component,** το οποίο το αναλύσαμε και πιο πάνω στην ενότητα 2.6.1. Επίσης το Component είναι το μόνο directive που έχει και **Template.**

Ένα ακόμα είδος, είναι τα **Attribute Directives**, τα οποία «ακούν» και τροποποιούν την συμπεριφορά άλλων HTML element, ιδιότητες, χαρακτηριστικά αλλά και components. Ένα κοινό Attribute Directive είναι και αυτό που είδαμε στο Two-way binding, το **ngModel.** Μερικά ακόμα directives είναι τα **ngStyle** και **ngClass.**

Τέλος, έχουμε και τα **Structural Directives**, τα οποία είναι υπεύθυνα για την διάταξη της HTML, και μπορούν να διαμορφώσουν ή να αναδιαμορφώσουν την δομή των DOM, προσθέτοντας, αφαιρώντας και τροποποιώντας συνήθως τα στοιχεία που είναι συνδεδεμένα. Μερικά από τα πιο συνηθισμένα τέτοια directives είναι τα **ngIf** το οποίο δημιουργούν υπό-προβολές μέσα στο template μόνο να ισχύει η συνθήκη**, ngFor** το οποίο επαναλαμβάνει ενέργειες μέσα στο στοιχείο του template **και ngSwitch** το οποίο αλλάζει μεταξύ εναλλακτικών προβολών του template**.**

Η Angular δίνει την δυνατότητα στους χρήστες της να δημιουργήσουν και αυτοί τα δικά τους **custom** Attribute Directives, τα οποία μπορούν να χρησιμοποιήσουν όπου θέλουν μέσα στην εφαρμογή τους. Αυτό μπορεί να επιτευχθεί με την εκτέλεση του παρακάτω command.

```
ng generate directive highlight
```

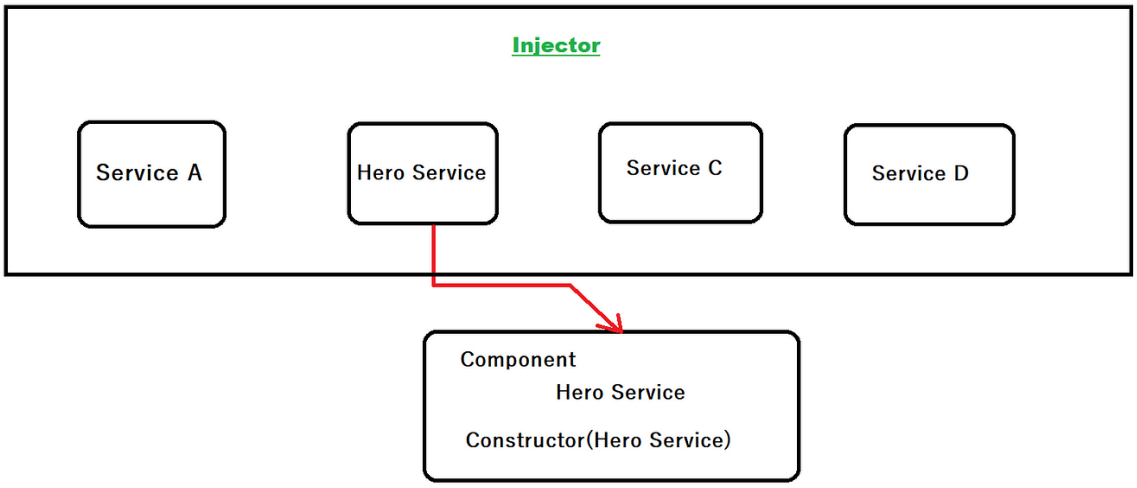
Αυτό θα δημιουργήσει ένα νέο directive με το όνομα **highlight**.

2.6.4 Dependency Injection

Ένας ακόμα σημαντικός μηχανισμός της αρχιτεκτονικής της Angular είναι το Dependency Injection (DI), με τα οποία η Angular μας δίνει την δυνατότητα εφαρμόσουμε άλλα κομμάτια της εφαρμογής μας σε άλλα αντίστοιχα, τα οποία χρειάζονται για την βελτίωση της ευελιξίας, της δομής αλλά και της δοκιμής (testing) της εφαρμογής.

Υπάρχουν δύο κεντρικοί ρόλοι στο σύστημα του DI, ο **dependency consumer** και ο **dependency producer.** Για να διευκολύνει την αλληλεπίδραση μεταξύ των consumers και producers, η Angular χρησιμοποιεί έναν αφαιρετικό μηχανισμό που ονομάζεται **Injection.** Όταν χρειάζεται ένα dependency, ο Injector αναζητεί αν υπάρχει ήδη κάποιο instance διαθέσιμο, αλλιώς αν δεν υπάρχει, τότε δημιουργείται ένα νέο instance και αποθηκεύεται στο μητρώο. Η Angular, φυσικά διαθέτει και ένα **root** injector που τον διοχετεύει σε όλη την εφαρμογή. Υπάρχει πολύ μεγάλη πιθανότητα να μην χρειαστεί να δημιουργήσει ο χρήστης χειροκίνητα Injectors.

Σε ένα ακόμα σημαντικό μέρος που βοηθάει το DI, είναι τα **Unit Testing**, δηλαδή τα components μπορούν να δοκιμαστούν εύκολα με την χρήση **mock** services, χωρίς να αλλάξει η εσωτερική τους λογική



Σχήμα 2.10: Διάγραμμα της λογικής του Dependency Injection

2.6.5 Services

Ένα από τα πιο σημαντικά Dependency Injection είναι τα **Services** ή *Υπηρεσίες*, που είναι αρχεία τύπου TypeScript τα οποία περιέχουν κλάσεις που χρησιμεύουν για να εκτελέσουν την λογική της εφαρμογής και δεν σχετίζονται με το User Interface (UI) της. Είναι ένα από τα κύρια κομμάτια μιας εφαρμογής Angular, καθώς διαδραματίζουν σημαντικό ρόλο στην ανάπτυξη και την οργάνωσή της.

Ένας από τους πιο σημαντικούς ρόλους του Service είναι ότι εκεί φιλοξενείται η **επιχειρησιακή λογική** της όλης εφαρμογής, καθώς αντί να επαναλαμβάνεται η ίδια λογική σε κάθε component, με την χρήση των Services, μεταφέρεται σε ένα μόνο μέρος, και αυτή γίνεται Inject σε οποιοδήποτε component επιθυμεί ο χρήστης να την χρησιμοποιήσει.

Επίσης στα Services, μπορεί να γίνει **επικοινωνία με τις εξωτερικές πηγές δεδομένων**, δηλαδή μπορεί να γίνει επικοινωνία με διάφορα **APIs** για την ανάκτηση δεδομένων. Μια σημαντική λειτουργία που χρησιμοποιείται σε αυτή την περίπτωση είναι το `**HttpClient**`, το οποίο χρησιμεύει για να μπορεί εφαρμογή, ή το Service στην περίπτωση αυτή, να μπορεί να κάνει κλήσεις σε μεθόδους HTTP, όπως το **GET**, **POST** κ.α.

Για να λειτουργήσουν σωστά τα Services, θα πρέπει να δηλωθούν σε έναν provider που καθορίζει πως θα μπορεί να χρησιμοποιηθεί το Service. Ο πιο συχνός provider είναι αυτός του **root,** ο οποίος κάνει διαθέσιμο το Service σε όλη την εφαρμογή για να γίνει Inject.

```typescript
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { AngularFirestore, AngularFirestoreDocument } from "@angular/fire/compat/firestore";
import { GmailUser, Rating } from "../models/firestore-schema/user.model";
import { HttpClient } from "@angular/common/http";
import { ForeignLanguage } from "../models/general/language-skills";

@Injectable({ Show usages  schatzis +1
  providedIn: 'root'
})

export class ProfileService {
  otherLanguages$: Observable<ForeignLanguage[]>;

  constructor(private firestore: AngularFirestore, private http: HttpClient) {} no usages  schatzis

  getCountries() : Observable<Object> { Show usages  StamChatzis
    const countryUrl : "../../assets/data/google-coun...  = "../../assets/data/google-countries.json";
    return this.http.get(countryUrl);
  }

```

Σχήμα 2.11: Παράδειγμα δομής ενός Service

2.6.6 Modules

Τα **Modules,** ή αλλιώς *ενότητες*, είναι ένας θεμελιώδης πυλώνας της Angular, καθώς ορίζει την διαφορά μεταξύ των κομματιών στην εφαρμογή. Ορίζονται με τον διακοσμητή `**@NgModule**` και μπορούν να χρησιμοποιηθούν ως «δοχεία», καθώς μέσα τους δηλώνονται components, directives, services, κ.α. και βοηθάει στην σωστή οργάνωσή τους.

Με τα NgModules, επιτυγχάνεται η σωστή διαμόρφωση του Injector και του Compiler. Κάθε εφαρμογή της Angular, έχει ένα τουλάχιστον Module, το οποίο είναι το **Root Module,** και περιέχει το bootstrap component της εφαρμογής, που είναι απαραίτητο για την λειτουργία της κατά το ξεκίνημα.

Ένα σημαντικό στοιχείο επίσης είναι το **Lazy Loading**, με τα οποία η Angular υποστηρίζει την φόρτωση με καθυστέρηση, καθώς φορτώνει μόνο τα απαιτούμενα modules, βελτιώνοντας την απόδοση της εφαρμογής.

Τέλος υπάρχουν και τα **Shared Modules**, στα οποία ορίζονται κοινά components και directives που χρειάζονται σε διάφορα κομμάτια της εφαρμογής.

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({ Show usages  new *
  declarations: [AppComponent],
  imports: [BrowserModule, FormsModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Σχήμα 2.12: Δομή του Root Module

Στο Σχήμα 2.12 μπορούμε να παρατηρήσουμε την δομή του Root Module, στο οποίο φαίνονται τα **declarations**, τα οποία είναι η λίστα των components, directives, κ.α., τα **imports** τα οποία περιέχουν εξωτερικές μονάδες που χρειάζονται για να λειτουργεί το module, τους **providers** οι οποίοι περιέχουν τα services, και τέλος το **bootstrap,** το οποίο φορτώνει το κύριο component που ξεκινάει την εφαρμογή.

2.7 Testing στην Angular

Ένας από τους βασικούς κανόνες στην Angular είναι αυτός του **Testing,** καθώς έτσι διασφαλίζεται η ποιότητα και η αξιοπιστία του κώδικα που έχει υλοποιηθεί στην εφαρμογή. Η Angular έχει ενσωματωμένα εργαλεία για testing, όπως το **Unit Testing** και το **End-to-End Testing.**

Η εντολή που πρέπει να εκτελεστεί για να μπει η Angular εφαρμογή σε επίπεδο δοκιμασίας, είναι το **ng test** το οποίο γίνεται μέσω του **Angular CLI** (2.5).

Το **Unit Testing,** χρησιμεύει για την δοκιμή μεμονωμένων κομματιών κώδικα μέσα στην εφαρμογή όπως τα components. Για τη εκτέλεση των Unit Tests, χρησιμοποιούνται δύο εργαλεία, για τα οποία θα αναλύσουμε παρακάτω, το **Jasmine** και το **Karma**. Οι δοκιμές αυτές μπορούν να τρέξουν κατά την διάρκεια της ανάπτυξης της εφαρμογής.

Το **End-to-End Testing** από την άλλη, ελέγχει την συνολική ροή της εφαρμογής, και για τη εκτέλεση των συγκεκριμένων δοκιμών χρησιμοποιείται το **Protractor**, το οποίο ελέγχει τα διάφορα κομμάτια της εφαρμογής για το πως συνεργάζονται μεταξύ τους. Τα συγκεκριμένα tests είναι πιο αργά στην υλοποίησή τους από τα Unit Tests, αλλά καλύπτουν μεγαλύτερο μέρος της χρήστης της εφαρμογής με περισσότερα σενάρια δοκιμών. Για το E2E Testing, χρειάζεται να τρέξει η εντολή **ng e2e** στο Angular CLI

Τα εργαλεία τα οποία χρησιμοποιεί εξ ‘ορισμού η Angular είναι τα:

- **Jasmine:** Το Jasmine είναι ένα framework στην Angular το οποίο χρησιμεύει για τα Unit Tests στην εφαρμογή.
- **Karma:** Το Karma είναι ένας test runner ο οποίος τρέχει στον browser ώστε να μας εμφανίσει τα αποτελέσματα.
- **Protractor:** Το Protractor επίσης ένα framework στην Angular, το οποίο αφορά τα End-to-End Tests. Συνεργάζεται με το **Selenium WebDriver,** το οποίο είναι ένα open-source εργαλείο για την αυτοματοποίηση των web browsers.

2.8 Reactive Extension

Η Reactive Extension της JavaScript ή **RxJs** για συντομογραφία, είναι μια βιβλιοθήκη η οποία παρέχει πολλές δυνατότητες για προγραμματισμό με χρήση παρατηρήσιμων ακολουθιών, τα αποκαλούμενα **observables**, επιτρέποντας την ασύγχρονη χρήση δεδομένων. Για την Angular, είναι ένας σημαντικός παράγοντας, καθώς αποτελεί τον πυρήνα για την διαχείριση των δεδομένων σε ασύγχρονες συνθήκες, όπως στις περιπτώσεις των **HTTP Requests** που συμβαίνουν εντός των Services.

```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({  no usages
  providedIn: 'root'
})
export class DataService {
  constructor(private http: HttpClient) {}  no usages

  getData(): Observable<any> {  no usages
    return this.http.get( url: 'https://api.example.com/data');
  }
}
```

Σχήμα 2.13: Χρήση της RxJS σε ένα Service της Angular

Στο παραπάνω Σχήμα 2.13, μπορούμε να παρατηρήσουμε την χρήση της RxJS σε ένα Service μια εφαρμογής Angular. Βλέπουμε ότι η μέθοδος **getData(),** επιστρέφει ένα **Observable**. Σε αυτό ο **Observer** θα το παρατηρήσει και θα κάνει **Subscribe** ώστε να πάρει την ροή δεδομένων που επιστρέφει αυτή η **GET** HTTP μέθοδος. Όλα αυτά περιγράφονται πιο αναλυτικά στις επόμενες υποενότητες.

2.8.1 Observables

Τα Observables, ή αλλιώς τα *Παρατηρήσιμα,* είναι μια ροή δεδομένων ή γεγονότων στα οποία μπορεί να γίνει «παρατήρηση» κατά την διάρκεια της εκτέλεσή τους. Αυτά μπορεί να μεταδίδουν (emit) τιμές, να στέλνουν σφάλματα και, φυσικά, να ολοκληρώνονται.

2.8.2 Observers

Οι Observers, ή αλλιώς οι *Παρατηρητές,* στην ουσία «απορροφούν» τις τιμές που εκπέμπει ένα Observable. Αποτελείται από τρεις διαφορετικές μεθόδους: το **next(),** το οποίο είναι αυτό που λαμβάνει της επόμενη τιμή που εκπέμπεται από το Observable, το **error(),** το οποίο λαμβάνει τα σφάλματα που μπορεί να προκύψουν κατά την εκπομπή του Observable, και τέλος έχουμε το **complete(),** το οποίο εκτελείται όταν η διαδικασία της εκπομπής του Observable, ολοκληρώνεται.

2.8.3 Subscriptions

Τα Subscriptions, ή αλλιώς οι *Συνδρομές,* δημιουργούνται από την σύνδεση ενός Observer με το Observable. Το Subscription επιτρέπει στον Observer να ξεκινήσει να παρακολουθεί ένα Observable, και να το διακόψει όταν πλέον δεν το χρειάζεται άλλο.

2.8.4 Operators

Τα Operators, ή αλλιώς οι *Τελεστές,* χρησιμοποιούνται για να χειριστούν τα Observables και κυρίως τα streams ή τις ροές των δεδομένων που παράγονται. Μερικοί από τους τελεστές είναι τα: **map,** όπου γίνεται mapping των δεδομένων των ροών με τις μεταβλητές της κλάσης, το **filter,** όπου στην ουσία

φιλτράρει τις ροές και διαβάζει μόνο τα δεδομένα που χρειάζεται, το **mergeMap,** το **switchMap,** και πολλοί άλλοι.

Ένας σημαντικός operator είναι και το **pipe**, που στην ουσία είναι συνάρτηση που μπορεί να χρησιμοποιηθεί και όπως οι συνηθισμένες συναρτήσεις, με εξαίρεση ότι η συνάρτηση Pipe() που υπάρχει στην RxJS, δεν επιτρέπει το να συνενώνεται μεταξύ της και να γίνονται δυσανάγνωστοι.

2.8.5 Subjects

Τα Subjects, ή αλλιώς τα *Θέματα,* είναι ένας ειδικός τύπος Observable, τα οποία επιτρέπουν τις τιμές των Observables να μεταδίδονται σε πολλαπλούς Observers. Κάθε Subject είναι και από ένα Observable που μπορείς να κάνεις κανονικά Subscription, και θα αρχίσει να λαμβάνει τιμές.

Ταυτόχρονα ένα Subject μπορεί να είναι και Observer που έχει και τις μεθόδους που έχει ένας Observer όπως περιεγραφήκαν πιο πάνω.

2.9 Επίλογος

Με την ολοκλήρωση αυτού του κεφαλαίου που αφορά την **Angular**, μπορεί να παρατηρηθεί πως αυτό το framework είναι ένα από τα πιο σημαντικά framework που κυκλοφορούν αυτή την στιγμή. Αποτελεί ένα από τα πιο ολοκληρωμένα open-source εργαλεία, για να μπορέσει ο οποιοσδήποτε χρήστης της να δημιουργήσει από το μηδέν μια web εφαρμογή. Επίσης έγινε κατανόηση της αρχιτεκτονική της, το πως είναι ένα component-based framework και το πως είναι δομημένη. Στην συνέχεια έγινε ανάλυση και σε έναν από τους σημαντικούς πυλώνες της, που είναι τα Dependency Injection και τα Directives, και τέλος επεξηγήθηκε και η RxJS, και το πόσο χρήσιμη είναι για να λειτουργήσει σωστά μια εφαρμογή γραμμένη σε Angular.

Κεφάλαιο 3ο: Firebase

Σχήμα 3.1: Το λογότυπο της Firebase

3.1 Εισαγωγή στη Firebase

Η Firebase είναι μια διαδικτυακή εφαρμογή που παρέχεται από την Google, και προσφέρει πολλά εργαλεία για την ανάπτυξη εφαρμογών σε πολλαπλές πλατφόρμες όπως **web**, **Android** και **iOS**. Παρέχει μια πλούσια γκάμα από δυνατότητες, που βοηθούν τον εκάστοτε χρήστη να δημιουργήσει μια εφαρμογή, να την παρακολουθεί, να την συντηρεί, να την ασφαλίσει και πολλές ακόμα δυνατότητες που στην ουσία «λύνουν» τα χέρια των προγραμματιστών.

Η Firebase είναι μια cloud-hosted εφαρμογή που σημαίνει ότι η Google αναλαμβάνει να διαχειρίζεται τα εργαλεία που προσφέρει η Firebase. Στην ουσία η Firebase λειτουργεί σαν το **back-end** μιας εφαρμογής, και αυτό είναι που την μετατρέπει σε ένα εργαλείο που προσελκύει όλο και περισσότερους χρήστες μιας και η Google «κάνει» την δουλειά του back-end κομματιού μιας εφαρμογής, ενώ οι χρήστες μπορούν να επικεντρωθούν περισσότερο στο front-end κομμάτι της χωρίς να καταναλώνει περισσότερους πόρους, και βρίσκεται σε **client**, οπότε αυτό χαρακτηρίζει την Firebase ως **Back-end as a Service**, δηλαδή που προσφέρει υπηρεσίες.

3.2 Ιστορία της Firebase

Η Firebase εξελίχθηκε από την **Envolve,** που ήταν μια start-up η οποία ιδρύθηκε από τους James Tamplin και Andrew Lee το 2011 [10]. Σε αυτή είχαν δημιουργήσει ένα API το οποίο υλοποιούσε μια λειτουργία διαδικτυακών μηνυμάτων σε μορφή chat. Στην συνέχεια, το εξέλιξαν ώστε το σύστημα chat που δημιούργησαν να το αποσπάσουν και να το κάνουν μια real-time αρχιτεκτονική. Έτσι τον Απρίλιο του 2012 ίδρυσαν την Firebase σαν ξεχωριστή εταιρία και την δημοσιοποίησαν ένα χρόνο αργότερα. Τον Οκτώβρη του 2014, η Firebase αποκτήθηκε από την **Google**.

3.3 Προσθέτοντας την Firebase

Για να μπορέσει η Firebase να λειτουργήσει σε μια εφαρμογή, όπως, για παράδειγμα, στην εφαρμογή που αναφέρεται και αυτή η Π.Ε., την SnD, θα πρέπει να ακολουθηθεί μια διαδικασία που θα μπορέσει να «ενώσει» την Angular εφαρμογή στην Firebase και να προσθέσει το **SDK** της**.**

Για αρχή θα πρέπει ένας χρήστης να δημιουργήσει ένα project στην Firebase και να δηλώσει την εφαρμογή του. Αφότου κάνει την εγγραφή, ο χρήστης θα λάβει ένα configuration που θα του επιτρέπει να συνδέσει την εφαρμογή του στο Firebase, και να μπορεί να κάνει χρήστη των υπηρεσιών που προσφέρει.

Έπειτα θα πρέπει να κάνει εγκατάσταση το **SDK** της Firebase και στην συνέχεια να το αρχικοποιήσει. Ο τρόπος που γίνεται αυτό είναι με την χρήση του **npm**, που το αναλύσαμε και στην ενότητα 2.4, εκτελώντας την εντολή: **`npm install firebase`,** στο Command Prompt.

Στην συνέχεια, και αφού γίνει η εγκατάσταση της Firebase στο σύστημα μας, τότε θα μπορεί να έχει να αποκτήσει πρόσβαση στις υπηρεσίες της Firebase, για τις οποίες θα μιλήσουμε αργότερα, στην εφαρμογή του.

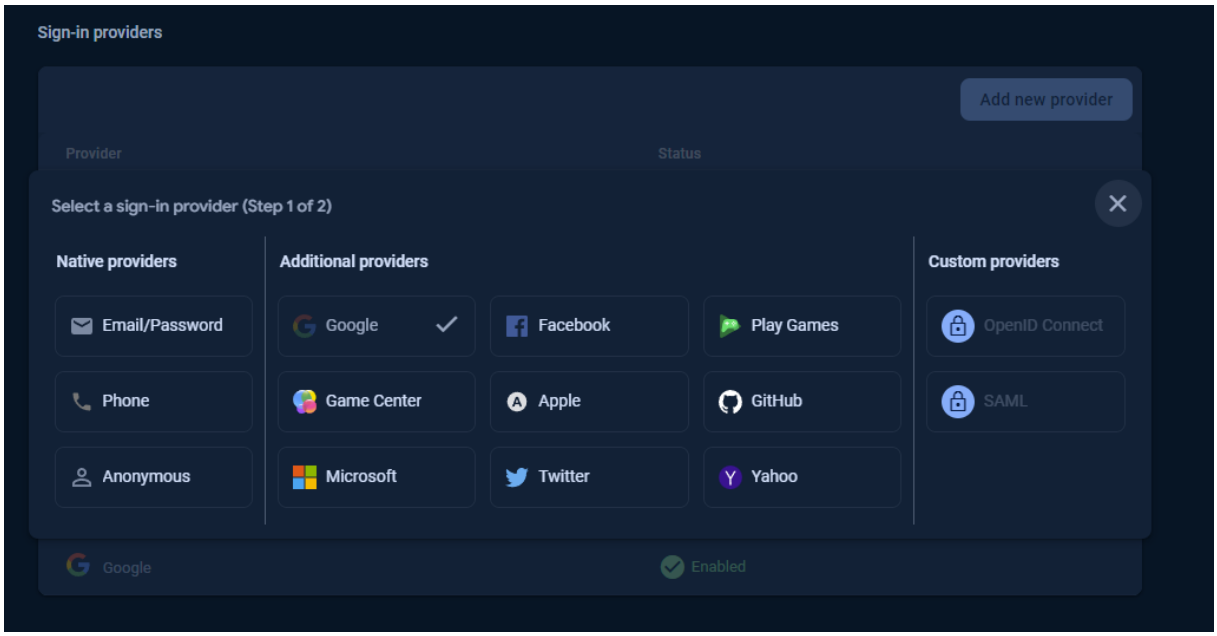
3.4 Υπηρεσίες της Firebase

Όπως προαναφέραμε και πιο πάνω, η Firebase παρέχει μια μεγάλη ποικιλία από **υπηρεσίες – εργαλεία** που είναι διαθέσιμα για να προστεθούν σε web εφαρμογές. Παρακάτω θα αναλύσουμε μερικές από τις πιο σημαντικές υπηρεσίες που υπάρχουν, καθώς και αυτές που έχουν χρησιμοποιηθεί στην εφαρμογή της Π.Ε.

3.4.1 Firebase Authentication

Στις μέρες μας η ασφάλεια και η αξιοπιστία των διαδικτυακών εφαρμογών είναι, ίσως, το πιο σημαντικό κομμάτι κατά την υλοποίηση μιας web app. Ανά διαστήματα παρατηρούμε πως γίνονται κακόβουλες επιθέσεις από τρίτους, ώστε να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα των χρηστών μιας εφαρμογής. Επομένως η ανάγκη για ασφάλεια είναι μεγάλη σε όλες τις web apps. Βασικοί πυλώνες για την ασφάλεια μιας web εφαρμογής είναι η **Αυθεντικοποίηση (Authentication)** και η **Εξουσιοδότηση (Authorization).**

Και κάπου εδώ έρχεται η Firebase με την υπηρεσία της **Firebase Authentication,** όπου στην ουσία είναι ένα σύστημα back-end που προσφέρεται για τις web εφαρμογές, και όχι μόνο. Χρησιμοποιεί ένα εύκολο στην χρήση SDK, και βιβλιοθήκες οι οποίες έχουν έτοιμα UI για την χρήση της αυθεντικοποίησης. Με αυτή την υπηρεσία, η εφαρμογή αποκτά υποστήριξη αυθεντικοποίησης με την χρήση κωδικών, κινητών τηλεφώνων και εξουσιοδότηση μέσω των πιο γνωστών παρόχων όπως η **Google,** το **Facebook,** το **X (πρώην Twitter),** και πολλούς ακόμα.



Σχήμα 3.2: Τρόποι αυθεντικοποίησης του Firebase Authentication

Η χρήση της Firebase Authentication, μπορεί να συνεργαστεί άμεσα και χωρίς προβλήματα και με άλλες υπηρεσίες της Firebase. Φυσικά για την αξιοπιστία της συγκεκριμένης υπηρεσίας, το Firebase Authentication χρησιμοποιεί όλα τα πιστοποιημένα πρωτόκολλα ασφάλειας, όπως το **OAuth 2.0**, και μπορεί να χρησιμοποιηθεί εύκολα και με άλλα custom back-end συστήματα που μπορεί να έχει υλοποιήσει ο χρήστης.

Το Firebase Authentication SDK, παρέχει τις πολλές δυνατότητες αυθεντικοποίησης σε μια εφαρμογή και μερικές από τις πιο σημαντικές είναι:

- Απλή αυθεντικοποίηση μέσω **email** και **password** (κωδικό πρόσβασης)
- Αυθεντικοποίηση μέσω **παρόχων μέσων κοινωνικής δικτύωσης** όπως Google, Facebook, κ.α.
- Αυθεντικοποίηση μέσω **αριθμού τηλεφώνου**
- **Προσαρμοσμένη** ενσωμάτωση αυθεντικοποίησης συστήματος
- **Ανώνυμη** αυθεντικοποίηση.
- Και τέλος, **Multi-factor authentication,** δηλαδή αυθεντικοποίηση χρήστη μέσω πολλαπλών παραγόντων, όπως μια εφαρμογή Authenticator στο smartphone του.

Ο χρήστης αφού κάνει σύνδεση στην εφαρμογή του με έναν από τους τρόπους που προαναφέραμε, τότε αποκτά τα διαπιστευτήρια της αυθεντικοποίησης. Τότε αυτά τα διαπιστευτήρια περνάν στο Firebase Authentication SDK, και η back-end υπηρεσία που προσφέρει, επαληθεύουν ότι πρόκειται για έναν πιστοποιημένο χρήστη, και επιστρέφεται αυτή η απάντηση στην εφαρμογή. Τότε ο χρήστης αποκτά πρόσβαση στην εφαρμογή.

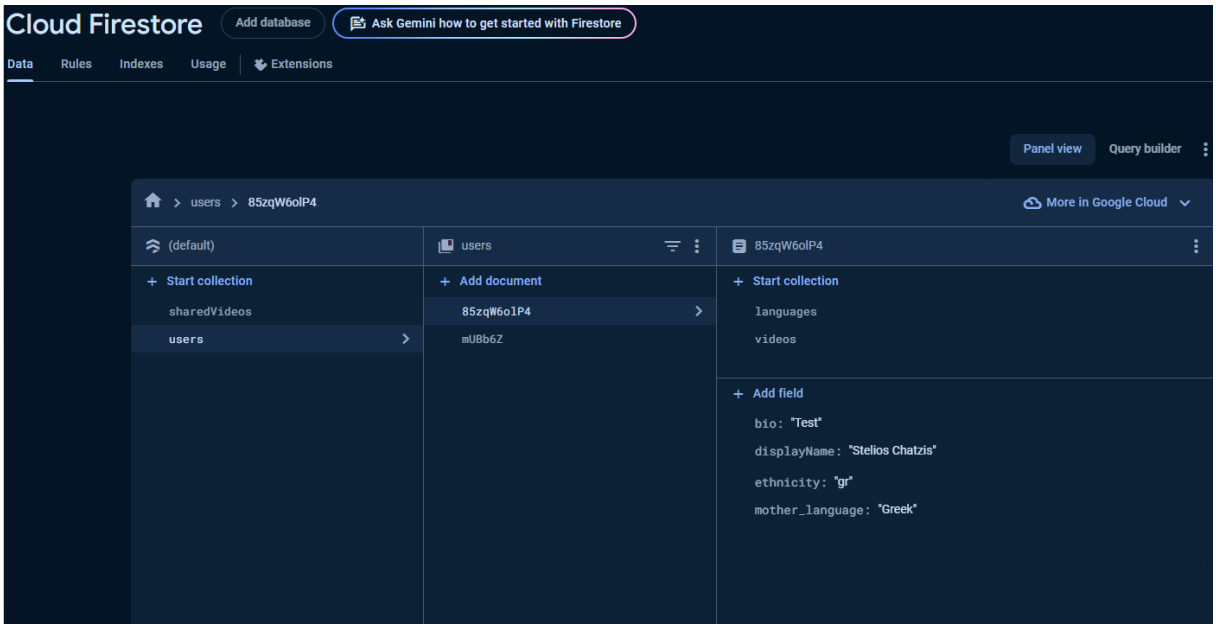
3.4.2 Cloud Firestore Database

Ένα ακόμα σημαντικό μέρος μια εφαρμογής είναι η **βάση δεδομένων,** καθώς εκεί αποθηκεύονται όλα τα δεδομένα των χρηστών, δεδομένα απαραίτητα για την εφαρμογή και πολλές άλλες πληροφορίες. Οι βάσεις δεδομένων χωρίζονται σε 2 βασικές μορφές: **α) SQL,** όπου είναι βάσεις δεδομένων οι οποίες είναι βασισμένες σε μορφή πινάκων και είναι σχεσιακές, και **β) NoSQL,** όπου είναι βάσεις δεδομένων που βασίζονται σε documents, key-values και είναι μη σχεσιακές.

Φυσικά, και για το κομμάτι της βάσης δεδομένων, η Firebase προσφέρει μια υπηρεσία, την **Cloud Firestore,** η οποία είναι μια βάση δεδομένων σε cloud, μορφής **NoSQL.** Αυτό σημαίνει ότι η βάση του είναι μη-σχεσιακή. Αυτό προσφέρει στον χρήστη περισσότερη ευελιξία και επεκτασιμότητα, καθώς τα δεδομένα είναι πιο εύκολα και γρήγορα προσβάσιμα.

Η Firestore μπορεί εύκολα να γίνει προσβάσιμη από τις εφαρμογές από όλες τις πλατφόρμες, όπως Android και Web, μέσω ενός έμφυτου SDK. Η συγκεκριμένη βάση υποστηρίζει πολλούς διαφορετικούς τύπους μεταβλητών όπως string, number, και ακόμα και πιο σύνθετους τύπους όπως εμφωλευμένα αντικείμενα.

Η Firestore έχει δημιουργηθεί με τέτοιο τρόπο στο cloud, έτσι ώστε να μπορεί να λειτουργεί γρήγορα, αποδοτικά και ευέλικτα. Λειτουργώντας με **Documents,** τα οποία περιέχουν πολλά **Collections και πεδία** τα οποία Collections με την σειρά τους έχουν πολλά Documents και ούτω καθεξής**,** όταν γίνονται queries στην Firestore, αυτή με την σειρά της επιστρέφει δεδομένα σε επίπεδο Document, χωρίς να χρειάζεται να γίνει ανάκτηση ολόκληρων Collections ή εμφωλευμένων **Subcollections**.



Σχήμα 3.3: Δομή του Firestore στο Console της Firebase

Στο παραπάνω Σχήμα 3.3 μπορούμε να παρατηρήσουμε την δομή του Firestore, και το πως λειτουργούν τα Documents και τα Collections. Η βάση ξεκινάει με Collections τα οποία έχουν Documents που το καθένα περιέχει fields, και Subcollections. Έτσι με αυτόν τον τρόπο όταν γίνονται queries, επιστρέφονται μόνο τα fields του Document. Για παράδειγμα αν κάνουμε ένα query για να μας επιστρέψει τον **user** με id `*85zqW6olP4*`, αυτό θα μας επιστρέψει τα παρακάτω δεδομένα σε μορφή **JSON:**

```
{
    "bio": "Test",
    "displayName" : "Stelios Chatzis",
    "ethnicity" : "gr",
    "mother_language": "Greek"
}
```

Η Firestore επίσης προσφέρει την δυνατότητα των **Security Rules** στο Console της Firebase, από όπου εκεί παρέχετε στους χρήστες να μπορούν να διαχειριστούν την πρόσβαση των δεδομένων της βάσης. Εκεί με μερικές γραμμές εντολών ο χρήστης μπορεί να περιορίσει την πρόσβαση ανάλογα σε όποιον επιθυμεί. Ένα παράδειγμα αυτών των εντολών είναι οι παρακάτω:

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Σε αυτό των κομμάτι εντολών στο Security Rules, ο χρήστης περιορίζει την πρόσβαση των δεδομένων της Firestore, όταν ο χρήστης δεν έχει συνδεθεί μέσω του **Firebase Authentication.** Έτσι με αυτόν τον τρόπο μπορούμε να δούμε ότι, όπως αναφέραμε και στην προηγούμενη ενότητα, ότι όλες οι υπηρεσίες της Firebase συνδέονται με έναν άμεσο τρόπο για την εύρυθμη λειτουργία της εφαρμογής.

3.4.3 Cloud Storage

Μια ακόμα χρήσιμη δυνατότητα για μια εφαρμογή, είναι να μπορεί να αποθηκεύει ολόκληρα αρχεία στο cloud. Με αυτόν τον τρόπο, εφαρμογές όπως του δημοσίου που περιέχουν πολλή γραφειοκρατία, η αποθήκευση των αρχείων σε ένα σύστημα cloud είναι απαραίτητη. Όπως και στην εφαρμογή για την οποία ασχολείται η Π.Ε., που πρόκειται για υπότιτλους βίντεο, σημαίνει ότι θα πρέπει να μπορεί να γίνεται διαχείριση τέτοιου είδους αρχείων και να γίνεται αποθήκευση στο cloud.

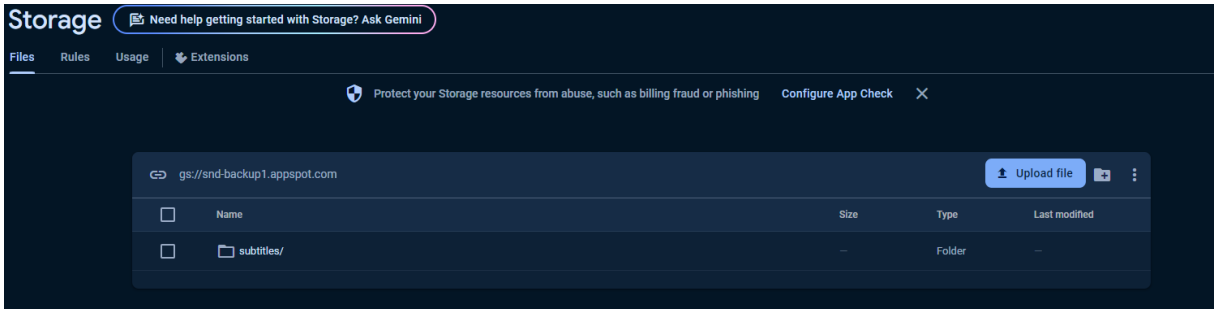
Η Firebase, δεν θα μπορούσε να μην έχει αυτήν την δυνατότητα, και το κάνει με την υπηρεσία της **Cloud Storage.** Η συγκεκριμένη υπηρεσία προσφέρει στους χρήστες της την δυνατότητα να αποθηκεύουν αρχεία διάφορων τύπων στο Cloud. Φυσικά και αυτό χρησιμοποιεί το δικό του SDK, ώστε να μπορεί να προστεθεί στην εκάστοτε εφαρμογή η δυνατότητα της χρήσης του Storage.

Πως όμως λειτουργεί το Storage; Στην ουσία στο Storage, υπάρχουν buckets ή αλλιώς *κάδοι*, οι οποίοι περιέχουν μέσα τα αρχεία που αποθηκεύει ο χρήστης. Τα buckets μπορούν φυσικά να περιέχουν και φακέλους και υποφακέλους, που μέσα τους το καθένα περιέχει αρχεία. Με αυτόν τον τρόπο το Storage προσφέρει την ευελιξία στους χρήστες της να μπορούν να αποθηκεύσουν αρχεία στην δομή που θέλουν αυτοί, και φυσικά να τα ανακτήσουν κάνοντας download, όποτε αυτοί το επιθυμούν

Για να μπορεί κάποιος χρήστης να κατεβάσει ή να ανεβάσει αρχεία από την εφαρμογή του, θα πρέπει να χρησιμοποιήσει τους τύπους μεταβλητών **Blob** που είναι μία δυαδικού τύπου μεταβλητή που αποθηκεύει μεγάλα μεγέθη δεδομένων, ή τύπου **File**, ώστε να μπορεί να μετατρέψει τα δεδομένα του από string, number, κ.α., σε μορφή αρχείου.

Επιπρόσθετα η Storage, δίνει την δυνατότητα να μπορεί κάποιος να κάνει επεξεργασία των αρχείων σε επίπεδο server, καθώς σε συνεργασία με το **Google Cloud Storage API,** παρέχεται η δυνατότητα του να φιλτράρει ή να κάνει κωδικοποίηση ενός βίντεο.

Και σε αυτήν την περίπτωση υπάρχουν τα **Security Rules,** που αναλαμβάνουν τον ρόλο του να μπορεί ο χρήστης να δίνει εξουσιοδότηση σε όποιον επιθυμεί αυτός, να κατεβάζει ή να ανεβάζει τα αρχεία στο Storage χωρίς να χρειάζεται να υλοποιήσει κάποιο authentication στην εφαρμογή του, με την χρήση μόνο του Firebase Authentication.



Σχήμα 3.4: Δομή του Storage στο Firebase Console της εφαρμογής

Παραπάνω μπορούμε να δούμε και την δομή που έχει η υπηρεσία του Storage στην Firebase. Παρατηρούμε πως μπορούμε να έχουμε και φακέλους εντός του bucket

3.4.4 Λοιπές Υπηρεσίες

Στις παραπάνω ενότητες αναπτύξαμε τρεις από τις πιο βασικές λειτουργίες της Firebase, το **Authentication,** το **Firestore** και το **Storage.** Φυσικά εκτός από αυτές τις υπηρεσίες, υπάρχουν και άλλες, που κάνουν την υλοποίηση μιας εφαρμογής πιο εύκολη για τους χρήστες της. Παρακάτω θα δούμε συνοπτικά μερικές από αυτές.

Μια από αυτές είναι το **Hosting,** όπου με αυτή δίνεται η δυνατότητα στον χρήστη να κάνει deploy την εφαρμογή του σε Hosting Servers της Firebase. Έτσι με αυτόν τον τρόπο μπορεί να τρέξει την εφαρμογή του σε cloud και να προβάλει δυναμικά το περιεχόμενο του. Εξυπηρετείται μέσω σύνδεσης SSL.

Επίσης υπάρχει και η **Realtime Database,** που είναι μια διαφορετικού τύπου βάση δεδομένων από αυτή της Firestore, και είναι τύπου JSON, σαν τις κλασσικές βάσεις NoSQL. Είναι για πιο απλές χρήσεις.

Τέλος η Firebase δίνει την δυνατότητα οι χρήστες της να προσθέσουν διάφορα **Extensions** επιθυμούν αυτοί, για να μπορέσουν να αναπτύξουν ακόμα περισσότερο τις δυνατότητες τις εφαρμογής τους.

3.5 Κοστολόγηση

Όλες οι παροχές της Firebase, παρέχονται δωρεάν, και το μόνο που χρειάζεται κανείς είναι να κάνει εγγραφή στη πλατφόρμα της Google. Βέβαια υπάρχουν περιορισμοί για την χρήση της κάθε υπηρεσίας που μπορεί να χρησιμοποιεί ο καθένας.

Για αυτό τον λόγο η Google έχει προσθέσει και πλάνα πληρωμών στην Firebase, όπου οι χρήστες πληρώνουν ένα μικρό αντίτιμο κάθε μήνα ώστε να αυξήσουν τον όριο της χρήσης της κάθε υπηρεσίας.

Για παράδειγμα, στην Firestore υπάρχει όριο **50 χιλιάδων document** που μπορεί ο χρήστης να ανακτήσει σαν δωρεάν πλάνο. Αν θέλει να αυξήσει αυτό το όριο, ας πούμε γιατί η εφαρμογή απευθύνεται σε περισσότερους πελάτες του, τότε θα πρέπει να καταβάλει ένα μικρό ποσό ώστε να αυξήσει το όριο. Με μόνο **6 cents του δολαρίου**, για παράδειγμα, μπορεί να αυξήσει το όριο αυτό στις **100 χιλιάδες**.

Για αυτό η Firebase από την αρχή σου προτείνει να προσθέσεις μια τραπεζική κάρτα ώστε να υπάρχει ελευθερία στο όριο αυτό. Αυτό γίνεται στο **Google Cloud** το οποίο θα το αναπτύξουμε παρακάτω.

3.6 Google Cloud

Όπως προαναφέραμε και πριν, η Firebase ανήκει στην Google, και βρίσκεται σε cloud επίπεδο. Έτσι όταν ένας χρήστης κάνει εγγραφή μια εφαρμογή στην Firebase, τότε αυτόματα δημιουργείτε μια εφαρμογή και στην πλατφόρμα της **Google Cloud**.

Η Google Cloud, είναι μια πλατφόρμα που προσφέρει υπηρεσίες cloud στο κοινό. Μπορεί να αποκτήσει ο οποιοσδήποτε πρόσβαση σε αυτή, όπως οι προγραμματιστές, οι διαχειριστές cloud, και άλλοι επαγγελματίες enterprise.

Μερικές από τις υπηρεσίες που προσφέρει η Google Cloud είναι:

- **Compute Engine:** Το οποίο είναι μια υποδομή υπηρεσίας η οποία προσφέρει στους χρήστες της **Virtual Machines (VM)** δηλαδή *Εικονικές Μηχανές*, με τις οποίες ο καθένας μπορεί να φιλοξενήσει τον φόρτο εργασίας του.
- **Cloud Storage:** Το οποίο χρησιμεύει για την αποθήκευση μεγάλων μη δομημένων data set. Επίσης προσφέρει και επιλογές για αποθήκευση βάσεων δεδομένων, όπως το **Google Datastore,** το οποίο είναι μια μη σχεσιακή βάση NoSQL. Την συγκεκριμένη υπηρεσία, την χρησιμοποιεί και το Storage της Firebase, που αναλύσαμε πιο πάνω

- **Kubernetes Engine:** Η συγκεκριμένη υπηρεσία χρησιμεύει για την διαχείριση εφαρμογών που βρίσκονται σε containers.
- **Gemini AI:** Φυσικά ένα από τα πιο σημαντικά χαρακτηριστικά της Google Cloud είναι και το σύστημα AI, το Gemini της Google, το οποίο φυσικά χρησιμεύει για την διευκόλυνση των χρηστών με την χρήση της Τεχνικής Νοημοσύνης.

3.6.1 Google Cloud APIs

Η χρήση του Google Cloud, δεν αφορά μόνο τις υπηρεσίες, αλλά προσφέρει και APIs που χρησιμεύουν για τη ανάκτηση δεδομένων που βρίσκονται στο περιβάλλον της Google.

Ένα από τα πιο γνωστά και χρήσιμα, ειδικά για αυτήν την Π.Ε., είναι το **API** του **YouTube.** Χάρη σε αυτό το API, η εφαρμογή μπορεί να λαμβάνει πολλά δεδομένα για τα βίντεο που υπάρχουν στην πλατφόρμα του YouTube. Δεδομένα όπως οι προβολές του βίντεο, η ονομασία του, η ημερομηνία κυκλοφορίας και άλλα χρήσιμα δεδομένα, ανακτώνται στην εφαρμογή της Π.Ε. χάρη σε αυτό το API που προσφέρει η Google Cloud.

Επίσης ένα χρήσιμο για την εφαρμογή της Π.Ε., είναι το API του **Google Translate.** Το συγκεκριμένο API προσφέρει την δυνατότητα στους χρήστες να μπορούν να μεταφράσουν λέξεις η και ολόκληρα κείμενα με την χρήση του Google Translate. Επίσης μπορεί να κάνει αναγνώριση της γλώσσας του κειμένου, και έτσι βοηθάει τους χρήστες να αναγνωρίζουν την γλώσσα, για παράδειγμα των υποτίτλων, και να μπορούν να τους μεταφράσουν στην γλώσσα που επιθυμούν.

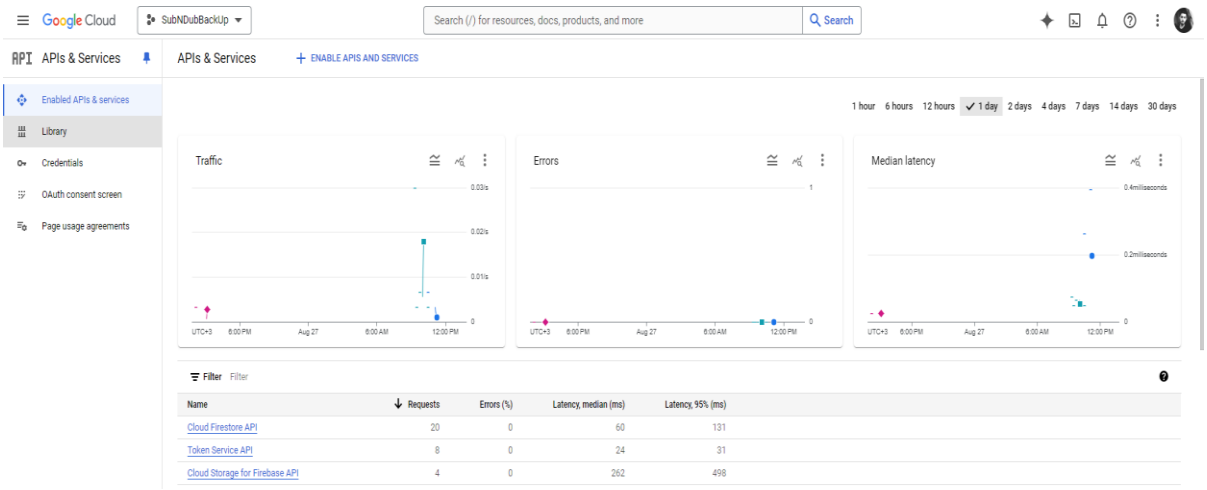
Φυσικά το πιο σημαντικό API που χρησιμοποιείται, είναι αυτό της Firebase, καθώς χωρίς αυτό το API δεν θα υπήρχε τρόπος επικοινωνίας της εφαρμογής και της Firebase.

Για να μπορέσει κάποιος να ανακτήσει δεδομένα σε ένα από αυτά τα API, αρκεί να μεταβεί στην σελίδα του Console του Google Cloud, και στην συνέχεια να δημιουργήσει ένα **API Key** στην κατηγορία **Credentials.** Αυτό το API Key της Google, είναι απαραίτητο για να μπορέσει κανείς να χρησιμοποιήσει οποιοδήποτε API θέλει μέσω του Google Cloud. Φυσικά αυτό το κλειδί είναι προσωπικό και σε καμία περίπτωση δεν θα πρέπει κάποιος να το δημοσιοποιήσει, καθώς κινδυνεύει από το να διαρρεύσουν ευαίσθητες πληροφορίες.

3.6.2 Google Cloud Shell

Το Google Cloud Shell είναι ένα εργαλείο της Google, το οποίο είναι προεγκατεστημένο στο Cloud, και προσφέρει περιβάλλον για την εκτέλεση εντολών shell. Με αυτό το εργαλείο οι χρήστες μπορούν να εκτελέσουν εντολές για να μπορέσουν αν διαχειριστούν της υπηρεσίες και τους πόρους που υπάρχουν στο Google Cloud, για να το αξιοποιήσουν για την εφαρμογή τους.

Σε αυτό υπάρχει και ένας online editor, που μοιάζει αρκετά με το Visual Studio, με το οποίο ο καθένας μπορεί να δει τα αρχεία του Cloud του. Φυσικά υπάρχει και περιορισμός στον αποθηκευτικό χώρο του Cloud που είναι τα **5 GB.**



Σχήμα 3.5: Προεπισκόπηση του Console του Google Cloud

3.7 Επίλογος

Εν κατακλείδι, κατανοήθηκε πλήρως η Firebase και το πόσο χρήσιμη είναι για την ανάπτυξη μιας εφαρμογής που μπορεί να είναι σε οποιαδήποτε πλατφόρμα όπως web, Android, iOS. Με τις υπηρεσίες που προσφέρει, όπως το Firestore, το Storage, το Authentication, μπορεί να γίνει ένα πανίσχυρο εργαλείο για την υλοποίηση ενός back-end, και δεν είναι τυχαίο που τον τελευταίο καιρό γίνεται ολοένα και πιο δημοφιλές και προτιμάται από όλο και περισσότερους χρήστες, μιας και μπορεί να τους «γλυτώσει» πολλή εργασία, αφού την δουλεία αυτή την αναλαμβάνει η Firebase. Επίσης επεξηγήθηκε και το Google Cloud και το πόσο χρήσιμο είναι για να λειτουργήσει η Firebase, και όχι μόνο, καθώς προσφέρει πολύτιμα εργαλεία και APIs που μπορεί ένας να χρησιμοποιήσει για να υλοποιήσει την εφαρμογή του.

Κεφάλαιο 4ο: Git

Σχήμα 4.1: Το λογότυπο του Git

4.1 Τι είναι το Git

Για να υλοποιηθεί μια εφαρμογή, συνήθως απαιτούνται πολλοί developers, που θα γράψουν ο καθένας κομμάτια κώδικα σε πολλά διαφορετικά αρχεία, που όλα μαζί θα πρέπει να ενωθούν ώστε να γίνει μια ολοκληρωμένη εφαρμογή. Για να γίνει αυτό θα πρέπει να υπάρχει ένα σύστημα το οποίο θα μπορεί να διαχειρίζεται τις τροποποιήσεις των αρχείων, αλλά και να μπορεί να τις ελέγχει ανάλογα με τις εκδόσεις του ώστε να γνωρίζει πότε τροποποιήθηκε και από ποιον.

Έτσι λοιπόν έχουν δημιουργηθεί συστήματα διαχείρισης εκδόσεων. Ένα από αυτά είναι το **Git,** το οποίο είναι ένα τέτοιο version control εργαλείο το οποίο διευκολύνει τους developers να διαχειρίζονται τις εκδόσεις από την εφαρμογή τους.

Το Git είναι ένα δωρεάν open-source εργαλείο που προσφέρει μια πλούσια γκάμα από εντολές υψηλού επιπέδου που μπορεί ο καθένας να τις χρησιμοποιήσει ώστε να έχει πρόσβαση στο εσωτερικό κομμάτι μιας εφαρμογής.

4.2 Ιστορία του Git

Το Git δημιουργήθηκε από τον Linus Torvalds τον Απρίλιο του 2005, και ο αρχικός του σκοπός ήταν για να χρησιμοποιηθεί για την ανάπτυξη του Linux kernel [15]. Ο Torvalds ήθελε να υλοποιήσει ένα σύστημα σαν το **BitKeeper**, το οποίο ήταν ένα επίσης ένα σύστημα διαχείρισης εκδόσεων το οποίο όμως πλέον βρίσκεται στην φάση του **End-of-Life.**

Πλέον ο Git έχει γίνει το de-facto σύστημα για την διαχείριση εκδόσεων, καθώς είναι το πιο δημοφιλές σύστημα τέτοιου είδους, καθώς το 95% των developers το χρησιμοποιούν για να υλοποιήσουν τις εφαρμογές τους

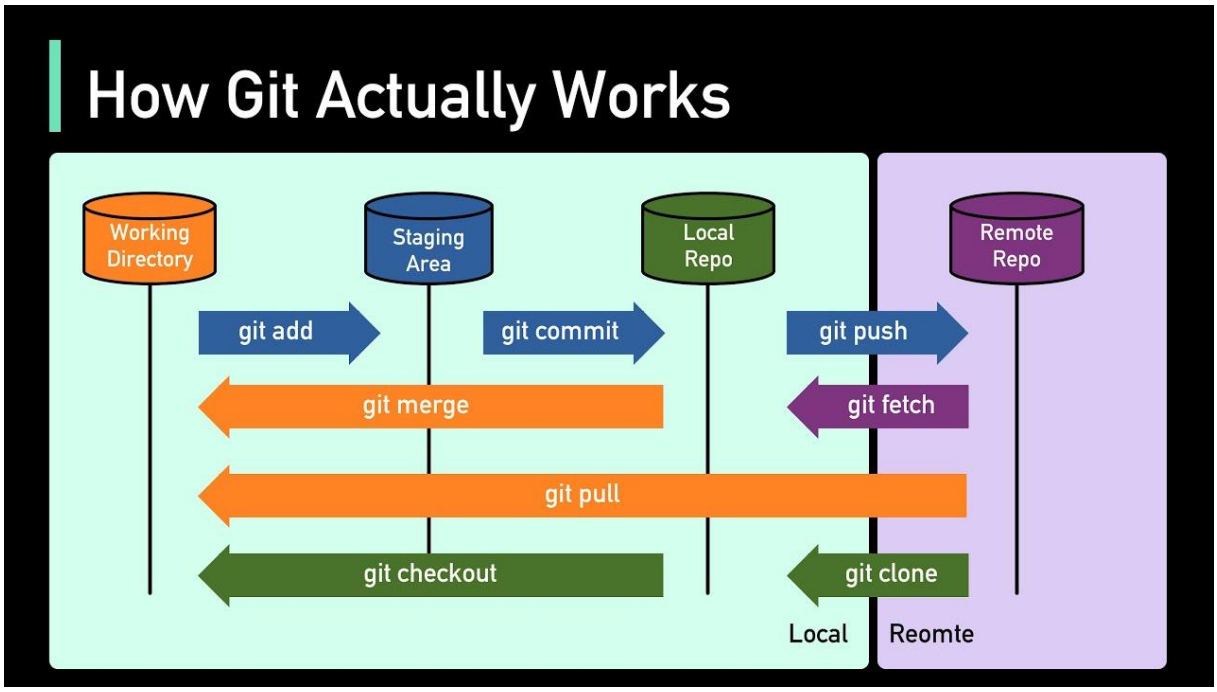
4.3 Μηχανική του Git

Το Git είναι ένα κατανεμημένο σύστημα διαχείρισης εκδόσεων, που αυτό σημαίνει ότι υπάρχει ένας κεντρικός server που περιέχει την τελευταία έκδοση του κώδικα μιας εφαρμογής. Ενώ ταυτόχρονα ο κάθε χρήστης έχει από ένα δικό του αντίγραφο στον προσωπικό του υπολογιστή. Ο κώδικας αποθηκεύεται σε ένα **remote repository** και περιλαμβάνει όλο το ιστορικό του κώδικα και φυσικά τον χρήστη ο οποίος τον τροποποίησε.

Με αυτόν τον μηχανισμό, το Git επιτρέπει στους χρήστες του να δουλεύουν παράλληλα στον κώδικα. Αυτό βέβαια μπορεί να δημιουργήσει προβλήματα, καθώς δύο ή περισσότεροι developers μπορεί να

δουλεύουν στο ίδιο αρχείο. Έτσι το Git προσφέρει τα **branches,** τα οποία χρησιμεύουν για αυτόν ακριβώς σκοπό. Ο κάθε developer δουλεύει στα δικά του branches, που περιέχουν την δική του έκδοση αρχείων στο τοπικό του repository, και όταν έρθει η ώρα, για παράδειγμα να βγει μια νέα έκδοση της εφαρμογής, τότε οι developers ενώνουν αυτά τα branches με την μέθοδο του **merge**, ώστε να τα συγχωνευτούν όλα μαζί στο κεντρικό branch, το **main.** Κατά την συγχώνευση μπορούν να αναλύσουν τις αλλαγές αλλά και τα **conflicts** που μπορεί να υπάρξουν κατά την συγχώνευση μεταξύ των αρχείων και έτσι να αποφασίσουν ποιες αλλαγές θα κρατήσουν και ποιες δεν θα τις δεχτούν.

4.4 Git Commands



Σχήμα 4.2: Εικονική απεικόνιση των commands του Git

Για να μπορέσει όμως να δουλέψει η μηχανική του Git, έχουν δημιουργηθεί εντολές, τα λεγόμενα **Git Commands,** όπου με την χρήση αυτών οι χρήστες μπορούν να αντιγράφουν, ενημερώνουν και να συγχωνεύουν τον κώδικά τους με το remote repository. Η χρήση των commands γίνεται μέσω του **git bash,** το οποίο είναι ένα emulator, το οποίο μπορεί να τρέξει τις συγκεκριμένες εντολές του git. Το bash emulator, στην ουσία συμπεριφέρεται σαν να βρίσκεται σε περιβάλλον **Linux,** που είναι απαραίτητο για την εκτέλεση των εντολών**.**

Καθώς υπάρχουν πάρα πολλά commands παρακάτω θα αναλύσουμε μόνο μερικά από τα πιο σημαντικά, μιας και αυτά χρησιμοποιήθηκαν για τις ανάγκες της Π.Ε.

- **git init:** Με την χρήση αυτής της εντολής, ο developer μπορεί να δημιουργήσει ένα νέο τοπικό repository στον υπολογιστή του.
- **git clone:** Αυτή η εντολή χρησιμοποιείται για να μπορέσει ο χρήστης να αποκτήσει και να αντιγράψει ένα remote repository, χρησιμοποιώντας το URL του repo.
- **git add:** Με αυτή την εντολή γίνεται η προσθήκη ενός η περισσοτέρων αρχείων στο staging area. Με την του αστερίσκου * γίνεται προσθήκη όλων των αρχείων στο staging.
- **git commit:** Στη συνέχεια με την συγκεκριμένη εντολή, ο χρήστης δημιουργεί ένα snapshot των αρχείων που είναι στο staging, ώστε να περαστεί στο ιστορικό των εκδόσεων. Με την χρήση του **-m** , ο χρήστης προσθέτει και ένα σχόλιο στο commit, που είναι απαραίτητο. Ενώ

με την εντολή **-a** , γίνεται commit όλων των αρχείων που έχουν προστεθεί μετά την εντολή git add.

- **git push:** Αυτή η εντολή χρησιμοποιείται ώστε να στείλει όλες τις αλλαγές που έχουν γίνει commit στο remote repository που έχει ο χρήστης. Δίπλα από την εντολή ο χρήστης πληκτρολογεί και το όνομα του branch που θα στείλει τις αλλαγές, π.χ. αν πληκτρολογήσει το main, τότε θα αποσταλούν στο main branch του repository.
- **git remote:** Με αυτήν την εντολή οι χρήστες συνδέουν το local repository με το remote repository που έχουν. Δίπλα πρέπει να πληκτρολογήσουν και το URL του remote repository.
- **git pull:** Με αυτό το command οι χρήστες φέρνουν και συγχωνεύουν τον κώδικα που υπάρχει στο remote repository, με αυτόν που υπάρχει στο local repository. Κατά την διάρκεια του merge, υπάρχουν και οι περιπτώσεις των conflicts, δηλαδή να έχουν επηρεαστεί τα ίδια αρχεία και στο local αλλά και στο remote repository. Έτσι θα πρέπει να γίνει σωστή συγχώνευση των αρχείων.
- **git checkout:** Μιας και στο repository μπορούν να υπάρχουν πολλά branches όπως αναφέραμε πιο πάνω, με αυτήν την εντολή ο χρήστης μπορεί να μεταβεί στο branch που επιθυμεί αυτός. Φυσικά δίπλα στην εντολή θα πρέπει να πληκτρολογήσουν και το όνομα του branch.

Χάρη σε αυτές και άλλες περισσότερες εντολές, οι developers μπορούν να χρησιμοποιήσουν με ευελιξία το git, για να κάνουν έλεγχο στα versions της εφαρμογής τους.

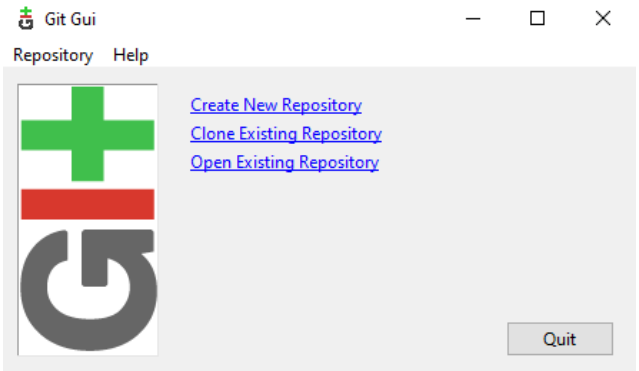
4.5 Γραφικές διεπαφές του Git

Το Git, όπως αναφέρθηκε και πριν, έχει το δικό του emulator, το git bash, το οποίο χρησιμεύει για την εκτέλεση εντολών του git. Καθώς όμως οι χρήστες, ειδικά των Windows, ολοένα και αυξάνονται, υπήρχε η ανάγκη για την δημιουργία μιας γραφικής διεπαφής, δηλαδή εφαρμογής, που να μπορούν οι χρήστες να εκτελούν τις εντολές του Git με μεγαλύτερη ευκολία και κατά κύριο λόγο, να έχουν την οπτική του τι αλλαγές έχουν πραγματοποιηθεί στο repository τους.

Έτσι παρακάτω θα μιλήσουμε για μερικές από τις εφαρμογές αυτές που δημιουργήθηκαν με σκοπό να αντικαταστήσουν το git bash.

4.5.1 Git GUI

Μια από αυτές τις γραφικές διεπαφές που δημιουργήθηκαν, είναι το **Git GUI.** Το πρόγραμμα αυτό προστίθεται την ώρα που οι χρήστες εγκαθιστούν το Git στον υπολογιστή τους, και είναι desktop εφαρμογή. Οι δυνατότητές του είναι να μπορεί να απεικονίσει με γραφικά, φιλικά προς τον χρήστη, τις αλλαγές που πραγματοποιούνται κατά την διάρκεια των commit, merge, αλλαγών repository, καθώς μπορεί να δείξει ξεκάθαρα τις όποιες τροποποιήσεις έχουν πραγματοποιήσει οι χρήστες.



Σχήμα 4.3: Απεικόνιση του Git GUI

4.5.2 GitHub

Μια από τις πιο σημαντικές, αλλά και πιο χρησιμοποιημένες εφαρμογές διεπαφής για το git, είναι αυτή **GitHub.** Το GitHub είναι μια web εφαρμογή στην οποία οι χρήστες μπορούν να προβάλουν τα remote repository τους αλλά και τα branches τους. Είναι βασισμένο στο cloud, καθώς εκεί αποθηκεύονται όλα τα repository από όλους τους χρήστες. Είναι, ίσως το πιο δημοφιλές web app που αφορά την διαχείριση των εκδόσεων των εφαρμογών.

Σε αντίθεση με το git, το οποίο χρησιμοποιείται πιο πολύ για την διαχείριση του local repository, το GitHub προσθέτει στην ουσία τα επίπεδα της συνεργασίας, διαχείρισης ενός project αλλά και αποθηκευτικού χώρου στο cloud.

Το GitHub προσφέρει πολλά χρήσιμα εργαλεία όπως αυτό του **CI/CD,** το οποίο αφορά την συνεχή ανάπτυξη και ενοποίηση, που στην ουσία αυτοματοποιεί τα testing, builds και deployments μιας εφαρμογής

Επίσης προσφέρει και τα εργαλεία της συνεργασίας που πρέπει να υπάρχουν για project με πολλούς χρήστες, όπως αυτό του **Pull Request**, το οποίο δίνει την δυνατότητα στους developers να μπορούν να αποφασίσουν και να προτείνουν τις αλλαγές στον κώδικα που θέλουν να αναθεωρήσουν.

Το GitHub χρησιμοποιήθηκε και για τις ανάγκες τις **Π.Ε.**, καθώς τα εργαλεία που προσφέρει με βοήθησαν προσωπικά να εργαστώ με ευκολία στις αλλαγές και στις εκδόσεις που πραγματοποίησα κατά την υλοποίηση της εφαρμογής αυτής, καθώς επίσης με βοήθησε και να μπορέσω να παρατηρήσω όλα τα προβλήματα και bugs που είχαν δημιουργηθεί, ώστε να μπορέσω να δω το ιστορικό των αλλαγών και να τα διορθώσω με επιτυχία.

4.5.3 GitLab

Ένα ακόμα παρόμοιο πρόγραμμα το οποίο μοιάζει πολύ με το GitHub, είναι το **GitLab**, το οποίο είναι εξίσου μια εφαρμογή στο web και βασίζεται στο cloud, καθώς εκεί αποθηκεύονται όλα τα remote repositories.

Σε αντίθεση με το GitHub, το GitLab προσφέρει μια πιο ολοκληρωμένη πλατφόρμα all-in-one που αφορά ολόκληρο τον κύκλο ανάπτυξης ενός λογισμικού, από την αρχή του, έως και το στάδιο του end-of-life του.

Το GitLab κατά κύριο λόγο προτιμάται γιατί προσφέρει μια πιο ολοκληρωμένη DevOps πλατφόρμα, και για αυτό αρκετές εταιρίες ανάπτυξης λογισμικού, το χρησιμοποιούν.

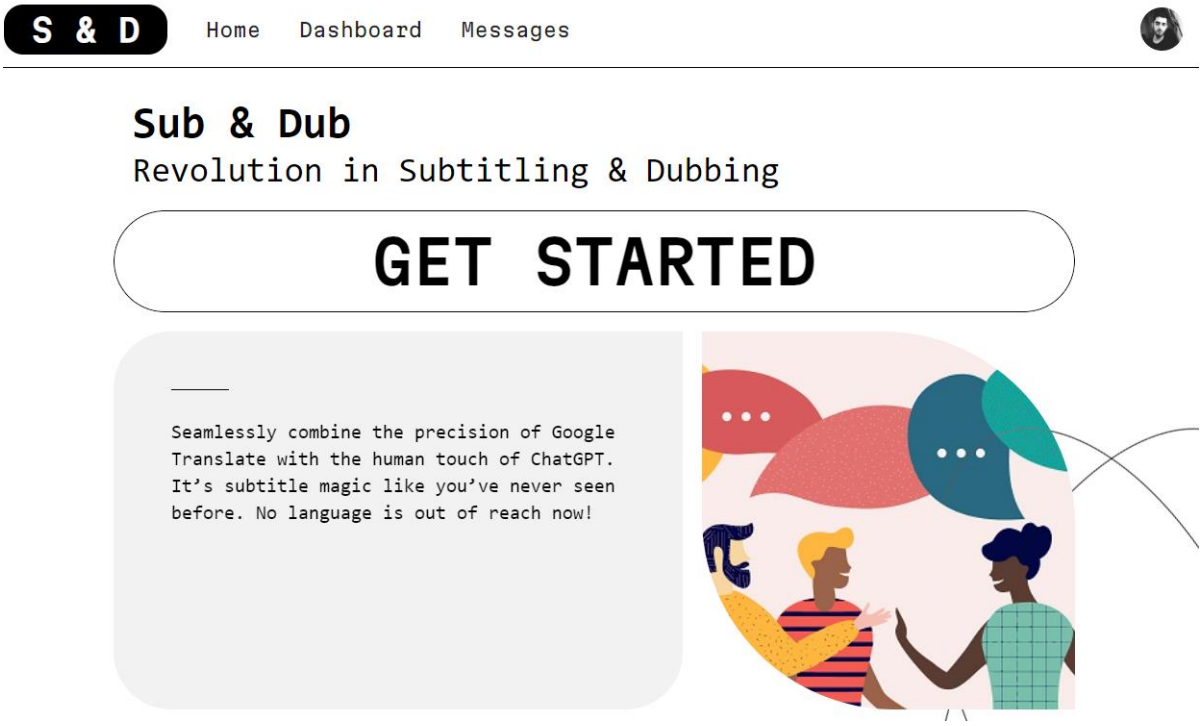
4.5.4 GitHub Desktop

Αν και πολλά **IDE,** δηλαδή ολοκληρωμένα περιβάλλοντα ανάπτυξης, όπως το **IntelliJ,** προσφέρουν την δυνατότητα εκτέλεσης εντολών git, ακόμα και πιο αυτοματοποιημένα, υπήρχαν πολλά άλλα προγράμματα τα οποία δεν παρείχαν αυτή την υποστήριξη, και έτσι το GitHub δημιούργησε και μια desktop εφαρμογή, το **GitHub Desktop,** το οποίο προσφέρει αυτήν την υποστήριξη με γραφικά φιλικά προς τον χρήστη τα οποία τον βοηθούν να διαχειριστεί τα versions από την εφαρμογή του.

4.6 Επίλογος

Όπως μπορούμε να καταλάβουμε, το να μπορεί μια ομάδα ανάπτυξης λογισμικού να ελέγχει τις εκδόσεις των αρχείων και του κώδικά της, αλλά και να μπορεί να παρατηρεί το ποιος έχει πραγματοποιήσει την εκάστοτε αλλαγή, είναι ένα πολύ σημαντικό μέρος για την ανάπτυξη του λογισμικού. Επομένως η ανάγκη του ενός version control system σαν το **Git** είναι άμεση. Με το Git είδαμε ότι μπορεί ο κάθε developer να έχει πλήρη έλεγχο στις εκδόσεις της εφαρμογής του, και με τις εντολές του git, αυτό γίνεται ακόμα πιο εύκολο και ευέλικτο. Φυσικά είδαμε και για τις γραφικές διεπαφές του git, δηλαδή εφαρμογές σαν το GitHub που προσφέρουν οπτική απεικόνιση του τι έχει αλλάξει στον κώδικα μιας εφαρμογής, και φυσικά τον σημαντικό ρόλο του στην ανάπτυξη της εφαρμογής της Π.Ε.

Κεφάλαιο 5ο: Περιγραφή της Εφαρμογής



Σχήμα 5.1: Αρχική σελίδα της εφαρμογής

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναπτύξουμε και το πιο σημαντικό μέρος αυτής της Πτυχιακής Εργασίας, την υλοποίηση και περαιτέρω εξέλιξη της εφαρμογής **Sub & Dub.** Ο σκοπός της υλοποίησης αυτής της Π.Ε., ήταν να προστεθούν νέες δυνατότητες στην εφαρμογή, αλλά και να διορθωθούν μερικά σφάλματα τα οποία εμπόδιζαν την σωστή ροή της λειτουργίας της εφαρμογής.

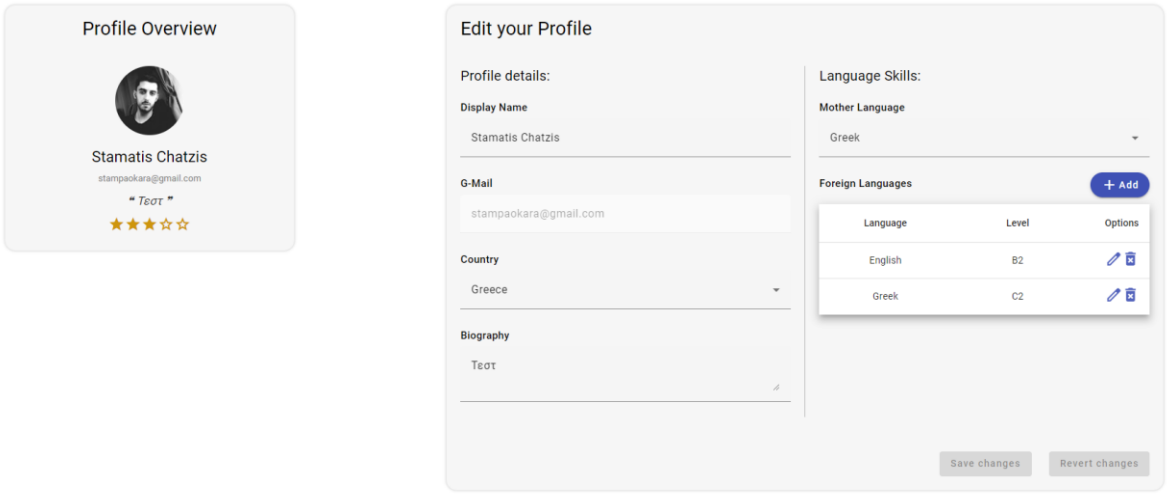
Μερικές από τις προσθήκες που υλοποιήθηκαν ήταν η προσθήκη μιας σελίδας διαχείρισης του προφίλ του χρήστη που συνδέετε στην εφαρμογή, η αυτόματη αναγνώριση της γλώσσας του βίντεο και των υποτίτλων του, και άλλα που θα αναλυθούν πλήρως στις επόμενες υποενότητες.

Επίσης υπήρξαν και μερικές τροποποιήσεις στην λειτουργία και εμφάνιση της εφαρμογής ώστε να γίνει πιο φιλική προς τον χρήστη, αλλά και να διορθωθούν προβλήματα αποθήκευσης και μεταφόρτωσης των υποτίτλων.

5.2 Profile page

Θα ξεκινήσουμε μιλώντας για ένα νέο χαρακτηριστικό που προστέθηκε στην εφαρμογή, την σελίδα που είναι της **Διαχείρισης του Προφίλ του χρήστη.** Ο χρήστης αφού συνδεθεί στην εφαρμογή για πρώτη φορά μέσω του λογαριασμού του στο **Google**, θα παραπεμφθεί στο να συμπληρώσει μερικές απαραίτητες πληροφορίες στο προφίλ του.

Το προφίλ, για μια εφαρμογή που έχει να κάνει με συνδεδεμένους χρήστες, είναι πολύ σημαντικό καθώς πρόκειται για την «ταυτότητα» του εκάστοτε χρήστη που είναι συνδεδεμένος, καθώς θα πρέπει να δείξει και στους υπόλοιπους χρήστες στην κοινότητα της εφαρμογής σημαντικές πληροφορίες για αυτόν.



Σχήμα 5.2: Άποψη της σελίδας του προφίλ του χρήστη

Όπως μπορούμε να παρατηρήσουμε και στο Σχήμα 5.2, η σελίδα είναι χωρισμένη σε δύο τμήματα.

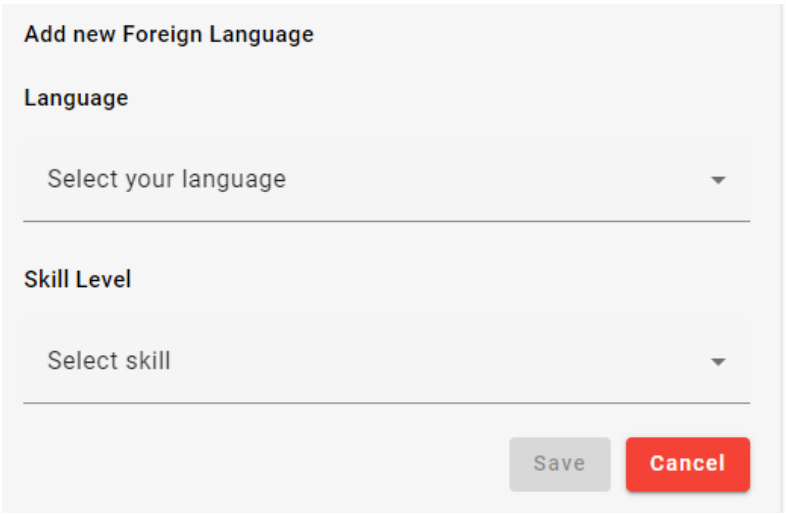
Το ένα, το δεξί τμήμα, αφορά το **Profile Overview,** το οποίο είναι και η προεπισκόπηση του προφίλ του χρήστη που είναι συνδεδεμένος, και το πως θα εμφανίζεται στους υπόλοιπους χρήστες. Εκεί υπάρχουν βασικές πληροφορίες όπως το **όνομα** του χρήστη, το **Gmail** του, το **βιογραφικό** του το οποίο εμφανίζεται μόνο αν υπάρχει, και τέλος το **rating,** το οποίο επίσης είναι μια νέα λειτουργία η οποία θα αναπτυχθεί σε επικειμένη υποενότητα.

Στο αριστερό τμήμα, υπάρχουν τα στοιχεία του προφίλ που ο κάθε χρήστης μπορεί να προσθέσει ή και να τροποποιήσει. Χωρίζεται σε δύο υποτμήματα, τις γενικές πληροφορίες του profile και τις γλωσσικές δυνατότητες του χρήστη.

Στα γενικά στοιχεία του προφίλ που βρίσκεται στα δεξιά του, υπάρχουν διάφορα πεδία για να διαμορφώσει το προφίλ του, όπως το **display name,** το οποίο είναι το όνομα του χρήστη που εμφανίζεται. Μετά υπάρχει το πεδίο του **Gmail,** το οποίο όμως δεν γίνεται να τροποποιηθεί καθώς πρόκειται για το Gmail του λογαριασμού της Google, που στην είναι ο τρόπος που συνδέεται ο χρήστης στην εφαρμογή. Έπειτα υπάρχει ένα dropdown, το οποίο περιέχει όλες τις χώρες του πλανήτη, που ανακτώνται μέσω ενός αρχείου JSON που υπάρχει στην εφαρμογή, ώστε ο χρήστης να διαλέξει την **χώρα** του. Έπειτα υπάρχει ένα text area πεδίο στο οποίο χρήστης μπορεί να προσθέσει μερικές πληροφορίες για τον εαυτό του σαν ένα μικρό **βιογραφικό.**

Μετά στο αριστερό μέρος, υπάρχουν τα πεδία που αφορούν τις γλωσσικές δυνατότητες του χρήστη. Το πρώτο πεδίο αφορά την **μητρική γλώσσα**, το οποίο επίσης είναι ένα dropdown που περιέχει όλες τις

διαθέσιμες γλώσσες του πλανήτη, που ανακτώνται μέσω του **API** του Google Translate. Ενώ τέλος υπάρχει ένας πίνακας που περιέχει όλες τις γλωσσικές του γνώσεις πέρα της μητρικής του. Κάνοντας κλικ στο κουμπί `+ **Add`,** ο χρήστης μπορεί να προσθέσει την ξένη γλώσσα που γνωρίζει αλλά και το επίπεδο γνώσης που έχει για αυτήν. Στο παρακάτω Σχήμα 5.3, μπορεί να παρατηρηθεί η φόρμα με την οποία ο χρήστης μπορεί να προσθέσει μια ξένη γλώσσα. Φυσικά υπάρχουν τα κουμπιά **Save** που αποθηκεύουν την ξένη γλώσσα, και το **Cancel,** το οποίο ακυρώνει την διαδικασία.



Σχήμα 5.3: Φόρμα προσθήκης νέας ξένης γλώσσας

Στο πίνακα που περιέχει τις γλώσσες υπάρχουν και οι επιλογές να τροποποιήσεις μια γλώσσα ή και να την αφαιρέσεις από τον πίνακα του Foreign Languages.

Αφού ο χρήστης πραγματοποιήσει τις όποιες αλλαγές ή και προσθήκες στο προφίλ του, με το πάτημα του κουμπιού **Save Changes,** θα μπορεί να αποθηκεύσει το προφίλ του, το οποίο θα ενημερώσει και την βάση στο **Firestore**. Επίσης υπάρχει και ένα κουμπί **Revert Changes,** το οποίο αναιρεί τις όποιες αλλαγές έχει κάνει ο χρήστης, και το γυρνάει στην μορφή που είχε αποθηκευτεί τελευταία.

Όλες αυτές οι πληροφορίες του προφίλ υλοποιήθηκαν σε ένα component της Angular το οποίο και έχει όλη την λογική της απεικόνισης της σελίδας αυτής. Το component κατά τον κύκλο **ngOnInit**(), φορτώνει όλα τα στοιχεία του προφίλ του χρήστη από τα services. Στο component, εκτός από το service του προφίλ που υλοποιείται η λογική της αποθήκευσης των δεδομένων για τις γλωσσικές δυνατότητες και του προφίλ, γίνονται inject και άλλα services. Το service του auth, το οποίο είναι απαραίτητο για την ανάκτηση των στοιχείων του συνδεδεμένου χρήστη και το GoogleTranslateService το οποίο είναι απαραίτητο για να γίνονται ανάκτηση οι γλώσσες.

Η φόρμα για τις πληροφορίες του προφίλ, υλοποιείται με το **FormGroup**, το οποίο είναι βιβλιοθήκη της Angular για να προσθέτει φόρμες εισαγωγής δεδομένων και να περιορίζει τον χρήστη στο τι δεδομένα μπορεί να προσθέσει στο κάθε πεδίο με τη χρήση των **Validators**.

profile.service.ts:

```
export class ProfileService {
  otherLanguages$: Observable<ForeignLanguage[]>;

  constructor(private firestore: AngularFirestore, private http: HttpClient) {}

  getCountries(){ Show usages  StamChatzis
    const countryUrl = "../../assets/data/google-countries.json";
    return this.http.get(countryUrl);
  }
```

profile.component.ts :

```
ngOnInit() { Show usages  StamChatzis +1
  this.user$.subscribe({
    next: data => {
      this.loadUserDetails(data);
      this.uid = data.uid;
      this.photoUrl = data.photoURL;
      this.email = data.email;
      this.loadUserRatings(data.uid)
      this.proService.getAllLanguages(data.uid).subscribe({
        next: data => {
          this.loadForeignLanguages(data)
        }
      });
    }
  });

  this.proService.getCountries().subscribe({
    next: data => {this.loadCountries(data);}
  });
```

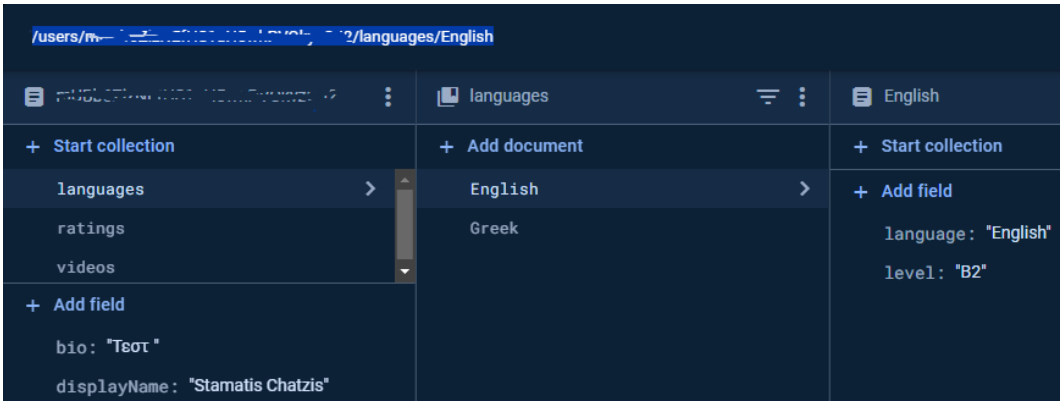
Σχήμα 5.4: Κομμάτια κώδικα για την ανάκτηση των χωρών και πληροφοριών του user

Στα παραπάνω κομμάτια κώδικα στο Σχήμα 5.4, φαίνεται το πώς έχει υλοποιηθεί η ανάκτηση των δεδομένων των χωρών και των στοιχείων του χρήστη. Τα στοιχεία του user ανακτώνται από το **user$** το οποίο είναι ένα observable, το οποίο φορτώνεται από το injected service του auth. Ενώ οι χώρες, φορτώνονται από ένα αρχείο json που περιέχει όλες τις χώρες, με την μέθοδο του **http get.** Και στα δύο γίνεται subscribe για να ανακτηθούν οι ροές των δεδομένων τους και γίνεται στον κύκλο ngOnInit της ζωής του component, καθώς θέλουμε να γίνεται στην αρχή της δημιουργίας του component.

```
addForeignLanguages(uid: any, foreignLang: ForeignLanguage[]) { Show usages  schatzis +1
  for(let lang of foreignLang){
    const language = lang.language;
    const langRef: AngularFirestoreDocument<ForeignLanguage> = this.firestore.doc(`users/${uid}/languages/${language}`);
    langRef.set(lang, {merge: true}).catch(ex => {
      console.log(ex.message)
      return false
    });
  }
  return true
}
```

Σχήμα 5.5: Υλοποίηση της αποθήκευσης των ξένων γλωσσών στην Firebase

Στο Σχήμα 5.5 απεικονίζεται ο τρόπος με τον οποίο γίνεται η αποθήκευση των ξένων γλωσσών στην Firebase στο **profile service**. Για να μπορέσει να υπάρχει επικοινωνία του service με την Firestore, γίνεται inject το **AngularFirestore,** το οποίο είναι από το SDK που εγκαταστάθηκε στην εφαρμογή. Έπειτα η μέθοδος **addForeignLanguages**, δέχεται δύο παραμέτρους, αυτή του id του user και ένα Array τύπου **ForeignLanguage**. Στην αρχή γίνεται το reference στην Firestore και το URL όπου θα αποθηκευτούν οι γλώσσες, και μετά καλείτε το **.set** του συγκεκριμένου reference, που στην ουσία είναι αυτό που αποθηκεύει τις ξένες γλώσσες. Έτσι γενικότερα, αποθηκεύονται τα δεδομένα στην Firebase, και συγκεκριμένα στην Firestore Database όπως φαίνεται και στο παρακάτω Σχήμα 5.6.

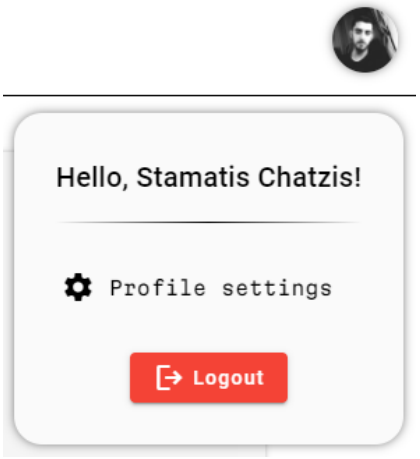


Σχήμα 5.6: Απεικόνιση των αποθηκευμένων γλωσσών στην Firestore.

Αυτά φυσικά είναι μερικά μόνο από τα κομμάτια του κώδικα που υλοποιήθηκαν, καθώς θα ήταν αδύνατο να προβληθούν όλοι οι τρόποι με τους οποίους υλοποίησα την εφαρμογή.

5.2.1 Profile Dropdown

Για να μπορέσει ο χρήστης να μεταβεί στο προφίλ του από οποιοδήποτε σημείο της εφαρμογής, αρκεί να πατήσει πάνω δεξιά στην εικόνα του προφίλ. Εκεί θα εμφανιστεί ένα dropdown με τις επιλογές ο χρήστης να μεταβεί στο προφίλ του, και να κάνει **log out** από την εφαρμογή. Αυτό επίσης είναι ένα νέο χαρακτηριστικό για να διευκολύνει τους χρήστες να χρησιμοποιούν τις συγκεκριμένες επιλογές, μιας και είναι αρκετά παρόμοιο με το profile dropdown της **Google**. Παρακάτω στο Σχήμα 5.7, φαίνεται η μορφή του συγκεκριμένου dropdown.



Σχήμα 5.7: Άποψη του dropdown όταν γίνεται κλικ στην profile picture

5.3 Σύστημα αξιολόγησης

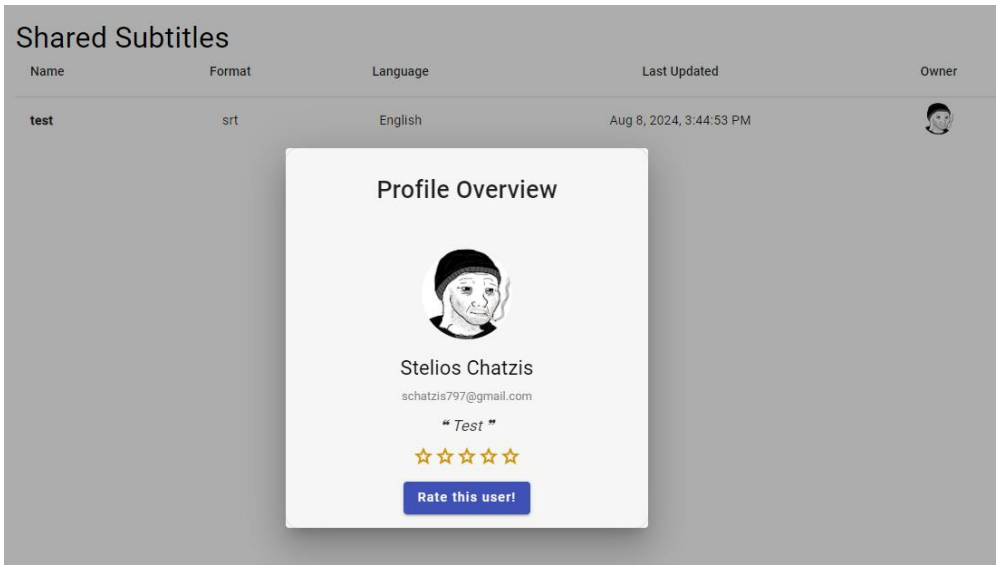
Καθώς η εφαρμογή αφορά την δημιουργία και μεταγλώττιση υποτίτλων με την βοήθεια του **community** που υπάρχει στην εφαρμογή, αλλά και την δημιουργία προσφορών για να μπορέσει ένας χρήστης να αναλάβει την δημιουργία υποτίτλων για άλλους χρήστες που ζήτησαν αυτήν την υπηρεσία, ήταν απαραίτητο να δημιουργηθεί και ένα **σύστημα αξιολόγησης** των χρηστών της κοινότητας.

Με το σύστημα της αξιολόγησης, οι χρήστες θα μπορούν να βλέπουν την βαθμολογία του rating ενός χρήστη για να μπορέσουν να αξιολογήσουν αν θα πρέπει να βασιστούν σε αυτόν τον χρήστη για να υλοποιήσει την δημιουργία των υποτίτλων που ζήτησαν ή όχι.

Η αξιολόγηση βασίζεται σε ένα σύστημα 5 αστέρων, με το 1 αστέρι να αφορά την χαμηλότερη βαθμολογία, και το 5 την υψηλότερη. Εκεί ο χρήστης μπορεί να αξιολογήσει με ένα από τα 5 αστέρια που υπάρχουν. Επίσης δύναται στον χρήστη η δυνατότητα να προσθέσει ένα σχόλιο που θα αφορά την αξιολόγηση του χρήστη.

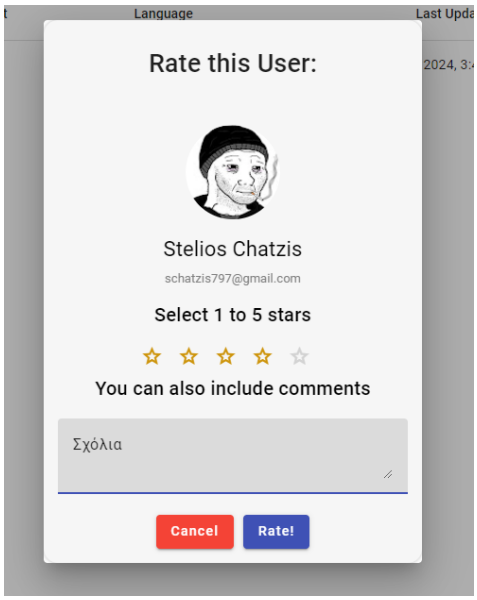
Για να αξιολογήσει κάποιος έναν χρήστη θα πρέπει να μεταβεί στο **Dashboard** και έπειτα στα **Shared Videos,** δηλαδή στα βίντεο που του έχουν κοινοποιηθεί από κάποιον άλλον χρήστη για να τα μεταφράσει ή υποτιτλίσει. Εκεί, αφού επιλέξει ένα βίντεο θα του εμφανιστεί μια οθόνη με έναν πίνακα με όλους τους διαθέσιμους υπότιτλους του βίντεο. Σε μία από τις στήλες του πίνακα, υπάρχει και η στήλη **Owner,** που υποδηλώνει τον ιδιοκτήτη του βίντεο. Εκεί εμφανίζεται η εικόνα του ιδιοκτήτη, και, αν κάνει κλικ πάνω της, θα εμφανιστεί ένα πλαίσιο με τα στοιχεία του χρήστη, τις αξιολογήσεις του, και ένα button το οποίο γράφει **`Rate this user!`,** όπως φαίνεται και στο Σχήμα 5.8 παρακάτω, για να του ανοίξει η φόρμα συμπλήρωσης της αξιολόγησης

Φυσικά αν ο ιδιοκτήτης είναι ο ίδιος χρήστης που έχει συνδεθεί στην εφαρμογή, το κουμπί αυτό δεν υπάρχει, καθώς δεν δύναται η δυνατότητα να αξιολογήσει κάποιος τον εαυτό του.



Σχήμα 5.8: Άποψη του πλαισίου με τις πληροφορίες του ιδιοκτήτη

Έτσι ο χρήστης αφού πατήσει στο button “Rate this user” τότε τον μεταφέρει σε μια φόρμα για την συμπλήρωση της αξιολόγησής του. Εκεί μπορεί να επιλέξει από **1 έως 5 αστέρια**, ανάλογα αν τον ικανοποίησε η όχι, και να προσθέσει κάποια **σχόλια**, αν φυσικά το επιθυμεί, όπως φαίνεται και στο Σχήμα 5.9.



Σχήμα 5.9: Φόρμα συμπλήρωσης της αξιολόγησης

Έτσι με αυτόν τον τρόπο οι χρήστες μπορούν να αξιολογούν άλλους χρήστες και οι ίδιοι να παίρνουν ένα feedback από τον τρόπο που εξυπηρετούν στην δημιουργία των υποτίτλων.

Όλες οι αξιολογήσεις φυσικά αποθηκεύονται στο Firestore Database, και από εκεί ανακτώνται όλες, για να γίνει ο υπολογισμός του μέσου όρου των αξιολογήσεων και να εμφανιστούν τα ανάλογα αστέρια στον κάθε χρήστη. Αν για παράδειγμα ο μέσος όρος των αξιολογήσεων ενός χρήστη είναι 3,2 αστέρια, τότε θα εμφανιστούν 3 αστέρια.

Ο υπολογισμός γίνεται κατά την ανάκτηση των πληροφοριών και στο προφίλ του χρήστη, αλλά και στην προβολή των πληροφοριών του ιδιοκτήτη του υπότιτλου. Στο παρακάτω Σχήμα 5.10, στην μέθοδο **loadRatings,** γίνεται απεικόνιση του πως υπολογίζεται ο μέσος όρος. των αξιολογήσεων ώστε να εμφανιστούν τα ανάλογα αστέρια, και φυσικά το μήνυμα με τον μ.ο. που θα εμφανιστεί στον χρήστη αν κάνει hover πάνω από τα αστέρια. Αν δεν έχει καμία αξιολόγηση, του αναφέρει πως δεν υπάρχουν αξιολογήσεις ακόμα.

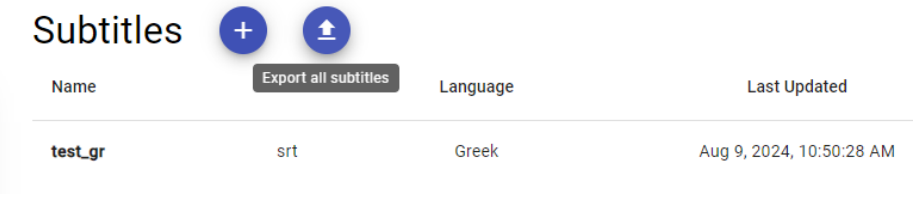
```
loadRatings(data: Rating[]){  Show usages   schatzis
  let total = 0;
  let sum = 0;
  let index = 5
  this.stars = []

  for(let i = 0; i < data.length; i++){
    total++
    sum += data[i].rating
  }

  if(total > 0){
    this.averageRate = sum/total;
    if(this.averageRate > 5) {
      this.averageRate = 5;
    }

    let flagAvg = this.averageRate;
    this.ratingTooltip = `Your rating is: ${this.averageRate}`

    while(index > 0){
      if(flagAvg ⩾ 1){
        this.stars.push(1)
      }else{
        if(flagAvg ⩾ 0.5){
          this.stars.push(2)
        }else{
          this.stars.push(3)
        }
      }
      flagAvg--
      index--
    }
  }else{
    this.ratingTooltip = `You don't have been rated yet`
    for(let i = 0; i < 5; i++){
      this.stars.push(3)
    }
  }
}
```

Σχήμα 5.10: Μέθοδος υπολογισμού του μέσου όρου αστεριών αξιολόγησης.

5.4 Μαζικό download των υποτίτλων

Μία ακόμη χρήσιμη λειτουργία που προστέθηκε στην εφαρμογή, είναι αυτή της **μαζικής εξαγωγής** των υποτίτλων ενός βίντεο. Με αυτήν την λειτουργία ο χρήστης μπορεί να κατεβάσει όλους τους υπότιτλους ενός βίντεο, σε ένα συμπιεσμένο αρχείο τύπου **.zip**. Έτσι με αυτόν τον τρόπο ο χρήστης έχει την δυνατότητα να εξάγει όλους τους υποτίτλους που έχει δημιουργήσει για ένα βίντεο, χωρίς να χρειάζεται να μεταβαίνει σε κάθε υπότιτλο και να το κάνει download από εκεί.

Ο τρόπος για να το κάνει αυτό είναι μέσω της σελίδας του πίνακα των υποτίτλων του βίντεο. Εκεί πάνω από τον πίνακα δίπλα από του κουμπί προσθήκης νέου υποτίτλου, έχει προστεθεί ένα νέο button, με την επιλογή «**Export all subtitles**», όπως γίνεται αντιληπτό και στο παρακάτω Σχήμα 5.11.



Σχήμα 5.11: Το σημείο του button για το μαζικό download

Ο τρόπος που υλοποιήθηκε η συγκεκριμένη λειτουργία ήταν με την δημιουργία ενός νέου **service**, το οποίο για κάθε υπότιτλο που υπάρχει βρίσκει την τοποθεσία που είναι αποθηκευμένο στο **Firebase**

Storage, που με την σειρά του διαβάζει το περιεχόμενο του αρχείου, το μετατρέπει, και το αποθηκεύει σε μια μεταβλητή τύπου **Blob**, που είναι για αποθήκευση μεγάλων αρχείων.

Έπειτα το προσθέτει σε ένα array από blob μεταβλητές, και στην συνέχεια δημιουργεί ένα αρχείο με την ονομασία του υπότιτλου και σε τετράγωνες αγκύλες, την γλώσσα του. Μετά όλα αυτά τα αρχεία, με την βοήθεια της βιβλιοθήκης του **JSZip,** τα προσθέτει σε ένα συμπιεσμένο αρχείο τύπου .zip.

Τέλος, με την χρήση της βιβλιοθήκης **saveAs,** του **file-saver**, δίνει εντολή στον browser να κατεβάσει το αρχείο με όνομα τον τίτλο του βίντεο.

```
massExportSubtitles(data: any[], videoId: any, uid: any){  Show usages   schatzis
  const zip = new JSZip();
  let fileNames: string[] = [];
  let fileBlobs: Blob[] = [];

  const downloadPromises = data.map((item, index) => {
    return new Promise<void>((resolve, reject) => {
      const fileName = item.fullFileName;
      const storageRef = this.storage.ref(`subtitles/${uid}/${videoId}/${item.iso}/${fileName}`);
      fileNames.push(fileName.split('.')[0] + '[' + item.iso + ']' + '.' + fileName.split('.')[1]);

      storageRef.getDownloadURL().subscribe({
        next: url => {
          this.http.get(url, { responseType: 'text' }).subscribe({
            next: sub => {
              const fileBlob = new Blob([sub], { type: 'text/' + fileName.split('.').pop() + ';charset=utf8' });
              fileBlobs.push(fileBlob);
              resolve();
```

Σχήμα 5.12: Μέρος της μεθόδου massExportSubtitles

Στο παραπάνω Σχήμα 5.12 υπάρχει ένα κομμάτι της μεθόδου massExportSubtitles που υλοποιήθηκε για να εκτελέσει την δυνατότητα της μαζικής εξαγωγής. Το συγκεκριμένο βρίσκεται στο service **download-file-handler.service.ts**, που μπορεί να γίνει inject σε οποιοδήποτε component για να χρησιμοποιηθεί.

5.5 Δυνατότητες στην επεξεργασία των υποτίτλων

Ένα, επίσης μεγάλο, μέρος της υλοποίησης αυτής της Π.Ε., επικεντρώθηκε στην δημιουργία νέων δυνατότητων κατά την επεξεργασία ενός αρχείου υποτίτλων σε ένα βίντεο.

5.5.1 Αναγνώριση Γλώσσας

Μια από τις δυνατότητες της εφαρμογής είναι αυτή της αναγνώρισης της **γλώσσας του βίντεο** αλλά και της **γλώσσας των υποτίτλων**.

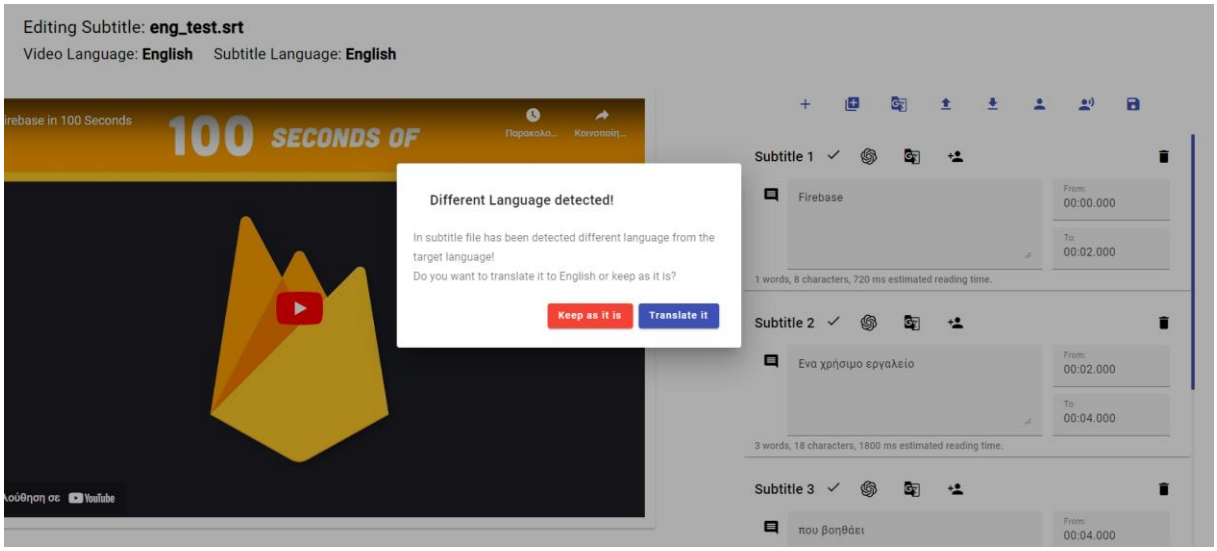
Πιο συγκεκριμένα, για την γλώσσα του βίντεο, επιστρατεύτηκε το **API του YouTube** από το Google Cloud. Το συγκεκριμένο API επιστρέφει πολλά δεδομένα που αφορούν το βίντεο του YouTube. Δίνοντας το id του βίντεο επιστρέφει data, όπως την ονομασία, τους διαθέσιμους υποτίτλους και το πιο σημαντικό για αυτήν την λειτουργία, την **γλώσσα** του βίντεο. Σε κάθε βίντεο το YouTube έχει εντοπίσει αυτόματα την γλώσσα του βίντεο. Έτσι με την βοήθεια του συγκεκριμένου API έχει προστεθεί η δυνατότητα στον χρήστη να βλέπει σε τι γλώσσα είναι το βίντεο, η οποία γλώσσα αναγράφεται πάνω από το βίντεο στην επεξεργασία των υποτίτλων.

Έπειτα πλέον γίνεται και αυτόματη αναγνώριση της γλώσσας των υποτίτλων. Αυτό γίνεται με την χρήστη του **API της Google Translate**, το οποίο «στέλνει» στο API όλα τα κείμενα από τους υπότιτλους και αναγνωρίζει την γλώσσα των κειμένων.

Φυσικά υπάρχουν και πολλές φορές που οι υπότιτλοι περιέχουν και άλλη γλώσσα εκτός από την συγκεκριμένη, επομένως το σύστημα εντοπίζει ότι είναι διαφορετική γλώσσα από την γλώσσα του

αρχείου του υποτίτλου που έχει δηλωθεί, όταν το ποσοστό των «ξένων» υποτίτλων είναι πάνω από **50%.**

Όταν γίνει αυτό ενημερώνει τον χρήστη ότι τα κείμενα του αρχείου του υποτίτλου που υπάρχουν, είναι **διαφορετικοί** από την γλώσσα που δηλώθηκε. Τα μηνύματα προειδοποίησης τα εμφανίζει όταν ο χρήστης πατάει για να επεξεργαστεί έναν υπότιτλο, και όταν ανεβάζει ένα αρχείο υποτίτλου.



Σχήμα 5.13: Άποψη του μηνύματος προειδοποίησης

Όπως φαίνεται και στο Σχήμα 5.13, το σήμα προειδοποίησης εμφανίστηκε καθώς εντόπισε διαφορετική γλώσσα. Έτσι δίνει στον χρήστη δύο επιλογές: **α)Keep as it is,** όπου ο χρήστης όταν το πατήσει οι υπότιτλοι θα παραμείνουν ως έχουν χωρίς καμία αλλαγή, και **β)Translate it**, όπου ο χρήστης όταν το πατήσει, όλα τα κείμενα του υποτίτλου θα μεταφραστούν αυτόματα στο target language του αρχείου υποτίτλου.

Στο σχήμα αυτό επίσης, μπορούν να παρατηρηθούν οι πληροφορίες του βίντεο και των υποτίτλων. Πάνω αριστερά έχει προστεθεί η γλώσσα του βίντεο, η γλώσσα των υποτίτλων και η ονομασία του υπότιτλου ώστε ο χρήστης να μπορεί να βλέπει αυτές τις χρήσιμες πληροφορίες.

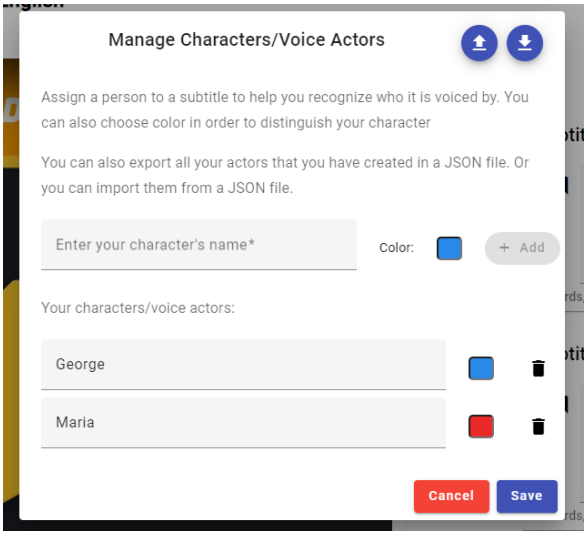
Το μήνυμα προειδοποίησης, στην ουσία είναι ένα **dialog-box component**, το οποίο εμφανίζεται στο component που θέλουμε να εμφανιστεί με την μέθοδο **dialog.open,** η οποία πραγματεύεται αυτόν ακριβώς τον σκοπό: να εμφανίζει components μέσα σε components, με την μορφή dialog. Έπειτα κάνουμε **subscribe** στην μέθοδο αυτή, ώστε να δούμε αν από το κλείσιμο του dialog, επιστρέφονται δεδομένα, με σκοπό να τα διαχειριστούμε. Επίσης με το **MAT_DIALOG_DATA,** το οποίο είναι μια βιβλιοθήκη της **Angular Material,** μπορούμε να κάνουμε Inject τα δεδομένα που θέλουμε να περάσουμε στο dialog component κατά το open.

```
openDialogForBatchDialogCreation(): void {  Show usages   schatzis +1
  this.dialog.open(BatchDialogModalComponent, {'width': '500px'})
    .afterClosed()  Observable<any>
    .pipe(take(1))
    .subscribe({
      next: interval => {
        if (interval > 0) {
          this.batchCreateDialog(interval);
        }
      },
      error: err => {
        console.error(err)
      }
    });
}
```

Σχήμα 5.14: Κώδικας για την κλήση dialog component

5.5.2 Ηθοποιοί / Χαρακτήρες υποτίτλων

Στην επεξεργασία των υποτίτλων έχει επίσης υλοποιηθεί η δυνατότητα να ορίζει ο χρήστης ηθοποιούς / χαρακτήρες, οι οποίοι μπορούν να αντιπροσωπεύουν κείμενα από τον υπότιτλο. Πιέζοντας το κουμπί **Manage Characters**, ο χρήστης μπορεί να προβάλει, επεξεργαστεί, προσθέσει και να διαγράψει τους χαρακτήρες που θέλει να υπάρχουν στους υποτίτλους.



Σχήμα 5.15: Φόρμα χαρακτήρων / ηθοποιών

Η φόρμα της διαχείρισης των ηθοποιών είναι επίσης ένα dialog-box component, στο οποίο ο χρήστης μπορεί να προσθέσει το όνομα του χαρακτήρα και ένα χρώμα, ώστε να διακρίνεται ο χαρακτήρας πιο εύκολα. Όπως μπορούμε να δούμε και στο Σχήμα 5.15, στο πάνω μέρος υπάρχει η φόρμα προσθήκης του χαρακτήρα και πιο κάτω υπάρχει ένας πίνακας όλους τους χαρακτήρες που έχουν προστεθεί. Κάθε νέος χαρακτήρας αποθηκεύεται φυσικά στην βάση της **Firestore**, με το **όνομα** και το **δεκαεξαδικό κωδικό** του **χρώματός** του.

Στον πίνακα ο χρήστης μπορεί να αφαιρέσει τον χαρακτήρα ή και να τροποποιήσει το όνομα του και το χρώμα του. Πατώντας το button **Save** όλες οι αλλαγές που πραγματοποίησε ο χρήστης θα αποθηκευτούν στην βάση.

Τέλος υπάρχει η δυνατότητα του χρήστη να κατεβάσει σε ένα αρχείο τύπου **Json,** τους χαρακτήρες που αποθήκευσε ώστε να τους κρατήσει σαν back-up ή να τους χρησιμοποιήσει και άλλους υποτίτλους.

Επίσης υπάρχει και η δυνατότητα της εισαγωγής των χαρακτήρων που φυσικά δέχεται αρχεία τύπου Json, και περιέχουν τα σωστά τα στοιχεία των χαρακτήρων.

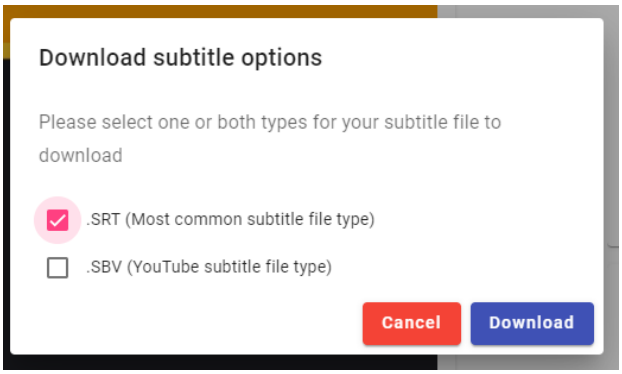
Το upload και το download των χαρακτήρων φυσικά το διαχειρίζονται τα services των **downloadHandler,** και **upload-file-handler,** που έχουν γίνει inject στο subtitling-component της επεξεργασίας των υποτίτλων.

5.5.3 Download υποτίτλου

Στον χρήστη δίνεται η δυνατότητα του να κατεβάσει τον υπότιτλο που διαχειρίζεται, πατώντας το κουμπί **Download Subtitles.** Με το κλικ πάνω στο κουμπί, εμφανίζεται στον χρήστη ένα ακόμα dialog component, με το οποίο ο χρήστη έχει την δυνατότητα να επιλέξει την μορφή του αρχείου του υποτίτλου που θέλει να κατεβάσει.

Στο dialog υπάρχουν δύο επιλογές μορφών: η μία είναι το **.srt,** όπου είναι ένας τύπος αρχείων για υποτίτλους που χρησιμοποιείται παντού, μιας και είναι ο πιο δημοφιλές τύπος. Σε αυτό μπορεί ο χρήστης ακόμα και να μορφοποιήσει τους υπότιτλους. Η δεύτερη επιλογή είναι ο τύπος **.sbv,** όπου είναι ο τύπος αρχείων υποτίτλων που χρησιμοποιείται για το **YouTube**.

Από προεπιλογή, επιλέγετε ο τύπος που ο χρήστης δήλωσε όταν δημιούργησε τον υπότιτλο. Φυσικά μπορεί να επιλέξει και τους δύο τύπους αρχείων και να κατεβάσει τον υπότιτλο με 2 διαφορετικές μορφές αρχείων. Αυτό βέβαια θα προσθέσει και τα 2 αρχεία σε ένα συμπιεσμένο αρχείο τύπου .zip πριν το κατεβάσει.



Σχήμα 5.16: Το dialog box για το download υποτίτλου.

Και σε αυτήν την περίπτωση χρησιμοποιήθηκε το service του download-file-handler, που αναφέραμε και πιο πάνω και χρησιμοποιείται για να μετατρέψει τα κείμενα υποτίτλου σε blob variable, και στην συνέχεια να τα κατεβάσει, είτε σαν ξεχωριστά αρχεία, είτε σε μορφή .zip.

5.5.4 Αυτόματη μετάφραση

Ένα ακόμα χαρακτηριστικό που έχει η επεξεργασία των υποτίτλων είναι να μεταφράζει τους υπότιτλους στην γλώσσα που επιθυμεί ο χρήστης. Βέβαια, κατά την δημιουργία ενός νέου υποτίτλου, ο χρήστης δηλώνει και την γλώσσα στην οποία θα είναι αυτός. Έτσι προστέθηκε και η δυνατότητα ο χρήστης να μεταφράζει αυτόματα τους υπότιτλούς του στην γλώσσα που έχει επιλέξει.

Επιλέγοντας το **Translate All Subtitles** στις γενικές επιλογές των υποτίτλων**,** ή επιλέγοντας το **Translate this Subtitle** στις επιλογές του ενός κειμένου του υποτίτλου, θα εμφανιστεί στην οθόνη ένα dialog-box που δίνει στον χρήστη δύο δυνατότητες

Η μία είναι να μεταφράσει τους υπότιτλους στην γλώσσα που έχει επιλεγεί για το συγκεκριμένο αρχείο. Έτσι προσφέρει στον χρήστη την ευκολία να μεταφράσει τους υπότιτλούς του κατευθείαν στην γλώσσα που έχει διαλέξει, χωρίς να χρειάζεται να κάνει αναζήτηση με το dropdown όλων των γλωσσών

Η δεύτερη επιλογή, επιτρέπει στον χρήστη να μεταφράσει τον υπότιτλό του σε οποιαδήποτε γλώσσα επιθυμεί αυτός.

Οι γλώσσες και οι μεταφράσεις, προσφέρονται από το **API** του **Google Translate**, που επικοινωνεί με το service **googletranslate.service**, όπου υλοποιήθηκε η λογική της επικοινωνίας αυτής. Έτσι στέλνοντας το κείμενο και την γλώσσα στην οποία θέλει να μεταφράσει ο χρήστης, το API στέλνει ένα response, που περιέχει το κείμενο μεταφρασμένο.

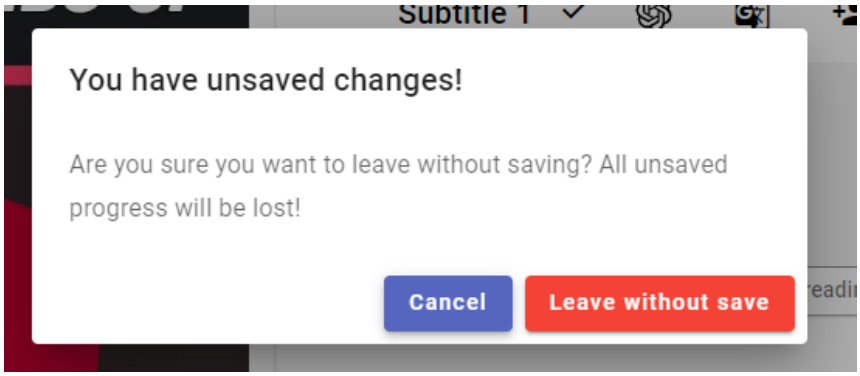
```
translate(translationObject: GoogleTranslateRequestObject) {  Show usages  StamChatzis
  const url = 'https://translation.googleapis.com/language/translate/v2?key=';
  const key = GOOGLE_API_KEY;
  return this.http.post(url + key, translationObject);
}

getSupportedLanguages() {  Show usages  StamChatzis
  const url = `https://translation.googleapis.com/language/translate/v2/languages?target=en&key=${GOOGLE_API_KEY}`;
  return this.http.get(url);
}

detectLanguage(text: string){  Show usages  schatzis
  const url = `https://translation.googleapis.com/language/translate/v2/detect?key=${GOOGLE_API_KEY}`;
  const body = {
    q: text
  };
  return this.http.post<GoogleDetectionResponseData>(url, body);
}
```

Σχήμα 5.17: Τρόπος επικοινωνίας με το API του Google Translate

5.5.5 Έλεγχος τροποποιήσεων

Σε κάθε αλλαγή που πραγματοποιείται στην επεξεργασία αλλαγών, το σύστημα γνωρίζει ότι το αρχείο έχει τροποποιηθεί. Έτσι κάθε φορά που ο χρήστης, ενώ έχει κάνει αλλαγές, προσπαθήσει να αποχωρήσει από την επεξεργασία του υποτίτλου, τότε εμφανίζεται ένα μήνυμα προειδοποίησης με την μορφή dialog-box component, που τον ενημερώνει ότι υπάρχουν αλλαγές που δεν έχουν αποθηκευτεί.

Του δίνει επομένως τις 2 επιλογές: να αποχωρήσει χωρίς να γίνει αποθήκευση και να χάσει όλη την πρόοδό του, ή πρώτα να αποθηκεύσει τις αλλαγές του πριν αποχωρήσει.



Σχήμα 5.18: Η προειδοποίηση για μη αποθηκευμένες αλλαγές

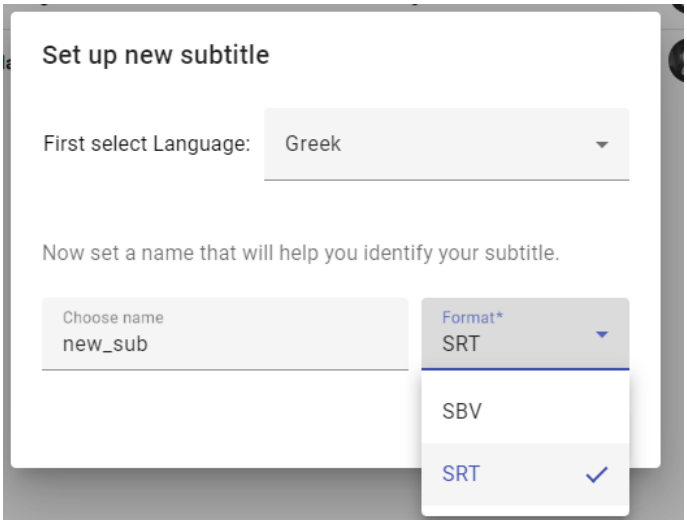
5.6 Τροποποιήσεις στη λίστα υποτίτλων

Για τις ανάγκες τις ολοκλήρωσης της πτυχιακής, έγιναν και οι απαραίτητες αλλαγές στην σελίδα με την λίστα των υποτίτλων ενός βίντεο του χρήστη, αλλά και στα Shared βίντεο.

5.6.1 Προσθήκη νέου υποτίτλου

Πιο συγκεκριμένα, έχει τροποποιηθεί η λειτουργία της δημιουργίας νέου υποτίτλου ώστε να διευκολύνει τον χρήστη, καθώς πλέον, αφού ο χρήστης επιλέξει την γλώσσα του υπότιτλου, τότε θα του εμφανιστεί η **φόρμα** που θα τον παραπέμπει να συμπληρώσει το όνομα και την μορφή του αρχείου υποτίτλου, που είναι μία από τις 2 επιλογές **.srt** και **.sbv**. Αν δεν συμπληρωθεί έστω και ένα από τα πεδία αυτά τότε το button για την δημιουργία του υπότιτλου παραμένει disabled. Αυτό γίνεται με την χρήστη του **FormGroup** και των **Validators** που έχει, καθώς αν παρατηρήσει ότι υπάρχει παραβίαση της φόρμας, τότε το button παραμένει disabled.

Επίσης κατά τον έλεγχο της δημιουργίας του νέου υποτίτλου, γίνεται και έλεγχος για το αν υπάρχει υπότιτλος με την **ίδια ονομασία** και **γλώσσα** στον πίνακα των υποτίτλων του εκάστοτε βίντεο. Αν εντοπιστεί ότι υπάρχει ίδια ονομασία, τότε εμφανίζει ανάλογο μήνυμα στο χρήστη που του αναφέρει ότι δεν μπορεί να υπάρξουν 2 πανομοιότυποι υπότιτλοι με το ίδιο όνομα και γλώσσα.

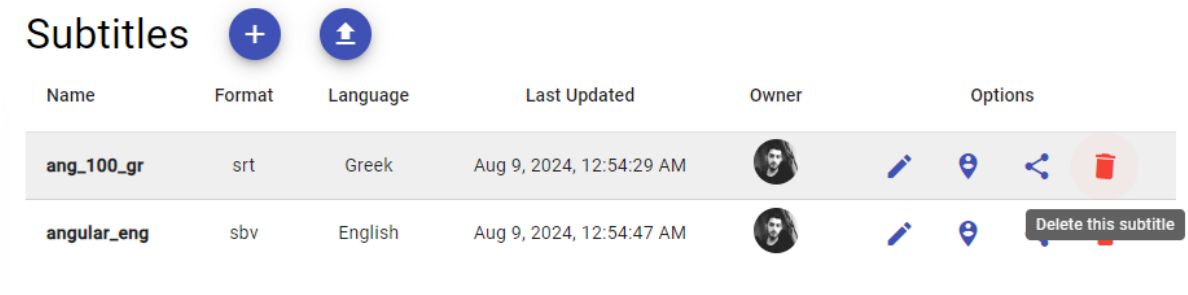


Σχήμα 5.19: Προεπισκόπηση της φόρμας της προσθήκης νέου υπότιτλου.

5.6.2 Μορφοποίηση του πίνακα και επιλογή διαγραφής

Επιπρόσθετα, η εμφάνιση του πίνακα έχει τροποποιηθεί ώστε να είναι πιο ευχάριστη στον χρήστη. Το όνομα του υποτίτλου πλέον έχει bold γράμματα για να είναι πιο ευδιάκριτο, ενώ, όπως αναφέραμε και πιο πάνω, έχει προστεθεί και μία στήλη ακόμα που εμφανίζει την εικόνα προφίλ του ιδιοκτήτη του εκάστοτε υποτίτλου. Όλα αυτά έγιναν χάρη στην **Angular Material** και την **CSS** που θα την αναλύσουμε σε επόμενη υποενότητα.

Επίσης στον πίνακα των υποτίτλων έχουν προστεθεί και νέες επιλογές για την διαχείριση των υποτίτλων. Πιο συγκεκριμένα προστέθηκε η δυνατότητα ενός χρήστη που έχει δικαιώματα ιδιοκτήτη, να διαγράφει τον υπότιτλό του.



Σχήμα 5.20: Εμφάνιση του πίνακα υποτίτλων

Στο παραπάνω Σχήμα 5.20 φαίνεται και η σχεδίαση του πίνακα, με την στήλη του ιδιοκτήτη και την επιλογή της διαγραφής στην στήλη **Options.** Όταν ο χρήστης πατήσει το συγκεκριμένο εικονίδιο, τότε του εμφανίζεται ένα dialog-box προειδοποίησης για το αν πραγματικά θέλει να διαγράψει τον συγκεκριμένο υπότιτλο. Αφού ο χρήστης επιβεβαιώσει ότι θέλει να διαγράψει τον υπότιτλο, τότε αυτός διαγράφεται οριστικά από τον πίνακα.

Φυσικά ο υπότιτλος αυτός διαγράφεται και από την βάση της Firestore, καθώς στο service **DetailsViewService,** που γίνεται inject στο συγκεκριμένο component, υλοποιήθηκε η μέθοδος **deleteSubtitle,** όπου διαγράφει τον υπότιτλο από την βάση στην Firestore, αλλά και το αρχείο του υποτίτλου στο Storage.

5.7 Βελτιώσεις στο UI

Ένας σκοπός της πτυχιακής, ήταν και η περαιτέρω βελτίωση της εμφάνισης, ή αλλιώς του UI, της εφαρμογής ώστε να είναι πιο φιλική προς τον χρήστη.

Οι αλλαγές που πραγματοποιήθηκαν, έγιναν με την βοήθεια της **Angular Material,** που είναι βιβλιοθήκη της Angular, που έχει προσχεδιασμένα components, όπως buttons, tables, και άλλα πολλά που αφορούν το UI μιας εφαρμογής στην Angular. Επίσης έγινε χρήση και της **CSS,** που υπάρχει σε κάθε component, για την αλλαγή λεπτομερειών σε κάθε κομμάτι της εφαρμογής.

Οι αλλαγές αφορούσαν κατά κύριο λόγο τα buttons της Angular, που τροποποιήθηκαν έτσι ώστε να είναι πιο ευδιάκριτα και φιλικά στον χρήστη. Πιο συγκεκριμένα έγινε χρήση των **mat-buttons** της Material, που έχουν έτοιμα προσχεδιασμένα buttons για κάθε σκοπό

Για παράδειγμα αν δηλωθεί ένα mat-button με color **warn,** τότε το button τροποποιείται έτσι ώστε να έχει την μορφή της προειδοποίησης.

```
<button mat-button color="warn" [mat-dialog-close]="null">Cancel</button>
<button mat-raised-button color="primary" [disabled]="newSubForm.invalid"
```

Σχήμα 5.21: Κομμάτι κώδικα HTML με τη χρήση mat-buttons

Στο παραπάνω Σχήμα 5.21 μπορεί να γίνει εμφανές η χρήση των mat-buttons και των χρωματισμών του, αλλά και του validator του FormGroup, το **newSubForm,** που απενεργοποιεί το button αν εντοπίσει ότι το form είναι **invalid.**

5.8 Αποθήκευση shared υποτίτλων

Μια από τις σημαντικές λειτουργίες που προστέθηκαν στην εφαρμογή, είναι η αποθήκευση των υποτίτλων στα **shared** βίντεο. Περνώντας σε λεπτομέρειες, οι χρήστες που έχουν εξουσιοδότηση στο να τροποποιούν έναν υπότιτλο ενός άλλου user, πλέον μπορούν και αποθηκεύονται. Η αποθήκευση γίνεται στο **Storage** στην Firebase, και έτσι ο ιδιοκτήτης μπορεί να δει και τις αλλαγές που έχει πραγματοποιήσει ένας άλλος χρήστης στον υπότιτλό του.

5.9 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκαν και αναπτύχθηκαν όλες οι νέες λειτουργίες της εφαρμογής της Sub & Dub, που αφορά τον υποτιτλισμό και μεταγλώττιση βίντεο από την πλατφόρμα του YouTube. Για αρχή αναλύθηκε η χρήση του προφίλ της εφαρμογής ενώ στην συνέχεια επεξηγήθηκε και το σύστημα αξιολόγησης των χρηστών. Επίσης μελετήθηκαν οι δυνατότητες που έχουν προστεθεί στην επεξεργασία των υποτίτλων κάθε βίντεο και οι αλλαγές που έγιναν στην διαχείρισή τους, όπως στην επιλογή της διαγραφής τους, αλλά και στον έλεγχο τροποποιήσεών τους. Ενώ τέλος έγινε και μια μικρή ανάλυση για την βελτίωση της σχεδίασης της εφαρμογής συνολικά. Για όλα τα παραπάνω, φυσικά υπήρξαν και εικόνες από κομμάτια κώδικα της εφαρμογής στην Angular, για την καλύτερη επεξήγηση και κατανόηση της λειτουργίας τους.

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

6.1 Συμπεράσματα από την εκπόνηση της Π.Ε.

Με την εκπόνηση αυτής την πτυχιακής εργασίας, έχω καταλήξει στα παρακάτω συμπεράσματα που αφορούν την χρήση των τεχνολογιών που χρησιμοποίησα, αλλά και την υλοποίηση της συγκεκριμένης εφαρμογής

Για αρχή θα ήθελα να αναφερθώ στις τεχνολογίες που χρησιμοποίησα για την ανάπτυξη της εφαρμογής. Χρησιμοποιώντας την Angular, ως ένα σύγχρονο και εύχρηστο εργαλείο, έκαναν την υλοποίηση της εφαρμογής μια ευχάριστη εμπειρία. Μου δίδαξε πολλά πράγματα που αφορούν το συγκεκριμένο εργαλείο και ταυτόχρονα με έκαναν να εξελιχθώ στο να χρησιμοποιώ αυτό το framework, ώστε στη πορεία της καριέρας μου να μπορώ να το χρησιμοποιήσω με μεγαλύτερη οικειότητα. Πράγματα όπως η δομή των components, η χρήση της Reactive JavaScript, αλλά και το design με την χρήση των Angular Material και CSS, με έκανα πιο ώριμο στον τρόπο της σκέψης για την σχεδίαση μελλοντικών project.

Ένα ακόμα χρήσιμο εργαλείο που το χρησιμοποίησα για πρώτη φορά ήταν αυτό της Firebase. Χρησιμοποιώντας αυτό το σημαντικό πλέον εργαλείο για την ανάπτυξη του back-end της εφαρμογής, με έκανε να το γνωρίσω καλύτερα και να «εγκλιματιστώ» σε αυτό το περιβάλλον της Google. Χάρη στην χρήση του, δεν θα ήταν ψέματα να πω πως πλέον και εγώ με την σειρά μου θα το χρησιμοποιώ πλέον και σε μελλοντικά web apps ή ακόμα και εφαρμογές σε Android και iOS.

Φυσικά δεν μπορώ να παραλείψω και την σπουδαία χρήση του Git, που πραγματικά λύνει τα χέρια του προγραμματιστή για να μπορεί να διαχειρίζεται, και να αποθηκεύει σε cloud, τον πηγαίο κώδικα που έχει υλοποιήσει. Και φυσικά με την χρήση του Git θα μπορώ να δουλέψω ομαδικά με μεγαλύτερη άνεση και σε project στις εργασίες μου που υπάρχουν πολλά διαφορετικά άτομα.

Όσο αφορά την εφαρμογή, η υλοποίησή της με έκανε να καταλάβω το πόσο σημαντικό είναι στις μέρες μας οι υπότιτλοι για τα βίντεο που υπάρχουν στο YouTube και όχι μόνο. Φυσικά μου δίδαξε τους τρόπους με τους οποίους δημιουργείται ένα αρχείο υποτίτλων και την σπουδαία λειτουργία της μετάφρασης που παρέχει το Google Translate. Βέβαια η υλοποίηση της εφαρμογής είναι και η αιτία που έμαθα να μπορώ να κατασκευάζω εφαρμογές σε Angular και Firebase και το πως να συνδυάζω αυτά τα δύο εργαλεία και το πως επικοινωνούν για να δημιουργήσουν ένα ολοκληρωμένο περιβάλλον για μια εξίσου ολοκληρωμένη εφαρμογή.

Κλείνοντας, η ολοκλήρωση της πτυχιακής αυτής είχε μόνο θετικά συμπεράσματα για μένα προσωπικά, και νιώθω τυχερός που βρέθηκα στην θέση να την υλοποιήσω.

6.2 Προτάσεις βελτίωσης της εφαρμογής

Φυσικά η υλοποίηση της εφαρμογής, έφερε και σκέψεις βελτίωσης που θα μπορούσαν να γίνουν στην εφαρμογή αυτή.

Για αρχή μια χρήσιμη βελτίωση θα ήταν αυτή της μετονομασίας των αρχείων των υποτίτλων. Με αυτή τη λειτουργία, θεωρώ, πως είναι ένα από τα βασικά κομμάτια, καθώς θα πρέπει να δύναται η δυνατότητα στον χρήστη να μπορεί να αλλάξει το όνομα των υποτίτλων του. Αυτό θα έδινε περισσότερη ευελιξία στην διαχείριση των αρχείων.

Μια ακόμα βελτίωση θα μπορούσε να γίνει και στα κομμάτια του κάθε υποτίτλου, όπως το να μπορούν να αλλάξουν θέση μεταξύ τους χωρίς όμως να χάνουν τα δευτερόλεπτα που έχουν δηλωθεί. Αυτό επίσης

θα έδινε μια μεγαλύτερη ευελιξία στον χρήστη ώστε να μην χρειάζεται να διαγράφει το dialog του υποτίτλου και να το προσθέτει σε ένα άλλο καινούργιο που θα εμφανιστεί στο τέλος της ουράς των κομματιών των υποτίτλων.

Επίσης, παραμένοντας στους υπότιτλους, θα μπορούσε να προστεθεί η δυνατότητα να μορφοποιηθούν. Με λίγα λόγια να μπορεί ένας χρήστης να αλλάζει την γραμματοσειρά του κειμένου, ή και να αλλάζει την γραφή σε bold και italic. Η ακόμα και να μπορούσε να χρωματίσει τα κείμενα, ανάλογα με το χρώμα του χαρακτήρα ώστε να είναι και πιο ευδιάκριτα στους χρήστες που χρησιμοποιούν τον υπότιτλο.

Ολοκληρώνοντας, πιστεύω πως όντως υπάρχουν περιθώρια βελτίωσης της εφαρμογής που θα μπορούσαν να «απογειώσουν» την χρήση της εφαρμογής όσο αφορά τους υπότιτλους αλλά και την εμφάνιση του UI.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Internet Sites

[1] Angular (web framework), *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Angular_(web_framework)

[2] What is Angular, *Angular Documentation* [Online]. Available: https://angular.dev/overview

[3] AngularJS, *Wikipedia.* [Online]. Available: https://en.wikipedia.org/wiki/AngularJS

[4] How To Install Angular on Windows, macOS, and Linux, *kinsta.* [Online]. Available: https://kinsta.com/knowledgebase/install-angular

[5] YouTube, *Wikipedia.* [Online]. Available: https://en.wikipedia.org/wiki/YouTube

[6] Top YouTube Channels in Greece, *HypeAuditor*. [Online]. Available: https://hypeauditor.com/top-youtube-all-greece/

[7] Understanding Angular, *Angular Documentation.* [Online]. Available: https://angular.io/guide/understanding-angular-overview

[8] Introduction to RxJs, *RxJS.* [Online]. Available: https://rxjs.dev/guide/overview

[9] Firebase Documentation, *Firebase.* [Online]. Available: https://firebase.google.com/docs

[10] Firebase History, Firebase, *Wikipedia.* [Online]. Available: https://en.wikipedia.org/wiki/Firebase#History

[11] SQL vs NoSQL: 5 Critical Differences, *Integrate.io*. [Online]. Available: https://www.integrate.io/blog/the-sql-vs-nosql-difference/

[12] Google Cloud Documentation, *Google Cloud.* [Online]. Available: https://cloud.google.com/docs

[13] Google Cloud, *TechTarget Cloud Computing.* [Online]. Available: https://www.techtarget.com/searchcloudcomputing/definition/Google-Cloud-Platform

[14] Git Documentation, *Git.* [Online]. Available: https://git-scm.com/docs/git

[15] Git, *Wikipedia.* [Online]. Available: https://en.wikipedia.org/wiki/Git

[16] Top 20 Git Commands With Examples, *DZone.* [Online]. Available: https://dzone.com/articles/top-20-git-commands-with-examples

ΠΑΡΑΡΤΗΜΑ Α : ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

Ο πηγαίος κώδικας της εφαρμογής υπάρχει σε ένα public repository που είναι διαθέσιμο για το καθένα να τον προβάλει ή και να τον κατεβάσει. Βρίσκεται στην διεύθυνση: **https://github.com/StamChatzis/SubNDub**.