ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Σχεδίαση & Ανάπτυξη Εφαρμογής iOS για ανάκτηση Θεατρικών δεδομένων

Του φοιτητή
Σκαρλή Ιωάννη
Αρ. Μητρώου: 154616

Επιβλέπων
Σαλαμπάσης Μιχαήλ
Βαθμίδα Καθηγητής

Φεβρουάριος 2023

Τίτλος Π.Ε. : Σχεδίαση & Ανάπτυξη εφαρμογής iOS για ανάκτηση θεατρικών δεδομένων
Κωδικός Π.Ε. : 21194
Ονοματεπώνυμο φοιτητή : Σκαρλής Ιωάννης
Ονοματεπώνυμο εισηγητή : Σαλαμπάσης Μιχαήλ
Ημερομηνία ανάληψης Π.Ε. : 18/03/2021
Ημερομηνία περάτωσης Π.Ε. : 19/01/2023

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.

Περιεχόμενα

Πρόλογος 5

Περίληψη 6

Abstract 7

1. Εισαγωγή 8

1.1 Περιγραφή εργασίας 8

1.2 Δομή εργασίας 9

2 Τι είναι το iOS 9

2.1 Εισαγωγή 9

2.2 Ιστορία 10

2.3 Αρχιτεκτονική 10

3 Εισαγωγή στο Flutter 12

3.1 Εισαγωγή 12

3.2 Η γλώσσα Dart 12

3.2.1 Τρόπος λειτουργίας της γλώσσας Dart 13

3.3 OOP Dart γνωρίσματα 16

3.4 Λόγοι χρήσης της Dart εκ μέρους του Flutter 17

3.5 Το Flutter 18

3.5.1 Παρόμοια frameworks 19

3.5.2 Η διαφορά συγκριτικά με τα υπόλοιπα frameworks 20

3.5.3 Εισαγωγή στα widgets 22

3.5.4 Η ύπαρξη του Widget στις εφαρμογές 23

3.5.5 Χρήστες του flutter 28

3.5.6 Διαδικασία εγκατάστασης του Flutter 28

3.6 Επίλογος 29

4 Κεφάλαιο: Υλοποίηση 29

4.1 Εισαγωγή 29

4.2 Δημιουργία Flutter project 29

4.2.1 Αρχικοποίηση κώδικα του Project 29

4.2.2 Δημιουργία iOS Emulator 31

4.3 Επεξήγηση των βιβλιοθηκών στο pubspec.yaml 31

4.3.1 Εισαγωγή στο pub.dev 31
4.3.2 Βιβλιοθήκες που χρησιμοποιήθηκαν 32
4.4 Παρουσίαση και ανάλυση σημαντικών σημείων του κώδικα 35
4.5 Επεξήγηση της βάσης δεδομένων 39
4.6 Επίλογος υλοποίησης της εφαρμογής 40
5 Κεφάλαιο: Παρουσίαση της εφαρμογής 41
5.1 Εισαγωγή στην παρουσίαση της εφαρμογής 41
5.2 Οθόνη Home 41
5.3 Οθόνη Actors 42
5.4 Οθόνη Movies 43
5.5 Οθόνη Theaters 44
6 Κεφάλαιο: Στάδια υλοποίησης της εφαρμογής 45
6.1 Απαιτήσεις 45
6.2 Scraping, δημιουργία βάσης δεδομένων και API 46
6.3 Σχεδιασμός οθονών 46
6.4 Συγγραφή κώδικα 48
7 Κεφάλαιο: Συμπέρασμα και βελτιώσεις 49
7.1 Προτάσεις βελτίωσης της εφαρμογής 49
7.2 Συμπεράσματα 49
8 ΒΙΒΛΙΟΓΡΑΦΙΑ 51
ΙΣΤΟΣΕΛΙΔΕΣ 52

Πρόλογος

Η ανάπτυξη εφαρμογής για λειτουργικό iOS που αφορά την ανάκτηση και παρουσίαση δεδομένων για θεατρικές παραστάσεις αποτελεί ένα μεγάλο συνεργατικό project αλλά και μια ατομική εργασία κάθε μέλους της ομάδας που υλοποίησε ένα διακριτό μέρος του συνολικού συστήματος. Η σύνθεση των εργασιών αποτέλεσε μία σημαντική συγκυρία εκμάθησης για συνεργατική και αποτελεσματική πρόοδο συστημάτων λογισμικού σε αρκετά μεγάλη κλίμακα, η οποία πολύ συχνά χρησιμοποιείται σε πτυχιακές και διπλωματικές μελέτες. Πιο συγκεκριμένα, στην παρούσα εργασία, με βάση την ανάλυση των χαρακτηριστικών που υπήρχαν σε μία βάση δεδομένων, απαιτούμενο ήταν να ανακαλυφθούν και να παρουσιαστούν ιδέες ώστε να προβληθούν κατάλληλα και να υπάρχει απόρροια λειτουργικής εφαρμογής. Για την ανάπτυξη της συγκεκριμένης εφαρμογής της πτυχιακής εργασίας μου, ήταν αναγκαίο, η εκμάθηση και χρήση του iOS, του οποίου η χρησιμότητα είναι πλέον αρκετά δημοφιλής και στο ευρύ κοινό αλλά και σε εταιρείες. Με αυτόν τον τρόπο, η δυνατότητα χρήσης του και επεξεργασίας του φέρνει άμεσα και έμμεσα επαγγελματικές ευκαιρίες και προοπτικές. Ο λόγος που καθιστά την παρούσας εργασίας ως ξεχωριστή είναι η επιλογή του Flutter framework για την υλοποίηση της εφαρμογής, το οποίο μας δίνει το πλεονέκτημα να δημιουργήσουμε σύγχρονες διεπαφές χρήστη αλλά και το cross-platform χαρακτηριστικό του, που μας επιτρέπει με έναν πηγαίο κώδικα, αν το επιθυμούμε, να παράγουμε εφαρμογές και για iOS και για android αλλά και για web.

Περίληψη

Αντικείμενο της συγκεκριμένης διατριβής είναι να σχεδιαστεί και να αναπτυχθεί μία εφαρμογή iOS για να επιτευχθεί η ανάκτηση θεατρικών δεδομένων. Σκοπός είναι να παρουσιαστούν τα δεδομένα πολύ πιο κατανοητά, έτσι ώστε να πληροφορούνται καλύτερα οι χρήστες γύρω από αυτά. Βασική προϋπόθεση ήταν οι εφαρμογές της ομάδας μας να χαρακτηρίζονται από την ύπαρξη ομοιογένειας και γι' αυτόν τον λόγο χρησιμοποιήθηκαν συγκεκριμένα εργαλεία, έτσι ώστε να σχεδιαστούν τα πρότυπα των οθονών και των λειτουργιών της κάθε μίας εξ' αυτών. Για να γίνει αυτό πράξη έγινε η επιλογή του Flutter, ως βασικού εργαλείου για να αναπτυχθεί η εφαρμογή για το iOS, λόγω του ότι ικανοποιούσε όλες τις ανάγκες που υπήρχαν και προσέφερε ακόμα πιο πολλά πλεονεκτήματα, όπως ήταν η μεγάλη ευκολία δημιουργίας μιας σύγχρονης εφαρμογής, με όμορφα γραφικά, και η δυνατότητα ύπαρξης ενός cross-platform framework.

Λέξεις κλειδιά: *IOS, Flutter, Dart, θεατρικά δεδομένα*

Abstract

The object of this thesis is to design and develop an iOS application to achieve theater data recovery. The purpose is to present the data much more comprehensibly, so that users are better informed about it. A basic condition was that the applications of our team were characterized by the existence of homogeneity and for this reason specific tools were used, so as to design the standards of the screens and functions of each of them. To make this a reality, Flutter was chosen as the main tool to develop the application for iOS, due to the fact that it satisfied all the needs that existed and offered even more advantages, such as the great ease of creating a modern application, with beautiful graphics, and the possibility of having a cross-platform framework.

Keywords: *IOS, Flutter, Dart, theater data*

1. Εισαγωγή

1.1 Περιγραφή εργασίας

Η παρούσα εργασία αποτελεί ένα πολύ βασικό κομμάτι ενός συνόλου εργασιών που πραγματοποιήθηκε από μια ομάδα φοιτητών. Κάθε ένας από τους φοιτητές της ομάδας αυτής υλοποιώντας διαφορετικά θέματα πτυχιακών εργασιών προσφέραμε, τόσο στο αποτέλεσμα μιας κοινής ιδέας για την δημιουργία ενός πληροφοριακού συστήματος ανάκτηση δεδομένων θεατρικών παραστάσεων, όσο και στο αποτέλεσμα μιας πλήρους ατομικής πτυχιακής. Η αρχή όλων έγινε με μια μέθοδο scraping με την οποία αντλήθηκαν κάποια δεδομένα για θεατρικές παραστάσεις. Αποθηκεύοντας αυτά τα δεδομένα σε μια βάση υπήρξε η δυνατότητα αλλά και η ανάγκη αυτά τα δεδομένα να μπορούν να παρουσιαστούν σε ένα πιο κατανοητό γραφικό περιβάλλον σε καλύτερη μορφή. Οι πλατφόρμες που υπήρχε ανάγκη για παρουσίαση των δεδομένων είναι τρείς και πιο συγκεκριμένα είναι για web, android και iOS όπου είναι και το θέμα της συγκεκριμένης πτυχιακής. Για να μπορέσουμε να επιτύχουμε αυτή την ανάπτυξη των τριών εφαρμογών αλλά με τα ίδια δεδομένα μας χρειαστήκαμε ένα API για να σερβίρει τα δεδομένα σε μορφή JSON και στις τρείς αυτές εφαρμογές. Σκοπός των τριών εφαρμογών είναι η παρουσίαση των δεδομένων με όσο το δυνατόν πιο απλό και κατανοητό τρόπο ώστε να προσφέρεται στον τελικό χρήστη η σαφέστερη πληροφόρηση. Έτσι λοιπόν οι τρείς αυτές εφαρμογές θα έπρεπε να έχουν κοινά χαρακτηριστικά και ομοιογένεια, οπότε για να επιτευχθεί αυτό χρησιμοποιήσαμε κάποια εργαλεία ώστε σε πρώτη φάση να σχεδιάσουμε κάποια πρότυπα των οθονών και των λειτουργιών της κάθε εφαρμογής. Με αυτόν τον τρόπο πετύχαμε ευκολότερα τον σκοπό αυτό αλλά και την συνεργασία μας. Ακόμη υπήρχε συνεχείς επικοινωνία και με τους φοιτητές που υλοποιούσαν το scraping και το API και έτσι αντλούσαν και μας σερβίρανε με τα κατάλληλα end points τα δεδομένα με τον τρόπο που χρειαζόμασταν για να τα παρουσιάσουμε. Έτσι επιλέχθηκε το Flutter framework ως κύριο εργαλείο για την ανάπτυξη της εφαρμογής για iOS, όπου αποτελεί το κυρίως θέμα της παρούσας εργασίας, γιατί ικανοποιούσε όλες τις παραπάνω ανάγκες και μας προσέφερε και ακόμα περισσότερα πλεονεκτήματα όπως είναι η ευκολία δημιουργίας μιας σύγχρονης και πιο γραφικά όμορφης εφαρμογή αλλά και την δυνατότητα ενός cross-platform framework.

1.2 Δομή εργασίας

Η εργασία αποτελείται από 7 κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζουμε συνοπτικά τη δομή και το περιεχόμενο της εργασίας. Στο δεύτερο κεφάλαιο μια σύντομη γνωριμία με το τι ακριβώς είναι το λειτουργικό iOS και πιο συγκεκριμένα ασχολούμαστε την ιστορία του και την αρχιτεκτονική του. Στο τρίτο, αναλύουμε την τεχνολογία που χρησιμοποιείται για την υλοποίηση της εφαρμογής, δηλαδή το Flutter. Αναλυτικότερα, θα δούμε τι είναι το flutter και πως δουλεύει, γιατί έχει επιλέξει την Dart ως κύρια γλώσσα. Στο τέταρτο κεφάλαιο μελετάμε την υλοποίηση της εφαρμογής σε επίπεδο κώδικα αλλά και την αρχικοποίηση των περιβαλλόντων που χρειαζόμαστε, ενώ στο πέμπτο παρουσιάζεται η ίδια η εφαρμογή σε επίπεδο οθονών και εξηγούμε τι ακριβώς συμβαίνει σε κάθε οθόνη. Το έκτο κεφάλαιο περιγράφει τα στάδια που ακολούθησε η ομάδα έτσι ώστε να είναι ξεκάθαρα τα συνολικά βήματα ενός τέτοιου project, αλλά και σε ποιο σημείο δραστηριοποιήθηκε ατομικά το κάθε ένα μέλος της, το κεφάλαιο αναφέρει κάποιες βελτιώσεις που θα μπορούσα να γίνουν συνολικά στην εφαρμογή αλλά και τα συμπεράσματα τα οποία προκύψανε κατά την διάρκεια της εκπόνησης της εργασίας.

2 Τι είναι το iOS

Σχήμα 1: Apple and iOS logo

2.1 Εισαγωγή

Το iOS, το οποίο προηγουμένως ονομαζόταν ως iPhone OS, είναι ένα λογισμικό για κινητά, που αναπτύσσεται και διανέμεται εκ μέρους της Apple Inc. Με βασικό χαρακτηριστικό του την ύπαρξη ενός ευχάριστου περιβάλλοντος εργασίας, εκπληκτικών δυνατοτήτων και τεράστιας ασφαλείας, το iOS αποτελεί τον πυρήνα των iPhone, των iPad και των iPod touch. Έχει σχεδιαστεί με τέτοιο τρόπο, ώστε να λειτουργεί άψογα και να δείχνει πολύ όμορφο, μετατρέποντας σε συναρπαστική οποιαδήποτε εργασία, όσο απλή κι αν είναι. Εξαιτίας του ότι έχει σχεδιαστεί για να αξιοποιεί εξ ολοκλήρου προηγμένη τεχνολογία, που είναι ενσωματωμένη στην Apple, οι συσκευές αυτές είναι πάντα πολύ πιο ανταγωνιστικές από τα υπόλοιπα κινητά.

2.2 Ιστορία

Το 2005 ο Steve Jobs ξεκινήσε τον σχεδιασμό του iPhone. Ο ίδιος θέλησε να ακολουθήσει μία καλύτερη προσέγγιση, από τα ήδη υπάρχον κινητά τηλέφωνα και έτσι δημιούργησε το iPhone OS. Αυτή του η απόφαση οδήγησε το iPhone στην επιτυχία και στη χρήση του ως ένα πολύ σημαντικό λειτουργικού συστάματος υπολογιστών, που επέτρεπε σε πολλούς προγραμματιστές να δημιουργούν λογισμικό για το iPhone με την ελάχιστη δυνατή κατάρτιση και εξειδίκευση. Ο Foster ήταν υπεύθυνος για να δημιουργήσει ένα λογισμικό για τους προγραμματιστές, έτσι ώστε να μπορούν να κατασκευάσουν εφαρμογές για iPhone και ένα APP Store μέσα στο iTunes (Satariano, 2011).

Το λειτουργικό αυτό σύστημα παρουσιάστηκε στο iPhone στο Macworld Conference & Expo μαζί με το iPhone τον Ιανουάριο του 2007 και κυκλοφόρησε τον Ιούνιο αυτού του έτους (Eadicicco, 2017). Ο Jobs τότε ισχυρίστηκε ότι το iPhone μπορεί θα τρέξει το OS 6 και να εκτελέσει εφαρμογές κατηγορίας των επιτραπέζιων υπολογιστών, αλλά τη στιγμή που κυκλοφόρησε το λειτουργικό αυτό σύστημα ονομαζόταν iPhone OS. Στην αρχή δεν υποστήριζε εγγενείς εφαρμογές τρίτων. Η βασική σκέψη του Jobs ήταν οι προγραμματιστές να έχουν την ικανότητα να δημιουργήσουν εφαρμογές μέσα από το πρόγραμμα περιήγησης Safari, οι οποίες θα συμπεριφέρονταν ως εγγενείς στο iPhone (Gonsalves, 2011). Τον Οκτώβριο του 2007, από την Apple ανακοινώθηκε ένα native Software Development Kit (SDK), το οποίο επρόκειτο να κατασκευαστεί τον Φεβρουάριο. Τον Μάρτιο 2008 τελικά Apple, στα πλαίσια μιας εκδήλωσης τύπου, ανακοίνωσε την δημιουργία του iPhone SDK (Eran, 2017) .

Το iPhone πρώτης γενιάς ήταν η πρώτη συσκευή που κυκλοφόρησε στο εμπόριο που χρησιμοποιούσε το iOS. Τον Ιούλιο του 2008 ξεκίνησε να λειτουργεί το iOS App Store, το οποίο αρχικά διέθετε 500 εφαρμογές. Σταδιακά αυτές έφτασαν τις 3000 τον Σεπτέμβριο του 2008, τις 15 χιλιάδες τον Ιούνιο του 2009, τις 50.000 τον Ιούλιο του ίδιου έτους, τις 100.000 τον Νοέμβριο αυτού του έτους, τις 250.000 τον Αύγουστο, τις 650.000 τον Ιούνιο 2012, τις 1 εκατομμύριο τον Οκτώβριο του 2013, τις 2 εκατομμύρια τον Ιούνιο του 2016 και τις 2,2 εκατομμύρια τον Ιανουάριο του 2017. Από τον Μάρτιο του 2016 περίπου 1 εκατομμύριο εφαρμογές είναι εγγενώς συμβατές με έναν υπολογιστή ή τάμπλετ iPad. Οι εφαρμογές αυτές έχουν ληφθεί από χρήστες πάνω από 130 δισεκατομμύρια φορές (Dignan, 2017).

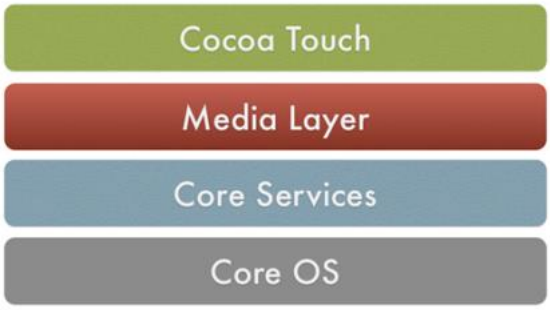
Το 2010 το iPhone OS ονομάστηκε iOS. Το σήμα αυτό χρησιμοποιήθηκε αρχικά από την Cisco για περισσότερα από 10 χρόνια και έτσι η Apple, για να αποφύγει κάθε αγωγή, πήρε την άδεια να το χρησιμοποιεί (Eran, 2017).

2.3 Αρχιτεκτονική

Το iOS, όπως αναφέρθηκε ήδη, αναπτύχθηκε εκ μέρους της Apple Inc. Αρχικά παρουσιάστηκε για το κινητό iPhone του 2007 και στη συνέχεια αναπτύχθηκε για ακόμη περισσότερες συσκευές όπως ήταν η Apple TV, τα iPad και τα iPod touch. Η Apple δεν δίνει άδεια εγκατασταθεί το λογισμικό της σε άλλες συσκευές, που δεν ανήκουν στην ίδια, κάτι το οποίο δεν ισχύει με το Android εκ μέρους της Google ή με το Windows Phone εκ μέρους της Microsoft.

Το iOS βασίζει τη λειτουργία του στον άμεσο χειρισμό μέσω της χρήσης πολλών ταυτόχρονα χειρισμών. Χρησιμοποιεί διάφορα στοιχεία για να χειριστεί το λογισμικό όπως είναι οι διακόπτες, τα κουμπιά και οι ολισθητές. Η αλληλεπίδραση μεταξύ των χρηστών και του λειτουργικού συστήματος περιλαμβάνει ακόμα και χειρονομίες, όπως είναι τα χτυπήματα στην οθόνη. Κάποιες από τις εφαρμογές που χρησιμοποιούνται περιλαμβάνουν και τη χρήση εσωτερικών επιταχυνσιόμετρων. Ορισμένες άλλες κυρίες λειτουργίες είναι η μεγέθυνση που γίνεται με τη χρήση των δύο δακτύλων και η εναλλαγή των φωτογραφιών με το δάχτυλο είτε προς τα αριστερά είτε προς τα δεξιά (Satariano, 2011).

Το iOS μοιράζεται ορισμένα κοινά Frameworks με το OS X, όπως είναι τα Core Foundation και Foundation Kit. Η εργαλειοθήκη του όμως είναι η Cocoa Touch και όχι η Cocoa, που χρησιμοποιείται στο OS X. Ο λόγος για τον οποίο γίνεται αυτό είναι για να μπορεί να προσφέρει το UIKit framework και όχι το Appkit framework. Εξαιτίας αυτού δεν μπορεί να είναι συμβατό με τις εφαρμογές OS X. Κάθε χρόνο δημοσιεύονται και ανακοινώνονται νέες εκδόσεις του λογισμικού αυτού. Η τελευταία έκδοση του είναι το iOS 16. Χρησιμοποιούνται περίπου 1,3 Gbyte από την μνήμη flash που περιέχουν οι συσκευές, ενώ υπάρχουν τέσσερα επίπεδα άντλησης, το Core OS, το Core Services, το Media και το Cocoa Touch. Όλες οι εργασίες του λογισμικού παρέχονται πλήρως δωρεάν. Η ανάπτυξη των εφαρμογών γίνεται με τη χρήση του εργαλείου Xcode, ενώ το τρέχον iOS τρέχει η αρχιτεκτονική 64-bit ARM. Για να γίνει δυνατή η πλοήγηση επιλέγεται ο browser Safari, που μπορεί να υποστηρίξει και τις web εφαρμογές (Adi Pranata et al., 2022).



Σχήμα 2: Επίπεδα αρχιτεκτονικής iOS.

Για να αναπτυχθεί η εφαρμογή επιλέγεται το λογισμικό ανάπτυξης των εφαρμογών iOS SDK, το οποίο δημιουργήθηκε από την Apple και δόθηκε σε κάθε πρόγραμμα το 2008. Με αυτόν τον τρόπο οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές με τη χρήση του εργαλείου Xcode και στη συνέχεια να προχωρήσουν στη δοκιμασία τους στον εξομοιωτή που προσφέρεται από το Xcode, που ονομάζεται iOS Simulator. Για να ανέβει μια εφαρμογή στο App Store και να γίνει διαθέσιμη στο κοινό, θα πρέπει οι ιδρυτές της να καταβάλουν περίπου 99€ ετησίως. Οι ιδρυτές αυτοί θα γνωρίζουν την τιμή, στην οποία θα πουλήσουν την εφαρμογή τους, το ελάχιστο ποσό της οποίας καθορίζεται από την Apple. Όλοι οι προγραμματιστές μπορούν να υλοποιήσουν τις εφαρμογές τους σε γλώσσες προγραμματισμού swift jet FCObjective-C, Swift και το Flutter το οποίο θα γνωρίσουμε στην εργασία αυτή. Οι απαιτήσεις που έχει το σύστημα είναι ένας υπολογιστής Mac, ο οποίος θα περιλαμβάνει το λειτουργικό σύστημα Mac OS X Leopard ή κάτι πιο σύγχρονο. Όλες οι προηγούμενες εκδόσεις δεν

μπορούν να υποστηριχθούν, ούτε όμως και λειτουργικά συστήματα όπως είναι τα Windows (https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/).

3 Εισαγωγή στο Flutter

3.1 Εισαγωγή

Στο κεφάλαιο αυτό αναλύεται Flutter και πιο συγκεκριμένα αρχικά η γλώσσα που χρησιμοποιεί, η Dart. Στη συνέχεια γίνεται λόγος για αυτή τη γλώσσα, για τον τρόπο λειτουργίας της, την εξέλιξη αυτής και τον λόγο, για τον οποίο χρησιμοποιείται από το Flutter. Ακολούθως γίνεται συγκεκριμένη αναφορά στον Flutter. Αναλυτικότερα παρουσιάζεται ο τρόπος δημιουργίας του και ο στόχος του σαν ένα εργαλείο για την ανάπτυξη εφαρμογών. Επιπλέον γίνεται αναφορά στα Widgets, που αποτελούν τα βασικά χαρακτηριστικά του και στον τρόπο, με τον οποίο μπορεί κάποιος να το εγκαταστήσει.

3.2 Η γλώσσα Dart

Επίκεντρο του Flutter είναι η γλώσσα Dart. Όλες οι καινούργιες τεχνολογίες, όπως είναι και το Flutter, χρειάζονται την υποστήριξη από μία σύγχρονη γλώσσα, πολύ υψηλού επιπέδου για να είναι τα αποτελέσματα όσο το δυνατόν πιο καλά. Έτσι θα μπορέσει ο προγραμματιστής να αποκτήσει την εμπειρία που χρειάζεται και να δημιουργήσει πολύ καλές εφαρμογές για τα κινητά τηλέφωνα. Οι προγραμματιστές που επιδιώκουν να ασχοληθούν με την τεχνολογία αυτή θα πρέπει να γνωρίζουν τα πάντα γύρω από τη γλώσσα Dart, από τη φιλοσοφία της, από τα πλεονεκτήματα που έχει και από τον λόγο, για τον οποίο θα πρέπει να επιλεχθεί και να αναπτυχθεί το Flutter (Biessek, 2019). Αναλυτικότερα η γλώσσα Dart έχει αναπτυχθεί εκ μέρους της Google και αποτελεί μία γλώσσα προγραμματισμού, ικανή να χρησιμοποιηθεί για να αναπτυχθούν Desktop, Mobile και Web εφαρμογές. Η γλώσσα αυτή χρησιμοποιείται για να αναπτυχθούν εφαρμογές με πολύ υψηλό επίπεδο και για να προσφερθεί εμπειρία στους προγραμματιστές. Για πρώτη φορά εμφανίστηκε το 2011 και εξελίσσεται διαρκώς. Το 2013 παρουσιάστηκε η σταθερή της έκδοση, η οποία περιελάμβανε ελάχιστες αλλαγές, όπως ήταν η κυκλοφορία του Dart 2.0 το 2018. Κατά τη διάρκεια των πρώτων ετών επίκεντρο ήταν το «web development», για να αναπτυχθούν και να συλληφθούν οι ιστοσελίδες, έτσι ώστε να μπορέσει να αντικατασταθεί η Javascript. Το τελευταίο διάστημα όμως η συγκεκριμένη γλώσσα δίνει έμφαση κυρίως στον τρόπο ανάπτυξης κινητών εφαρμογών και εξαιτίας αυτού επιλέγεται όλο και περισσότερο. Απώτερος στόχος λοιπόν είναι να επιλύσει προβλήματα, τα οποία παλαιότερα αντιμετωπίζονταν μέσω της Javascript. Ακόμα σημειώνει καλύτερες αποδόσεις και χρησιμοποιεί πιο αποτελεσματικά εργαλεία, όταν τα έργα είναι μεγαλύτερης κλίμακας. Τα συγκεκριμένα εργαλεία είναι σταθερά και σύγχρονα IDE plugins, ενώ σχεδιάζονται για να έχουν βέλτιστες αποδόσεις και ταυτόχρονα να διατηρούν την αίσθηση ότι πρόκειται για μία ξεκάθαρα δυναμική γλώσσα. Επιπλέον διαθέτουν και γνωρίσματα, που την κάνουν δυναμική είναι να χαρακτηρίζεται από ιδιαίτερη ευελιξία και στιβαρότητα, έτσι ώστε να μπορεί να ανταποκριθεί σε πολλαπλές πλατφόρμες και να βελτιώνεται συνεχώς (Napoli, 2019).

3.2.1 Τρόπος λειτουργίας της γλώσσας Dart

Η γλώσσα Dart, όπως αναφέρθηκε και παραπάνω, χαρακτηρίζεται από μεγάλη ευελιξία, χάρη στο περιβάλλον, στο οποίο δημιουργήθηκε. Ο κώδικας που περιλαμβάνει είναι δυνατόν να αναπτυχθεί μέσω πολλών και διαφορετικών τρόπων, όπως είναι οι ακόλουθοι:

1. Ανεξαρτητοποίηση.

Το σύστημα της Java, χρησιμοποιεί το Java Virtual Machine (JVM) για να μπορέσει να εκτελεστεί. Με παρόμοιο τρόπο η Dart προχώρησε στη χρήση του Dart Virtual Machine (DVM). Για να χρησιμοποιήσει, λοιπόν, κάποιος να εκτελέσει την Dart θα πρέπει αρχικά να κατεβάσει και να εγκαταστήσει το συγκεκριμένο σύστημα. Επιπλέον το Software Development Kit (SDK), εκτός από τις βιβλιοθήκες και τη μεταγλώττιση, παρέχει και πολλά, εργαλεία όπως είναι τα:

- the pub package
- managerdart2js, που χρησιμοποιείται για τη μεταγλώτιση σε JavaScript
- dartdoc,
- Dart Documentation generator
- dartfmt, πρόκειται για έναν μορφοποιητή κώδικα, που ακολουθεί συγκεκριμένους κανόνες (Payne, 2019a).

Αυτό σημαίνει ότι οποιοσδήποτε επιθυμεί να τρέξει τον κώδικα Dart, θα πρέπει να εγκαταστήσει αρχικά το DVM. Για να μπορέσει να πραγματοποιήσει μία εφαρμογή Flutter, είτε πρόκειται για κινητό είτε για υπολογιστή είτε για internet, θα πρέπει να εγκαταστήσει αρχικά το Flutter, που θα περιλαμβάνει αφενός το Dart SDK και αφετέρου το Flutter SDK (Payne, 2019a).

2. Ahead Of Time (AOT)

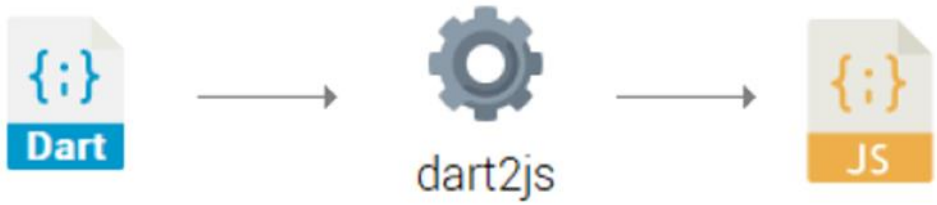
Μέσω του Ahead Of Time (AOT) μπορεί να επιτευχθεί η μεταγλώττιση μιας γλώσσας προγραμματισμού, η οποία χαρακτηρίζεται για το υψηλό επίπεδό της, σε εγγενή κώδικα μηχανής. Όπως γίνεται αντιληπτό και στο Σχήμα 1, αρχικά θα πρέπει να υπάρχει ένα αρχείο με προέλευση Dart, ικανό να μετατραπεί σε δυαδικό αρχείο και στη συνέχεια να εκτελεστεί μέσα στα πλαίσια συγκεκριμένου λειτουργικού συστήματος. Το AOT κάνει ακόμα πιο γρήγορο το Flutter και ταυτόχρονα το μετατρέπει σε φορητό (Payne, 2019b).

Σχήμα 3: AOT compilation

Με τη συμβουλή του AOT δεν είναι αναγκαίο κάποιος να εγκαταστήσει το DVM, λόγω του ότι πλέον το αρχείο είναι δυαδικό και έτσι μπορεί να τρέξει είτε σε Android με .apk, .aab, είτε σε IOS με .ipa είτε σε .exe για Windows. Σύμφωνα με το Flutter SDK, στο οποίο περιλαμβάνεται το AOT compile, μεταγλωτίζεται πολύ εύκολα ο κώδικας Dart σε έναν εγγενές δυαδικό για κινητές συσκευές, υπολογιστές και internet. Στο Flutter 1.21 περιλαμβάνεται πλέον το Dart SDK και έτσι δε χρειάζεται να γίνει η εγκατάστασά του ξεχωριστά. Στην έκδοση Flutter 2.6 η εντολή dart2native μπορεί να πραγματοποιήσει AOT μεταγλωττίσεις για οποιοδήποτε πρόγραμμα Dart στον εγγενή κώδικα x64 και αυτό να οδηγήσει σε εκτελέσιμα αρχεία(Payne, 2019b).

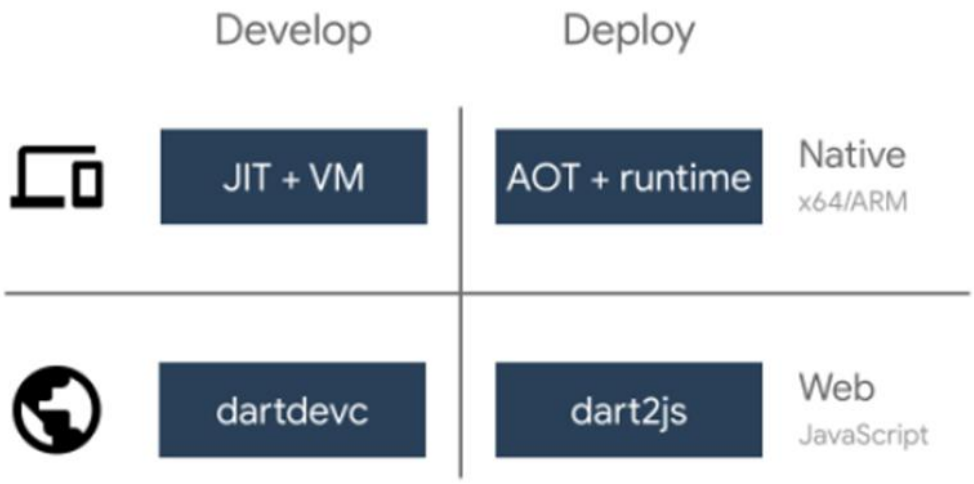
3. Ιστότοπος

Εξαιτίας του εργαλείου dart2js, ο κώδικας Dart μπορεί να μεταφραστεί με μεγάλη ταχύτητα και ευκολία σε κώδικα Javascript, όπως φαίνεται και στο Σχήμα 2. Έτσι το Flutter μπορεί να τρέξει Firefox, Chrome και το User Interface UI, ακριβώς όπως οι άλλες πλατφόρμες. Εξάλλου ένα από τα frameworks που χρησιμοποιεί η Google για να φτιάξει τις συγκεκριμένη ιστοσελίδες, όπως είναι τα «AdSense» και τα «AdWords». είναι και η AnjularDart (Payne, 2019b).



Σχήμα 4: Dart2js Compilation

Ακολούθως στο Σχήμα 5 παρουσιάζεται η ανάπτυξη (Deployment) του κώδικα Dart. Στη συγκεκριμένη εικόνα μελετώνται δύο περιπτώσεις, εκ των οποίων η πρώτη σχετίζεται με έναν υπολογιστή ή ένα κινητό και η δεύτερη με έναν ιστότοπο.



Σχήμα 5: Develop - Deploy Διαδικασία

4. Υπολογιστής/κινητό

Η Just In Time (JIT) τεχνική είναι μία συναλλαγή πραγματικού χρόνου, λόγω του ότι η συλλογή γίνεται την ώρα που εκτελείται ένα πρόγραμμα. Μέσα από τον συνδυασμό των δύο συγκεκριμένων τεχνολογιών, DVM + JIT , μπορεί να αποσταλεί ο κώδικας με δυναμικό τρόπο, χωρίς να λαμβάνεται υπόψη η αρχιτεκτονική που έχει το μηχάνημα των χρηστών. Έτσι τα αποτελέσματα είναι πολύ καλά στα πλαίσια της εξεύρεσης των σφαλμάτων και ταυτόχρονα εκτελείται ομαλότερα ο κώδικας (Zaccagnino, 2020).

5. Ιστότοπος

Στην περίπτωση του ιστοτόπου υπάρχει το εργαλείο «dartdevc» , που βοηθά να τρέξει ο ανιχνευτής λαθών σε ιστότοπους και σε εφαρμογές. Το συγκεκριμένο εργαλείο χρησιμοποιείται για την ανάπτυξη (Deploy), ενώ αντίστοιχα το «dart2js» χρησιμοποιείται για την ανάπτυξη (Deployment). Πέρα από αυτό υπάρχει και η δυνατότητα χρήσης ειδικών εργαλείων, στα οποία συμπεριλαμβάνεται και το «webdev», που βοηθά στην επεξεργασία των Dart αρχείων και στην οπτικοποίηση όλων των αλλαγών που γίνονται στο Browser (Zaccagnino, 2020).

3.3 OOP Dart γνωρίσματα

Η Dart, όπως και ορισμένες ακόμα γλώσσες, είναι μια αντικειμενοστρεφείς(OOP – Object Oriented Programming) γλώσσα. Εξαιτίας αυτού ακολουθεί κάποιες από τις βασικές αρχές τους, χαρακτηρίζεται ομως κι από μία ιδιαιτερότητα. Οι βασικές αυτές αρχές και δομές παρουσιάζονται ακολούθως.

- **Κλάσεις και αντικείμενα**

Μία από τις κύριες αρχές του OOP είναι αρχικά τα αντικείμενα και οι καθορισμένες κλάσεις. Όπως τονίστηκε προηγουμένως, στην Dart όλα είναι αντικείμενο. Αυτό σημαίνει ότι η αποθήκευση μιας οποιασδήποτε τιμής σε μεταβλητή συνοδεύεται πάντα από την ύπαρξη μιας κλάσης. Οι κλάσεις στη Dart διαδραματίζουν σπουδαίο ρόλο και έτσι μπορούν να σημαίνουν το ίδιο τόσο για τα μέλη παρουσίας, για τις μεθόδους δηλαδή και τα πεδία, όσο και για τα μέλη κλάσης, για τις στατιστικές δηλαδή μεθόδους και για τα πεδία. Ακόμα είναι ικανή να επιτύχει τη χρήση διαφορετικών τρόπων για να δημιουργηθεί μία κλάση, χωρίς να υπερφορτωθεί ο κατασκευαστής. Αυτό το επιτυγχάνει μέσω της δημιουργίας διαφοράς συναρτήσεων από την ίδια τη γλώσσα με τη συμβολή των κατασκευαστών (https://dart.dev/).

- **Ενθυλάκωση**

Ένα από τα πλέον πολλά πλεονεκτήματα που έχει είναι ότι δεν υπάρχει κάποιος περιορισμός πρόσβασης σχετικά με τις γνωστές λέξεις κλειδιά public, private και protected. Μία ακόμη μεγάλη διαφορά είναι και το ότι η ενθυλάκωση συμβαίνει στο επίπεδο της βιβλιοθήκης και όχι στο αντίστοιχο της τάξης, όπως γίνεται με άλλες γλώσσες, οι οποίες χρησιμοποιούν πρότυπα OOP. Επιπλέον οι μέθοδοι getters-setters στην Dart μπορούν να γίνουν προσβάσιμες και από άλλες κλάσεις, έτσι ώστε οποιαδήποτε στιγμή να γίνεται η άμεση ενημέρωσή τους. Επιπλέον χρησιμοποιείται το σύμβολο (_) για να ελέγχεται η ιδιωτικότητα της βιβλιοθήκης. Αυτό συναντάται κυρίως σε κλάσεις, συνάντηση συναρτήσεις, μεταβλητές και μέλη τάξης (https://dart.dev/).

- **Κληρονομικότητα**

Η κληρονομικότητα είναι η δυνατότητα να επεκταθεί ένα αντικείμενο με στόχο να χρησιμοποιηθούν ακόμα πιο εξειδικευμένες εκδόσεις μιας γλώσσας ή κάποιου τύπου πιο αφηρημένου. Η συγκεκριμένη διαδικασία στην Dart είναι αυτοματοποιημένη την ώρα που δηλώνεται μία τάξη. Αυτό σημαίνει ότι άμεσα επεκτείνεται ο τύπος του αντικειμένου. Η κληρονομικότητα λοιπόν είναι άμεση, ενώ αυτό σε συνδυασμό με την ύπαρξη της λειτουργίας mixins περιορίζει την άμεση κληρονομικότητα, έτσι ώστε να μπορεί να επαναχρησιμοποιηθεί ο κώδικας. Τέλος σημαντική είναι και η δυνατότητα επέκτασης μία κλάσης, που δεν υπάρχει σε άλλες γλώσσες (https://dart.dev/).

- **Abstraction**

Μέσα από την αφαίρεση είναι δυνατόν να οριστούν ένας τύπος και τα κύρια χαρακτηριστικά του με τη βοήθεια πιο προχωρημένων τύπων εκ μέρους τους γονέων των κλάσεων. Βασικό γνώρισμα της Dart είναι ότι δεν την ενδιαφέρει ο τρόπος εφαρμογής τους αλλά περιλαμβάνει τάξεις, που προσφέρουν όλα όσα μπορούν να κάνουν. Επιπλέον κάθε κατηγορία μετατρέπεται σε μία διεπαφή, γεγονός που υλοποιείται μέσω της σιωπηρούς διεπαφής, χωρίς όμως να χρειάζεται να επεκταθεί εκ μέρους άλλων (https://dart.dev/).

- **Πολυμορφισμός**

Ο πολυμορφισμός βοηθάει στην επίτευξη της κληρονομιάς. Το βασικό χαρακτηριστικό του είναι ότι χρησιμοποιείται κάθε φορά που επιδιώκεται ένα αντικείμενο να συμπεριφέρεται όπως ακριβώς ένα άλλο, που έχει παρόμοιες λειτουργίες. Η Dart μπορεί να υποστηρίξει κάτι τέτοιο, αλλά σε ορισμένα σημεία διαφέρει από τις υπόλοιπες γλώσσες.

3.4 Λόγοι χρήσης της Dart εκ μέρους του Flutter

Υπάρχουν διάφοροι λόγοι, εξαιτίας των οποίων η Google απόφασισε να επιλέξει τη γλώσσα Dart. Ένας από τους κυριότερους είναι η αντικειμενοστρέφεια που περιλαμβάνει. Οι πιο πολλοί προγραμματιστές της σύγχρονης εποχής ξέρουν να γράφουν σε γλώσσες αντικειμενοστρεφείς, κάτι το οποίο σημαίνει ότι η γλώσσα Dart είναι πολύ εύκολο και γρήγορο να μαθευτεί από τους ίδιους. Οι προγραμματιστές λοιπόν δε χρειάζεται να ασχοληθούν με έναν κώδικα που θα είναι εντελώς καινούργιος για αυτούς, εφόσον με τις γνώσεις τους μπορούν να διαχειριστούν κάθε ιδιαιτερότητα της Dart. Άλλος ένας σημαντικός παράγοντας για να παραχθεί και να υλοποιηθεί μία εφαρμογή είναι οι πολύ υψηλές επιδόσεις που μπορεί να δώσει η γλώσσα προγραμματισμού, με την οποία εργάζονται στα πλαίσια εκτέλεσης ενός προγράμματος. Η Dart το εγγυάται αυτό μέσω ενός πολύ ισχυρού κατανεμητή μνήμης, που μπορεί να χειριστεί κάθε μικρή κατανομή. Αυτό είναι η πιο καλή λύση για το λειτουργικό Flutter. Εξάλλου η παραγωγικότητα είναι ένα κύριο γνώρισμα που κάνει την Dart επιθυμητή εκ μέρους του Flutter. Το ίδιο μπορεί να παράγει περιεχόμενο για Android, Web και Desktop σύμφωνα με μία βάση κώδικα, που έχει τις ίδιες όψεις και αναπαραστάσεις για όλες τις πλατφόρμες. Αυτό σημαίνει ότι μία γλώσσα τόσο παραγωγική όσο η Dart κάνει πολύ πιο γρήγορη τη διαδικασία και αυξάνει την ελκυστικότητα του framework. Επιπλέον ακόμα ένας λόγος είναι το ότι οι δύο συγκεκριμένες τεχνολογίες δημιουργήθηκαν εκ μέρους της Google και έτσι μπορεί πολύ πιο ελεύθερα να τις διαχειρίζεται και να της εξελίσσει (Napoli, 2019).

Υπάρχουν, όμως, και διάφοροι ακόμα λόγοι για να προχωρήσει κάποιος στην επιλογή της γλώσσας Dart ως βασική γλώσσα. Αναλυτικότερα η ίδια έχει ένα πολύ αυστηρό τυπολόγιο. Ακόμα οι χρόνοι της εκτέλεσης είναι πολύ πιο γρήγοροι συγκριτικά με άλλες γλώσσες προγραμματισμού, ενώ περιλαμβάνει κι έναν εύκολο και γρήγορο στη χρήση ανιχνευτή λαθών. Επιπροσθέτως περιέχει και διάφορα, εργαλεία όπως είναι τα ακόλουθα:

- Tree-shaking optimazion
- Hot reload feature
- DartPad, ένα playground για να μπορέσει κάποιος να εξασκηθεί στη γλώσσα
- DevTools, μια συλλογή από ανιχνευτές λαθών
- Code documentation generator tool

· Υποστηρίζει το JIT και AOT

3.5 To Flutter

Το Flutter αποτελείται από διάφορα εργαλεία για την ανάπτυξη των εφαρμογών για κινητά τηλέφωνα και δημιουργήθηκε εκ μέρους της Google. Είναι ένας ανοιχτός κώδικας προσβάσιμος από κάθε προγραμματιστή που θέλει να ασχοληθεί με αυτόν. Πρόκειται για ένα εργαλείο, με το οποίο μπορούν πολύ εύκολα και γρήγορα να δημιουργηθούν εφαρμογές υψηλού επιπέδου για κινητά τηλέφωνα. Μία από τις βασικές αρχές του είναι το «write once, and deploy everywhere». Αυτό σημαίνει ότι μπορεί να δημιουργηθεί περιεχομένου για Chrome OS, iOS και Android. Στόχος του Flutter για το μέλλον είναι να μπορεί να τρέξει σε desktop και web εφαρμογές σε πολλά λειτουργικά συστήματα. Φαίνεται ότι είναι ένα ολοκληρωμένο εργαλείο για να παραχθούν οι εφαρμογές και μία πλατφόρμα που προσφέρει οτιδήποτε χρειάζεται κάποιος για να φτιάξει μία εφαρμογή. Έτσι μπορεί κανείς να βρει έναν rendering engine, testing frameworks, εξαρτήματα διεπαφής χρήστη και πολλά ακόμα εργαλεία. Το μόνο λοιπόν που έχει να κάνει ένας προγραμματιστής είναι να καθορίσει τη λειτουργικότητα της εκάστοτε εφαρμογής (Alessandria, 2020).

Το Flutter είναι μία νέα τεχνολογία, η οποία κατά τη διάρκεια του τελευταίου χρόνου εξελίσσεται ραγδαία. Για πρώτη φορά συναντήθηκε με την ονομασία Sky στο Dart Developer Summit 2015 εκ μέρους του Eric Seidel. Τότε η Google ήθελε να δημιουργήσει μία τεχνολογία, που θα χαρακτηριζόταν από πιο υψηλό επίπεδο και από την καλύτερη εμπειρία χρήσης. Το Flutter παρουσιάστηκε έτσι στο κοινό με αυτήν την ονομασία το 2016, ενώ η alpha έκδοση έγινε τον Μάιο του 2017. Η ίδια Είχε κατασκευαστεί για συστήματα Android και iOS. Προς τα τέλη του 2018, εξαιτίας του μεγάλου ενδιαφέροντος που υπήρξε εκ μέρους της κοινότητας, παρουσιάστηκε και η σταθερή εκδοχή της συγκεκριμένης τεχνολογίας (Alessandria, 2020).

Μαζί με το Flutter υπάρχουν και ορισμένα frameworks, τα οποία έκαναν αυτό που στην ουσία θα πρέπει τώρα να κάνει το Flutter. Όπως έχει ήδη τονιστεί, προοριζόταν για την παροχή μιας καλύτερης εμπειρίας στον χρήστη και μιας υψηλού επιπέδου λειτουργίας και εκτέλεσης. Φυσικά δεν είναι μόνο αυτά, τα οποία υπόσχεται. Βασικό μέλημα των δημιουργών του ήταν να φτιάξουν μία νέα πλατφόρμα για τα πολλαπλά συστήματα των κινητών συσκευών. Γι' αυτόν τον λόγο υπήρξαν ορισμένα χαρακτηριστικά, για τα οποία έγιναν προσπάθειες βελτίωσης και ενσωμάτωσης στον εξοπλισμό αυτού. Τα προηγούμενα χρόνια για να μπορέσει μία εταιρεία ή μία ομάδα να καλύψει τη ζήτηση της αγοράς, όσον αφορά στις εφαρμογές των κινητών συσκευών, έπρεπε να διαθέσει τεράστια χρηματικά ποσά και πάρα πολλές εργατοώρες για να καλύψει κάθε σύστημα. Εδώ ακριβώς πλέον μπορεί να επέμβει το Flutter και να παρέχει μία ενιαία πλατφόρμα για να παραχθούν πολλαπλές συσκευές και λειτουργικά συστήματα (Biessek, 2019).

Κάθε προγραμματιστής που επιδιώκει να ασχοληθεί με τις εφαρμογές για κινητές συσκευές θα πρέπει να γνωρίζει πολλές γλώσσες προγραμματισμού, για να μπορεί να καλύψει κάθε λειτουργικό σύστημα, όπως είναι το Swift για το iOS και το Java ή το Kotlin για το Android. Το Flutter μέσα από την γλώσσα Dart προσφέρει την καινοτομία να αναπτυχθούν εφαρμογές για πάρα πολλά συστήματα κινητών συσκευών. Επίσης ένας από τους βασικούς παράγοντες που επηρεάζει την παραγωγικότητα των προγραμματισμών είναι ο μεγάλος χρόνος

μεταγλώττισης και κατασκευής οποιασδήποτε εφαρμογής. Η υλοποίηση στο λειτουργικό Android δείχνει ότι οι κύκλοι παραγωγής είναι μεγαλύτεροι κι αυτό ταλαιπωρεί τους προγραμματιστές που επιλέγουν να το χρησιμοποιήσουν. Το Flutter έκανε προσπάθειες να μειωθούν οι χρόνοι της μεταγλώττισης, έτσι ώστε να αυξηθεί η παραγωγή του υλικού και να βοηθηθούν οι προγραμματιστές να απαλλαγούν από τις πολύωρες αναμονές (Miola, 2020).

3.5.1 Παρόμοια frameworks

Μαζί με το Flutter υπήρξαν και ορισμένα, frameworks, τα οποία έκανα πράξη όσα ήθελε να κάνει το αυτό, τα οποία παρουσιάζονται στο Σχήμα 4. Μερικά από τα κυριότερα είναι τα εξής:

- Xamarin
- React Native
- Ionic
- Native Script

Σχήμα 6: Frameworks

1. Xamarin

Για πρώτη φορά το **Xamarin** δημοσιεύτηκε το 2011 εκ μέρους της Microsoft. Πιο συγκεκριμένα πρόκειται για ένα εργαλείο ανάπτυξης για κινητές συσκευές και αποτελεί μια πλατφόρμα ανοικτού κώδικα. Με αυτό το framework μπορούν να υλοποιηθούν συσκευές για Android, για Windows με .net και για iOS. Οι γλώσσες που χρησιμοποιούνται εκ μέρους τους είναι οι HTML, CSS και JavaScript. Το συγκεκριμένο framework βρίσκεται στην 10η θέση των πιο δημοφιλών frameworks για να αναπτυχθούν λογισμικά οι πλατφόρμες των κινητών συσκευών (https://medium.com/@devathon_/10-best-android-frameworks-for-app-development-in-2020-98f5afb300e9 .

2. React Native

Για πρώτη φορά δημοσιεύτηκε το 2015 και αποτελεί ένα open source framework, το οποίο στην κατάταξη δημοφιλίας κατέχει την 8η θέση μεταξύ των πιο αγαπημένων frameworks. Πρόκειται για μία πλατφόρμα που αναπτύσσεται πολύ γρήγορα και χρησιμοποιείται κυρίως από προγραμματιστές που επιδιώκουν εύκολες και γρήγορες εφαρμογές. Χρησιμοποιεί τις

γλώσσες framework React και JavaScript. Ορισμένες από τις πιο γνωστές εταιρείες που συνεργάζεται με το συγκεκριμένο framework είναι η Uber, η Tesla, η Instagram και η Facebook (https://medium.com/@devathon_/10-best-android-frameworks-for-app-development-in-2020-98f5afb300e9).

3. Ionic

Το **Ionic** για πρώτη φορά εμφανίστηκε το 2012 και είναι ένα από τα πιο δημοφιλή frameworks για να αναπτύξουν εφαρμογές cross-platform. Πρόκειται για ένα open source framework, το οποίο υλοποιήθηκε εκ μέρους του MIT και κάνει χρήση σύγχρονων τεχνολογιών όπως είναι οι HTLM5, CSS3 και η JavaScript. . Στόχος είναι να δημιουργηθούν υβριδικές και διαδραστικές εφαρμογές για τα κινητά τηλέφωνα. Τα συστήματα, που μπορούν να υλοποιηθούν μέσα από το συγκεκριμένο framework αποτελούν εφαρμογές για τις κινητές συσκευές, υπολογιστικές εφαρμογές και ιστότοπους εφαρμογών (https://medium.com/@devathon_/10-best-android-frameworks-for-app-development-in-2020-98f5afb300e9).

4. Native Script

Το Native Script είναι ένα από τα πιο γνωστά framework, το οποίο χρησιμοποιείται για να δημιουργηθούν εφαρμογές για τις πλατφόρμες των κινητών συσκευών και για πολλές ακόμα. Μέσα από τη χρήση γνωστών τεχνολογιών όπως είναι οι Angular, Vue.js , TypeScript και η JavaScrpit, είναι ικανή η δημιουργία εφαρμογών διεπαφής του χρήστη. Με αυτό το framework οι εφαρμογές φτιάχνονται σε πολύ σύντομο διάστημα (https://medium.com/@devathon_/10-best-android-frameworks-for-app-development-in-2020-98f5afb300e9).

3.5.2 Η διαφορά συγκριτικά με τα υπόλοιπα frameworks

Εξαιτίας των πολλών και διαφορετικών cross-platform frameworks, δεν είναι καθόλου εύκολο οι νέες τεχνολογίες, όπως άλλωστε είναι και το Flutter, να ξεχωρίσουν. Παρόλα αυτά το ίδιο παρουσιάζει ορισμένες διαφοροποιήσεις, που το κάνουν να διαχωρίζεται από τα υπόλοιπα frameworks. Κάποια από τα κύρια σημεία, στα οποία παρατηρούνται αυτές οι διαφοροποιήσεις είναι το εξής:

- Υψηλή απόδοση
- Καλύτερη διαχείριση στα πλαίσια της διεπαφής του χρήστη
- Ύπαρξη της γλώσσας Dart
- Υλοποίηση από την Google
- Είναι open source framework

1. Υψηλή απόδοση

Η Google τονίζει ότι η απόδοση που έχει το Flutter συγκριτικά με τα υπόλοιπα frameworks, είναι πολύ καλύτερη. Το γεγονός αυτό βέβαια είναι πολύ νωρίς να επιβεβαιωθεί, λόγω του ότι αποτελεί μία τεχνολογία των τελευταίων ετών. Από το Flutter χρησιμοποιείται μία

συγκεκριμένη μηχανή γραφικών, που ονομάζεται Skia. Η ιδιαιτερότητά της είναι ότι μπορεί να αποδώσει την εφαρμογή pixel by pixel. Συγκριτικά με τα υπόλοιπα frameworks δεν χρειάζεται να καλέσει την Original Equipment Manufacturer (OEM) widgets και έτσι αποφεύγει τη συμφόρηση του συστήματος, που μπορεί να προκαλέσει καθυστερήσεις στον κύκλο παραγωγής αυτής της εφαρμογής. Επιπλέον με τη χρήση του Dart AOT μεταγλωττιστή εκ μέρους του Flutter, είναι ουσιαστικά σαν να παράγεται ένας native code. Το γεγονός αυτό κάνει καλύτερη τη ροή της παραγωγής και της απόδοσης κατά τη διαδικασία υλοποίησης των εφαρμογών, αλλά και κατά τη διαδικασία παραγωγής τους. Το Flutter λοιπόν φαίνεται ότι είναι πιο αποτελεσματικό και πιο γρήγορο (Napoli, 2019).

2. Καλύτερη διαχείριση στα πλαίσια της διεπαφής του χρήστη

Σχετικά με τη διεπαφή του χρήστη, το Flutter δεν μπορεί να περιοριστεί ούτε από συμβάσεις ούτε από κανόνες. Αυτό οφείλεται στο ότι προβάλλει τα οπτικά στοιχεία απευθείας πάνω στον καμβά. Τα πιο πολλά frameworks αναπαράγουν τις εικόνες που προσφέρει η κάθε πλατφόρμα, με διαφορετικό απλά τρόπο. Πολλά εξ' αυτών περιορίζονται μόνο στα γραφικά στοιχεία OEM, με αποτέλεσμα να μπορούν να αναπαράγουν τις δυνατότητες που τους προσφέρει η πλατφόρμα και μόνο, χωρίς να κάνουν τις δικές τους παρεμβάσεις. Επιπλέον τα frameworks, τα οποία στηρίζονται στις τεχνολογίες του ιστού, περιλαμβάνουν πιο πολλές αλλαγές και δεν διαθέτουν τόσο πολλούς περιορισμούς. Περιορίζονται όμως ανάλογα με την κινητή μηχανή του ιστού, που θα χρησιμοποιηθεί, λόγω του ότι κάθε μία από αυτές έχει τα δικά της ιδιαίτερα χαρακτηριστικά. Το Flutter δίνει στον προγραμματιστή τη δυνατότητα να επεμβαίνει σε κάθε περίπτωση διεπαφής του χρήστη. Όλα αυτά βασίζονται στην ύπαρξη ενός πολύ πλούσιου API widgets, το οποίο έχει τη δυνατότητα να επεκταθεί και να καταγράψει μεγάλες αποδόσεις. Περιέχει έναν ικανό αριθμό Widget σχεδιασμού πλατφόρμας, που αποτελούν ένα υλικό που χρησιμοποιείται από το Android και από το Cupertino για να δημιουργηθεί το λειτουργικό iOS. Επιπλέον χρησιμοποιούνται και διάφοροι σημασιολογικοί κανόνες, που μπορούν να αποδώσουν αποτελεσματικότερες και καλύτερες διατάξεις συγκριτικά με τα υπόλοιπα frameworks, τα οποία χρησιμοποιούν πιο περίπλοκους κανόνες για την διάταξη CSS (Zaccagnino, 2020).

3. Ύπαρξη της γλώσσας Dart

Όπως αναφέρθηκε και προηγουμένως, ένας από τους κύριους στόχους που θέτει το Flutter είναι η δημιουργία μιας πιο εξελιγμένης πλατφόρμας σε σύγκριση με αυτές που υπάρχουν. Αυτό το καταφέρνει με τη χρήση της γλώσσας Dart. Πιο συγκεκριμένα με τη γλώσσα αυτή μπορεί να εκτελείται ένας κώδικας πολύ πιο εύκολα. Το Flutter επιλέγει να χρησιμοποιεί το Dart AOT για να εκδώσει μία εφαρμογή και ταυτόχρονα το JIT, ως έναν δεύτερο μεταγλωττιστή. Όλα αυτά τα χαρακτηριστικά κάνουν πολύ πιο γρήγορο τον κύκλο εργασίας και επιφέρουν διάφορες αλλαγές στον κώδικα. Εξαιτίας της ύπαρξης του AOT μεταγλωττιστή, είναι γρηγορότερη η μεταφορά από non-native σε native , κάτι που παρέχει τη δυνατότητα ύπαρξης ικανοποιητικότερων ταχυτήτων. Επιπλέον το γεγονός ότι η γλώσσα Dart είναι πολύ γρήγορη στην εκμάθηση της, βοηθά ακόμη περισσότερο το Flutter. Πέρα από αυτά είναι μία πολύ προχωρημένη και σύγχρονη γλώσσα, που χρησιμοποιεί την αντικειμενοστρέφεια σε

συνδυασμό με άλλες λειτουργίες γνωστές στις στατικές και στις δυναμικές γλώσσες. Πέρα από αυτά διαθέτει και μία κοινότητα πολύ ενεργή. Τέλος και η σύνταξή του διαφέρει από τα υπόλοιπα frameworks, λόγω του ότι χαρακτηρίζεται από μεγαλύτερη ευελιξία. Αυτό εξασφαλίζει το ότι ο προγραμματιστής μπορεί όπως αυτός θέλει να αναδιαμορφώνει τον κώδικά του, με διάφορα εργαλεία που διαθέτει και να εντοπίζει τα σφάλματα (Kuzmin et al., 2020).

4. Υλοποίηση από την Google

Παρά του ότι το Flutter είναι ένα σύγχρονο λογισμικό, μέσα σε πολύ λίγο χρόνο έχει αγαπηθεί από τον κόσμο. Πιο συγκεκριμένα εξαιτίας του ότι υλοποιείται εκ μέρους της Google, έχει μία τεράστια κοινότητα που το υποστηρίζει. Επιπλέον μέσω της υλοποίησης διαφόρων εκδηλώσεων σε πολύ σύντομα χρονικά διαστήματα αυξάνει συνεχώς τη δημοτικότητά του. Όπως έχει ήδη τονιστεί, η πρώτη έκδοση beta ανακοινώθηκε στο Google IO 2018 και στη συνέχεια το ίδιο έτος εμφανίστηκε και η πρώτη σταθερή έκδοση του Flutter Live Event. Στη σύγχρονη εποχή υπάρχουν περισσότεροι από 200 εκατομμύρια χρήστες των Flutter εφαρμογών, περισσότεροι από 250.000 προγραμματιστές που χρησιμοποιούν αυτήν την τεχνολογία και 3000 εφαρμογές Flutter, οι οποίες περιλαμβάνονται μέσα στο Play Store (Napoli, 2019).

5. Είναι open source framework

Το γεγονός ότι το Flutter ανήκει στα open source frameworks βοήθησε το ίδιο να εξελιχθεί. Αναλυτικότερα μέσα από τη συμβολή της κοινότητας της Google βελτιώνονται όλα τα προβλήματα που μπορούν εμφανιστούν. Ακόμα με τη συνεργασία αυτή παρατηρείται ακόμα πιο μεγάλη εξέλιξη σχετικά με το documentation και γενικότερα με την τεχνολογία που χρησιμοποιείται. Τέλος το πιο βασικό είναι η ύπαρξη μίας αλυσίδας ανατροφοδότησης τόσο των θετικών όσο και των αρνητικών απόψεων, κάτι που βοηθάει στην περαιτέρω ανάπτυξή του (Napoli, 2019).

3.5.3 Εισαγωγή στα widgets

Όλοι οι προγραμματιστές που θέλουν να ασχοληθούν με την τεχνολογία Flutter, οφείλουν να γνωρίζουν τον τρόπο λειτουργίας του Widget. Πιο συγκεκριμένα η δομή που περιλαμβάνει ο κώδικας Flutter αποτελείται ακριβώς από τα Widget. Τα ίδια αναπαριστούν διάφορα μηνύματα της εφαρμογής και γι' αυτόν τον λόγο είναι πολύ σημαντικά για αυτήν. Βασικό χαρακτηριστικό των εφαρμογών είναι το ότι περιλαμβάνουν πολλαπλά Widget, το καθένα αποτελεί τμήμα τους. Αυτά έχουν τη δυνατότητα να επεκτείνονται και να προσαρμόζονται πολύ εύκολα, με αποτέλεσμα οι προγραμματιστές να τα εκμεταλλεύονται όπως θέλουν. Βασικός στόχος του Widget είναι να υλοποιούνται με όσο το δυνατό λιγότερο κώδικα, να επεκτείνονται και να δομούνται σωστά. Ένα ακόμα χαρακτηριστικό τους είναι η συνδεσιμότητα ανάμεσα τους και η επίτευξη πολλών πολλαπλών συνδυασμών (Payne, 2019a).

3.5.4 Η ύπαρξη του Widget στις εφαρμογές

Έχει τονιστεί ότι οτιδήποτε περιλαμβάνει η εφαρμογή Flutter ανήκει στην κατηγορία των Widget. Έτσι αυτά μπορεί να αποτελούνται από εικονικά στοιχεία, εμφανή στους χρήστες, και από άλλα, τα οποία δεν είναι εμφανή. Πιο αναλυτικά ένα από τα πιο κύρια στοιχεία της εφαρμογής είναι τα κομμάτια του κειμένου ή τα κουμπιά, καθένα από τα οποία είναι Widget. Επιπλέον η διάταξή τους, το εάν δηλαδή αυτά θα δομούνται σε στήλες ή σε σειρές, αποτελεί ένα ακόμα Widget, το οποίο όμως δεν είναι αντιληπτό εκ μέρους των χρηστών. Στη συνέχεια η διαμόρφωση και ο σχεδιασμός κειμένου ή άλλων στοιχείων της εφαρμογές μαζί με τις ενέργειες που μπορούν οι χρήστες να κάνουν με το πάτημα ενός κουμπιού και γενικότερα οι αλληλεπιδράσεις με αυτούς αποτελούν ένα Widget. Τέλος κάθε Widget μέσα στην εφαρμογή αποτελεί τη δομή αυτής, έτσι ώστε για να μπορέσουν να οργανωθούν με σωστό τρόπο και να ομαδοποιηθούν σχηματίζοντας έτσι ένα Widget tree (Napoli, 2019).

3.5.4.1 Κατηγορίες Widget στις εφαρμογές

Ήδη τονίστηκε ότι υπάρχουν διάφορες κατηγορίες των Widget, συνολικά 160, αλλά για λόγους ευκολίας χωρίζονται σε τρεις κύριες: στα Widget τιμής, στα γραφικά στοιχεία διάταξης και στα γραφικά στοιχεία πλοήγησης.

Value widgets

Αυτά τα Widgets σχετίζονται με τη διατήρηση μιας συγκεκριμένης τιμής. Πιο αναλυτικά είναι εργαλεία που χρησιμοποιούνται για να φανερωθούν οι τιμές που προέρχονται είτε από ένα στάνταρ πλαίσιο υπηρεσιών, που υπάρχει στο διαδίκτυο, είτε από τους ίδιους τους χρήστες, οι οποίοι χρησιμοποιούν συγκεκριμένα εργαλεία για να κάνω λήψη των τιμών. Ένα από τα πιο κύρια Value widgets, τα οποία συναντώνται συχνά είναι το Text, μέσα από το οποίο μπορεί κάποιος να εμφανίσει τα κείμενα που θέλει ή να λάβει από τους χρήστες μία τιμή κειμένου όποτε απαιτείται στην εφαρμογή. Κάποια από τα άλλα Widget, τα οποία συναντώνται είναι τα ακόλουθα: Checkbox, CircularProgressIndicator, Date & Time, Pickers, DataTable, DropdownButton, FlatButton, FloatingActionButton, FlutterLogo, Form, FormField, Icon, IconButton, Image, LinearProgressIndicator, PopupMenuButton, Radio, RaisedButton RawImage, RefreshIndicator, RichText, Slider, Switch, Text, TextField και Tooltip (Zaccagnino, 2020).

Layout widgets

Στην κατηγορία των Layout widgets ανήκουν εργαλεία, τα οποία βοηθούν στη διαμόρφωση του περιβάλλοντος μιας εφαρμογής. Αναλυτικότερα τα εργαλεία αυτά βοηθούν τους Προγραμματιστές να έχουν τον έλεγχο των γραφικών που θα τοποθετήσουν πάνω σε κάθε περιβάλλον. Με αυτόν τον τρόπο οι ίδιοι θα έχουν τη δυνατότητα να φτιάξουν όπως έχουν φανταστεί τα γραφικά της εφαρμογής και να χρησιμοποιήσουν κάθε εργαλείο, που θα τους βοηθήσει να στοιχειοθετήσουν σωστά το περιβάλλον είτε κάθετα είτε οριζόντια. Ακόμα μπορούν να δημιουργήσουν και πολλά σχέδια για την διεπαφή χρήστη. Κάποια από αυτά τα εργαλεία είναι τα ακόλουθα: Align, AppBar, AspectRatio Baseline, BottomSheet, ButtonBar, Card, Center, Column, ConstrainedBox, Container CustomMultiChildLayout, Divider,

Expanded, ExpansionPanel, FittedBox, Flow, FractionallySizedBox, GridView, IndexedStack, IntrinsicHeight, IntrinsicWidth, LayoutBuilder LimitedBox, ListBody, ListTile, ListView, MediaQuery, NestedScrollview, OverflowBox, Padding, PageView, Placeholder, Row, Scaffold, Scrollable, Scrollbar, SingleChildScrollView, SizedBox, SizedOverflowBox, SliverAppBar, SnackBar, Stack, Table και Wrap (Payne, 2019a).

Navigation widgets

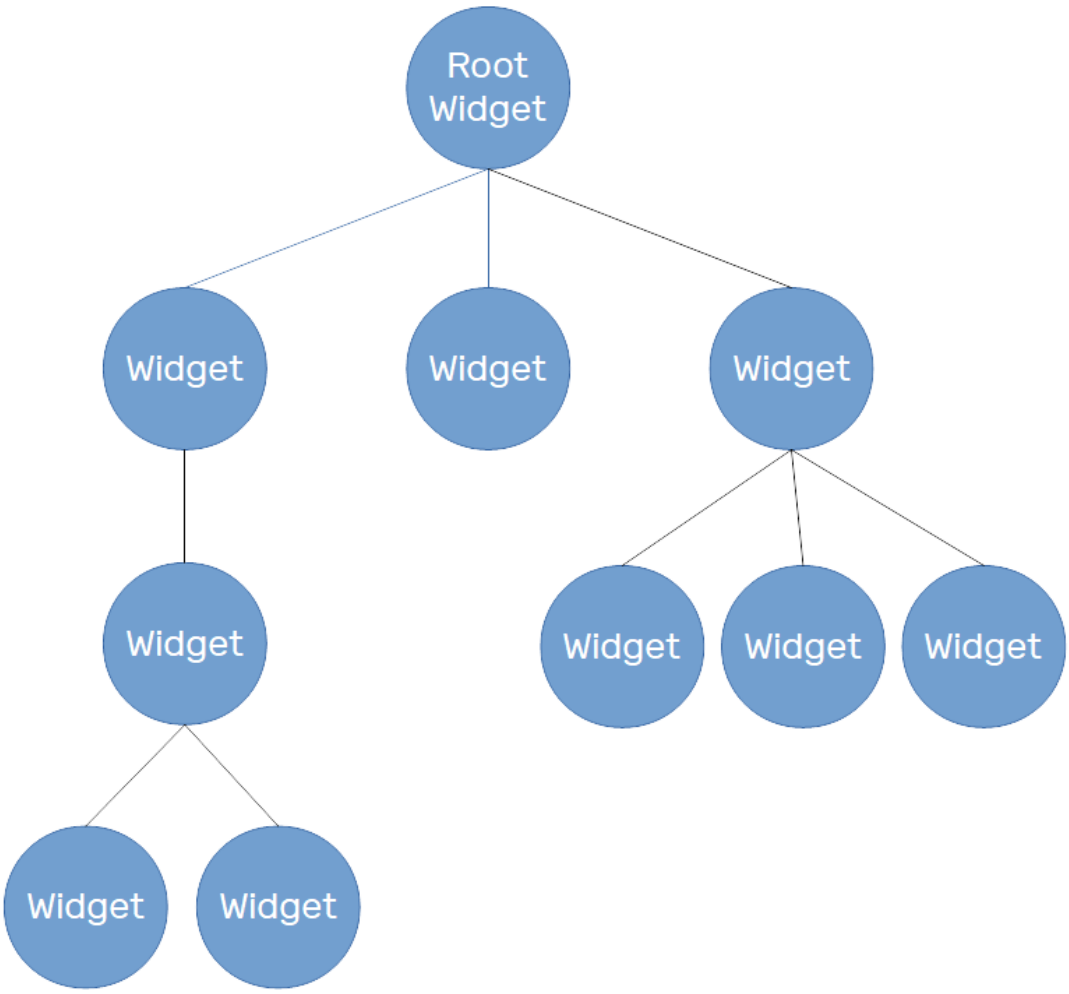
Η συγκεκριμένη κατηγορία βοηθά στην πλοήγηση των χρηστών στην εφαρμογή. Πιο συγκεκριμένα όταν οι εφαρμογές αποτελούνται από πολλές σελίδες, για τη διευκόλυνση των χρηστών, οι προγραμματιστές τοποθετούν ένα μενού, μέσα από το οποίο μπορούν πολύ πιο εύκολα να εξασφαλίσουν την πρόσβασή τους σε συγκεκριμένες σελίδες. Τα συγκεκριμένα εργαλεία περιλαμβάνουν είτε κουμπιά είτε γραμμές καρτελών είτε κάποιο είδος συρταριού, το οποίο γλιστρά από την αριστερή προς τη δεξιά πλευρά. Κάποια από αυτά τα εργαλεία τα εξής: AlertDialog, BottomNavigationBar, Drawer, MaterialApp, Navigator, SimpleDialog, TabBar και TabBarView (Payne, 2019a).

Other widgets

Στη συγκεκριμένη κατηγορία ανήκουν όλα τα Widgets, για τα οποία δεν είναι δυνατή η ένταξή τους σε κάποια από τις προηγούμενες κατηγορίες. Παρόλα αυτά η χρησιμότητά τους είναι μεγάλη. Κάποια από αυτά είναι τα εξής: GestureDetector, Dismissible, Cupertino, Theme, Transitions και Transforms (Payne, 2019b).

3.5.4.2 Widget tree

Το Widget tree αποτελεί μία πολύ σημαντική έννοια όσο αναφορά τη διάταξη για το Flutter. Σε αυτό διατάσσονται και αναπαριστώνται όλα τα γραφικά και μη στοιχεία και η διεπαφή χρήστη κάθε εφαρμογής. Τα γραφικά στοιχεία Widget έχουν την εικόνα ενός δέντρο και παρουσιάζουν κόμβους. Οποιαδήποτε μεταβολή μπορεί να καταγραφεί σε κάποιο Widget θα επηρεάσει όλο αυτό το δέντρο και όλους τους κόμβους. Η δομή του Widget tree δεν είναι σταθερή, αλλά καθορίζεται συνεχώς από τους κόμβους που προστίθενται ή αφαιρούνται. Διαδραματίζει λοιπόν πολύ καίριο ρόλο σε κάθε εφαρμογή Flutter, εφόσον βοηθά να χαρτογραφηθεί ένα δέντρο Widget στην οθόνη. Κάθε φορά που είναι επιθυμητή η ενημέρωση ενός Widget τότε θα πρέπει να καθοριστεί και ο τρόπος αναδημιουργίας του (Σχήμα 7) (Zaccagnino, 2020).



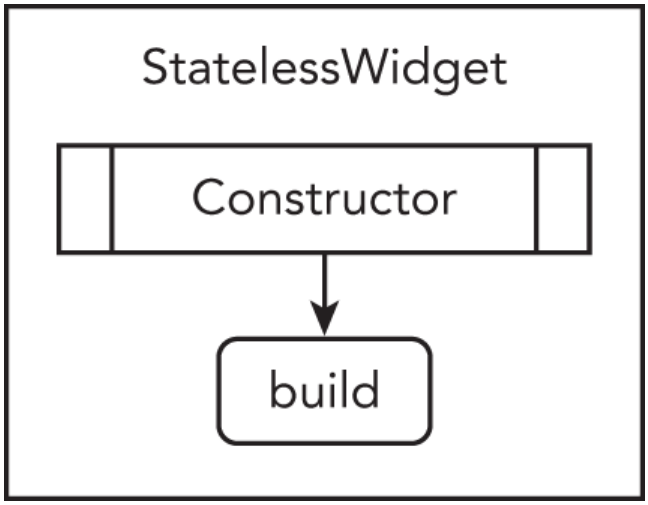
Σχήμα 7: Widget tree

3.5.4.3 Κύκλος ζωής ενός widget

Κατά τη διάρκεια του προγραμματισμού εμφανίζονται αρκετοί διαφορετικοί κύκλοι ζωής, τους οποίους θα πρέπει να μάθει κανείς για να μπορέσει να χρησιμοποιήσει όσο το δυνατό σωστότερα και αποτελεσματικότερα μία γλώσσα. Οι πιο πολλοί κύκλοι ζωής ανήκουν στον γραμμικό τύπο, κάτι που σημαίνει ότι ακολουθείται μία συγκεκριμένη σειρά από ενέργειες. Στη συγκεκριμένη ενότητα θα γίνει αναφορά του κύκλου ζωής του Widget. Για να γίνει δυνατή η δημιουργία μιας διεπαφής χρήστη στις εφαρμογές Flutter, χρησιμοποιούνται κυρίως δύο τύποι Widget: το Stateless Widget και το Stateful Widget. Το πρώτο χρησιμοποιείται κάθε φορά που οι τιμές στα πλαίσια της κλήσης του παραμένουν σταθερές, ενώ το δεύτερο για την αντίθετη περίπτωση. Επιπλέον είτε το το Stateless Widget και το Stateful Widget περιλαμβάνουν μία build μέθοδο με ένα BuildContext, μέσα από τα οποία καθορίζουν την τοποθεσία που έχει το κάθε Widget στο Widget tree. Στην ουσία πρόκειται για αντικείμενα element (Wee & Kea, 2021).

3.5.4.4 Stateless widget

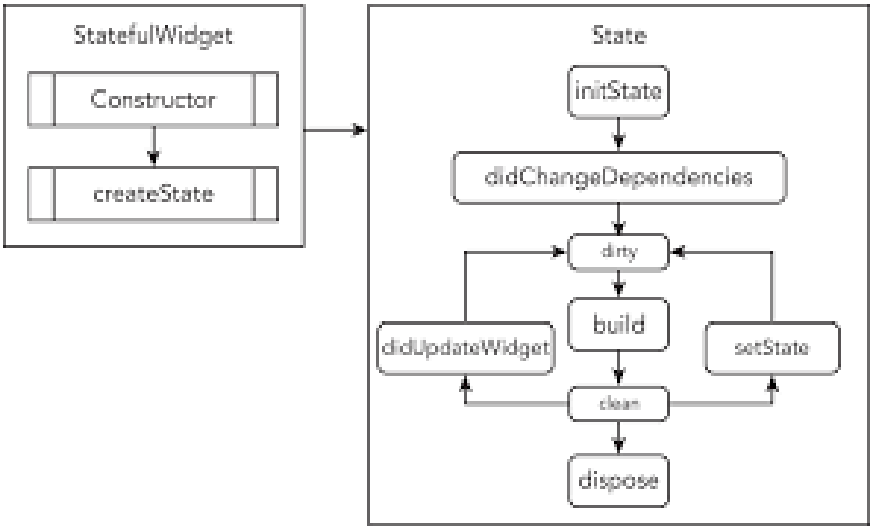
Βασικό χαρακτηριστικό του **Stateless Widget** είναι η σταθερότητα στη μορφή του. Αυτό σημαίνει ότι δεν παρατηρούνται δυναμικές αλλαγές μέσα στο Widget tree. Χαρακτηριστικό παράδειγμα είναι ο τίτλος, που θα μπορούσε να δοθεί σε μία λίστα. Σε αυτήν την περίπτωση η δομή και το περιεχόμενο είναι σταθερά. Ένα **Stateless Widget** μπορεί να καλεστεί με τρεις εναλλακτικούς τρόπους. Αρχικά καλείται κατά τη διάρκεια της πρώτης φοράς που δημιουργείται, στη συνέχεια κάθε φορά που αλλάζει ο γονέας του Widget και τέλος όταν αλλάζει ένα Ingretied Widget. Η κύρια δομή που συναντάται όταν δημιουργείται για πρώτη φορά ένα τέτοιο είδους Widget είναι η εξής: (Σχήμα 8).



Σχήμα 8: Stateless widget lifecycle

3.5.4.5 Statefull widget

Η δομή που έχει ένα τέτοιο Widget αναδιαμορφώνεται δυναμικά κάθε φορά που κάποιος το καλεί ή όταν αυτό καλείται μόνο του. Ένα τέτοιο παράδειγμα είναι ένα ρολόι, το οποίο εμφανίζεται στην οθόνη και αλλάζει διαρκώς τη μορφή του με την αλλαγή της ώρας. Αυτό σημαίνει ότι πρόκειται για ένα μεταβλητό Widget. Η δήλωση του γίνεται σύμφωνα με δύο κατηγορίες. Η πρώτη από αυτές είναι Stateful widget class και η δεύτερη η state class.. Όταν αλλάζει αυτό το Widget στην πρώτη κατηγορία παρατηρείται μία ολιστική αλλαγή σε αυτό, ενώ στη δεύτερη μένει σταθερό στο state, αλλά αλλάζει κάθε φορά που καλείται. Το state μπορεί να κλειστεί με τη χρήση της μεθόδου setState. Η δομή που υπάρχει όταν δημιουργείται πρώτη φορά ένα τέτοιο είδους Widget είναι η ακόλουθη (Σχήμα 9).



Σχήμα 9: Stateful widget lifecycle

Στη συνέχεια ακολουθεί ένας πίνακας, όπου παρουσιάζονται όλες οι τιμές που συναντώνται στον κύκλο ζωής του stateful widget. Πολλές εξ' αυτών δεν είναι ανάγκη να χρησιμοποιηθούν σε κάθε περίπτωση δημιουργίας αυτού του Widget.

Μέθοδοι	Περιγραφή	Κώδικάς

initState()	Καλείται μια φορά με το που εισάγεται το αντικείμενο εστο δέντρο.	@override
		void initState() {
		super.initState();
		print('initState');
		}

dispose()	Καλείται μόλις το αντικείμενο αφαιρεθεί μόνιμα από το δέντρο.	@override
		void dispose() {
		print('dispose');
		super.dispose();
		}

didChangeDependencies()	Καλείται μόλις το state αντικείμενο αλλάξει.	@override
		void didChangeDependencies() {
		super.didChangeDependencies();
		print('didChangeDependencies');
		}

didUpdateWidget(Contacts oldWidget)	Καλείται μόλις αλλάξει η διαμόρφωση του widget.	@override
		void didUpdateWidget(Contacts oldWidget) {
		super.didUpdateWidget(oldWidget);
		print('didUpdateWidget: $oldWidget');
		}

deactivate()	Καλείται μόλις αφαιρεθεί το αντικείμενο από το δέντρο.	@override
		void deactivate() {
		print('deactivate');
		super.deactivate();
		}

build(BuildContext context)	Καλείται πολλές φορές για να δημιουργηθεί η διεπαφή χρήστη και το BuildContext χειρίζεται τη θέση που έχει το γραφικό αυτό στοιχείο στο widget tree.	@override
		Widget build(BuildContext context) {
		print('build');
		return Container();
		}

setState()	Χρησιμοποιείται για να μεταβληθεί η τιμή.	setState(() {
		name = _newValue;
		});

Πίνακας 1: Μέθοδοι του Stateful Widget

3.5.5 Χρήστες του flutter

Οποιοσδήποτε θέλει να χρησιμοποιήσει την τεχνολογία και να μπορέσει να φτιάξει εύκολα και πολύ γρήγορα εφαρμογές, τότε σίγουρα θα προτιμήσει και το Flutter.

3.5.5.1 Εταιρείες και ομάδες

Το Flutter αποδεδειγμένα αυξάνει σημαντικά τη συνεργασία και την παραγωγικότητα εταιρειών και ομάδων. Κάθε φορά που μία εταιρεία θέλει να προσθέσει καινούργια χαρακτηριστικά στα προϊόντα της, τα οποία υλοποιούνται εκ μέρους διαφορετικών ομάδων, είναι δύσκολο να επικοινωνήσουν εξαιτίας των δεξιοτήτων που περιλαμβάνουν. Αυτό το ζήτημα λύθηκε εκ μέρους του Flutter. Οι ομάδες λοιπόν που χωρίστηκαν, ενώνονται σε μία και έτσι επικοινωνούν καλύτερα μεταξύ τους και αυξάνεται η παραγωγικότητα της εφαρμογής. Επιπλέον το αποτέλεσμα είναι πιο ολοκληρωμένο και η ανταπόκρηση πιο άμεση σχετικά με την ενημέρωση και με την προσθήκη κάποιου καινούργιου χαρακτηριστικού (Boukhary & Colmerates, 2019).

3.5.5.2 Μεμονωμένοι προγραμματιστές

Είναι πολλοί οι προγραμματιστές, που επιλέγουν το Flutter σαν τη βασική τους ενασχόληση. Ένας από τους βασικούς λόγους, για τον οποίο γίνεται αυτό, είναι το ότι αποτελεί μία πολύ γρήγορη γλώσσα, όσον αφορά στην εκμάθησή της. Επιπλέον πρόκειται για ένα ισχυρό εργαλείο, με το οποίο μπορεί κανείς να δομήσει εφαρμογές υψηλού επιπέδου. Για τους προγραμματιστές, που ασχολούνται με τεχνολογίες ανοικτού κώδικα, το Flutter περιλαμβάνεται μεταξύ των επιλογών τους. Επιπλέον όλο και περισσότερες δουλειές freelancer χρησιμοποιούν το Flutter και αυξάνεται διαρκώς (Cheng, 2019).

3.5.6 Διαδικασία εγκατάστασης του Flutter

Η διαδικασία της εγκατάστασης του Flutter είναι σχετικά απλή και αναλύεται ακολούθως. Στο λειτουργικό σύστημα των Windows θα πρέπει να κατεβάσει καταρχήν ο χρήστης το Flutter zip και στη συνέχεια να δημιουργήσει έναν φάκελο με το όνομα src στο root του δίσκου C:/, όπως είναι για παράδειγμα το src. Ακολούθως θα πρέπει να κάνει extract εκεί το zip που κατέβασε. Στη συνέχεια χρειάζεται να κατεβάσει και το Node.js και να το εγκαταστήσει. Τέλος θα ήταν καλό να κατεβάσει και το Git για τα Windows και να το εγκαταστήσει. Όταν πρόκειται για το λειτουργικό σύστημα Mac τότε κατεβάζει το αρχείο, κάνει το extract και το τοποθετεί στο σύστημα χρησιμοποιώντας το path $ export PATH=`pwd`/flutter/bin:$PATH. Ακολούθως τρέχει το $flutter doctor για να ελέγξει εάν έχει ολοκληρωθεί επιτυχώς η διαδικασία της εγκατάστασης, κάτι το οποίο γίνεται και στο λειτουργικό σύστημα των Windows. Και σε αυτό το λειτουργικό θα πρέπει να εγκαταστήσει τα links για το Node.js και Git για Mac (Zaccagnino, 2020).

Για να μπορέσει να δουλέψει κάποιος με το Flutter θα πρέπει να χρησιμοποιεί το Android Studio/IntelliJ είτε το Visual Studio (VS) code για Windows ή Mac αντίστοιχα. Αυτά τα IDEs είναι πιο αποτελεσματικά για τον προγραμματισμό των κινητών συσκευών. Στη συνέχεια θα πρέπει να εγκατασταθούν και τα plugins σύμφωνα με το IDE που χρησιμοποιήθηκε (Zaccagnino, 2020).

3.6 Επίλογος

Το κεφάλαιο αυτό ήταν πολύ σημαντικό για να κατανοηθεί ακριβώς η λειτουργία της γλώσσα προγραμματισμού, που χρησιμοποιείται εκ μέρους του Flutter. Επιπλέον έγινε φανερός και ο λόγος, για τον οποίο έχει επιλέξει αυτή τη γλώσσα σαν το κύριο εργαλείο του, ενώ κατανοήθηκε περισσότερο και η έννοια αυτού.

4 Κεφάλαιο: Υλοποίηση

4.1 Εισαγωγή

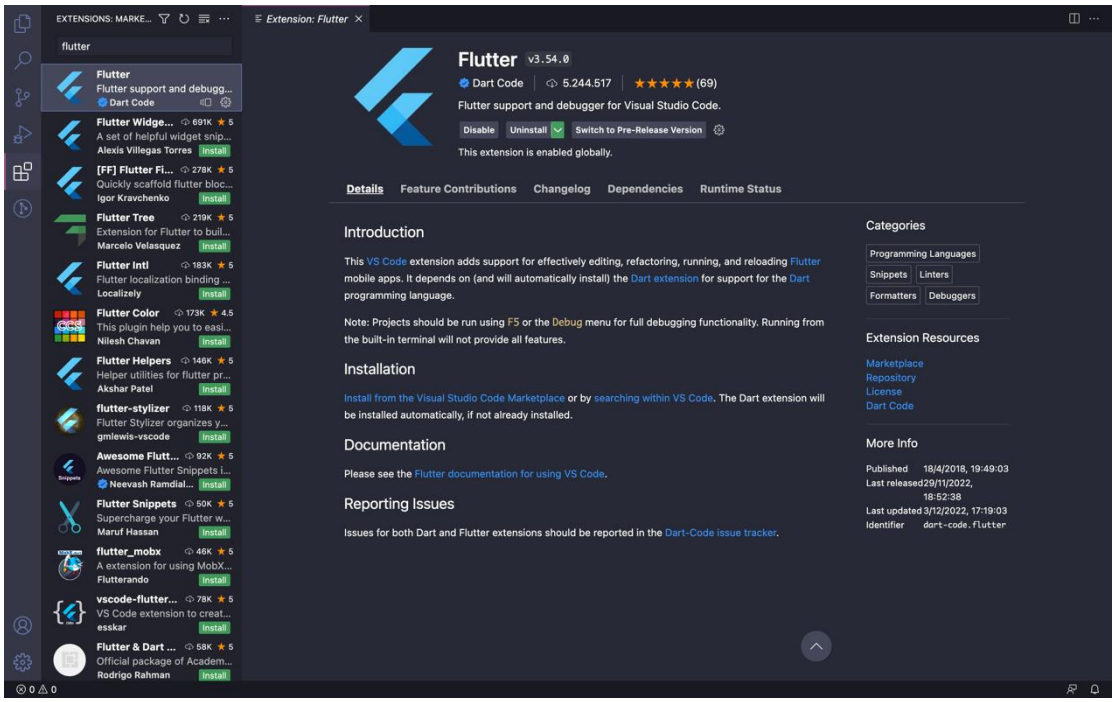
Στο παρακάτω κεφάλαιο θα αναφερθούμε στα τεχνικά κομμάτια της εργασίας. Πιο αναλυτικά, θα εξηγήσουμε τον τρόπο με τον οποίο μπορεί κανείς να δημιουργήσει ένα καινούργιο Flutter project. Επιπλέον, θα αναφερθούμε στις εξωτερικές βιβλιοθήκες που έχουν χρησιμοποιηθεί και θα αναλύσουμε κάποια βασικά σημεία του κώδικα και θα εξηγήσουμε την δομή και το σκεπτικό συγγραφής ενός Flutter Project. Τέλος, θα περιγράψουμε την βάση της εργασίας και θα εξηγήσουμε, το πως αυτά τα δεδομένα θα μπορέσουν να φτάσουν στο δική μας εφαρμογή ώστε να τα αξιοποιήσουμε και να τα εμφανίσουμε με τρόπο που εμείς θέλουμε στον δικό μας κώδικα.

4.2 Δημιουργία Flutter project

4.2.1 Αρχικοποίηση κώδικα του Project

Όπως έχουμε αναφέρει και σε προηγούμενο κεφάλαιο, η υλοποίηση της συγκεκριμένης εφαρμογής έγινε με το Flutter framework μέσω του περιβάλλοντος ανάπτυξης λογισμικού Visual Studio Code(VS Code) το οποίο κυκλοφόρησε για πρώτη φορά το 2015 από την Microsoft η οποία το δημιούργησε κιόλας. Ένα απαραίτητο εργαλείο που χρειαζόμαστε για την ανάπτυξη μιας εφαρμογής για iOS είναι το Xcode. Το Xcode είναι ένα περιβάλλον ανάπτυξης λογισμικού της Apple το οποίο χρησιμοποιείται για την δημιουργία εφαρμογών για macOS, iOS, iPadOS, watchOS, και tvOS. Το Xcode υποστηρίζει αρκετές γλώσσες προγραμματισμού όπως είναι οι C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), and Swift, όμως εμείς θα χρησιμοποιήσουμε το Visual Studio Code γιατί η γλώσσα προγραμματισμού που χρησιμοποιούμε με το Flutter υποστηρίζεται στο VS Code και διαθέτει πληθώρα επιλογών σε plugins έτσι ώστε να επιτύχουμε πιο γρήγορη ανάπτυξη του λογισμικού μας. Έτσι λοιπόν ο λόγος που χρειαζόμαστε το Xcode είναι για να μπορέσουμε να δημιουργήσουμε ένα iOS emulator, πιο συγκεκριμένα, ένα iPhone emulator για να μπορέσουμε να τρέξουμε και να δοκιμάσουμε την εφαρμογή, ή να συνδέσουμε μια συσκευή iPhone στον υπολογιστή μας. Σε όποια από τις δύο περιπτώσεις και αν επιλέξουμε χρειαζόμαστε έναν υπολογιστή με λειτουργικό iOS είτε για να συνδέσουμε την iPhone συσκευή μας είτε για να εγκαταστήσουμε το Xcode όπου μας επιτρέπει να δημιουργήσουμε

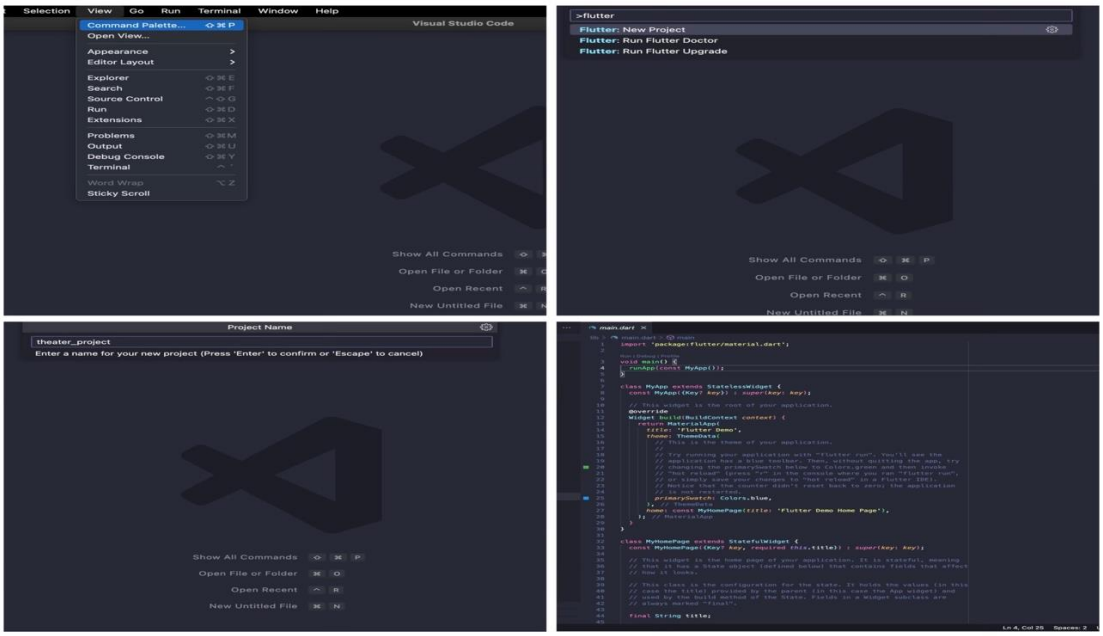
μια virtual iPhone συσκευή, διότι το Xcode δεν μπορεί να εγκατασταθεί και να λειτουργήσει σε υπολογιστή με λειτουργικό σύστημα διαφορετικό από macOS. Αντίθετα όμως αν διαθέτουμε έναν υπολογιστή με macOS λειτουργικό, και λόγο του ότι το Flutter είναι ένα cross-platform framework, αν θέλουμε να τρέξουμε να τρέξουμε την εφαρμογή σε μια συσκευή android μπορούμε να είτε να συνδέσουμε την συσκευή μας στον macOS υπολογιστή μας είτε να εγκαταστήσουμε το android studio και να δημιουργήσουμε ένα android emulator και να δούμε την εφαρμογή εκεί. Στην παρούσα εργασία έχουμε ακολουθήσει την επιλογή να χρησιμοποιήσουμε το Xcode ώστε να δημιουργήσουμε έναν iPhone emulator και να εγκαταστήσουμε εκεί και να προβάλουμε την εφαρμογή μας. Έτσι λοιπόν για να δημιουργήσουμε ένα project έχοντας τελειώσει την εγκατάσταση των flutter,Xcode,VS Code όπως αναφέρεται παραπάνω μπορούμε είτε με εντολές από τερματικό είτε μέσω του VS Code. Εμείς θα ακολουθήσουμε την επιλογή με το VS Code λόγο ευκολίας. Ξεκινώντας λοιπόν, ανοίγουμε το VS Code πατάμε στο αριστερό μενού τα extensions και κάνουμε αναζήτηση και εγκατάσταση το «flutter» extension,όπως βλέπουμε στην φωτογραφία παρακάτω, όπου το ένα από τα πιο σημαντικά πλεονεκτήματα που μας δίνει είναι να μπορούμε να εκτελέσουμε flutter commands μέσα από τον editor (Σχήμα 8).



Σχήμα 8: Εγκατάσταση του Flutter extension στο VS Code editor.

Μόλις εγκατασταθεί το flutter extension επιλέγουμε την επιλογή «View» και από το μενού που θα ανοίξει την επιλογή «Command Palette» και στο πλαίσιο που ανοίγει πληκτρολογούμε την λέξη «flutter» και μας δείχνει τις διαθέσιμες εντολές. Επιλέγουμε την επιλογή που λέει «New Project» και στο νέο μενού επιλέγουμε την επιλογή «Application». Ύστερα, μπροστά μας θα εμφανιστεί μια λίστα με τους φακέλους του συστήματός μας και διαλέγουμε το που θέλουμε να δημιουργήσουμε το νέο μας Project, το οποίο μετέπειτα του δίνουμε ένα όνομα της επιλογής μας. Τέλος, θα πρέπει να έχει εμφανιστεί μπροστά μας η οθόνη με τους βασικούς φακέλους

και αρχεία ενός Sample που είναι προεπιλογή του Flutter ως ένα αρχικό βασικό πλαίσιο(Σχήμα 9).



Σχήμα 9: Βήματα δημιουργίας βασικού Sample Project.

4.2.2 Δημιουργία iOS Emulator

Για να τρέξουμε λοιπόν ένα application σε έναν iOS emulator επιλέγουμε κάτω δεξιά στο παράθυρο του VS Code editor το κουμπί που λέει ≪No Device≫ και θα εμφανιστεί το παράθυρο με τις διαθέσιμες επιλογές. Διαλέγοντας λοιπόν την επιλογή ≪Start iOS Simulator≫ μια εφαρμογή διαχείρισης ψηφιακών συσκευών iOS όπου έχουμε την δυνατότητα να επιλέξουμε όποια συσκευή θέλουμε ανάμεσα σε μια πλήρη γκάμα iPhone, iPad αλλά και διάφορες εκδόσεις του λειτουργικού iOS. Αυτή η διαδικασία μπορεί να γίνει χωρίς την χρήση διαδικτύου, έχοντας προηγηθεί η εγκατάσταση του Xcode το οποίο μας προσφέρει την εφαρμογή των iOS emulator. Αντίστοιχα λοιπόν, θα εμφανιστεί και η επιλογή για android emulator εάν έχουμε εγκατεστημένο το android studio, ή αν θέλουμε να το τρέξουμε σε κάποιο περιηγητή μπορούμε να εγκαταστήσουμε το Google Chrome και έτσι θα μας εμφανίσει το ≪chrome≫ σαν επιλογή ώστε το application να τρέξει σε κάποιον web browser. Έτσι λοιπόν μπορούμε να αξιοποιήσουμε το cross-platform πλεονέκτημα που αναφέρθηκε παραπάνω και μας προσφέρει το Flutter Framework.

4.3 Επεξήγηση των βιβλιοθηκών στο pubspec.yaml

4.3.1 Εισαγωγή στο pub.dev

Όπως και σε πολλά άλλα framework, μπορούμε να χρησιμοποιήσουμε και διάφορες βιβλιοθήκες τρίτων (third party libraries) με τις οποίες στην ουσία επιτυγχάνουμε επαναχρησιμοποίηση κώδικα αυτών των τρίτων παρόχων, αλλά και να επιταχύνουμε την διαδικασία παραγωγής κώδικα με αρκετά μεγάλη αξιοπιστία αλλά και ευελιξία. Ένα πολύ σημαντικό προτέρημα του Flutter είναι ότι έχει την Google αλλά και μια αρκετά μεγάλη και συνεχώς αναπτυσσόμενη κοινωνία να το υποστηρίζει και να δημιουργεί νέες βιβλιοθήκες ώστε

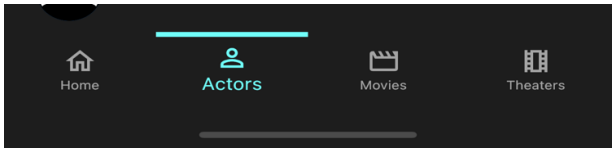
να κάνει το Flutter ένα πολύ ανταγωνιστικό εργαλείο στην αγορά. Αν όχι η πιο δημοφιλής, η επίσημη και πιο διαδεδομένη ιστοσελίδα παροχής τρίτων βιβλιοθηκών είναι η «https://pub.dev». Σε αυτή την ιστοσελίδα μπορούμε να βρούμε μια τεράστια γκάμα βιβλιοθηκών τρίτων, οι οποίες είναι ασφαλείς και έμπιστες. Ένα μεγάλο δίκτυο προγραμματιστών συντηρεί, αναβαθμίζει και προσθέτει συνεχώς νέες. Κάθε μια βιβλιοθήκη που υπάρχει με στο αποθετήριο μας προσφέρει οδηγίες για το πως να την χρησιμοποιήσουμε στο δικό μας project, αλλά και πληροφορίες για τις διαθέσιμες παραμέτρους που μα προσφέρει ώστε να μπορέσουμε να την αξιοποιήσουμε και να την προσαρμόσουμε στο σημείο του κώδικα που θέλουμε με το καλύτερο δυνατό τρόπο. Οι βιβλιοθήκες, ή αλλιώς «packages» όπως ονομάζονται στα αγγλικά, είναι απολύτως ασφαλείς ως προς την χρήση τους διότι δεν μπορούν να σβηστούν και έτσι να δημιουργηθεί πρόβλημα στην λειτουργεία εφαρμογών που τις χρησιμοποιούν.

4.3.2 Βιβλιοθήκες που χρησιμοποιήθηκαν

Στην συγκεκριμένη παράγραφο, θα αναφερθούμε στις βιβλιοθήκες που χρησιμοποιήσαμε για την υλοποίηση της συγκεκριμένης εφαρμογής και θα εξηγήσουμε την χρησιμότητα τους στο project.

- **flutter_snake_navigationbar**

Η βασικότερη λοιπόν βιβλιοθήκη που αποτελεί ουσιαστικά και την βάση της εφαρμογής είναι η «flutter_snake_navigationbar» η οποίο στην ουσία είναι μια όμορφη, με κινούμενα γραφικά στοιχεία, μπάρα περιήγησης όπου στην ουσία μας δίνει την δυνατότητα να κάνουμε πιο ξεκάθαρο αλλά και εντυπωσιακό στο χρήστη το πως μπορεί να αλλάξει οθόνες και να περιηγηθεί στο περιβάλλον της εφαρμογής μας καθώς επίσης και να καταλάβει τι είναι αυτό που έχει επιλέξει να δει. Επίσης με την χρήση των διαφορετικών χρωμάτων μεταξύ της επιλεγμένης καρτέλας και των διαθέσιμων επιλογών κάνει τον χρήστη να καταλαβαίνει ευκολότερα, εκτός από το τι βλέπει στην επιλεγμένη καρτέλα, αλλά και τι άλλες πιθανές καρτέλες του δίνει η εφαρμογή μας για προβολή όπως φαίνεται στην εικόνα παρακάτω (Σχήμα 10). Η συγκεκριμένη μπάρα περιήγησης μας προσφέρει αρκετές παραμέτρους για αλλαγή έτσι ώστε να κάνουμε την μπάρα με την μορφή που μας αρέσει αλλά και τα χρώματα που έχει επιλέξει η ομάδα μας. Ακόμα, εκτός από την επιλογή των χρωμάτων έχουμε την δυνατότητα να διαλέξουμε το σχήμα της μπάρας αλλά και το σχήμα και τον τρόπο του δείκτη που δείχνει την επιλεγμένη καρτέλα.



Σχήμα 10: Το snake bottom navigation bar της εφαρμογής.

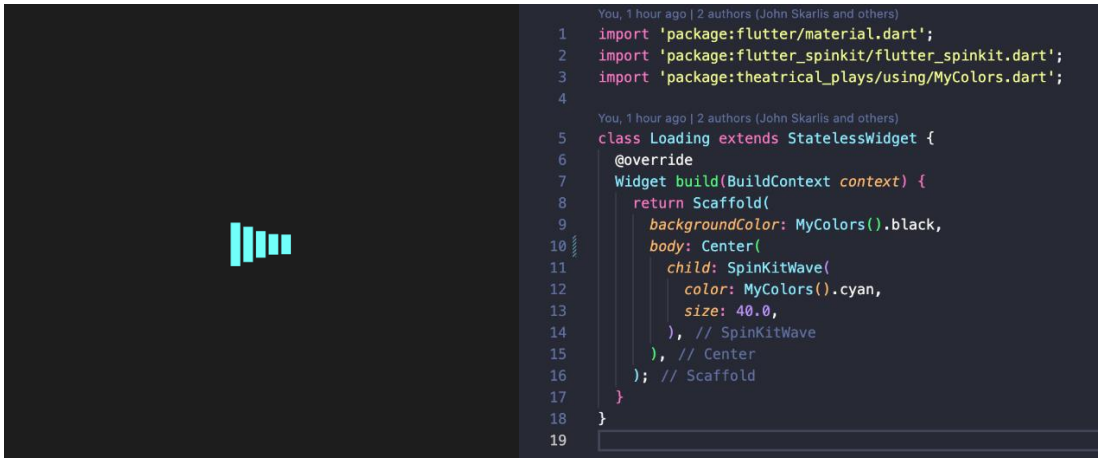
- **animated_splash_screen**

Αυτή η βιβλιοθήκη χρησιμοποιείται στην αρχική μας οθόνη για να μπορέσουμε να δώσουμε μια ελκυστική πρώτη εικόνα στο χρήστη με το όνομα της εφαρμογής και να δείξουμε το βασικό

μενού με μια όμορφη εναλλαγή. Στην ουσία είναι μια βιβλιοθήκη που μας προσφέρει ένα component το οποίο μπορούμε να τροποποιήσουμε σύμφωνα με το πως θέλουμε να κινείτε. Μπορούμε να δώσουμε στο widget ένα δικό μας image ή ακόμα να σχεδιάσουμε τι δικό μας widget πλήρως όπως θέλουμε. Επίσης μας δίνει την δυνατότητα να επιλέξουμε, μέσα από μια ικανοποιητική γκάμα, τον τρόπο με τον οποίο θέλουμε το εικονίδιό μας να κινηθεί στην οθόνη ώστε να δημιουργήσει ένα όμορφο εφέ καθώς και τον τρόπο με τον οποίο η οθόνη εισαγωγής θα αποσυρθεί για να έρθει στο προσκήνιο η αρχική μας οθόνη. Με πιο απλά λόγια μπορούμε να καθορίσουμε το εφέ κίνησης κάποιου στοιχείου στην οθόνη εισαγωγής και το εφέ μετάβασης από αυτή την εισαγωγική οθόνη σε μια πρώτη οθόνη της εφαρμογής μας.

- **flutter_spinkit**

Το «flutter_spinkit» είναι μια αρκετά απλή βιβλιοθήκη που μας προσφέρει κάποια κινούμενα components για να τα χρησιμοποιήσουμε ως μπάρα φόρτισης. Στην δική μας περίπτωση έχουμε φτιάξει μια ξεχωριστή dart κλάση, η οποία στην ουσία αποτελεί ένα αυτούσιο widget και το υλοποιούμε όπως μας αρέσει. Έτσι έχουμε την δυνατότητα να κάνουμε επαναχρησιμοποίηση του κώδικα σε κάθε σημείο της εφαρμογής που χρειαζόμαστε. Μπορούμε να τροποποιήσουμε την το σχήμα του ή το εικονίδιο που θα έχει η μπάρα καθώς επίσης το μέγεθος του το χρώμα του, ακόμα και την διάρκειά του. Παρακάτω στο σχήμα μπορούμε να δούμε την υλοποίηση και το γραφικό στοιχείο που έχουμε δημιουργήσει(Σχήμα 11).



Σχήμα 11: Γραφικό «wave» στοιχείο μπάρα φόρτισης και κώδικας υλοποίησης.

- **http**

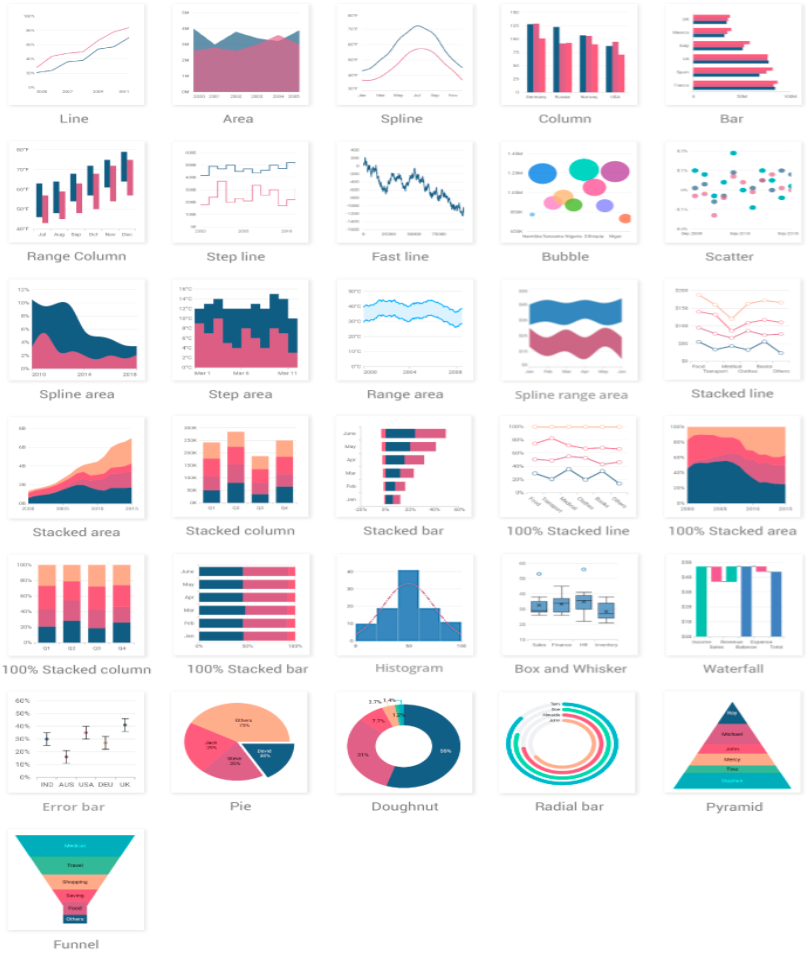
Όπως μπορούμε να εύκολα διακρίνουμε και από την ονομασία του συγκεκριμένου πακέτου, καταλαβαίνουμε ότι είναι μια βιβλιοθήκη που μα διευκολύνει να πραγματοποιήσουμε εύκολα, γρήγορα και απλά «http» αίτημα προς κάποιον άλλον εξυπηρετητή. Μας δίνεται η δυνατότητα να εκτελέσουμε πάνω από εννιά διαφορετικά αιτήματα μεταξύ άλλων και κάποια από τα πιο συχνά χρησιμοποιούμενα και σημαντικά για την υλοποίηση μιας εφαρμογής, όπως είναι τα delete, get, post, put και head.

- **maps_launcher**

Ένας από τους πιο σημαντικούς τομείς της παρούσας εφαρμογής που αναλύεται είναι τα θέατρα. Δουλεύοντας λοιπόν με τα θέατρα εμφανίζεται η ανάγκη να παρουσιάσουμε την πληροφορία της τοποθεσίας του κάθε θεάτρου σε ένα χάρτη. Η βιβλιοθήκη maps_launcher μας διευκολύνει στο να μπορέσουμε να ανοίξουμε τον χάρτη ενός iOS emulator και να παρουσιάσουμε στον χρήστη την τοποθεσία του θεάτρου που έχει επιλέξει. Χρησιμοποιώντας μια άλλη βιβλιοθήκη που ονομάζεται url_launcher εκκινεί μια αναζήτηση με βάση την τοποθεσία και ανάλογα την συσκευή που τρέχει η εφαρμογή ανοίγουν οι αντίστοιχοι χάρτες με και εμφανίζεται η τοποθεσία του θεάτρου που ψάχνουμε εάν υπάρχει η συγκεκριμένη πληροφορία στην βάση. Το συγκεκριμένο πακέτο μας παρέχει την δυνατότητα να δημιουργήσουμε ένα τέτοιο σύνδεσμο με μια τοποθεσία είτε στέλνοντας τις συντεταγμένες και το όνομα της περιοχής και να τον ανοίξουμε όποτε εμείς θέλουμε είτε μόνο με ένα όνομα περιοχής ή διεύθυνση. Αντίστοιχα με τις ίδιες παραμέτρους, μπορούμε να ανοίξουμε τους χάρτες όπως αναφέρθηκε και προηγουμένως, άμεσα χωρίς να χρειαστεί να αποθηκεύσουμε το σύνδεσμο και σε επόμενο βήμα να το ανοίξουμε.

- **syncfusion_flutter_charts**

Το πακέτο «syncfusion_flutter_charts» είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων γραμμένη εγγενώς στο Dart για τη δημιουργία όμορφων, κινούμενων γραφημάτων και υψηλής απόδοσης, τα οποία χρησιμοποιούνται για τη δημιουργία διεπαφών χρήστη εφαρμογών για κινητά υψηλής ποιότητας χρησιμοποιώντας το Flutter. Είναι ένα πακέτο που προσφέρει δεκάδες λύσεις, στους προγραμματιστές που χρησιμοποιούν το Flutter, όσον αφορά την στατιστική ανάλυση δεδομένων και την παρουσίαση αυτών σε διάφορα στατιστικά γραφήματα. Μπορούμε να δημιουργήσουμε διάφορους τύπους καρτεσιανών, κυκλικών και άλλων διαφόρων διαγραμμάτων τα οποία είναι διαδραστικά και έχουν ποικιλία κινούμενων εφέ. Έχει ένα πλούσιο σύνολο χαρακτηριστικών, ενώ είναι πλήρως προσαρμόσιμο και επεκτάσιμο. Για παράδειγμα κάποιες από τις παραμέτρους που μπορούμε να ορίσουμε είναι ο τύπος του γραφήματος, όπως είναι η πίτα το καρτεσιανό και πολλά άλλα. Επίσης μπορούμε να ορίσουμε χρώματα για την κάθε οντότητα για του γραφήματος αλλά και διάφορες άλλες επιλογές για μέγιστα σημεία στο γράφημα, ταμπέλες βοηθήματος, μέγεθος γραμματοσειράς και χρώμα γραμματοσειράς(Σχήμα 12).



Σχήμα 12: Διαγράμματα «syncfusion_flutter_charts»

4.4 Παρουσίαση και ανάλυση σημαντικών σημείων του κώδικα

Σε αυτή την παράγραφο θα παρουσιάσουμε και θα εξηγήσουμε κάποια σημαντικά σημεία στον κώδικα για να εξηγήσουμε κάποιες βασικές λειτουργικότητες της εφαρμογής. Όπως έχουμε ήδη εξηγήσει η βασική ιδέα που χρειάζεται κάποιος να καταλάβει για να προχωρήσει στην υλοποίηση μια εφαρμογής με το Flutter είναι ότι όλα τα components μια εφαρμογής θεωρούνται widgets. Το βασικότερο και πρωταρχικό κομμάτι της υλοποίησης είναι η λειτουργικότητα της κύριας μπάρας πλοήγησης της εφαρμογής η οποία χρησιμοποιεί έναν Page Controller το οποίο είναι ένα widget που μας προσφέρεται από το πακέτο Page View του Flutter και ένα PageView element το οποίο αποτελεί και τον βασικό container της εφαρμογής. Στην ουσία αυτά τα δύο widgets είναι υπεύθυνα για την εμφάνιση και εναλλαγή ανάλογα με το ποια από τις πιθανές καρτέλες έχει επιλέξει ο χρήστης. Έτσι με κάθε επιλογή που γίνεται σε κάποια από τις κεντρικές καρτέλες κρατάμε τον αριθμό id της καρτέλας που επιλέχθηκε και εμφανίζουμε την αντίστοιχη οθόνη στον χρήστη(Σχήμα 13).

```dart
final PageController controller = PageController(initialPage: 0);

//bottom navigation bar size colors an snake shape
bottomNavigationBar: SnakeNavigationBar.color(
  height: 60,
  backgroundColor: MyColors().black,
  snakeShape: snakeShape,
  snakeViewColor: MyColors().cyan,
  selectedItemColor: SnakeShape.indicator == SnakeShape.indicator
      ? MyColors().cyan
      : null,
  unselectedItemColor: Colors.white,
  showUnselectedLabels: true,
  showSelectedLabels: true,
  currentIndex: _selectedItemPosition,
  onTap: (index) {
    setState(() {
      _selectedItemPosition = index;
      controller.jumpToPage(index);
    });
  },
  items: [
    const BottomNavigationBarItem(
        icon: Icon(Icons.home_outlined), label: 'Home'), // BottomNavigationBarItem
    const BottomNavigationBarItem(
        icon: Icon(Icons.person_outline), label: 'Actors'), // BottomNavigationBarItem
    const BottomNavigationBarItem(
        icon: Icon(Icons.movie_outlined), label: 'Movies'), // BottomNavigationBarItem
    const BottomNavigationBarItem(
        icon: Icon(Icons.theaters_outlined), label: 'Theaters') // BottomNavigationBarItem
  ],
  selectedLabelStyle: const TextStyle(fontSize: 14),
  unselectedLabelStyle: const TextStyle(fontSize: 10),
), // SnakeNavigationBar.color
body: PageView(
    controller: controller,
    children: screens,
    scrollDirection: Axis.horizontal,
    onPageChanged: (index) {
      setState(() {
        _selectedItemPosition = index;
      });
    }), // PageView
); // Scaffold
}
```

Σχήμα 13: Δήλωση Page Controller και εμφάνιση οθονών με το Page View.

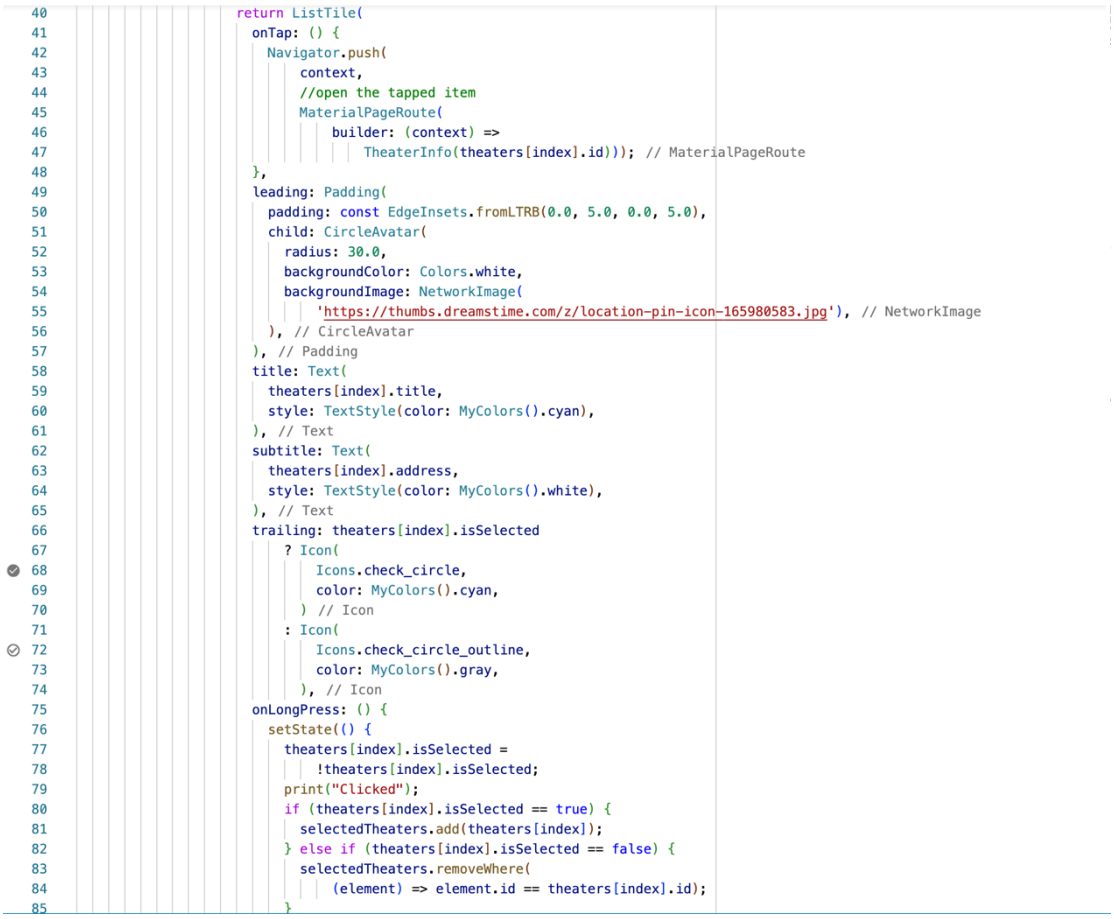
Ένα ακόμα σημαντικό σημείο του κώδικα, είναι το πως μια εφαρμογή μπορεί να πάρει δεδομένα για να τα παρουσιάσει στον χρήστη. Όπως και στις υπόλοιπες γλώσσες και frameworks για να πάρουμε δεδομένα εκτελούμε κάποια αιτήματα προς κάποιον σέρβερ για να μας δώσει δεδομένα. Στην δική μας περίπτωση εκτελούμε αιτήματα προς τον σέρβερ που έχει παραχωρήσει η σχολή μας στην ομάδα μας για την παρούσα εργασία, όπου βρίσκεται το το API που έχει υλοποιήσει η ομάδα μας για να κατευθύνει τα αιτήματα και μας επιστρέφει τα κατάλληλα δεδομένα από την βάση που έχουμε φτιάξει(Σχήμα 14). Η διεύθυνση που εκτελούμε τα αιτήματα είναι η «http://195.251.123.174:8080/api» και τα αποτελέσματα που μας επιστρέφονται βρίσκονται σε μορφή JSON. Το JSON είναι μια μορφοποίηση των δεδομένων όπου ένας πελάτης μπορεί να λάβει δεδομένα από τον εξυπηρετητή είτε να στείλει σε αυτόν. Μόλις πάρουμε αυτά τα δεδομένα που θέλουμε σε μορφή JSON τα μοντελοποιούμε, δηλαδή τα ορίζουμε σε κλάσεις-μοντέλα έτσι ώστε να μπορούμε να τα χρησιμοποιήσουμε με ευκολία στην εφαρμογή μας. Με την έννοια «κλάση- μοντέλο» εννοούμε μια κλάση στην οποία δηλώνουμε τόσα πεδία όσα μας επιστρέφει το JSON και μέσω του δομητή αρχικοποιούμε την κλάση και μπορούμε να την χρησιμοποιήσουμε μέσα στον κώδικα. Όπως αναφέρθηκε και πιο πάνω για να εκτελεστούν αυτά τα αιτήματα προς τον εξυπηρετητή χρησιμοποιείται η

εξωτερική βιβλιοθήκη «http» όπου μας διευκολύνει να εκτελέσουμε το αίτημα αυτό αλλά και διάφορα άλλα αιτήματα όπως είναι το post, put, delete και άλλα.

```dart
Future<List<Actor>> loadActors(String query) async {
  try {
    Uri uri = Uri.parse("http://195.251.123.174:8080/api/people");
    Response data = await get(uri, headers: {"Accept": "application/json"});
    if (data.statusCode == 200) {
      var jsonData = jsonDecode(data.body);

      for (var oldActor in jsonData['data']['content']) {
        if (oldActor['image'] == null || oldActor['image'] == '') {
          oldActor['image'] =
              'http://www.macunepimedium.com/wp-content/uploads/2019/04/male-icon.jpg';
        }
        Actor actor = new Actor(
            oldActor['image'], oldActor['id'], oldActor['fullName']);
        actors.add(actor);
      }

      return actors.where((actor) {
        final actorNameToLowerCase = actor.fullName.toLowerCase();
        final queryToLowerCase = query.toLowerCase();

        return actorNameToLowerCase.contains(queryToLowerCase);
      }).toList();
    } else {
      print("Api status code error");
    }
  } on Exception catch (e) {
    print('error data: $e');
  }
}
```

Σχήμα 14: Εκτέλεση αιτήματος για δεδομένα και μοντελοποίηση των αποτελεσμάτων.

Σε πολλά σημεία της εφαρμογής, όπως θα δούμε παρακάτω στην παρουσίαση της εφαρμογής στο επόμενο κεφάλαιο, χρησιμοποιούμε λίστες. Οι λίστες χρησιμοποιούνται για να παρουσιάσουμε πολλά δεδομένα με μια σύντομη μορφή παρουσιάζοντας της βασικές πληροφορίες κάθε αντικειμένου που θέλουμε να δείξουμε στον χρήστη. Έτσι καταφέρνουμε να εμφανίσουμε όλα τα αντικείμενα που μας ενδιαφέρουν χωρίς περιττές πληροφορίες έτσι ώστε να έχει την δυνατότητα ο χρήστης να αναζητήσει και να επιλέξει ένα ώστε να προβάλλει περισσότερες λεπτομέρειες για αυτό. Το κάθε ένα πλακίδιο(ListTile) που θα εμφανίζει της πληροφορίες του κάθε αντικειμένου μας, μπορούμε να ορίσουμε εμείς την δομή του. Στην παρούσα εργασία έχουμε επιλέξει με την ιδιότητα «leading» να χρησιμοποιήσουμε μια εικόνα του εκάστοτε αντικειμένου η οποία θα περιέχεται στο αίτημα που έχει εκτελεστεί και θα έρχεται από την βάση. Σε περίπτωση που δεν υπάρχει αποθηκευμένη στην βάση μια φωτογραφία για κάποιο αντικείμενο το έχουμε καθορίσει στον κώδικα να δίνεται μια προκαθορισμένη φωτογραφία γενικού περιεχομένου. Έχουμε ορίσει την ιδιότητα «title» ώστε να εκεί να δείξουμε το βασικό όνομα του αντικειμένου μας και την ιδιότητα «onTap» η οποία μας βοηθάει να καθορίσουμε στον κώδικα τι θα συμβαίνει όταν ένα από τα πλακίδια της λίστας επιλεχθεί με ένα απλό άγγιγμα. Ωστόσο λόγο της ανάγκης που υπάρχει για μια έξτρα λειτουργικότητα σε κάποιες από τις λίστες που έχουμε χρησιμοποιήσει προσθέσαμε την ιδιότητα «onLongPress». Με αυτή την ιδιότητα μπορούμε να ορίσουμε ακόμη μια διαδικασία να συμβεί σε περίπτωση που ο χρήστης επιλέξει παρατεταμένα ένα πλακίδιο. Έτσι λοιπόν σε κάθε παρατεταμένη επιλογή αποθηκεύουμε το αντίστοιχο αντικείμενο σε μια λίστα .Σε συνδυασμό με την επιλογή «trailing» όπου έχουμε ένα εικονίδιο ώστε σε κάθε παρατεταμένη επιλογή του χρήστη το αντίστοιχο εικονίδιο να παίρνει χρώμα ή να γίνεται άχρωμο αντίστοιχα σύμφωνα με το αν το αντικείμενο αυτό είναι στην λίστα ή όχι.



```
return ListTile(
  onTap: () {
    Navigator.push(
        context,
        //open the tapped item
        MaterialPageRoute(
            builder: (context) =>
                TheaterInfo(theaters[index].id))); // MaterialPageRoute
  },
  leading: Padding(
    padding: const EdgeInsets.fromLTRB(0.0, 5.0, 0.0, 5.0),
    child: CircleAvatar(
      radius: 30.0,
      backgroundColor: Colors.white,
      backgroundImage: NetworkImage(
          'https://thumbs.dreamstime.com/z/location-pin-icon-165980583.jpg'), // NetworkImage
    ), // CircleAvatar
  ), // Padding
  title: Text(
    theaters[index].title,
    style: TextStyle(color: MyColors().cyan),
  ), // Text
  subtitle: Text(
    theaters[index].address,
    style: TextStyle(color: MyColors().white),
  ), // Text
  trailing: theaters[index].isSelected
      ? Icon(
          Icons.check_circle,
          color: MyColors().cyan,
        ) // Icon
      : Icon(
          Icons.check_circle_outline,
          color: MyColors().gray,
        ), // Icon
  onLongPress: () {
    setState(() {
      theaters[index].isSelected =
          !theaters[index].isSelected;
      print("Clicked");
      if (theaters[index].isSelected == true) {
        selectedTheaters.add(theaters[index]);
      } else if (theaters[index].isSelected == false) {
        selectedTheaters.removeWhere(
            (element) => element.id == theaters[index].id);
      }
```

Σχημα 15: Δομή ενός ListTile και όλων των ιδιοτήτων που χρησιμοποιήθηκαν.

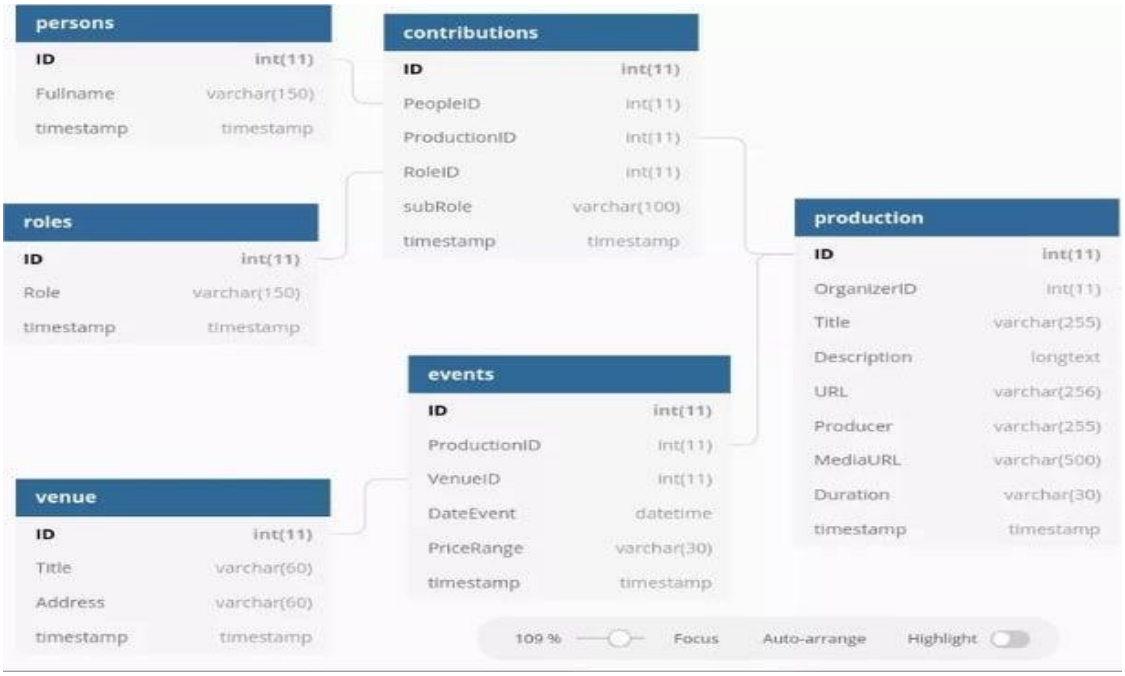
Αξιοσημείωτο κομμάτι του κώδικα που πρέπει να αναφέρουμε είναι η δημιουργία στατιστικών διαγραμμάτων που χρησιμοποιούνται σε κάποια σημεία της εφαρμογής, όπως θα παρουσιάσουμε παρακάτω στο επόμενο κεφάλαιο. Όπως αναφέραμε παραπάνω για την δημιουργία και την σύγκριση των στατιστικών γραφημάτων χρησιμοποιούμαι την βιβλιοθήκη «syncfusion_flutter_charts» η οποία έχει πολλές επιλογές. Για να δημιουργήσουμε ένα Widget γράφημα καλούμε την ανάλογη μέθοδο από την βιβλιοθήκη. Για να γίνει αυτή η κλήση χρησιμοποιούμε τα δύο αρχικά γράμματα «Sf»(Syncfusion), το είδος του γραφήματος που θέλουμε να χρησιμοποιήσουμε και στο τέλος την λέξη Chart. Για παράδειγμα τα δύο γραφήματα που χρησιμοποιήσαμε είναι δύο από τα δημοφιλέστερα το καρτεσιανό και το κυκλικό. Για παράδειγμα για να καλέσουμε αυτά τα δύο διαγράμματα καλέσαμε τις δύο μεθόδους «SfCartesianChart()» και «SfCircularChart()». Στην ιδιότητα «series» δηλώνουμε τον τύπο του γραφήματος που θέλουμε σύμφωνα με τη μέθοδο που έχουμε επιλέξει πιο πριν «CartesianChart» ή «CircularChart» και δηλώνουμε το αντικείμενο με τα δεδομένα που θα χρησιμοποιήσουμε. Στο xValueMapper και yValueMapper αντίστοιχα δηλώνουμε την τιμή που θέλουμε για τους δύο άξονες. Και τέλος, χρησιμοποιώντας το «dataLabelSettings» παραμετροποιούμε το πώς ακριβώς επιθυμούμε να φαίνεται το κείμενο με τις πληροφορίες του στρογγυλού διαγράμματος. Όσο αναφορά το καρτεσιανό διάγραμμα έχουμε προσθέσει μια ιδιότητα που λέγεται «tooltipBehavior» έτσι ώστε με κάθε άγγιγμα σε κάθε στήλη του εμφανίζεται ένα αναδυόμενο παράθυρο με πληροφορίες(Σχήμα 16).

Σχήμα 16: Παράδειγμα στατιστικού διαγράμματος(Καρτεσιανό).

Τέλος, πολύ σημαντικό κομμάτι για την εφαρμογή μας είναι η αναζήτηση που βρίσκεται και στις τρείς καρτέλες, των ηθοποιών, των παραστάσεων και των θεάτρων. Πιο συγκεκριμένα λοιπόν στην αναζήτηση μόλις πληκτρολογήσουμε τα τρία πρώτα γράμματα που θέλουμε ξεκινάει ένα ερώτημα προς την βάση το οποίο πρακτικά μας ανανεώνει τα αποτελέσματα και εφαρμόζεται επάνω σε αυτά η αναζήτηση μας. Έτσι κερδίζουμε την αξιοπιστία των αποτελεσμάτων διότι αν έχει προστεθεί εκείνη την στιγμή ή έχει αφαιρεθεί ένα αποτέλεσμα η αναζήτηση θα εφαρμοστεί πάνω στην ενημερωμένη λίστα. Το στοιχείο της αναζήτησης είναι ένα Widget που έχουμε δημιουργήσει εμείς και το επαναχρησιμοποιούμε σε κάθε μια από τις καρτέλες απλώς η διαδικασία της αναζήτησης εφαρμόζεται στα εκάστοτε δεδομένα που θέλουμε.

4.5 Επεξήγηση της βάσης δεδομένων

Η υλοποίηση της βάσης δεδομένων έγινε σε συνεργασία μελών της ομάδας μας σύμφωνα με απαιτήσεις και προδιαγραφές του συστήματος. Όσο αναφορά την υλοποίηση της εφαρμογής είναι ένα από τα πιο βασικά βήματα. Στην συνέχεια παρουσιάζεται μια εικόνα με το διάγραμμα ER της βάσης και μια επεξήγηση(Σχήμα 17).



Σχήμα 17: Παρουσίαση διαγράμματος ER για την βάση δεδομένων της εφαρμογής.

Ο βασικότερος πίνακας του σχήματος είναι ο πίνακας με τις παραστάσεις, ο οποίος στο σχήμα ονομάζεται «production», ο οποίος περιέχει της βασικές πληροφορίες για κάθε μια παράσταση όπως για παράδειγμα είναι το όνομα της, η περιγραφή της, η διάρκειά της, κάποιον σύνδεσμο για το τρέιλερ της παράστασης και άλλα. Κάθε μια παράσταση αποτελεί μια διοργάνωση όπου αποθηκεύονται στον πίνακα «events». Στον πίνακα events μπορούμε να δούμε πληροφορίες όπως είναι η τιμή του εισιτηρίου, το θέατρο που θα παιχτεί η παράσταση και άλλα. Συνεπώς αυτό μας οδηγεί σε ακόμη ένα βασικό πίνακα της εφαρμογής που είναι τα θέατρα και ο πίνακας στο διάγραμμα αναφέρεται ως «venue». Ο τρίτος σημαντικότερος πίνακας είναι αυτός που αναφέρεται στο ER σχήμα ως «persons». Ο πίνακας αυτός αναπαριστά τα δεδομένα της τρίτης βασικής οντότητας της εφαρμογής όπου είναι οι ηθοποιοί. Σε συνδυασμό με τον πίνακα «roles» και τον πίνακα «contributions» συντελείται η πληροφορία του κάθε ηθοποιού με το τι ρόλο συνεισφέρει σε κάθε παράσταση.

4.6 Επίλογος υλοποίησης της εφαρμογής

Συνοψίζοντας λοιπόν, για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το Visual Studio Code(VS Code) ως περιβάλλον ανάπτυξης κώδικα σε συνδυασμό με το Xcode, το οποίο και αυτό είναι περιβάλλον ανάπτυξης κώδικα σε γλώσσα Swift, από το οποίο αξιοποιούμε την δυνατότητα δημιουργίας iOS προσομοιωτή κινητής συσκευής και έτσι έχουμε τα δύο βασικά εργαλεία για ανάπτυξη κώδικα και εκτέλεση της εφαρμογής. Κατά την διάρκειας ανάπτυξης του κώδικα για δική μας ευκολία χρησιμοποιήσαμε κάποιες βιβλιοθήκες από την επίσημη σελίδα παροχής βιβλιοθηκών τρίτων παρώχων «https://pub.dev», τις οποίες τις δηλώνουμε όπως αναφερθήκαμε παραπάνω, στο pubspec.yaml αρχείο.

5 Κεφάλαιο: Παρουσίαση της εφαρμογής

5.1 Εισαγωγή στην παρουσίαση της εφαρμογής

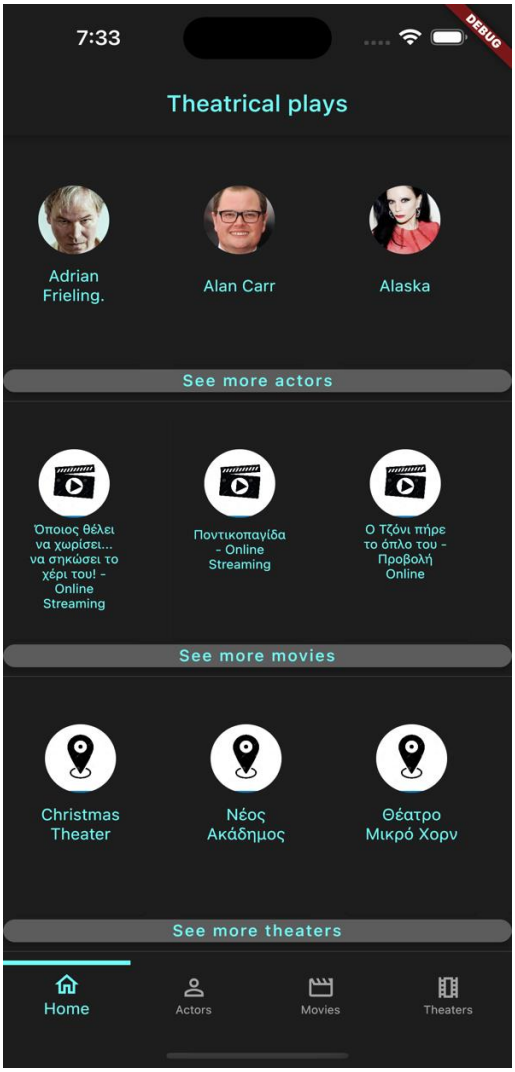
Ο τελικός στόχος που θέλαμε να επιτύχουμε με την υλοποίηση της παρούσας εφαρμογής θεατρικών παραστάσεων με ονομασία «theatrical analytics», είναι η ανάκτηση και η παρουσίαση πληροφοριών σχετικά με θεατρικές παραστάσεις, ηθοποιούς και θέατρα. Το βασικό μενού της εφαρμογής αποτελείται από 4 βασικές καρτέλες οι οποίες είναι ευκρινές μόλις ανοίξει η εφαρμογή στην κεντρική μπάρα πλοήγησης. Κάθε μια καρτέλα μας προσφέρει πληροφορίες για διαφορετική οντότητα και χρησιμοποιώντας την αναζήτηση μπορούμε να ψάξουμε κάτι συγκεκριμένο.

Η πρώτη και αρχική καρτέλα μας δείχνει τι πρόκειται να αντικρίσουμε στην συγκεκριμένη εφαρμογή και μας δίνει ένα μικρό δείγμα των τριών βασικών οντοτήτων που χρησιμοποιούμε στην εφαρμογή, των ηθοποιών, των θεατρικών παραστάσεων και των θεάτρων. Με αυτό τον τρόπο κάνουμε πιο οικεία την εφαρμογή στον χρήστη και τον πληροφορούμε καλύτερα για το περιεχόμενό της.

Οι τρείς καρτέλες είναι λίστες όπου η κάθε μια παρουσιάζει όλα τα αντικείμενα για τα θέατρα τις παραστάσεις και τους ηθοποιούς. Στην οθόνη των ηθοποιών μπορούμε να προβάλλουμε λεπτομέρειες για κάθε ηθοποιό καθώς επίσης και μια αναζήτηση για να αναζητήσουμε συγκεκριμένα κάτι από την λίστα. Στην οθόνη των παραστάσεων αντίστοιχα μπορούμε να προβάλλουμε τις λεπτομέρειες κάθε παράστασης και να αναζητήσουμε μια αλλά μπορούμε να επιλέξουμε και κάποιες για να συγκρίνουμε τις τιμές των εισιτηρίων τους όπου είναι εφικτό. Παρομοίως λοιπόν και στην τελευταία καρτέλα μπορούμε να επιλέξουμε από την λίστα κάποιο από τα θέατρα για λεπτομέρειες είτε να αναζητήσουμε κάποιο όπως και στις άλλες καρτέλες και μπορούμε να κάνουμε και εδώ σύγκριση μεταξύ των θεάτρων για το ποιο θέατρο έχει περισσότερες προβολές παραστάσεων. Επίσης μας δίνεται η δυνατότητα στην εφαρμογή να περιηγηθούμε μεταξύ των καρτελών, εκτός από το άγγιγμα στην κεντρική μπάρα πλοήγησης, σέρνωντας προς τα δεξιά ή αριστερά την κεντρική οθόνη έτσι ώστε η εφαρμογή να αναγνωρίσει την κίνηση του δακτύλου και να μετακινηθεί αντίστοιχα στην επόμενη ή προηγούμενη καρτέλα.

5.2 Οθόνη Home

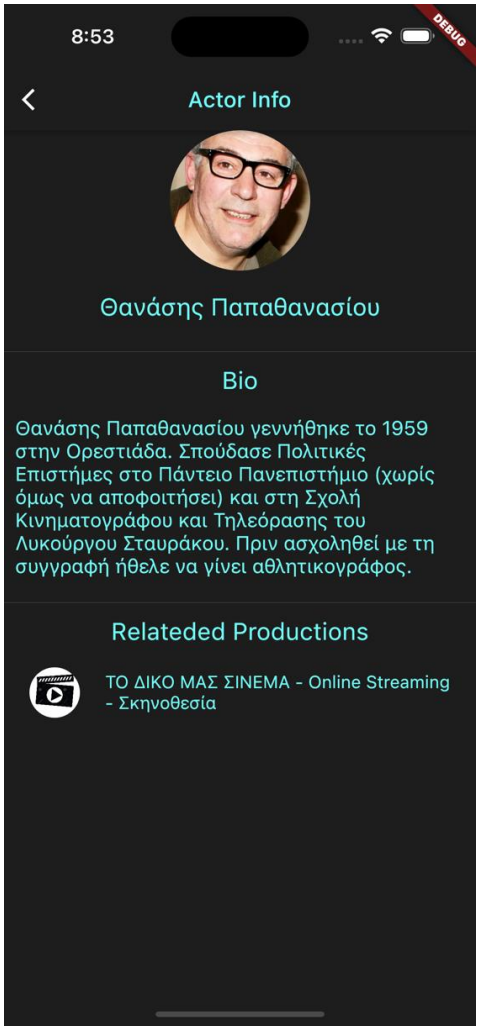
Στην αρχική οθόνη λοιπόν της εφαρμογής πιο συγκεκριμένα βλέπουμε τρία οριζόντια ListView και κάθε ένα από αυτά αντιπροσωπεύει μια από τις βασικές οντότητες τους ηθοποιούς, παραστάσεις και θέατρα. Στην κορυφή βρίσκεται η λίστα με τους ηθοποιούς την οποία ο χρήστης μπορεί να ανασύρει προς τα δεξιά και να δει και τους 6 ηθοποιούς ενδεικτικά που υπάρχουν και να επιλέξει όποιον θέλει ώστε να προβάλλει τις πληροφορίες του. Ακριβώς από κάτω από την συγκεκριμένη λίστα υπάρχει ένα κουμπί με την λέξη «more» το οποίο πατώντας το μπορεί να μας μεταφέρει στην καρτέλα τον ηθοποιών για να συνεχίσουμε εκεί την περιήγηση μας στις πληροφορίες των ηθοποιών. Αντίστοιχα από κάτω, με την ίδια λογική ακολουθεί και μια λίστα για τις παραστάσεις και μια για τα θέατρα.



Σχήμα 18: Αρχική οθόνη

5.3 Οθόνη Actors

Στην οθόνη των ηθοποιών λοιπόν συναντάμε, όπως ήδη αναφέρθηκε, την λίστα με τους ηθοποιούς και στο πάνω μέρος της οθόνης το πεδίο αναζήτησης. Πατώντας με το δάκτυλο μας σε κάποιο στοιχείο της λίστας θα μεταβούμε στην οθόνη με τις λεπτομέρειες του ηθοποιού. Στο πάνω μέρος μπορούμε να δούμε το όνομα του ηθοποιού και μια φωτογραφία του ή σε περίπτωση που δεν υπάρχει θα δούμε προεπιλεγμένη φωτογραφία του συστήματος. Ακριβώς από κάτω ακολουθεί μια σύντομη βιογραφία, ενώ αμέσως από κάτω ακολουθεί μια λίστα στην οποία απεικονίζονται οι παραστάσεις όπου κάθε ηθοποιός έχει συμμετέχει και με τι ρόλο ακριβώς.

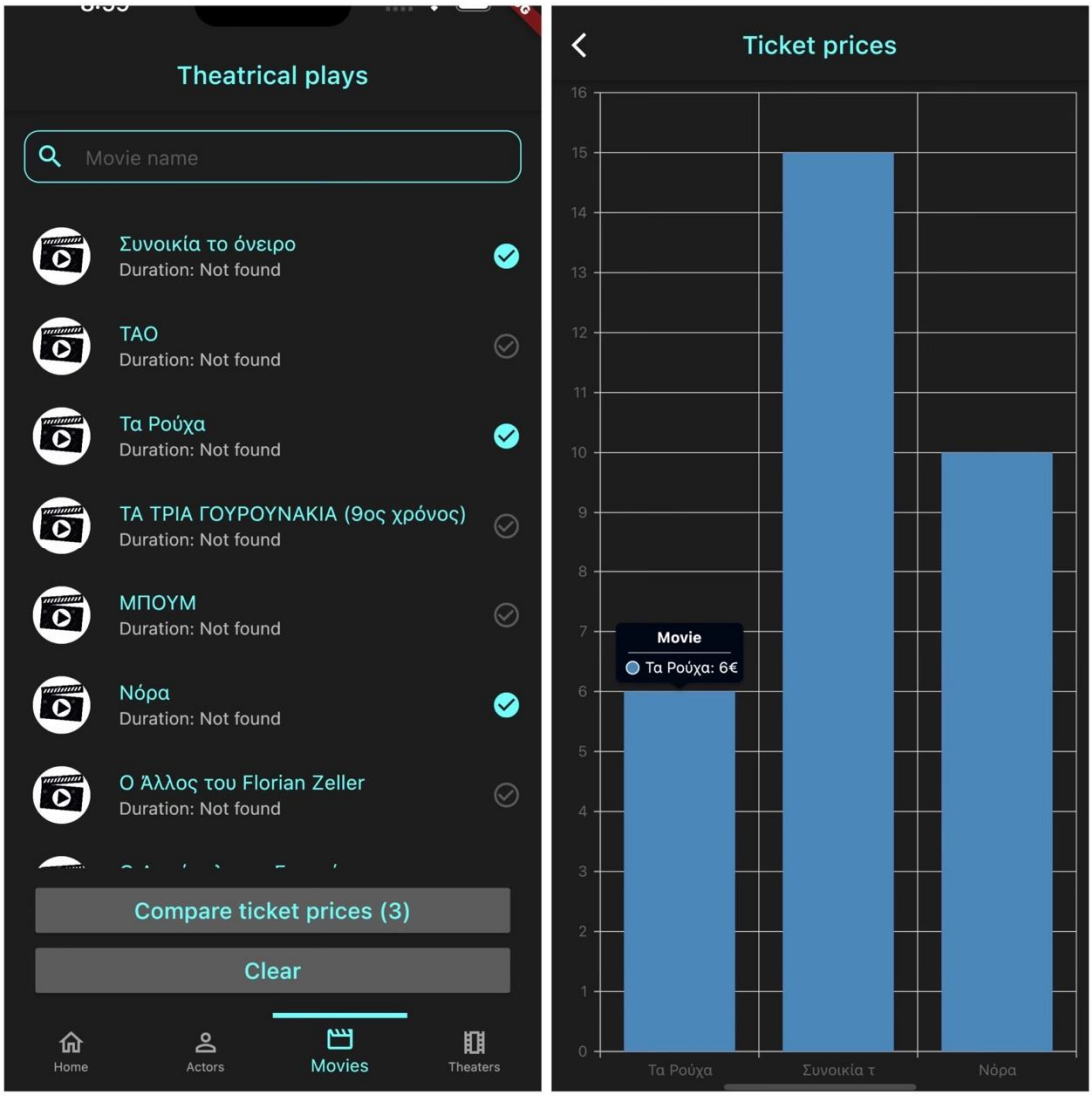


Σχήμα 19: Οθόνη λεπτομερειών ηθοποιού.

5.4 Οθόνη Movies

Στην τρίτη καρτέλα λοιπόν υπάρχει η οθόνη των παραστάσεων. Στην οθόνη αυτή το περιβάλλον μας είναι αρκετά γνώριμο διότι μοιάζει με την λίστα των ηθοποιών και όπως είναι προφανές έχει υλοποιηθεί με τον ίδιο τρόπο. Και εδώ λοιπόν χρησιμοποιείται μια λίστα φτιαγμένη με ListView component για την δημιουργία της λίστας και το ίδιο widget που χρησιμοποιήθηκε και στους ηθοποιούς για την αναζήτηση. Επιλέγοντας μια παράσταση η οθόνη των λεπτομερειών μιας παράστασης διαφέρει σε κάποια σημεία από την οθόνη λεπτομερειών των ηθοποιών. Και εδώ υπάρχει μια προκαθορισμένη φωτογραφία στο επάνω μέρος ενώ από κάτω υπάρχει το όνομα της παράστασης και αμέσως ακολουθεί μια περιγραφή. Επίσης στην συνέχεια υπάρχει το όνομα του παραγωγού της παράστασης και κουμπί που φέρει επάνω την λέξη «trailer». Το συγκεκριμένο κουμπί σε μεταφέρει στο Link που περιέχεται στην βάση για κάθε αντικείμενο. Στο κάτω μέρος της οθόνης εμφανίζεται και μία λίστα με τους ηθοποιούς που έχουν συμμετάσχει στην συγκεκριμένη παράσταση. Παρόλα αυτά στην δεξιά πλευρά κάθε πλακιδίου της λίστας μπορούμε να διακρίνουμε ένα χαρακτηριστικό σηματάκι το οποίο προσδιορίζει το αν το συγκεκριμένο στοιχείο της λίστας είναι επιλεγμένο για να σύγκριση ή όχι. Για να μας ανοίξει η επιλογή να προχωρήσουμε σε σύγκριση κάποιον παραστάσεων πρέπει να επιλέξουμε παρατεταμένα από την λίστα ένα αντικείμενο ώστε. Έτσι

μπορούμε να επιλέξουμε μέχρι τέσσερις παραστάσεις και να πιέσουμε το κουμπί «compare» για να κάνουμε μια σύγκριση των τιμών μεταξύ τους. Παρακάτω ακολουθεί μια φωτογραφία με την λίστα και την οθόνη σύγκρισης.



Σχήμα 20: Οθόνη λίστας ταινιών και οθόνη σύγκρισης τιμών.

5.5 Οθόνη Theaters

Στην τελευταία θέση του κεντρικού μενού πλοήγησης βρίσκεται η καρτέλα με τα θέατρα. Στην καρτέλα αυτή θα συναντήσουμε την ίδια δομή με την κεντρική λίστα και το πεδίο αναζήτησης όπως και στην καρτέλα των παραστάσεων που περιγράψαμε παραπάνω. Λειτουργεί ακριβώς με τον ίδιο τρόπο. Όσον αφορά την οθόνη με τις λεπτομέρειες κάθε θεάτρου η δομή των στοιχείων μοιάζει αρκετά με τις οθόνες λεπτομερειών των υπολοίπων οντοτήτων. Περιέχει πάνω πάνω μια εικόνα προκαθορισμένη με το όνομα και την διεύθυνση του θεάτρου και ακριβώς από κάτω ένα κουμπί το οποίο μας μεταφέρει στους χάρτες του κινητού τηλεφώνου εμφανίζοντας την τοποθεσία του θεάτρου. Τέλος, στο κάτω μέρος υπάρχει μια λίστα με τις θεατρικές παραστάσεις που παίζονται στο συγκεκριμένο θέατρο. Ακόμα, μας δίνεται και εδώ η δυνατότητα να πατήσουμε παρατεταμένα στοιχεία στην λίστα, μέχρι να ανάψει η ένδειξη

που το καθιστά ως «επιλεγμένο» δεξιά του αντικειμένου της λίστας, και να επιλέξουμε το κουμπί της σύγκρισης. Και εδώ η οθόνη που θα μεταβούμε επιλέγοντας το κουμπί «compare» είναι μια οθόνη σύγκρισης που γίνεται με την βοήθεια ενός γραφήματος σε μορφή πίτας(στρογγυλό). Έτσι τα θέατρα που έχουμε επιλέξει στην λίστα εμφανίζονται με διαφορετικό χρώμα με στο γράφημα και συγκρίνεται ο αριθμός παραστάσεων που παίζεται σε κάθε θέατρο. Ακριβώς από πάνω υπάρχει ένα υπόμνημα που δείχνει ποιο χρώμα ακριβώς αντιπροσωπεύει κάθε στοιχείο του γραφήματος.

Σχήμα 21: Οθόνη λεπτομερειών και οθόνη σύγκρισης θεάτρων.

6 Κεφάλαιο: Στάδια υλοποίησης της εφαρμογής

6.1 Απαιτήσεις

Η ανάγκη υλοποίησης της παρούσας εφαρμογής προήλθε από ένα συνδυασμό άλλων πτυχιακών εργασιών έτσι ώστε να αποτελέσει ένα κομμάτι από το συνολικό αποτέλεσμα της βασικής ιδέας. Ωστόσο λοιπόν πρόκειται για ένα αποτέλεσμα και που περιέχει το στοιχείο της ομαδικής δουλειά αλλά σίγουρα δεν αποτελεί μια εργασία πλήρως εξαρτημένη από άλλες. Η βασική ιδέα ήταν η υλοποίηση ενός πλήρους πληροφοριακού συστήματος όπου θα παίρνει δεδομένα από το internet, θα αποθηκεύονται σε μια βάση και στην συνέχεια αυτά τα δεδομένα θα διοχετεύονται σε τρία περιβάλλοντα χρηστών και πιο συγκεκριμένα σε ένα web application, σε ένα android application και σε ένα iOS application. Παρακάτω θα ασχοληθούμε και αναλύσουμε τα στάδια που ακολουθήθηκαν μέχρι την υλοποίηση και το τελικό αποτέλεσμα.

6.2 Scraping, δημιουργία βάσης δεδομένων και API

Το πρώτο και βασικότερο βήμα λοιπόν που έγινε για την της υλοποίησης της βασικής ιδέας είναι η διαδικασία του Scraping. Το Web Scraping, είναι μία μέθοδος, μέσω της οποίας μπορούμε να συλλέξουμε δεδομένα σε μια μη δομημένη μορφή, όπως για παράδειγμα HTML και να τα μεταφέρουμε σε δομημένη μορφή σε ένα αρχείο, όπως ένα φύλλο εργασίας του Excel.[1]

Έτσι λοιπόν το πρώτο βήμα ήταν η συλλογή δεδομένων θεατρικών παραστάσεων, ενώ το αμέσως επόμενο βήμα είναι η αποθήκευση των δεδομένων αυτών στην βάση που αναφέραμε στο κεφάλαιο 4.5. Για να μπορέσουμε να διοχετεύσουμε και στις τρείς εφαρμογές που πρέπει να υλοποιηθούν τα δεδομένα αυτά, κρίθηκε απαραίτητη από την ομάδα μας η ανάγκη δημιουργίας ενός API. Το API (Application Programming Interface) ή αλλιώς Διεπαφή Προγραμματισμού Εφαρμογών αναφέρεται στις εντολές (διεπαφή) των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, μια βιβλιοθήκη ή μια εφαρμογή έτσι ώστε να επιτρέπει από άλλα προγράμματα να κάνουν αιτήσεις προς αυτά για ανταλλαγή ή επεξεργασία δεδομένων.[2] Μια τέτοια διεπαφή δίνει τις λειτουργίες της σε άλλες εφαρμογές χωρίς αυτές να γνωρίζουν τον κώδικα ο οποίος εκτελείται. Με την χρήση λοιπόν του API πετυχαίνουμε την πλήρη επικοινωνία της βάσης μας με τις εφαρμογές παρουσίασης των δεδομένων που υλοποιήθηκαν. Έτσι λοιπόν μέσω της μορφής JSON (βλ. κεφάλαιο 4.4) και η Web εφαρμογή αλλά και οι άλλες δύο που είναι για mobile η android και η iOS ανταλλάζουν δεδομένα μέσω του API με την βάση της εφαρμογής. Κάποια από τα πρώτα βασικά end points του API που δημιουργήθηκαν μετά από συναντήσεις με τον διαχειριστή του API είναι τα παρακάτω:

- Με το id του ηθοποιού να εμφανίζονται οι παραστάσεις που έχει συμμετάσχει.
- Να εμφανίζονται οι πιο πρόσφατες ημερολογιακά παραστάσεις
- Με το id του ηθοποιού να υπάρχει η λίστα των φωτογραφιών του
- Με το id της παράστασης να έρχονται οι διοργανώσεις που έγιναν με αυτήν και σε ποια θέατρα
- Όταν είναι γνωστό το id παράστασης, ηθοποιού ή θεάτρου να έρχονται πληροφορίες μόνο για αυτό το id

Έτσι λοιπόν μετά την δημιουργία των end points μπορούσαμε και οι τρείς που υλοποιούσαμε ο κάθε ένας μια διαφορετική εφαρμογή να έχουμε πρόσβαση στα δεδομένα.

6.3 Σχεδιασμός οθονών

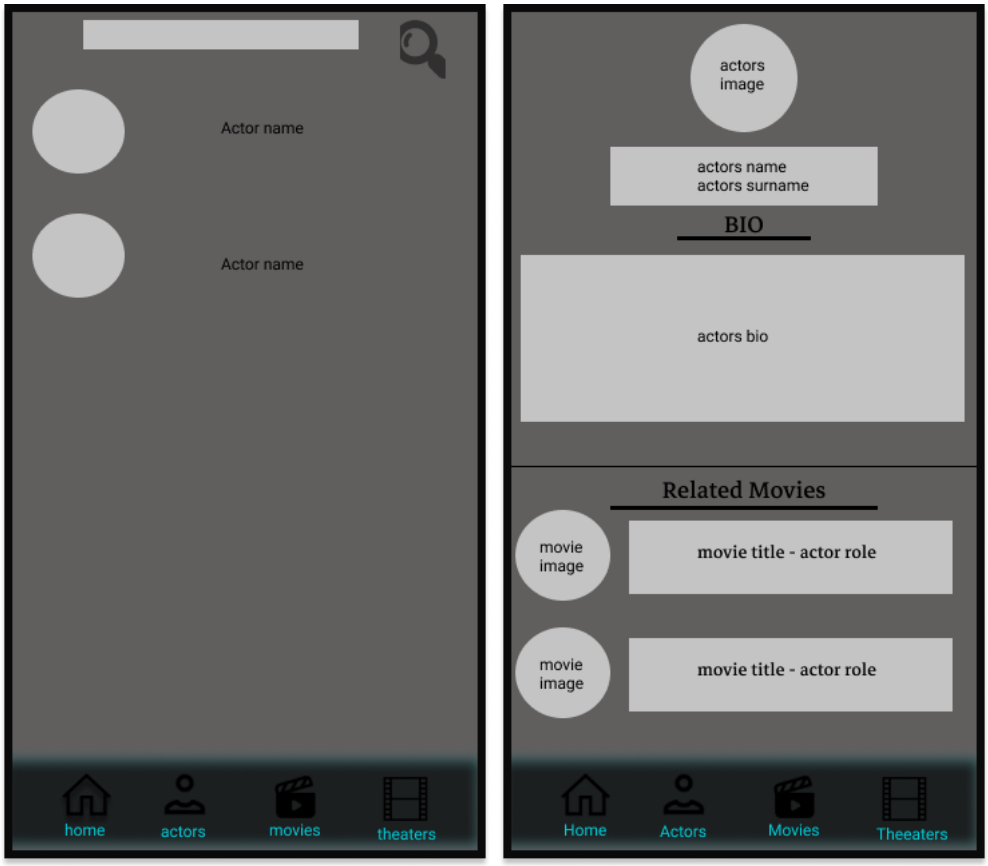
Ο σχεδιασμός οθονών είναι ένα από τα σημαντικότερα βήματα της ομάδας για να μπορέσει να συντονιστεί η δουλειά των προγραμματιστών που ασχολούνταν με την υλοποίηση των εφαρμογών που θα παρουσιάζουν τα δεδομένα. Σε πρώτη φάση υπήρξαν συναντήσεις μεταξύ και των τριών προγραμματιστών για την επιλογή των χρωμάτων αρχικά της εφαρμογής όπου σαν βάση επιλέχθηκε μια απόχρωση του μαύρου από έναν color picker το οποίο σε RGB code

[1] https://bigblue.academy/gr/ti-einai-to-web-scraping

[2] https://www.ip.gr/el/dictionary/378-API__Application_Programming_Interface?gclid=EAIaIQobChMIvfS0oKSz_AIVUAGLCh3wyAQ2EAAYASAAEgJeV_D_BwE

είναι RGB(29, 29, 29) και τα γράμματα που θα χρησιμοποιούνται επιλέχθηκε μια απόχρωση του γαλάζιου με RGB(113, 255, 250).

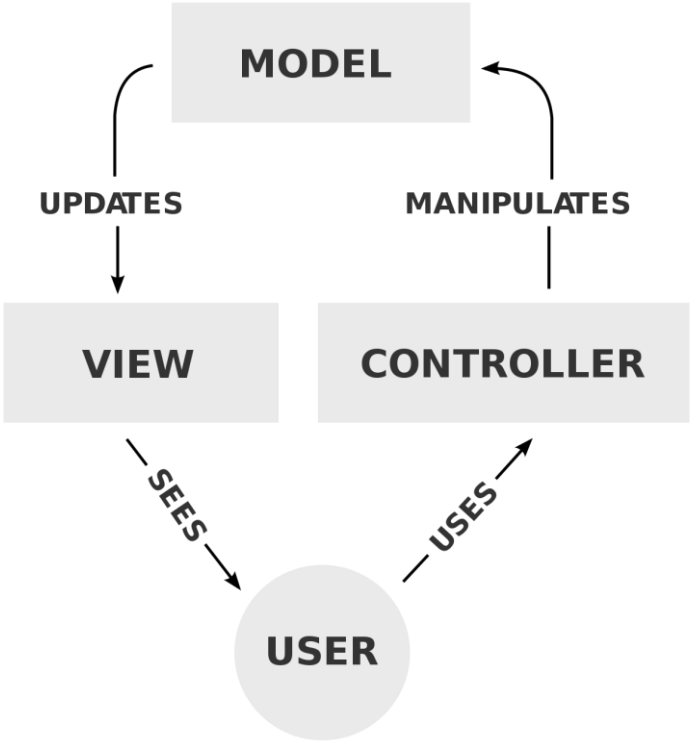
Στην συνέχεια λόγο της ομοιογένειας που έπρεπε να υπάρχει μεταξύ των δύο mobile εφαρμογών γίνανε κάποιες συναντήσεις μεταξύ των δύο προγραμματιστών για να μπορέσει να οριστεί το περιβάλλον των δύο εφαρμογών να είναι κοινό διότι παρουσιάζουν τα ίδια δεδομένα για διαφορετικό λειτουργικό. Έτσι λοιπόν προέκυψε η ανάγκη για σχεδίαση κάποιον οθονών ώστε ο στόχος που θέλαμε να πετύχουμε στην υλοποίηση της iOS εφαρμογής, όπου παρουσιάζουμε την παρούσα εργασία, και της android εφαρμογής. Υπάρχουν αρκετά εργαλεία που μπορούν να εξυπηρετήσουν τον σκοπό αυτό, παρόλα αυτά στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε ένα online που ονομάζεται Figma. Το Figma λοιπόν μας βοήθησε να σχεδιάσουμε κάποια ψηφιακές οθόνες έτσι να κάνουμε πιο κατανοητό το UI (user interface) που θέλουμε να πετύχουμε. Το βασικότερο πλεονέκτημα που μας προσφέρει είναι η συνεργασία. Αυτό επιτυγχάνεται κάνοντας share την αρχική δουλειά που ξεκινάει κάποιος έτσι ώστε να δώσει δικαίωμα προβολής ή και συγγραφής σε άλλους χρήστες. Εκτός από την ευκολία στην συνεργασία που κερδίζουμε με το εργαλείο αυτό, μπορούμε να επιλέξουμε κάποιες οθόνες ανάμεσα σε πληθώρα επιλογών όπου το καθιστά πιο αποδοτικό ως προς το έργο του σχεδιασμού οθονών. Παρακάτω υπάρχει ένας πρώτος σχεδιασμός της οθόνης των ηθοποιών και της αρχικής οθόνης με την χρήση του Figma.



Σχήμα 22: Παράδειγμα οθονών σχεδιασμένων με Figma

6.4 Συγγραφή κώδικα

Τελευταίο στάδιο λοιπόν, εφόσον έχουμε ολοκληρώσει την σχεδίαση των οθονών και έχουμε έτοιμα κάποια end points από το API έτσι ώστε να έχουμε δεδομένα, είμαστε έτοιμοι να προετοιμάσουμε και να ξεκινήσουμε την συγγραφή κώδικα. Για την συγγραφή του κώδικα επιλέξαμε να χρησιμοποιήσουμε ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη και ονομάζεται Model-View-Controller όπου σαν συντομογραφία το αποκαλούμε MVC. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήστη από την μορφή που έχει αποθηκευτεί στο σύστημα. Το κύριο μέρος του μοντέλου είναι το αντικείμενο Model το οποίο διαχειρίζεται την ανάκτηση/αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο View χρησιμοποιείται μόνο για να παρουσιάζεται η πληροφορία στον χρήστη (π.χ. με γραφικό τρόπο). Το τρίτο μέρος είναι ο Controller ο οποίος δέχεται την είσοδο και στέλνει εντολές στο αντικείμενο Model και στο View(Wikipedia.MVC). Έτσι λοιπόν και στην παρούσα υλοποίηση αντίστοιχα, κάθε μια οθόνη αποτελείται από μια ή και δύο Dart κλάσεις που εκτελούν το ρόλο του View σύμφωνα με το μοντέλο που ακολουθούμε. Για τις λειτουργίες που παρουσιάζονται στα Views χρησιμοποιούνται κάποιοι Controllers οι οποίοι είναι υπεύθυνοι, χρησιμοποιώντας τις κλάσεις Model, να εκτελέσουν τα ανάλογα αιτήματα στην βάση, να πάρουν τα δεδομένα στην μορφή JSON, να τα μοντελοποιήσουν στις κλάσεις Model και να στείλουν στις κλάσεις View όπου παρουσιάζονται με όμορφο γραφικό τρόπο στον χρήστη. Αντίστοιχα οποιαδήποτε αλληλεπίδραση του χρήστη με την εφαρμογή μας το κάθε View καλεί τον κατάλληλο Controller για να διαχειριστεί την δράση αυτή.



Σχήμα 23: Αναπαράσταση MVC μοντέλου αρχιτεκτονικής λογισμικού

7 Κεφάλαιο: Συμπέρασμα και βελτιώσεις

7.1 Προτάσεις βελτίωσης της εφαρμογής

Παρά την δουλειά που έχει γίνει από όλα τα μέλη της ομάδας θα μπορούσαν να γίνουν ατελείωτες βελτιώσεις στην εμφάνιση και στην απόδοση καθώς και κάποιες προσθήκες στις λειτουργίες της εφαρμογής, όπως άλλωστε συμβαίνει σε όλες της εφαρμογές να έχουν συνεχή ενημέρωση με νέες λειτουργίες αλλά και βελτιστοποίηση των ήδη υπάρχων. Στην παρακάτω παράγραφο θα αναφερθούμε και θα εξηγήσουμε κάποιες από αυτές.

Μια πολύ σημαντική και χρήσιμη βελτιστοποίηση της εφαρμογής θα ήταν να προστεθεί μία οθόνη ημερολόγιο ώστε να εμφανίζει τις παραστάσεις που είναι πιο κοντά στην σημερινή ημερομηνία του χρήστη καθώς επίσης να χρησιμοποιηθεί και η τοποθεσία του, έτσι ώστε να μπορέσει να ψάξει πληροφορίες για αυτή και να πάει στο κοντινότερο θέατρο της περιοχής του. Ακόμη μια βελτίωση που θα μπορούσε να γίνει, είναι η βελτίωση του logo της εφαρμογής σε συνδυασμό με την προσθήκη μιας πιο animated οθόνης στην εκκίνηση της εφαρμογής. Τέλος μια ακόμη αλλαγή που θα μπορούσε να μας προσφέρει περισσότερα όσο αναφορά την απόδοση της εφαρμογής είναι η βελτίωση του τρόπου αναζήτησης. Σε αντίθεση με τον τρόπο που είναι η υλοποιημένη θα μπορούσαμε με ένα μικρό κόστος αξιοπιστίας να βελτιώσουμε την ταχύτητα της αναζήτησης κρατώντας ένα στιγμιότυπο της λίστας που έχει ήδη φορτώσει από τον σέρβερ και να εκτελέσουμε την αναζήτηση σε επίπεδο εφαρμογής. Αυτό θα ανέβαζε την ταχύτητα όμως σε μια αλλαγή δεδομένων ταυτόχρονα ανάμεσα στην διαδικασία δύο αναζητήσεων αυτή η αλλαγή δεδομένων δεν θα φαινόταν. Παρόλα αυτά, η αλλαγή αυτή θα μας βοηθούσε εάν ο στόχος μας ήταν η ταχύτητα της αναζήτησης.

7.2 Συμπεράσματα

Συνοψίζοντας θα λέγαμε ό,τι τα οφέλη και οι γνώσεις που αποκομίσαμε από την διαδικασία της εκπόνησης της παρούσας εργασίας είναι πολλές. Όσον αναφορά το Flutter είναι ένα εργαλείο που απαιτεί στην αρχή να επενδύσει κάποιος λίγο χρόνο για να καταλάβει το τρόπο λειτουργίας του και την δομή του ώστε να μπορεί να συγγράψει κώδικα, όμως σίγουρα προσφέρει αρκετά μεγάλο πλεονέκτημα στην ταχύτητα και παραγωγικότητα ενός πολύ καλού και σύγχρονου User Interface. Ακόμα η τεράστια γκάμα εξωτερικών βιβλιοθηκών, που χρησιμοποιούνται και συνεχώς αναβαθμίζονται, μας προσφέρουν ευκολία σε πολλές συνηθισμένες προγραμματιστικές μεθόδους που χρειαζόμαστε για την υλοποίηση κάποιας εφαρμογής. Παρά το γεγονός τα συνεχούς ενημέρωσης των εξωτερικών βιβλιοθηκών που μας έφερε αρκετές φορές σε μια κατάσταση να αντιμετωπίσουμε μεθόδους που καταργούνταν, το Flutter είναι από τα καλύτερα και πιο καινοτόμα εργαλεία της αγοράς με συνεχούς ανάπτυξη και εφαρμογή σε πολλούς τομείς.

Αξίζει να σημειωθεί πως ένα από τα σημαντικότερα οφέλη που μας προσέφερε το project είναι συνεργασία. Το κάθε μέλος της ομάδας προσέφερε με των δικό του τρόπο και καταφέραμε να δημιουργήσουμε ένα αποτέλεσμα ομαδικής αλλά και ατομικής προσπάθειας. Επίσης πολύ

σημαντική ήταν και η χρήση διαφόρων εργαλείων που αναφέρθηκαν παραπάνω για την ταχύτερη και πιο αποτελεσματική απόδοση της ομάδας, διαχωρίζοντας τα σημεία όπου καταβλήθηκε ομαδική δουλειά αλλά και τονίζοντας την προσωπική δουλεία του κάθε μέλους της ομάδας και σε ποιό κομμάτι δραστηριοποιήθηκε ο κάθε ένας. Τέλος, όλη αυτή η συνεργασία ήταν μια πραγματική εμπειρίας από μια ομάδα ή από ένα εργασιακό περιβάλλον που πολλοί θα κριθούμε να αντιμετωπίσουμε στο μέλλον.

8 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Adi Pranata, A., Selviandro, N., and Adrian, M., Analysis and Implementation of Presenter Layer on MVC Architecture iOS Application Development∗. In *2022 The 5th International Conference on Software Engineering and Information Management (ICSIM)* (pp. 82-86), 2022.

[2] Alessandria, S., *Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games*, Packt Publishing Ltd, 2020.

[3] Biessek, A., *Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2*, Packt Publishing Ltd., 2019.

[4] Boukhary, S., and Colmenares, E., A clean approach to flutter development through the flutter clean architecture package. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1115-1120), IEEE, 2019.

[5] Cheng, F., State Management. In *Flutter Recipes* (pp. 365-412), Apress: Berkeley, CA, 2019.

[6] Kuzmin, N., Ignatiev, K. and Grafov, D. (2020). "Experience of developing a mobile application using flutter", In *Information Science and Applications* (pp. 571-575). Springer, Singapore, 2019.

[7] Mainkar, P., and Giordano, S., *Google Flutter Mobile Development Quick Start Guide: Get Up and Running with IOS and Android Mobile App Development*, Packt Publishing Ltd., 2019.

[8] Miola, A., *Flutter Complete Reference: Create Beautiful, Fast and Native Apps for Any Device*, Independently published, 2020.

[9] Napoli, M. L., *Beginning flutter: a hands on guide to app development*, John Wiley & Sons, 2019.

[10] Payne, R., *Beginning App Development with Flutter: Create Cross-Platform Mobile Apps*, Apress, 2019a.

[11] Payne, R., Developing in Flutter. In *Beginning App Development with Flutter* (pp. 9-27), Apress: Berkeley, CA, 2019b.

[12] Wee, T. C. and Kee, K. N. C., On-Device Image Labelling Photo Management System Using Flutter and ML Kit. In *International Conference on Intelligent Technologies for Interactive Entertainment* (pp. 208-222), Springer, Cham, 2021.

[13] Zaccagnino, C., "Programming Flutter: Native, Cross-Platform Apps the Easy Way", *Programming Flutter*, 1-275, 2020.

Zammetti, F., *Practical Flutter*, Berkeley, CA: Apress, 2019.

ΙΣΤΟΣΕΛΙΔΕΣ

[14] Dignan, L., *Apple's App Store 2016 revenue tops $28 billion mark, developers net $20 billion*, 2017. Ανακτήθηκε από https://www.zdnet.com/article/apples-app-store-2016-revenue-tops-28-billion-mark-developers-net-20-billion/

[15] Eadicicco, L., *Watch Steve Jobs Unveil the First iPhone 10 Years Ago Today*, 2017Ανακτήθηκε από https://time.com/4628515/steve-jobs-iphone-launch-keynote-2007/

[16] Eran, D., *Nine Years of Apple's iOS SDK generated $60 billion, 1.4 million jobs*, 2017. Ανακτήθηκε https://appleinsider.com/articles/17/03/07/nine-years-of-apples-ios-sdk-generated-60-billion-14-million-jobs

[17] Gonsalves, A., *Apple Launches iPhone Web Apps Directory*, 2007. Ανακτήθηκε από https://www.informationweek.com/government/apple-launches-iphone-web-apps-directory

[18] Satariano, A., *Scott Forstall, the Sorcerer's Apprentice at Apple*, 2011. Ανακτήθηκε από https://www.bloomberg.com/news/articles/2011-10-12/scott-forstall-the-sorcerers-apprentice-at-apple

https://medium.com/@devathon_/10-best-android-frameworks-for-app-development-in-2020-98f5afb300e9

https://blog.pirago.vn/dart-flutter-bai1/

https://www.minitzon.com/blogs/cross-platform-mobile-app-comparison/

https://www.oreilly.com/api/v2/epubs/9781788996082/files/assets/c4764590-7c54-4afb-8a25-464d67605d7d.png

https://www.sitepoint.com/premium/books/beginning-flutter/read/1/

https://mobikul.com/lifecycle-of-a-flutter-app/

https://intellipaat.com/blog/tutorial/ios-tutorial/ios-architecture/

https://bigblue.academy/gr/ti-einai-to-web-scraping

https://www.ip.gr/el/dictionary/378-API___Application_Programming_Interface?gclid=EAIaIQobChMIvfS0oKSz_AIVUAGLCh3wyAQ2EAAYASAAEgJeV_D_BwE