Σελίδα τίτλου

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΚΑΙ ΔΙΟΙΚΗΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ MOBILE ΕΦΑΡΜΟΓΗΣ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ ΚΑΙ ΕΓΚΑΙΡΗΣ ΕΙΔΟΠΟΙΗΣΗΣ ΠΕΡΙΒΑΛΛΟΝΤΙΚΩΝ ΔΕΔΟΜΕΝΩΝ - ΑΙΘΑΛΟΜΙΧΛΗΣ ΜΕ ΧΡΗΣΗ CROSS-PLATFORM ΤΕΧΝΟΛΟΓΙΩΝ

Δημήτριος Ηλιόπουλος

Επιβλέπων: Αλέξανδρος Τζάλλας

Άρτα, Ιούνιος, 2023

DEVELOPMENT OF A MOBILE APP FOR MONITORING AND ON-TIME UPDATES ABOUT ENVIRONMENTAL DATA-SMOG USING CROSS-PLATFORM TECHNOLOGIES

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Άρτα, 20/06/2023

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Αλέξανδρος Τζάλλας

2. Μέλος επιτροπής

   Ευριπίδης Γλαβάς

3. Μέλος επιτροπής

   Δημήτριος Ηλιόπουλος

© Ηλιόπουλος, Δημήτριος, 2023.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Ηλιόπουλος, Δημήτριος

Υπογραφή

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία με τίτλο «Ανάπτυξη mobile εφαρμογής παρακολούθησης και έγκαιρης ειδοποίησης περιβαλλοντικών δεδομένων – αιθαλομίχλης με χρήση cross-platform τεχνολογιών» παρουσιάζεται η διαδικασία υλοποίησης μίας εφαρμογής για smartphones, η οποία εστιάζει στην απεικόνιση περιβαλλοντικών μετρήσεων από μετεωρολογικούς σταθμούς προς ενημέρωση του χρήστη. Η εφαρμογή αναπτύχθηκε στο cross-platform framework της Microsoft, .NET MAUI (.NET Multi-platform App UI).

Λέξεις-κλειδιά: .net maui, cross-platform, mvvm, air quality index, AQI, αιθαλομίχλη

ABSTRACT

In this thesis on “Development of a mobile app for monitoring and on-time updates about environmental data-smog using cross-platform technologies” the process of developing an application for smartphone devices is presented, focusing on presenting and informing the user about environmental data received from weather stations. The application was developed on Microsoft’s cross-platform framework, .NET MAUI (.NET Multi-platform App UI).

Keywords: .net maui, cross-platform, mvvm, air quality index, AQI, smog

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ/ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΟΓΡΑΦΙΩΝ

MAUI..…….…….Multi-platform App UI

MVVM..……..……..Model-View-ViewModel

AQI……Air Quality Index

IDE…….……...Integrated Development Environment

JSON……..……..JavaScript Object Notation

XML……...eXtensible Markup Language

XAML…….eXtensible Application Markup Language

UI…….User Interface

DI……Dependency Injection

HTTP…….HyperText Transport Protocol

GUID……..Global Unique Identifier

VS……Visual Studio

MAC ……..Media Access Control

API……Application Programming Interface

ΑΠΟΔΟΣΗ ΟΡΩΝ / ΓΛΩΣΣΑΡΙΟ

Framework: Ένα σύνολο από βιβλιοθήκες, εργαλεία και κώδικα που χρησιμοποιείται για τη δημιουργία μιας εφαρμογής ή ενός συστήματος.

Architecture Pattern: Αναφέρεται στον σχεδιασμό ενός συστήματος λογισμικού μέσω μιας σειράς από αναγνωρίσιμα μοτίβα ή πρότυπα, τα οποία χρησιμοποιούνται για να οργανώσουν τον κώδικα και να βελτιώσουν την αναγνωσιμότητα, την συντηρησιμότητα και την επεκτασιμότητα του συστήματος.

Market Share: Αναφέρεται στο μερίδιο αγοράς που κατέχει μια εταιρεία, ένα προϊόν ή μια υπηρεσία σε σχέση με το συνολικό μέγεθος της αγοράς στην οποία δραστηριοποιείται.

Cross Platform: Αναφέρεται στη δυνατότητα ενός λογισμικού ή μιας εφαρμογής να λειτουργεί σε διαφορετικές πλατφόρμες ή λειτουργικά συστήματα.

Web App: Αναφέρεται σε μια εφαρμογή λογισμικού που λειτουργεί μέσω ενός web browser και είναι προσβάσιμη μέσω του διαδικτύου.

Application Programming Interface: Αναφέρεται στο σύνολο των προδιαγραφών και των μεθόδων που παρέχονται από μια εφαρμογή ή ένα λογισμικό για να επιτρέψουν σε άλλα προγράμματα να αλληλοεπιδρούν μαζί τους.

Codebase: Αναφέρεται στο σύνολο του κώδικα που αποτελεί μια εφαρμογή ή ένα πρόγραμμα.

Dependency Provider: Αναφέρεται στην κλάση ή το αντικείμενο που παρέχει τις εξαρτήσεις (dependencies) στο αντικείμενο που τις χρειάζεται.

User interface: Αναφέρεται στον τρόπο με τον οποίο ο χρήστης αλληλοεπιδρά με μια εφαρμογή, ιστοσελίδα ή οποιοδήποτε άλλο σύστημα που επιτρέπει την αλληλεπίδρασή του με την τεχνολογία.

Interface (.NET): Αναφέρεται σε μια δομή δεδομένων που περιγράφει τις δυνατότητες μιας κλάσης, χωρίς να περιγράφει τη συγκεκριμένη υλοποίηση αυτών των δυνατοτήτων. Συνήθως, μια διεπαφή περιέχει μόνο τις δηλώσεις των μεθόδων, ιδιοτήτων και γεγονότων.

Constructor: Αναφέρεται σε μια ειδική μέθοδο που χρησιμοποιείται για την αρχικοποίηση των μελών μιας κλάσης κατά τη δημιουργία ενός αντικειμένου της κλάσης.

Back-End: Αναφέρεται στο μέρος μιας εφαρμογής ή μιας ιστοσελίδας που δεν είναι ορατό στον χρήστη, αλλά αποτελεί την εσωτερική λειτουργία της εφαρμογής. Ασχολείται με την αποθήκευση, την επεξεργασία και την παροχή δεδομένων στο front-end της εφαρμογής που είναι ορατό στον χρήστη.

Front-End: Αναφέρεται στο μέρος μιας εφαρμογής που απευθύνεται στον τελικό χρήστη και αφορά την διεπαφή χρήστη (user interface) και τη λειτουργικότητα που είναι ορατή στον τελικό χρήστη

1 Εισαγωγή στο cross-platform λογισμικό

Τα cross-platform frameworks, έρχονται να δώσουν λύση στο πρόβλημα των ομάδων που υλοποιούν εφαρμογές για ευρύ κοινό και θέλουν να στοχεύσουν σε πάνω από μία πλατφόρμα. Μία παραδοσιακή εφαρμογή για να φτάσει σε στάδιο διανομής, απαιτείται η παραγωγή των εκτελέσιμων αρχείων ως προς την κάθε πλατφόρμα στην οποία στοχεύει η υλοποίηση. Με την αύξηση του Mac OS σε market-share τα τελευταία χρόνια, το ίδιο πρόβλημα που αντιμετωπίζουν οι ομάδες υλοποίησης για mobile apps, καλούνται να αντιμετωπίσουν και οι ομάδες των desktop apps, αλλά σε μικρότερο βαθμό. Η επιλογή της κατάλληλης πλατφόρμας που θα υποστηρίξει η ομάδα δεν είναι απλή, καθώς η επιλογή αυτή ορίζει τους developers που θα χρειαστούν και τις γνώσεις που απαιτούνται. Με τα δύο κυρίαρχα λειτουργικά συστήματα για κινητές συσκευές στην αγορά να είναι το Android και το iOS, οι ομάδες που υλοποιούν mobile εφαρμογές καλούνται να εξυπηρετήσουν και τις δύο πλατφόρμες. Με αυτό τον τρόπο δύνανται να αυξήσουν το πλήθος των χρηστών για τους οποίους θα είναι διαθέσιμη η εφαρμογή τους. Για να το πετύχουν αυτό καταφεύγουν σε ορισμένες λύσεις, ανάλογα τις ανάγκες και τις απαιτήσεις του project. Ορισμένες από τις πιο συνήθεις λύσεις, μεταξύ πολλών άλλων, είναι οι ακόλουθες:

- Υλοποίηση της εφαρμογής για την κάθε πλατφόρμα ξεχωριστά
- Υλοποίηση της εφαρμογής σε web-app
- Υλοποίηση της εφαρμογής σε cross-platform framework

1.1 Παραδοσιακή υλοποίηση (Native development)

Τα projects που επιλέγουν να υποστηρίξουν και τις δύο πλατφόρμες natively, εξασφαλίζουν τη μέγιστη δυνατή συμβατότητα της εφαρμογής για τις πλατφόρμες και τις συσκευές που στοχεύουν. Συνήθως, παρόλα αυτά, αντιμετωπίζουν το πρόβλημα ανάγκης απασχόλησης ομάδων που αφορούν την κάθε πλατφόρμα αποκλειστικά. Αυτό, δημιουργεί την ανάγκη απασχόλησης περισσότερων ανθρώπων και ως εκ τούτου, η εφαρμογή να απαιτεί διπλό χρόνο και κόστος, ως προς την υλοποίηση αλλά και την συντήρησή της - υλοποίηση νέων features, υλοποίηση change requests, διερεύνηση και διόρθωση bugs ανά πλατφόρμα/εφαρμογή, κλπ.

1.2 Web apps

Πολλές ομάδες για να αποφύγουν το πρόβλημα που προξενεί η παραδοσιακή υλοποίηση (βλ. κεφ. 1.1) και, εφόσον οι απαιτήσεις της εφαρμογής το επιτρέπουν, επιλέγουν να υλοποιήσουν την εφαρμογή σε κάποιο από τα διαθέσιμα web frameworks, όπως για παράδειγμα τα Angular, React, Rails και άλλα. Αυτό, λύνει το πρόβλημα της επιλογής για υλοποίηση και συνεχή υποστήριξη ανά πλατφόρμα, καθώς η μόνη ανάγκη για να εκτελεστεί η εφαρμογή σε οποιαδήποτε συσκευή, είναι η ύπαρξη ενός μοντέρνου web browser. Ωστόσο, παρόλο που λύνει ένα σημαντικό πρόβλημα, εξακολουθεί να μην αποτελεί τη βέλτιστη επιλογή σε κάθε project, αφού ως web εφαρμογή που εκτελείται σε browser φέρνει πολλούς περιορισμούς ως προς τις δυνατότητες, τις επιδόσεις αλλά και την πρόσβαση στους διαθέσιμους πόρους της εκάστοτε συσκευής (hardware components, κλπ.).

1.3 Υλοποίηση Cross-platform app

Τα cross-platform frameworks που παράγουν native εφαρμογές ανά πλατφόρμα, έρχονται να λύσουν τα προβλήματα που προαναφέραμε. Ένα από τα χαρακτηριστικά αυτής της επιλογής, είναι ότι δίνεται η δυνατότητα παραγωγής native εφαρμογών, από ένα κοινό codebase. Αυτό σημαίνει ότι μία ομάδα developers υλοποιεί σε κοινό κώδικα και όχι σε ξεχωριστό κώδικα ανά πλατφόρμα. Επιπλέον, έχει όλες τις δυνατότητες μίας native υλοποίησης και ως εκ τούτου, δεν περιορίζεται στις δυνατότητες ενός web browser.

2 Αιθαλομίχλη

Η αιθαλομίχλη αποτελεί έναν ατμοσφαιρικό ρύπο που αποτελείται από σωματίδια και αέρια που αιωρούνται στην ατμόσφαιρα. Μπορεί να επηρεάσει αρνητικά την υγεία, καθώς μπορεί να προκαλέσει αναπνευστικά προβλήματα ή να τα επιδεινώσει.

Το φαινόμενο της αιθαλομίχλης είναι αρκετά έντονο τα τελευταία χρόνια, ειδικά σε ορισμένες περιοχές του ελλαδικού χώρου (…), όπου το κλίμα αυτών βοηθάει στην παραμονή του φαινομένου στην ατμόσφαιρα. (…)

2.1 Air-Quality Index (AQI)

Το Air Quality Index ή AQI ή αλλιώς Δείκτης Ποιότητας Αέρα στα Ελληνικά, είναι μία ένδειξη η οποία δείχνει την ποιότητα του αέρα και την επίπτωση στην υγεία του ανθρώπου με την έκθεση σε αυτόν και υπολογίζεται βάσει μετρήσεων συγκέντρωσης ρύπων στον αέρα. Πιο συγκεκριμένα, οι βασικοί ρύποι που λαμβάνει υπόψιν το Common Air Quality Index (CAQI), το οποίο και χρησιμοποιείται στην Ευρώπη, είναι οι εξής:

- Αιωρούμενα Σωματίδια (PM_{10})
- Λεπτά Αιωρούμενα Σωματίδια ($PM_{2.5}$)
- Όζων (O_3)
- Διοξείδιο του Αζώτου (NO_2)
- Διοξείδιο του Θείου (SO2)

2.2 Ζώνες συγκέντρωσης ρύπων

Index Level (Based on pollutant concentrations in µg/m3)						
Pollutant	**Good**	**Fair**	**Moderate**	**Poor**	**Very Poor**	**Extremely Poor**
Particles less than 2.5 µm (PM2.5)	0-10	10-20	20-25	25-50	50-75	75-800
Particles less than 10 µm (PM10)	0-20	20-40	40-50	50-100	100-150	150-1200
Nitrogen dioxide (NO2)	0-40	40-90	90-120	120-230	230-340	340-1000
Ozone (O3)	0-50	50-100	100-130	130-240	240-380	380-800
Sulphur dioxide (SO2)	0-100	100-200	200-350	350-500	500-750	750-1250

Πίνακας 1 - Ζώνες συγκέντρωσης ρύπων

3 Τεχνολογίες που θα χρησιμοποιηθούν

Για την υλοποίηση της mobile εφαρμογής σε cross-platform framework, ο δημιουργός δύναται να χρησιμοποιήσει πληθώρα εργαλείων, από τα οποία μπορεί να επιλέξει όχι μόνο τα καταλληλότερα βάσει και των απαιτήσεων της εφαρμογής, αλλά και εκείνα με τα οποία νιώθει μεγαλύτερη άνεση και ασφάλεια να χρησιμοποιήσει.

Για τους σκοπούς της παρούσας εργασίας, σαν cross-platform framework επιλέχθηκε το .NET MAUI της Microsoft. Το IDE που θα χρησιμοποιηθεί είναι το *Visual Studio 2022*, στο οποίο υποστηρίζεται natively το *.NET MAUI*. Επιπλέον, θα χρησιμοποιηθεί το *Community Toolkit* και, πιο συγκεκριμένα, οι βιβλιοθήκες που αφορούν το Model-View-ViewModel architecture pattern (MVVM). Για τη διαγραμματική απεικόνιση των δεδομένων που θα εμπεριέχονται στην εφαρμογή, θα χρησιμοποιηθούν οι βιβλιοθήκες της *DevExpress* και *LiveCharts2*. Για την επικοινωνία των δεδομένων με το back-end θα χρησιμοποιηθεί η βιβλιοθήκη *Newtonsoft.Json*. Ακόμα, θα χρησιμοποιηθεί το Dependency Injection που έρχεται built-in στο .NET MAUI. Τέλος, τα εικονίδια που θα χρησιμοποιηθούν στην εφαρμογή προέρχονται από την δωρεάν συλλογή του *icons8.com*. Στα παρακάτω, ακολουθεί αναλυτική περιγραφή της κάθε προαναφερθείσας τεχνολογίας που πρόκειται να χρησιμοποιηθεί.

3.1 .NET MAUI

Το .NET MAUI ή Multi-platform App UI, είναι ένα cross-platform framework από την Microsoft. Αποτελεί τον απόγονο του Xamarin.Forms το οποίο δημιουργήθηκε από την Xamarin και ανακοινώθηκε πρώτη φορά το 2014. Το .NET MAUI χρησιμοποιεί ένα κοινό codebase, ενώ η δομή του επιτρέπει platform specific υλοποιήσεις, για τις ανάγκες ή τις ιδιαιτερότητες της κάθε πλατφόρμας όπου αυτό κρίνεται απαραίτητο.

3.1.1 Υποστηριζόμενες πλατφόρμες

Το .NET MAUI υποστηρίζει πολλές από τις πιο διαδεδομένες πλατφόρμες - Android, iOS, Windows και macOS -, ενώ δίνει τη δυνατότητα έκδοσης της εφαρμογής και ως TV App για την πλατφόρμα *Tizen*, της *Samsung*. Δίνει λοιπόν την δυνατότητα από ένα κοινό κώδικα να παράγουμε native εφαρμογές στις τέσσερις επικρατέστερες πλατφόρμες για desktops και smartphones. Πιο αναλυτικά, αναγράφονται και οι υποστηριζόμενες εκδόσεις ανά πλατφόρμα, όπως αυτές ορίζονται στο χρονικό πλαίσιο συγγραφής της παρούσας εργασίας:

- Android 5.0 (API 21) or higher.
- iOS 11 or higher, using the latest release of Xcode.
- macOS 10.15 or higher, using Mac Catalyst.
- Windows 11 and Windows 10 version 1809 or higher, using Windows UI Library (WinUI) 3.

3.1.2 Platform Specific υλοποιήσεις

Το .NET MAUI επιτρέπει platform specific υλοποιήσεις. Αν υπάρχει ανάγκη να χειριστεί κάποιος διαφορετικά κάποιο resource ανά πλατφόρμα, αυτό είναι το ιδανικό σημείο για να το κάνει. Έστω, για παράδειγμα, ότι η υλοποίηση μίας εφαρμογής δημιουργεί την ανάγκη να χειριστούμε διαφορετικά την επικοινωνία NFC και οι ήδη υπάρχουσες βιβλιοθήκες δεν μας δίνουν το επιθυμητό αποτέλεσμα. Ο developer, σε αυτή την περίπτωση, θα πρέπει να γράψει platform specific κώδικα, για να χειριστεί τις ιδιαιτερότητες της κάθε πλατφόρμας χρησιμοποιώντας τα native APIs που προσφέρει το .NET MAUI. Έπειτα, θα μπορέσει να αξιοποιήσει την platform specific υλοποίηση από το κοινό codebase με τη χρήση interface, εξαλείφοντας έτσι και την ανάγκη συγγραφής κώδικα με αυξημένη πολυπλοκότητα.

3.1.3 Γλώσσα προγραμματισμού

Το .NET MAUI ως framework στο οικοσύστημα του .NET χρησιμοποιεί την γλώσσα προγραμματισμού C#. Επιπλέον, όσον αφορά τον σχεδιασμό του User Interface (UI), δίνει τη δυνατότητα να γίνει χρήση της γλώσσας eXtensible Application Markup Language (XAML). Χρησιμοποιώντας συνδυαστικά XAML για το UI και C# για το business logic βάσει του MVVM architecture pattern, μπορεί εύκολα να αξιοποιηθεί η δυνατότητα Data-Binding που προσφέρει το .NET MAUI, όπως θα αναλυθεί και παρακάτω.

3.1.4 Data-Bindings

Η δυνατότητα Data-Binding στο .NET MAUI επιτρέπει την σύνδεση πεδίων δύο οντοτήτων. Με αυτό, η κάθε μεταβολή σε μία από τις δύο οντότητες προκαλεί την αντίστοιχη μεταβολή στην άλλη. Τα Data-Bindings, συνεπώς, είναι ιδιαίτερα χρήσιμα, καθώς διευκολύνουν τον προγραμματιστή, να εστιάσει στα δεδομένα που εμπεριέχονται στο ViewModel, χωρίς να δημιουργεί επιπρόσθετο όγκο δουλειάς ως προς την ενημέρωση του UI. Για να γίνει πιο κατανοητό το παραπάνω, έστω ότι έχουμε μία μεταβλητή A και ένα UI element απεικόνισης κειμένου. Με τη χρήση του data-bind μεταξύ του UI element και της μεταβλητής A, οποιαδήποτε ανάθεση τιμής γίνει στη μεταβλητή A, θα ενημερώσει αυτόματα και το UI element. Τέλος, τα data-bindings, μπορούν να χρησιμοποιηθούν ακόμα και αν το UI δεν είναι γραμμένο σε γλώσσα XAML, αλλά αποφασιστεί ο σχεδιασμός του να γίνει με C# στο code-behind.

3.1.5 Model-View-ViewModel

Το Model-View-ViewModel (MVVM) είναι μοτίβο αρχιτεκτονικής λογισμικού το οποίο διαχωρίζει την υλοποίηση του UI με το business logic, ορίζοντας με αυτό τον τρόπο το UI, ανεξάρτητο από τα μοντέλα του back-end και του business logic. Βάσει του παραπάνω, στο View γίνεται όλος ο σχεδιασμός του UI. Στο Model εμπεριέχεται όλο το business logic - δεδομένα και επικοινωνία με το back-end της εφαρμογής. Το ViewModel

λειτουργεί ως ενδιάμεσος ή μετατροπέας των δεδομένων του Model σε μορφή έτοιμη για απεικόνιση από το UI (View display logic), καθώς επίσης περιέχει και τις μεθόδους για επικοινωνία με το Model. Το MVVM είναι μία παραλλαγή του Presentation Model design pattern, το οποίο λειτουργεί παρόμοια. (…)

3.1.6 Code-Behind

Ο όρος Code-Behind αναφέρεται στον κώδικα που τρέχει πίσω από το XAML αρχείο. Ουσιαστικά είναι ένα C# αρχείο και λειτουργεί συνδυαστικά με το XAML αρχείο κατά το build της εφαρμογής. Συνήθως σε απλά UI δεν χρειάζεται να δουλέψουμε καθόλου με το code-behind αν έχουμε επιλέξει τον σχεδιασμό του UI σε XAML. Ωστόσο δίνει πολλές δυνατότητες για εκτέλεση εργασιών βάσει συγκεκριμένων events της σελίδας, καθώς και για υλοποίηση custom animations, αλλά και για εκτέλεση περίπλοκης λογικής στο UI.

3.1.7 Dependency Injection

Το dependency injection (DI) είναι μια τεχνική που χρησιμοποιείται στον προγραμματισμό λογισμικού για τη διαχείριση των εξαρτήσεων των αντικειμένων. Ουσιαστικά, με την τεχνική του DI, τα αντικείμενα δεν δημιουργούν ούτε δεσμεύουν τις εξαρτήσεις τους, αλλά τις λαμβάνουν από έναν εξωτερικό πάροχο (dependency provider). Η τεχνική του DI έχει τα πλεονεκτήματα της αποσύνδεσης του κώδικα, της ευκολίας στην αντιμετώπιση των αλλαγών και της εξασφάλισης της επαναχρησιμοποίησης του κώδικα.

3.2 Back-end

Το back-end της εφαρμογής αποτελεί μια βάση δεδομένων, στην οποία αποθηκεύονται περιβαλλοντικά δεδομένα από μετεωρολογικούς σταθμούς. Για την επικοινωνία της εφαρμογής με το back-end έχει δοθεί Web Service με τα κατάλληλα endpoints για την λήψη των δεδομένων. Τα δεδομένα θα μεταφέρονται σε μηνύματα με την μορφή JSON.

3.2.1 Web Services

Τα web services αποτελούν μια μέθοδο επικοινωνίας και ανταλλαγής δεδομένων μεταξύ διαφορετικών εφαρμογών και συστημάτων στο διαδίκτυο. Επιτρέπουν σε διαφορετικά συστήματα και πλατφόρμες να επικοινωνούν μεταξύ τους, χρησιμοποιώντας κοινές διεπαφές και πρωτόκολλα όπως το HyperText Transport Protocol (HTTP).

Τα web services ακολουθούν μια απλή αρχιτεκτονική που αποτελείται από τρία βασικά στοιχεία:

- Ένα συμβατικό πρωτόκολλο επικοινωνίας: όπως για παράδειγμα το HTTP.
- Ένα πρότυπο μορφοποίησης δεδομένων: Συνήθως χρησιμοποιείται το XML ή το JSON.
- Ένα σύνολο υπηρεσιών που προσφέρονται: Μπορεί να πρόκειται για υπηρεσίες όπως ανάγνωση και εγγραφή δεδομένων, εκτέλεση λειτουργιών και επιστροφή αποτελεσμάτων.

3.2.2 JavaScript Object Notation (JSON)

Το JSON αποτελεί μια απλή και ελαφριά μορφή που χρησιμοποιείται για την ανταλλαγή δεδομένων ανεξάρτητα από τη γλώσσα προγραμματισμού που χρησιμοποιείται για τη δημιουργία των εφαρμογών και χρησιμοποιείται ευρέως στην ανάπτυξη λογισμικού για τη μεταφορά δεδομένων μεταξύ διαφορετικών εφαρμογών.

Το JSON, αναπτύχθηκε από την εταιρεία Douglas Crockford, τη δεκαετία του 2000. Έχει τη δυνατότητα να περιγράψει δομές δεδομένων όπως αντικείμενα, πίνακες, αριθμούς, αλφαριθμητικές και λογικές τιμές. Η σύνταξη του JSON είναι παρόμοια με αυτή της JavaScript, με τις τιμές να αναπαρίστανται σε μορφή ζευγαριών "κλειδιού-τιμής". Για παράδειγμα, μια αναπαράσταση JSON ενός αντικειμένου μπορεί να φαίνεται ως εξής:

Εικόνα 1 – JSON Sample

3.2.3 Authentication

Η αυθεντικοποίηση, ή αλλιώς authentication, είναι ένα σημαντικό ζήτημα για την ασφάλεια των web services, καθώς επιτρέπει στο σύστημα να επαληθεύσει την ταυτότητα του χρήστη ή του συστήματος που αποστέλλει τα αιτήματα. Οι τρόποι που γίνεται το authentication σε ένα web service μπορεί να διαφέρουν ανάλογα με το πρωτόκολλο που χρησιμοποιείται, αλλά μερικοί από τους πιο κοινούς τρόπους είναι οι ακόλουθοι:

- HTTP Authentication: Το HTTP Authentication είναι μια διαδικασία που επιτρέπει την αυθεντικοποίηση του χρήστη μέσω του πρωτοκόλλου HTTP. Υπάρχουν διάφορες μέθοδοι HTTP Authentication, όπως η Basic, η Digest, η NTLM και η OAuth.
- Token-based Authentication: Το Token-based Authentication είναι μια διαδικασία που επιτρέπει στους χρήστες να επαληθεύονται μέσω ενός token που εκδίδεται από το σύστημα. Ο χρήστης πρέπει να στείλει το token μαζί με τα αιτήματά του για να πιστοποιηθεί.
- Certificate-based Authentication: Η Certificate-based Authentication είναι μια διαδικασία που επιτρέπει την αυθεντικοποίηση του χρήστη μέσω ενός ψηφιακού πιστοποιητικού που εκδίδεται από μια αξιόπιστη αρχή.

Το συγκεκριμένο web service που δόθηκε στα πλαίσια υλοποίησης της παρούσας εφαρμογής, χρησιμοποιεί τη μέθοδο Token-based. Το token ή αλλιώς κλειδί, έχει την μορφή GUID ή ολογράφως, Globally Unique Identifier και παρέχεται σαν παράμετρος στην διεύθυνση του web service.

Τα GUIDs είναι μοναδικά σε παγκόσμιο επίπεδο, καθώς υπολογίζονται με βάση την τρέχουσα ώρα και τη διεύθυνση MAC της συσκευής στην οποία δημιουργούνται. Αυτό τα καθιστά ιδανικά για τη δημιουργία μοναδικών αναγνωριστικών για αντικείμενα, όπως εγγραφές σε βάσεις δεδομένων, αρχεία ή σύνολα δεδομένων σε μια εφαρμογή. Τα GUIDs αποτελούνται από 32 ψηφία σε μορφή δεκαεξαδικού αριθμού, διαιρεμένου σε πέντε διακεκομμένα τμήματα, χωρισμένα με παύλες, έχοντας την παρακάτω μορφή:

```objectivec
6B29FC40-CA47-1067-B31D-00DD010662DA
```

Εικόνα 2 – GUID sample

3.3 Πρόσθετα εργαλεία που θα χρησιμοποιηθούν

3.3.1 Visual Studio 2022

Το Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment ή για συντομία IDE) που παρέχεται από τη Microsoft. Το πρόγραμμα αυτό χρησιμοποιείται κυρίως για την ανάπτυξη λογισμικού, όπως εφαρμογές για Windows, ιστοσελίδες και εφαρμογές για κινητές συσκευές.

3.3.2 Community Toolkit

Το Community Toolkit είναι ένα project ανοιχτού κώδικα (Open Source) της Microsoft. Πιο συγκεκριμένα, παρέχει μια συλλογή από επαναχρησιμοποιήσιμα εργαλεία για την ανάπτυξη εφαρμογών σε πλατφόρμες της Microsoft, όπως για παράδειγμα, το Universal Windows Platform (UWP), το Windows Presentation Foundation (WPF), το Xamarin και το .NET MAUI.

3.3.3 DevExpress

Η DevExpress είναι μια εταιρεία που αναπτύσσει εργαλεία λογισμικού για προγραμματιστές .NET αλλά και για άλλες πλατφόρμες. Η εταιρεία ιδρύθηκε το 1998 και από τότε έχει αναπτύξει μια ευρεία γκάμα επαγγελματικών εργαλείων λογισμικού, που καλύπτουν πολλούς τομείς της ανάπτυξης εφαρμογών.

Τα προϊόντα της DevExpress περιλαμβάνουν εργαλεία για την ανάπτυξη εφαρμογών σε .NET όπως το WinForms, WPF, ASP.NET, Xamarin.Forms και .NET MAUI. Η εταιρεία επίσης παρέχει εργαλεία για τη διαχείριση των δεδομένων, τη δημιουργία αναφορών και τον έλεγχο των δικαιωμάτων πρόσβασης.

3.3.4 LiveCharts2

Το LiveCharts2 είναι μια βιβλιοθήκη για τη δημιουργία διαγραμμάτων στην πλατφόρμα της Microsoft .NET, χρησιμοποιώντας την γλώσσα προγραμματισμού C#. Η βιβλιοθήκη χρησιμοποιείται συχνά για την ανάπτυξη επαγγελματικών εφαρμογών και προσφέρει μια ποικιλία από τύπους διαγραμμάτων, όπως γραμμικά, κυκλικά, στήλες και περιοχές. Επιπροσθέτως δίνει τη δυνατότητα για την προσαρμογή της εμφάνισης των διαγραμμάτων με παραμετροποίηση των χρωμάτων, της γραμματοσειράς, της ετικέτας του άξονα και άλλων χαρακτηριστικών.

3.3.5 Newtonsoft.Json

Το Newtonsoft.Json (πλέον γνωστό ως Json.NET) είναι μια βιβλιοθήκη για την ανάγνωση, την εγγραφή και τη διαχείριση δεδομένων JSON στην πλατφόρμα .NET. Η βιβλιοθήκη αυτή παρέχει μια εύκολη και ευέλικτη μέθοδο για τη μετατροπή αντικειμένων της πλατφόρμας .NET σε δεδομένα JSON και αντίστροφα, καθώς και για τον έλεγχο της εγκυρότητας αλλά και τη διαχείριση των σφαλμάτων.

3.3.6 Icons8

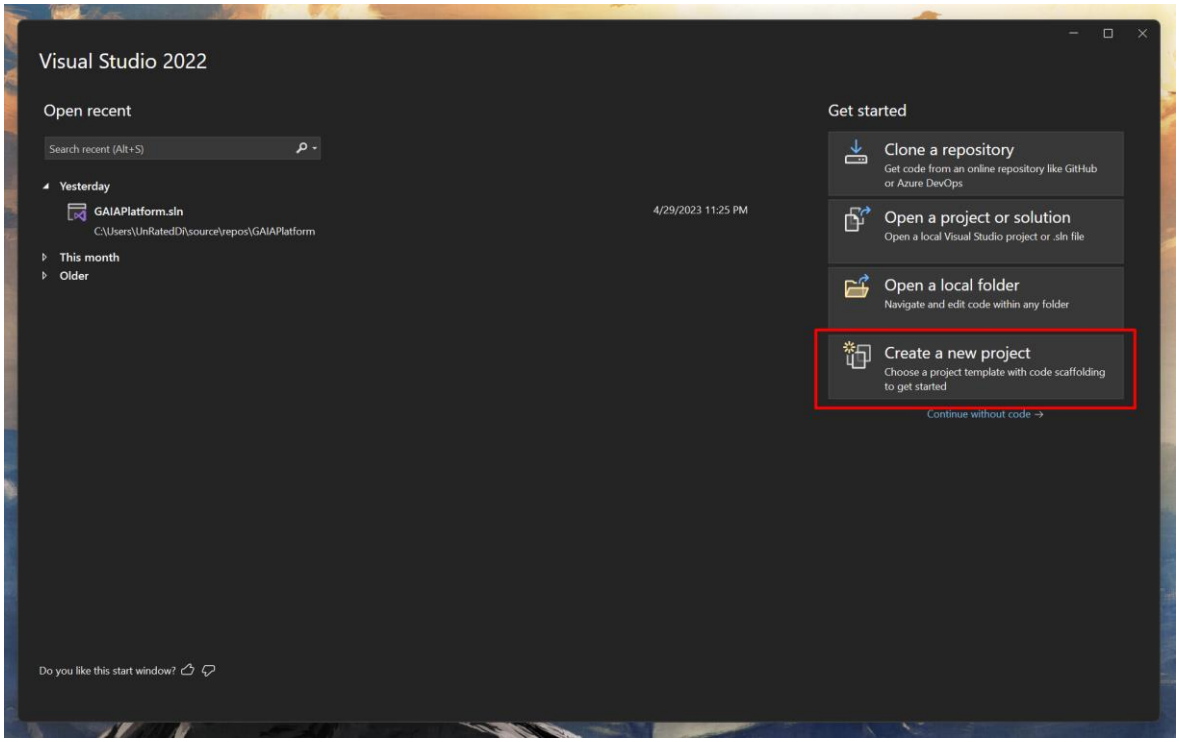
Το Icons8 είναι μια πλατφόρμα που παρέχει μεγάλη ποικιλία από εικονίδια, φωτογραφίες, βίντεο και ήχους, τα οποία διατίθενται δωρεάν για χρήση σε εφαρμογές, ιστοσελίδες και άλλα έργα. Τα εικονίδια είναι διαθέσιμα σε πολλά μεγέθη, υφές και χρώματα, συμπεριλαμβανομένων των διαφοροποιήσεων ως προς το υλικό, το επίπεδο σχεδίου, και παράλληλα διαθέτουν και επιλογές για κινούμενα ή στατικά σχέδια.

4 Δημιουργία της mobile εφαρμογής

Το βασικό εργαλείο που χρησιμοποιείται, όπως προαναφέρθηκε και σε προηγούμενο κεφάλαιο, είναι το Visual Studio 2022. Σε αυτό, πρόκειται να δημιουργηθεί το project για τη σύνταξη, τόσο του business logic κώδικα, όσο και του κώδικα για το UI της εφαρμογής.

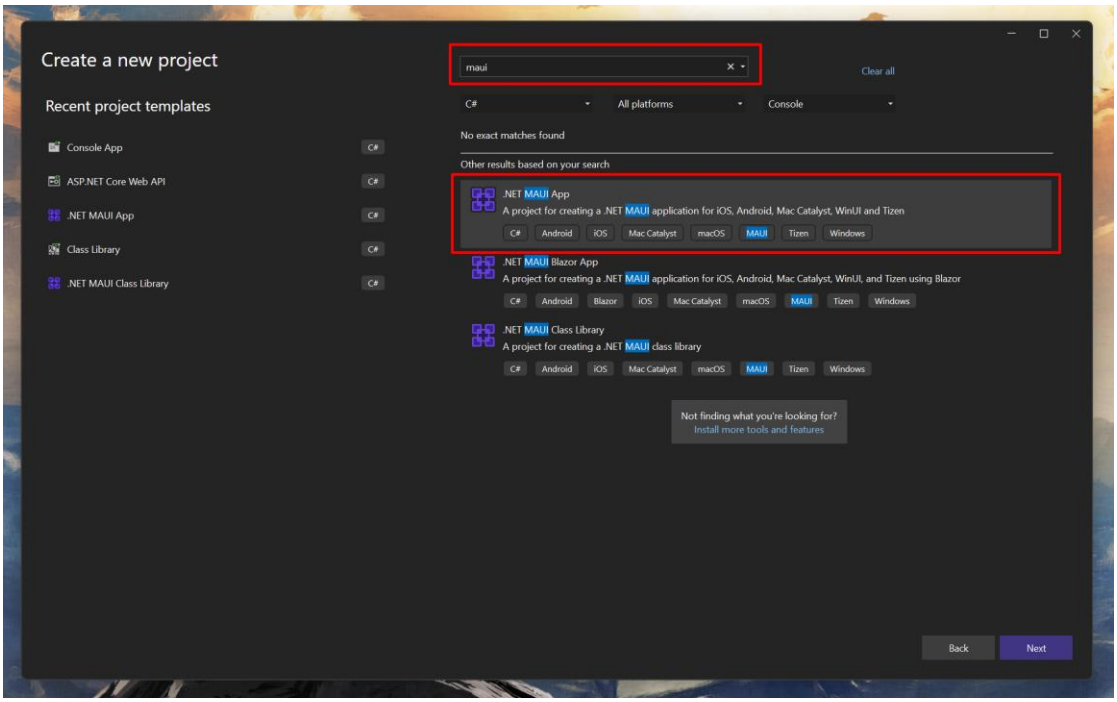
4.1 Δημιουργία του project στο Visual Studio 2022

Το Visual Studio 2022 ή VS2022 μας προσφέρει εύκολα και γρήγορα την δυνατότητα δημιουργίας νέου project. Κατά την εκκίνηση της εφαρμογής, εμφανίζεται μεταξύ άλλων στην οθόνη, η επιλογή “Create a new project” όπως παρουσιάζεται και στην παρακάτω εικόνα:



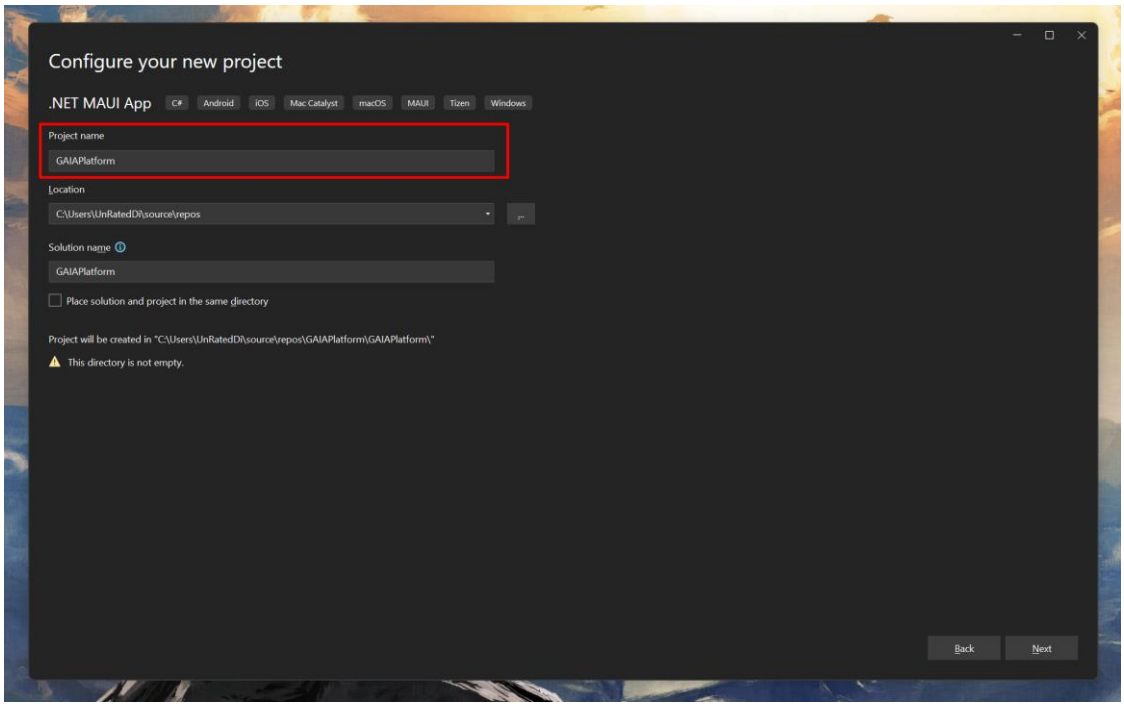
Εικόνα 3 – Οθόνη έναρξης του VS2022

Πατώντας σε αυτή την επιλογή, η επόμενη οθόνη που εμφανίζεται είναι τα διαθέσιμα templates για την δημιουργία project. Χρησιμοποιώντας τα templates, μπορούμε εύκολα να φτιάξουμε ένα project έτοιμο προς εκτέλεση για το είδος της εφαρμογής που μας ενδιαφέρει, καθώς σε αυτά προκαθορίζεται και η δομή του έργου σε φάκελους-υποφάκελους (π.χ. ASP.NET, WPF, .NET MAUI κλπ.). Θα χρησιμοποιήσουμε, λοιπόν, την αναζήτηση που παρέχεται με την λέξη κλειδί “maui” και, στην συνέχεια, θα επιλέξουμε το project template που μας απασχολεί. Το project template για την εφαρμογή που πρόκειται να υλοποιήσουμε, είναι το “.NET MAUI App” που όπως θα δούμε και στα tags (βλ. εικόνα 4), υποστηρίζει τις πλατφόρμες στις οποίες στοχεύουμε.



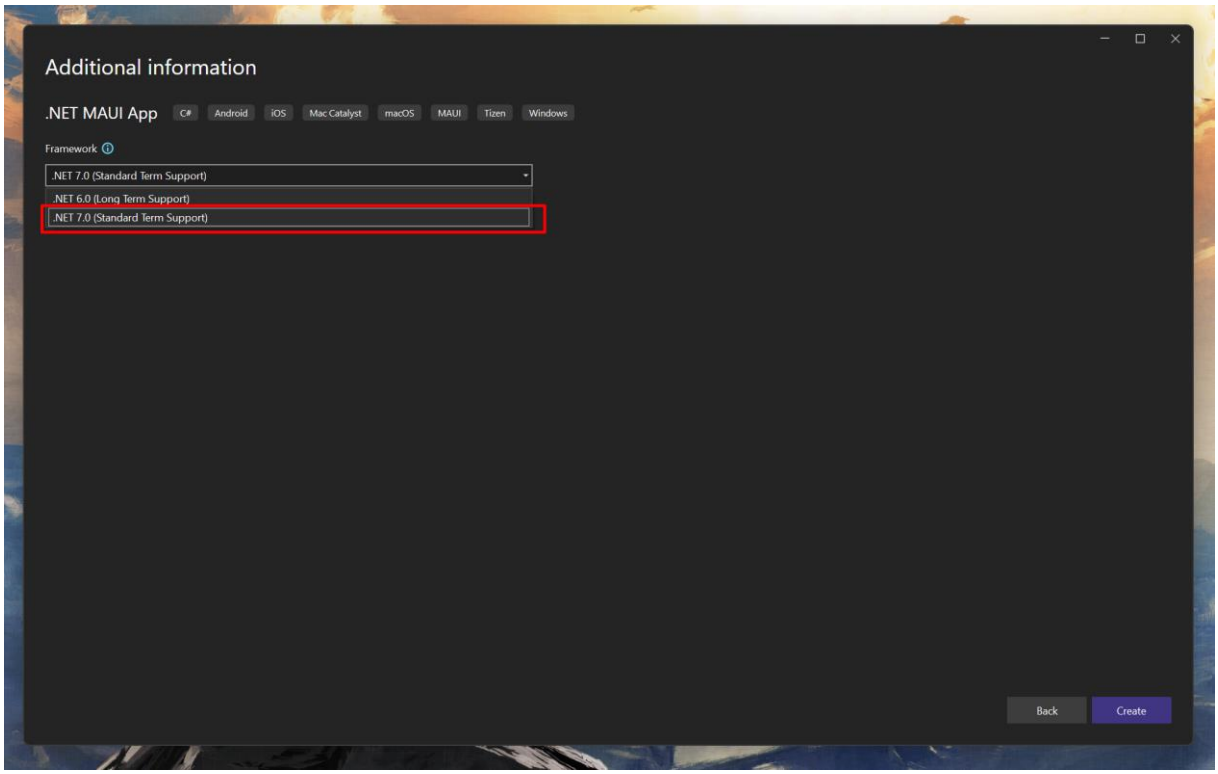
Εικόνα 4 – Οθόνη επιλογής project template στο VS2022

Ως επόμενο βήμα, καλούμαστε να ορίσουμε την ονομασία του project και του solution, καθώς επίσης και να αλλάξουμε την τοποθεσία που θα αποθηκευτεί το project εφόσον αυτό είναι επιθυμητό.



Εικόνα 5 – Οθόνη ονομασία του project και του solution και επιλογή τοποθεσίας αποθήκευσης VS2022

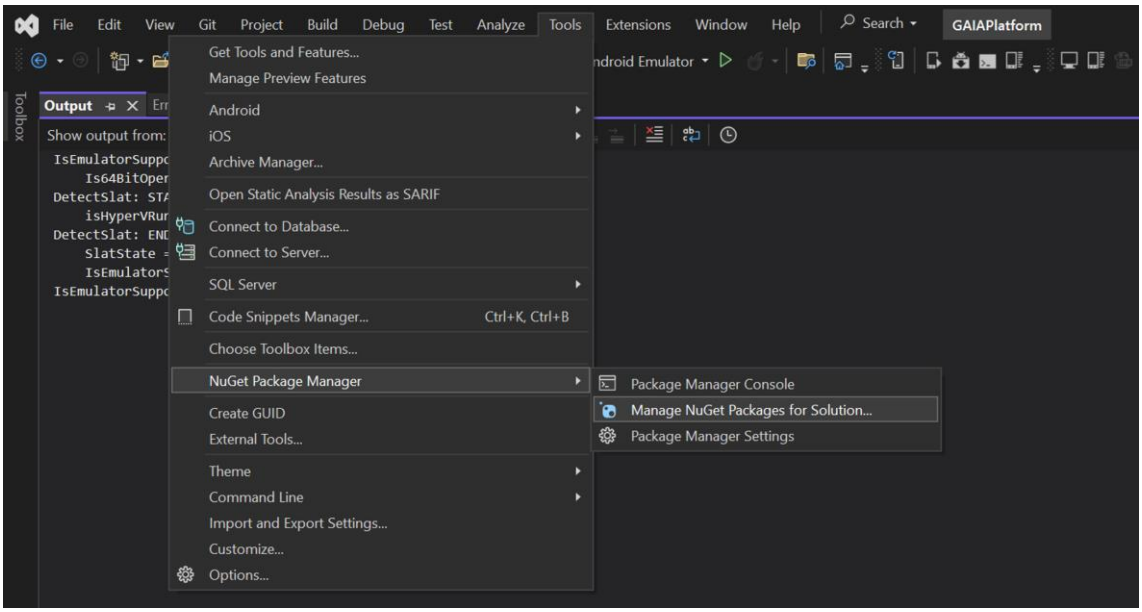
Ως τελευταίο βήμα για την δημιουργία νέου .NET MAUI project στο VS2022, είναι η επιλογή του .NET Framework που θέλουμε να χρησιμοποιήσουμε. Για τα .NET MAUI projects οι επιλογές είναι δύο: Το .NET 6 (Long Term Support) και το .NET 7(Standard Term Support). Για τους σκοπούς των δικών μας αναγκών, θα χρησιμοποιηθεί το .NET 7, καθώς φέρνει νέες δυνατότητες καθώς και πολλές βελτιώσεις σε επιδόσεις.



Εικόνα 6 – Οθόνη επιλογής .NET Framework για νέο project .NET MAUI VS2022

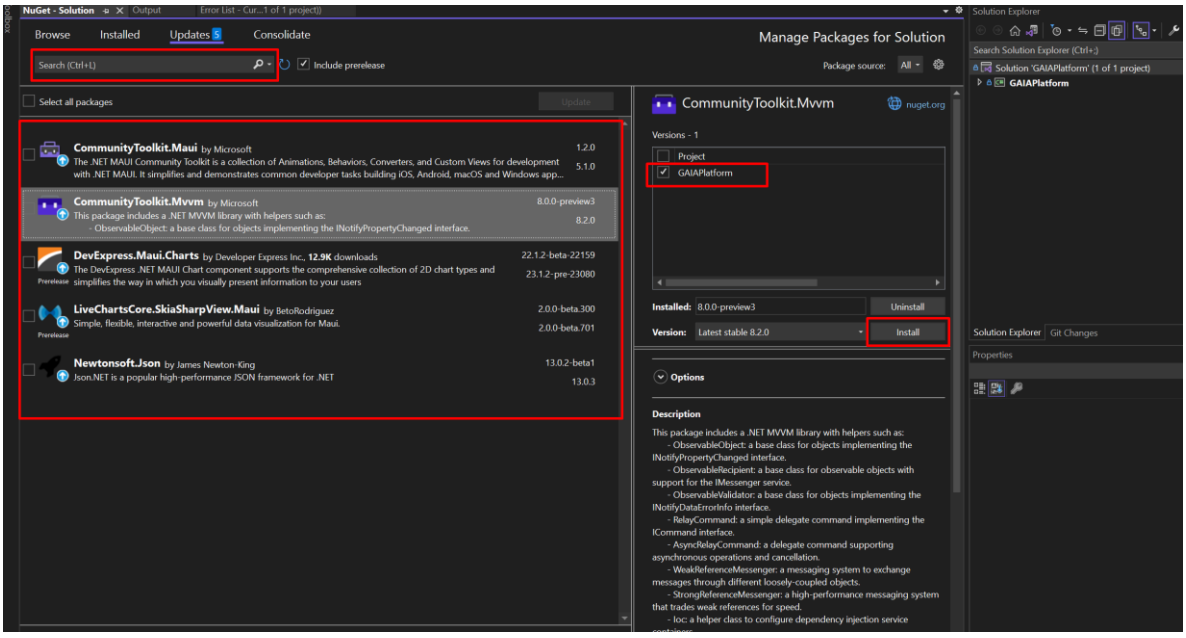
4.2 Προσθήκη βιβλιοθηκών

Για να προσθέσουμε στο project μας τις βιβλιοθήκες που χρειαζόμαστε, θα χρησιμοποιήσουμε το NuGet Package Manager του VS2022. Το τελευταίο, αποτελεί ένα πρόγραμμα για εύκολη εύρεση και εγκατάσταση βιβλιοθηκών στα projects μας, που προέρχονται είτε από το nuget.org είτε και από άλλες πηγές τρίτων. Θα το βρούμε στο μενού *Tools* και έπειτα στο *NuGet Package Manager*, όπου θα επιλέξουμε *Manage NuGet Packages for Solution…*, όπως παρουσιάζεται και στην παρακάτω εικόνα:



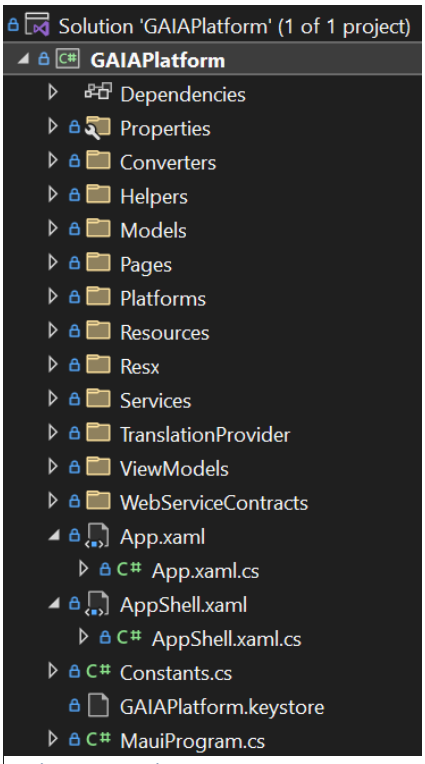
Εικόνα 7 – Εύρεση Nuget Package Manager VS2022

Τέλος, θα εγκαταστήσουμε τις βιβλιοθήκες που θα χρειαστούμε για να υλοποιήσουμε την εφαρμογή μας. Για να επιτευχθεί αυτό, θα πρέπει αρχικά να γίνει αναζήτηση της βιβλιοθήκης με λέξης κλειδιά και, ακολούθως, η επιλογή αυτής που μας εξυπηρετεί. Τέλος, πατώντας στο κουμπί εγκατάσταση, το NuGet Package Manager, θα πραγματοποιήσει αυτόματα τους ελέγχους συμβατότητας, ώστε να ολοκληρωθεί η εγκατάσταση της βιβλιοθήκης στο project (βλ. εικόνα 8):



Εικόνα 8 – NuGet Package Manager VS2022

4.3 Δομή του project



Εικόνα 9 – Δομή project

Το App.xaml / App.xaml.cs αποτελεί το σημείο εκείνο στο οποίο δημιουργείται η εφαρμογή βάσει του AppShell. Το AppShell.xaml / AppShell.xaml.cs αναλαμβάνει την διαδικασία πλοήγησης μέσα στην εφαρμογή, από οθόνη σε οθόνη. Στον φάκελο Resources κρατάμε πολυμέσα, όπως εικόνες, ήχους κλπ., που πρόκειται να χρησιμοποιηθούν στην εφαρμογή. Ακόμα, ο φάκελος Platforms θα αποτελέσει το μέρος που μπορούμε να κρατήσουμε Platform-specific υλοποιήσεις.

Τέλος, έχουμε οργανώσει το project σε φακέλους όπως ορίζει το μοντέλο αρχιτεκτονικής MVVM. Με αυτό τον τρόπο, έχουμε ορίσει τον φάκελο "Models", όπου εκεί θα περιγράψουμε τα μοντέλα που πρόκειται να χρησιμοποιηθούν στην εφαρμογή. Ακόμα, έχουμε δημιουργήσει τον φάκελο "Pages", όπου εκεί θα ζουν τα UI designs που θα σχεδιάσουμε και τα οποία ουσιαστικά αποτελούν τα Views κατά το MVVM. Επίσης, έχουμε τον φάκελο "ViewModels" που θα κρατάμε τα ViewModels των Views μας. Επιπλεόν, έχουμε και τον φάκελο "Services", όπου θα καταχωρίσουμε τις υλοποιήσεις όποιων Services μπορεί να χρειαστεί να δημιουργήσουμε. Τέλος, δημιουργήσαμε και τον φάκελο "WebServiceContracts", ο οποίος κρατάει τα μοντέλα επικοινωνίας με το back-end. Όλα τα παραπάνω, σε συνδυασμό με το MauiProgram.cs, το οποίο θα αναλυθεί και στη συνέχεια, θα αποτελέσουν και το βασικό κορμό του project.

Βοηθητικά, έχουν δημιουργηθεί και οι φάκελοι "Converters" και "Helpers", που αποτελούν ένα σετ από βοηθητικές μεθόδους, τόσο για το UI, όσο και για το business logic.

Ο φάκελος "TranslationProvider" και ο "Resx", κρατούν πρόσθετη υλοποίηση που δημιουργήθηκε για να εξυπηρετήσει ανάγκες πολυγλωσσίας στην εφαρμογή, σε περίπτωση που χρειαστεί.

4.3.1 MauiProgram.cs

Το αρχείο MauiProgram.cs αποτελεί και το σημείο εκκίνησης της εφαρμογής. Σε αυτό το αρχείο, δηλώνουμε τις πρόσθετες βιβλιοθήκες που χρειάζεται. Παράλληλα, αποτελεί επίσης και το σημείο στο οποίο θα ρυθμίσουμε το Dependency Injection και το MVVM. Όπως αναλύεται και στον κώδικα παρακάτω, αρχικά σετάρουμε τις βιβλιοθήκες που θα γίνουν χρήση από το MAUI (π.χ. η μέθοδος "UseMauiCommunityToolkit()"). Σε επόμενο στάδιο, δηλώνουμε τα services, τις σελίδες και τα ViewModels αυτών, για χρήση με DI.

```
using CommunityToolkit.Maui;
using GAIAPlatform.Pages;
using GAIAPlatform.Services;
using GAIAPlatform.ViewModels;
using SkiaSharp.Views.Maui.Controls.Hosting;
using DevExpress.Maui;
using Xamarin.CommunityToolkit;

namespace GAIAPlatform;

4 references | unrateddi, 264 days ago | 1 author, 3 changes
public static class MauiProgram
{
    4 references | unrateddi, 264 days ago | 1 author, 3 changes
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UseSkiaSharp(true)
            .UseMauiApp<App>()
            .UseDevExpress()
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
            })
            .UseMauiCommunityToolkit();

        // Register Services
        builder.Services.AddTransient<IGAIAService, GAIAService>();
        builder.Services.AddSingleton<ITranslationService, TranslationService>();

        // Register Pages
        builder.Services.AddTransient<MainPage>();
        builder.Services.AddTransient<ProjectPage>();
        builder.Services.AddTransient<NodePage>();
        builder.Services.AddTransient<NodeOverviewPage>();
        builder.Services.AddTransient<AboutPage>();

        // Register VMs
        builder.Services.AddTransient<MainPageViewModel>();
        builder.Services.AddTransient<ProjectPageViewModel>();
        builder.Services.AddTransient<NodePageViewModel>();
        builder.Services.AddTransient<NodeOverviewPageViewModel>();
        builder.Services.AddTransient<AboutPageViewModel>();

        return builder.Build();
    }
}
```

Εικόνα 10 – MauiProgram.cs

4.3.2 ViewModelBase.cs

Στο σημείο αυτό, δημιουργήσαμε ένα ViewModel το οποίο θα αποτελέσει τη βάση ώστε να στηθούν τα ξεχωριστά ViewModels για την κάθε σελίδα. Αυτό υλοποιήθηκε με σκοπό τη δημιουργία των μεθόδων OnNavigatedTo και OnNavigatedFrom σε ένα σημείο, ώστε τελικά να κληρονομήσουν από εκεί τα υπόλοιπα ViewModels. Αυτές οι μέθοδοι είναι απαραίτητες καθώς θα χρειαστεί να εκτελέσουμε εργασίες κατά την εμφάνιση μίας σελίδας. Πιο αναλυτικά, η λογική έχει ως εξής: όταν το View δημιουργηθεί στην οθόνη, αυτόματα θα εκτελεστεί η μέθοδος OnNavigatedTo του View, την οποία έχουμε κάνει override ώστε να καλέσει την OnNavigatedTo του ViewModel της σελίδας. Ακόμα, έχουμε δηλώσει ότι απαιτείται η παροχή του GAIAService – το οποίο και θα αναλύσουμε σε επόμενο κεφάλαιο - με DI (Dependency Injection) καθώς χρησιμοποιείται στις περισσότερες σελίδες. Το τελευταίο, γίνεται με τη δήλωση του GAIAService, ως παράμετρο στον constructor της κλάσης. Έτσι εξασφαλίζουμε πως θα έχουμε το GAIAService πάντα έτοιμο για χρήση.

```
1   using CommunityToolkit.Mvvm.ComponentModel;
2   using GAIAPlatform.Services;
3
4   namespace GAIAPlatform.ViewModels
5   {
6       [INotifyPropertyChanged]
        12 references | unrateddi, 333 days ago | 1 author, 2 changes
7       public partial class ViewModelBase
8       {
9           #region PRIVATES
10
11          protected readonly IGAIAService _gAIAService;
12
13          #endregion
14
            5 references | unrateddi, 335 days ago | 1 author, 1 change
15          public ViewModelBase(IGAIAService gAIAService)
16          {
17              _gAIAService = gAIAService;
18          }
19
            13 references | unrateddi, 333 days ago | 1 author, 1 change
20          public virtual void OnNavigatedTo()
21          {
22
23          }
24
            0 references | unrateddi, 333 days ago | 1 author, 1 change
25          public virtual void OnNavigatedFrom()
26          {
27
28          }
29      }
30  }
31
```

Εικόνα 11 – ViewModelBase.cs

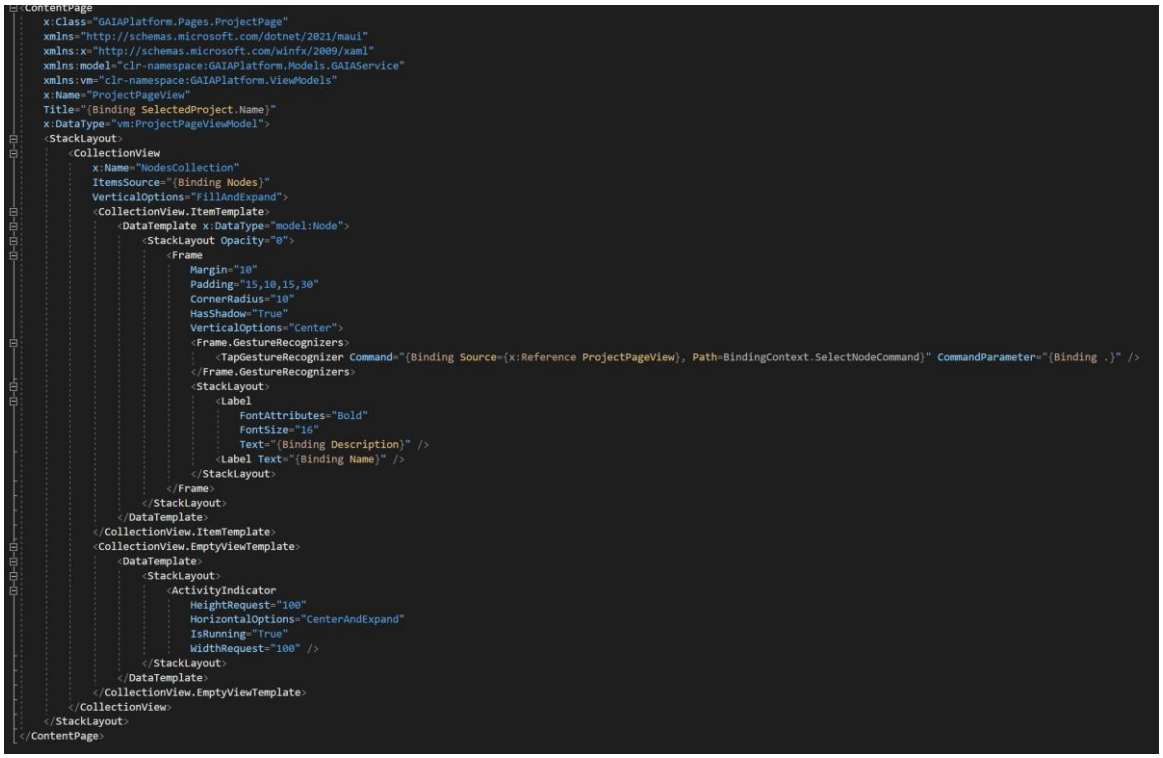
4.3.3 GAIAService

Το GAIAService είναι ένα service που υλοποιήθηκε στα πλαίσια διαχείρισης της επικοινωνίας της εφαρμογής με το back-end. Είναι υπεύθυνο για τις κλήσεις προς το back-end, αλλά και για τον τρόπο με τον οποίο εκτελούνται, καθώς και το authentication και τον μετασχηματισμό των δεδομένων στα μοντέλα της εφαρμογής. Με αυτόν τον τρόπο η υπόλοιπη εφαρμογή - δηλαδή τα Views και τα ViewModels - δεν είναι απαραίτητο να γνωρίζουν ούτε τον τρόπο επικοινωνίας με το back-end, ούτε τα δεδομένα που μπορεί να επιστρέφει το web service και τα οποία δεν ενδιαφέρουν το UI.

4.4 Δείγμα UI Design

Σε αυτό το σημείο, θα παρουσιάσουμε τον τρόπο με τον οποίο συντάσσουμε το UI σε γλώσσα XAML, καθώς και πώς χρησιμοποιούμε τα Data Bindings αλλά και τα Command Interfaces μέσα σε αυτή.

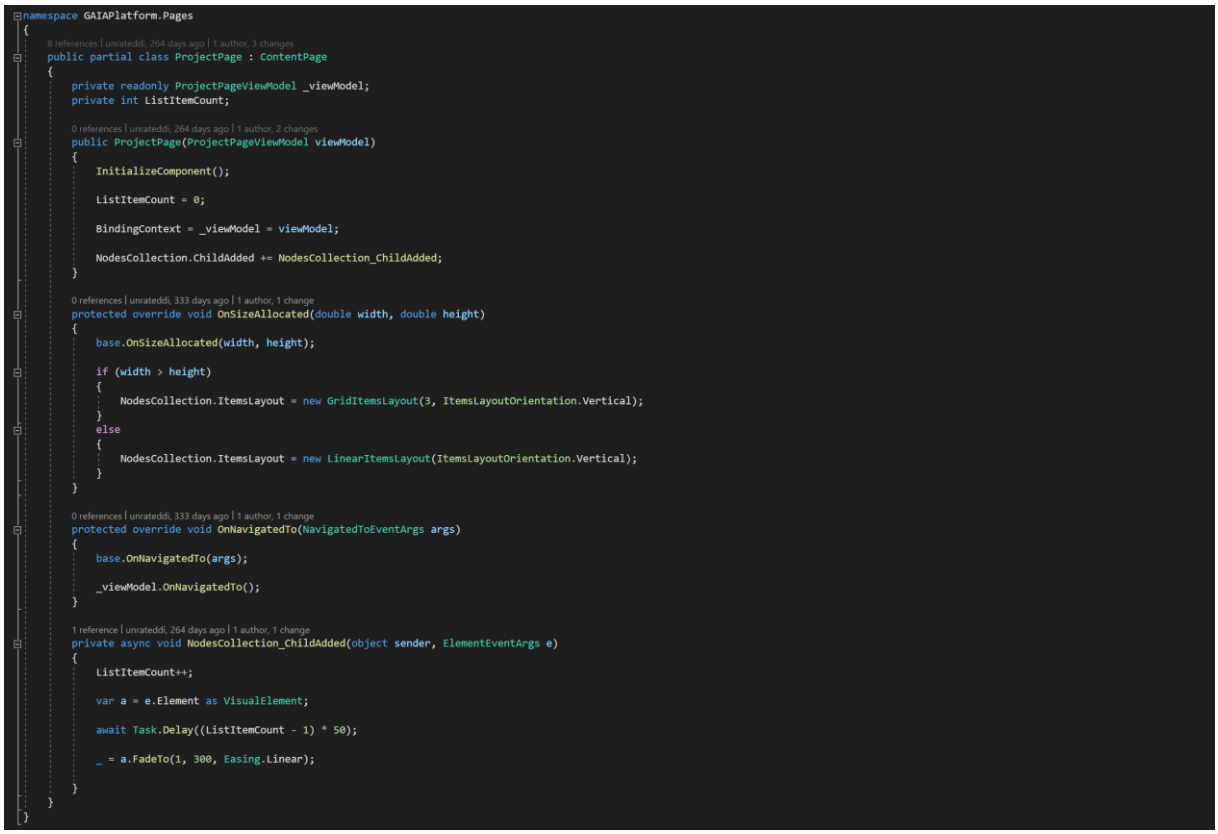
Για παράδειγμα, όπως επιδεικνύεται και στο απόσπασμα του κώδικα στην παρακάτω εικόνα, αφορά τη σελίδα που δείχνει τα nodes, δηλαδή τους μετεωρολογικούς σταθμούς, ενός Project. Αρχικά, χρησιμοποιούμε Data Binding στο πεδίο Title της σελίδας, με σκοπό να το συνδέσουμε με την οντότητα “SelectedProject” στο ViewModel και, συγκεκριμένα, με το πεδίο “Name”. Με αυτόν τον τρόπο, ο τίτλος της σελίδας θα είναι πάντα το όνομα του Project που έχει επιλεγεί από τον χρήστη σε κάθε περίπτωση. Ακόμα, χρησιμοποιούμε ένα UI element τύπου “CollectionView” με Data Binding στο πεδίο ItemsSource και, το δένουμε με την οντότητα Nodes του ViewModel. Κατ’ αυτό τον τρόπο, θα γεμίσει δυναμικά η σελίδα, με το πλήθος των Nodes και τις πληροφορίες τους, βάσει του template εμφάνισης που ορίζουμε στο πεδίο ItemTemplate. Τέλος, και με την αξιοποίηση των Command Interfaces, δημιουργούμε ένα TapGestureRecognizer - το οποίο και εκτελεί ένα προκαθορισμένο από εμάς action κατά το άγγιγμα (Tap) του στοιχείου - για κάθε node που θα εμφανιστεί στη σελίδα. Έπειτα, το δένουμε με το SelectNodeCommand του ViewModel, αποδίδοντάς του σαν παράμετρο την ίδια οντότητα του Node που επιλέχθηκε.



Εικόνα 12 – Δείγμα από το XAML αρχείο της σελίδας ProjectPage View

4.5 Δείγμα Code-Behind

Στον παρακάτω κώδικα επιδεικνύουμε την διαδικασία που πρέπει να ακολουθηθεί, ώστε να γίνει Bind το View με το ViewModel του. Φέρνοντας με DI το ViewModel στον constructor της σελίδας, το καταχωρούμε σε μία τοπική μεταβλητή για επαναχρησιμοποίηση και έπειτα στο BindingContext της σελίδας. Ακόμα, βλέπουμε το override της OnNavigatedTo ώστε να γίνει και η κλήση της μεθόδου OnNavigatedTo του ViewModel, όπως αναφέραμε και σε προηγούμενο κεφάλαιο. Παρατηρούμε επιπρόσθετα, την NodesCollection_ChildAdded, η οποία αποτελεί μία μέθοδο που υλοποιήθηκε για τις ανάγκες custom animation κατά την εμφάνιση στοιχείων στο CollectionView. Ουσιαστικά, η μέθοδος αυτή γίνεται attach στο Event ChildAdded του CollectionView και ο κώδικας που εμπεριέχει, θα τρέχει για κάθε στοιχείο που προστίθεται σε αυτό. Έτσι, δίνεται η δυνατότητα, μεταξύ άλλων, να επηρεάσουμε τον τρόπο εμφάνισης του στοιχείου.

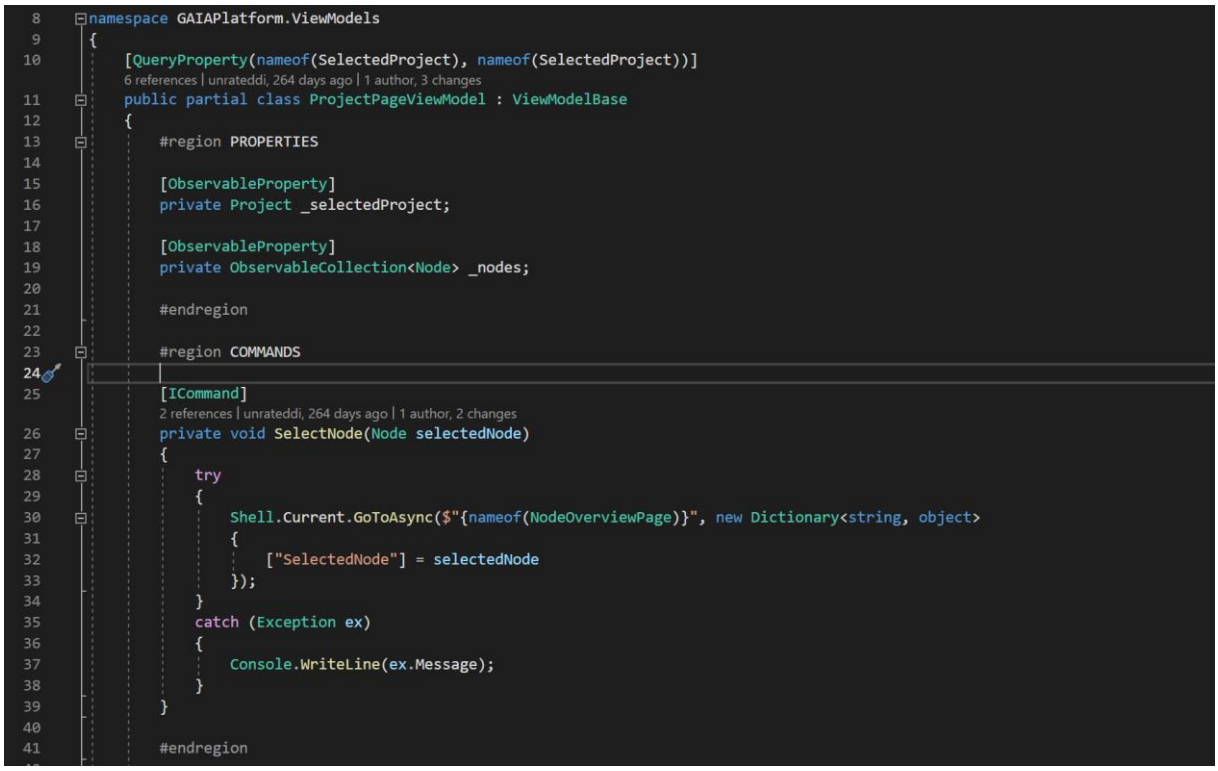


```
namespace GAIAPlatform.Pages
{
    8 references | unrateddi, 264 days ago | 1 author, 3 changes
    public partial class ProjectPage : ContentPage
    {
        private readonly ProjectPageViewModel _viewModel;
        private int ListItemCount;

        0 references | unrateddi, 264 days ago | 1 author, 2 changes
        public ProjectPage(ProjectPageViewModel viewModel)
        {
            InitializeComponent();

            ListItemCount = 0;

            BindingContext = _viewModel = viewModel;

            NodesCollection.ChildAdded += NodesCollection_ChildAdded;
        }

        0 references | unrateddi, 333 days ago | 1 author, 1 change
        protected override void OnSizeAllocated(double width, double height)
        {
            base.OnSizeAllocated(width, height);

            if (width > height)
            {
                NodesCollection.ItemsLayout = new GridItemsLayout(3, ItemsLayoutOrientation.Vertical);
            }
            else
            {
                NodesCollection.ItemsLayout = new LinearItemsLayout(ItemsLayoutOrientation.Vertical);
            }
        }

        0 references | unrateddi, 333 days ago | 1 author, 1 change
        protected override void OnNavigatedTo(NavigatedToEventArgs args)
        {
            base.OnNavigatedTo(args);

            _viewModel.OnNavigatedTo();
        }

        1 reference | unrateddi, 264 days ago | 1 author, 1 change
        private async void NodesCollection_ChildAdded(object sender, ElementEventArgs e)
        {
            ListItemCount++;

            var a = e.Element as VisualElement;

            await Task.Delay((ListItemCount - 1) * 50);

            _ = a.FadeTo(1, 300, Easing.Linear);

        }
    }
}
```

Εικόνα 13 – Code behind δείγμα του ProjectPage View

4.6 Πλοήγηση σε σελίδες και μεταφορά δεδομένων

Το .NET MAUI με την χρήση του AppShell μας δίνει την δυνατότητα για εύκολη πλοήγηση στις σελίδες της εφαρμογής μεταφέροντας ταυτόχρονα (με το navigation), δεδομένα από την μία σελίδα στην άλλη.



```
namespace GAIAPlatform.ViewModels
{
    [QueryProperty(nameof(SelectedProject), nameof(SelectedProject))]
    6 references | unrateddi, 264 days ago | 1 author, 3 changes
    public partial class ProjectPageViewModel : ViewModelBase
    {
        #region PROPERTIES

        [ObservableProperty]
        private Project _selectedProject;

        [ObservableProperty]
        private ObservableCollection<Node> _nodes;

        #endregion

        #region COMMANDS

        [ICommand]
        2 references | unrateddi, 264 days ago | 1 author, 2 changes
        private void SelectNode(Node selectedNode)
        {
            try
            {
                Shell.Current.GoToAsync($"{nameof(NodeOverviewPage)}", new Dictionary<string, object>
                {
                    ["SelectedNode"] = selectedNode
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        #endregion
```

Εικόνα 14 – Δείγμα από το ProjectPageViewModel

Στο παραπάνω δείγμα κώδικα του ProjectPageViewModel, παρατηρούμε πόσο εύκολα μπορούμε να περάσουμε δεδομένα στην επόμενη σελίδα που θέλουμε να πλοηγηθούμε, δημιουργώντας ένα απλό Dictionary, με κλειδί, τύπου String και value, τύπου object. Η επόμενη σελίδα θα πρέπει να ξέρει τι τύπου δεδομένα περιμένει να λάβει, καθώς το object type θα πρέπει να γίνει unboxed, ώστε να χρησιμοποιηθεί σωστά. Ωστόσο το unboxing το αναλαμβάνει το ίδιο το Navigation του AppShell. Για το λόγο αυτό, το μόνο που χρειάζεται να κάνουμε είναι να δηλώσουμε σαν attribute στην κλάση του ViewModel μας, σε ποιά μεταβλητή θα πρέπει να περάσει την τιμή που θα βρει με το όνομα που έχουμε ορίσει. Έτσι, το AppShell, θα ψάξει στις παραμέτρους που εισαγάγαμε από την προηγούμενη σελίδα, να βρει το κλειδί που του δηλώθηκε και θα προσπαθήσει να το κάνει unbox, στον τύπο που είναι ορισμένη η μεταβλητή που ορίσαμε. Επίσης, βλέπουμε πως οι μεταβλητές _selectedProject και _nodes είναι private και, θεωρητικά, δεν θα υπάρχει καν η πρόσβαση σε αυτές από άλλο σημείο της εφαρμογής – π.χ. το αντίστοιχο View - αφού είναι εκτός του scope του ViewModel. Ωστόσο, δηλώνοντας του, το attribute ObservableProperty, λειτουργεί από πίσω Code Generator ο οποίος θα δημιουργήσει τα public πεδία τους και επιπλέον θα δώσει την λειτουργικότητα που χρειάζεται για τα Data-Bindings. Συνεπώς, αφού στο XAML έχουμε δηλώσει Data-Binding με την μεταβλητή SelectedProject, το UI

της εφαρμογής θα ενημερωθεί αυτόματα, εφόσον το AppShell θα έχει καταφέρει να περάσει δεδομένα σε αυτή τη μεταβλητή.

5 Δυσκολίες που αντιμετωπίστηκαν κατά την υλοποίηση

Κατά την υλοποίηση της εφαρμογής αντιμετωπίστηκαν 3 κύριες δυσκολίες:

5.1 NET MAUI Breaking Changes

Την περίοδο που ξεκίνησε η υλοποίηση το .NET MAUI ήταν ακόμα σε pre-release. Αυτό σημαίνει ότι λάμβαναν χώρα πολλές αλλαγές στα APIs του, παράλληλα με τον σχεδιασμό. Συχνά, μάλιστα, αυτές οι αλλαγές ήταν σημαντικές και «έσπαγαν» την υλοποίηση. Ως αποτέλεσμα, ορισμένες φορές αδυνατούσε να γίνει compile το πρόγραμμα, καθώς και να διορθωθεί το εκάστοτε πρόβλημα που παρουσιαζόταν. Πιο συγκεκριμένα, και καθώς παρουσιαζόταν ασυμβατότητα σε ορισμένα πακέτα που χρησιμοποιούσε το .NET MAUI, χρειάστηκε να δημιουργηθεί το project εκ νέου και να μεταφέρουμε την υπάρχουσα υλοποίηση. Στο μεσοδιάστημα, ωστόσο, έγινε publish η Release Candidate έκδοση του MAUI, η οποία υποσχόταν ότι δεν θα υπάρξουν άλλες σοβαρές αλλαγές και ότι τα APIs του .NET MAUI είναι καταληκτικά, όπως και πράγματι διαπιστώθηκε.

5.2 Υποστήριξη διάφορων οθονών

Μία ακόμα δυσκολία που αντιμετωπίσαμε, ήταν ότι η εφαρμογή κυρίως στόχευε σε tablet οθόνες. Αυτό είχε ως αποτέλεσμα, το UI σε μικρότερες οθόνες, όπως αυτές των κινητών, να «σπάει» ή και να δείχνει παραμορφωμένο ή και σε άλλες περιπτώσεις να βγαίνει εκτός των ορατών ορίων της οθόνης. Αυτό αντιμετωπίστηκε, υλοποιώντας τεχνικές προσαρμογής του UI βάσει του aspect ratio και του μεγέθους της οθόνης. Κάνοντας override, λοιπόν, την OnSizeAllocated μέθοδο που μας δίνει το .NET MAUI σε κάθε σελίδα, έγινε υλοποίηση για αναδιάταξη ή αλλαγή εμφάνισης των σχετικών elements. Πρόσθετα, θα μπορούσε να χρησιμοποιηθεί η OnIdiom που προσφέρει επίσης το .NET MAUI, για να

αναγκάσουμε την εφαρμογή να σχεδιάσει διαφορετικά elements, καλύπτοντας κάθε τύπο συσκευής (Tablet/Phone).

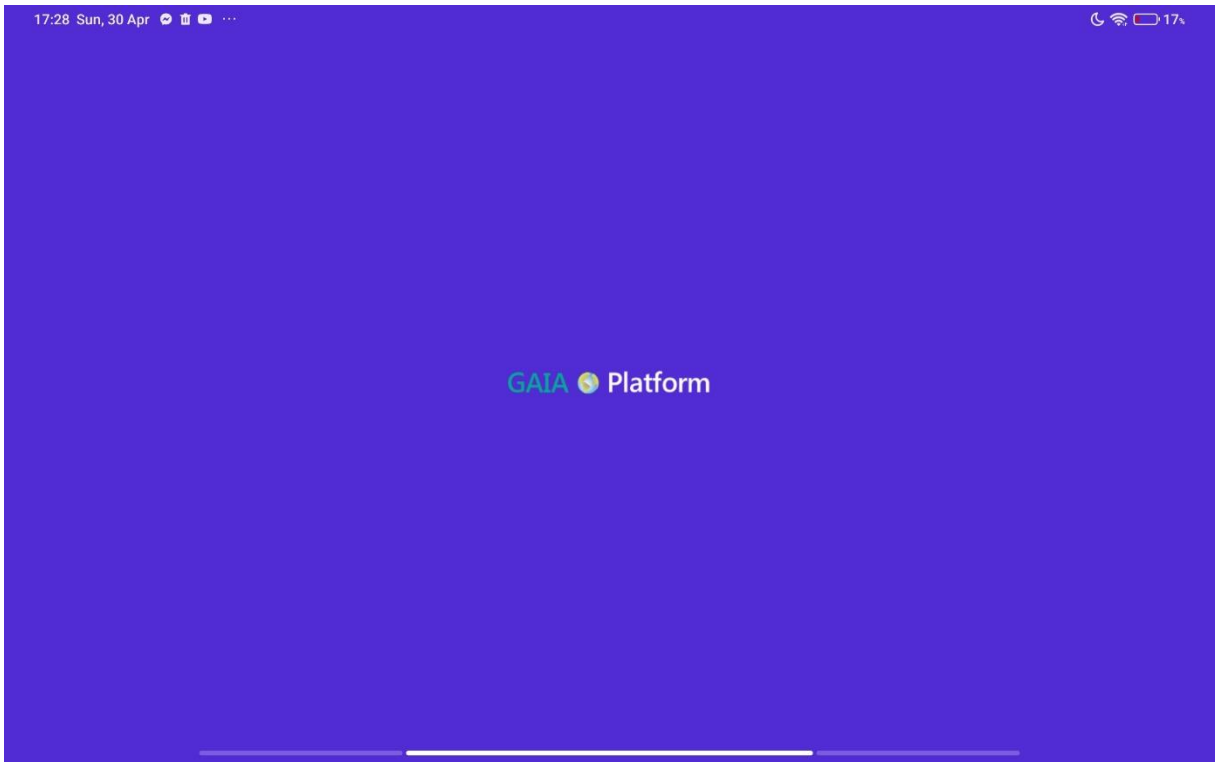
5.3 Bug στην βιβλιοθήκη LiveCharts2

Τελευταίο πρόβλημα που αντιμετωπίστηκε, είναι ένα bug που παρατηρήθηκε στη βιβλιοθήκη LiveCharts2, κατά το χρονικό διάστημα υλοποίησης της εφαρμογής. Στην σελίδα που εμφανίζονται αναλυτικά τα διαγράμματα για όλες τις μετρήσεις από τον επιλεγμένο μετεωρολογικό σταθμό, η βιβλιοθήκη αδυνατούσε να αποδόσει σωστά όλα τα δεδομένα, ως εκ τούτου πολλές φορές η σελίδα γινόταν μη λειτουργική. Για το λόγο αυτό, αξιοποιήθηκε η βιβλιοθήκη της DevExpress, που παρόλο που δεν προσφέρει την ποικιλία των features που διαθέτει η πρώτη, καλύπτει πλήρως τις ανάγκες της συγκεκριμένης σελίδας.

6 Παρουσίαση εφαρμογής

6.1 Splash Screen

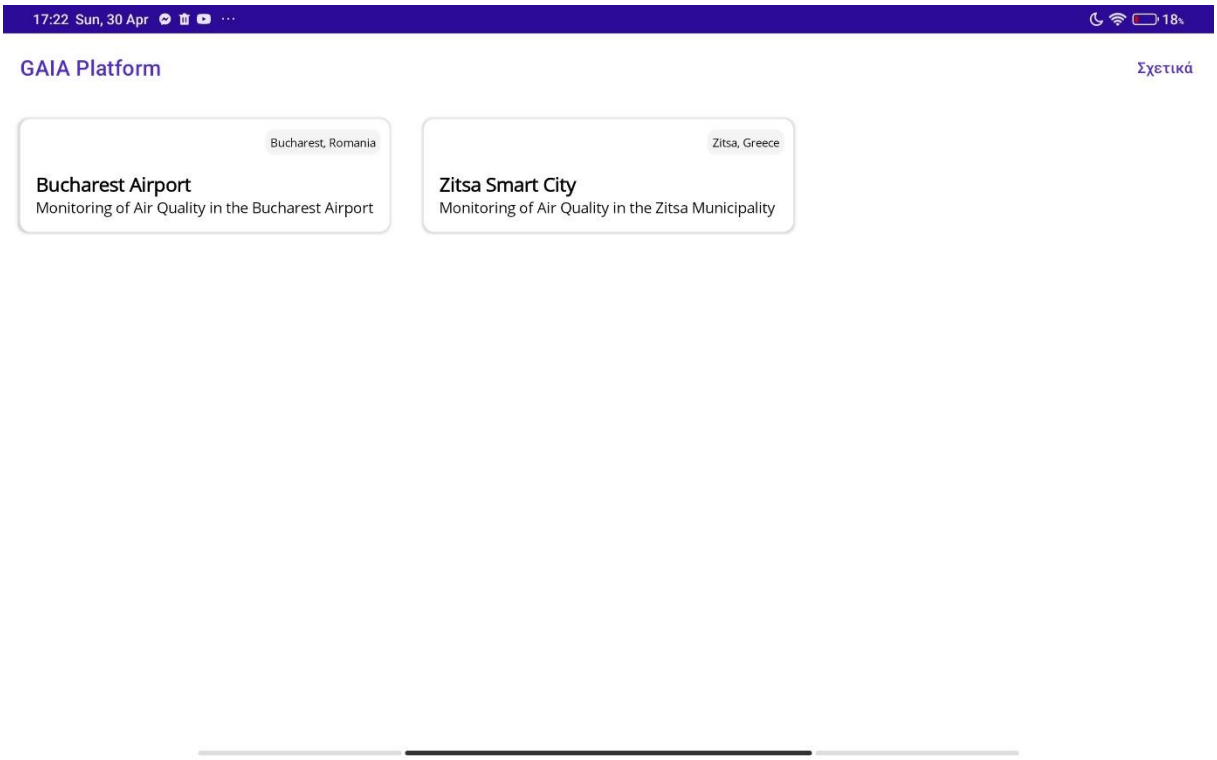
Για την οθόνη Splash Screen, που ουσιαστικά αποτελεί την οθόνη εκκίνησης της εφαρμογής σε μία συσκευή, χρησιμοποιήσαμε ένα απλό design με το logo του GAIAPlatform να ζει στο κέντρο της σελίδας.



Εικόνα 15 – Splash Screen

6.2 Main Page

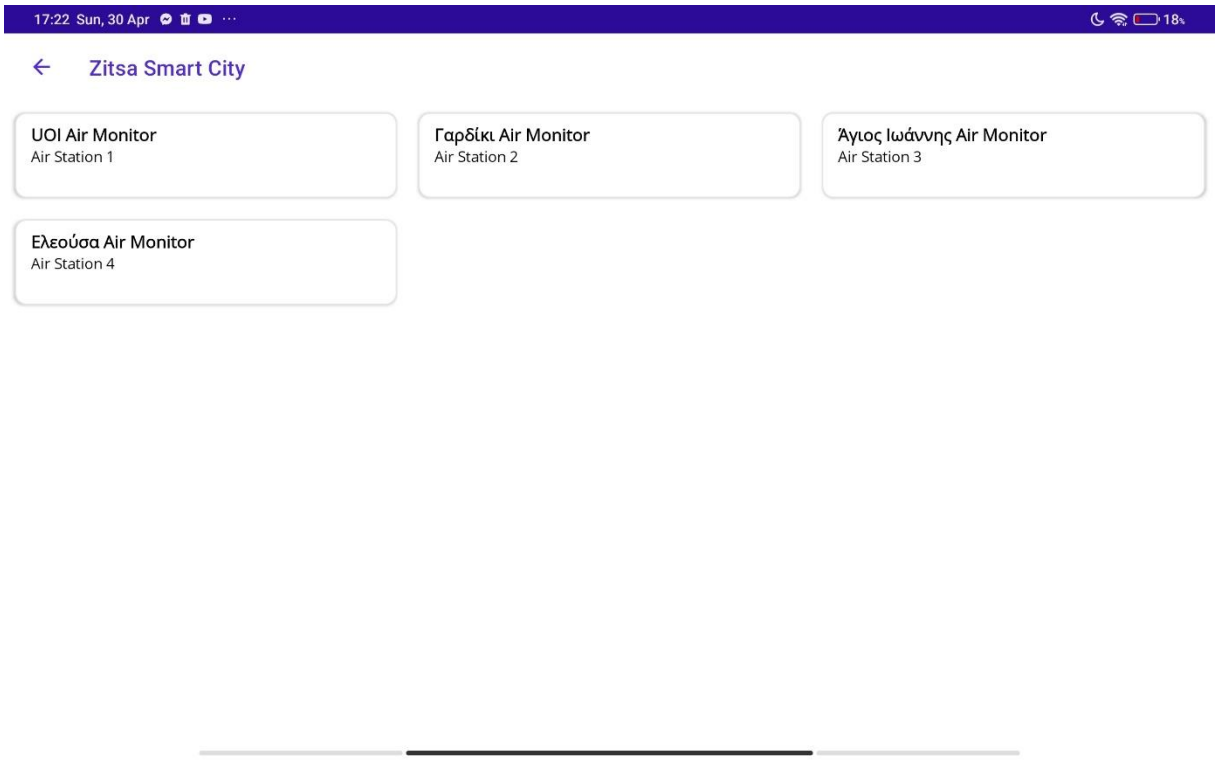
Στην πρώτη οθόνη, αφού φορτώσει η εφαρμογή μας, υπάρχει μία λίστα από τα διαθέσιμα Projects του GAIAPlatform. Κατά την εμφάνιση της σελίδας η εφαρμογή καλεί το web service του GAIAPlatform για να ζητήσει τα δεδομένα. Με την ολοκλήρωση της κλήσης, το GAIAService της εφαρμογής, επιστρέφει στην σελίδα τα δεδομένα για τα διαθέσιμα Projects και η οθόνη, αναλαμβάνει να τα αποδόσει με το design που έχουμε ορίσει. Κατά την αφή σε κάποιο από τα διαθέσιμα projects, η εφαρμογή θα μας ανακατευθύνει στην ProjectPage, μεταφέροντας της το Project που επιλέγεται από τον χρήστη.



Εικόνα 16 – Main Page (Projects)

6.3 Project Page

Η δεύτερη σελίδα της εφαρμογής, είναι η αυτή που δείχνει τα διαθέσιμα Nodes ή, με άλλα λόγια, τους μετεωρολογικούς σταθμούς του επιλεγμένου Project. Κατά το φόρτωμα, η σελίδα λαμβάνει το επιλεγμένο Project από το AppShell και, στη συνέχεια, καλεί το GAIAService με παράμετρο το Id του επιλεγμένου Project, ώστε να λάβει πίσω ως απάντηση τα διαθέσιμα Nodes. Κατά την επιλογή ενός από τα nodes, η εφαρμογή θα μας κατευθύνει στην οθόνη με τις λεπτομέρειες του μετεωρ. σταθμού, μεταφέροντάς της ως παράμετρο το επιλεγμένο Node.

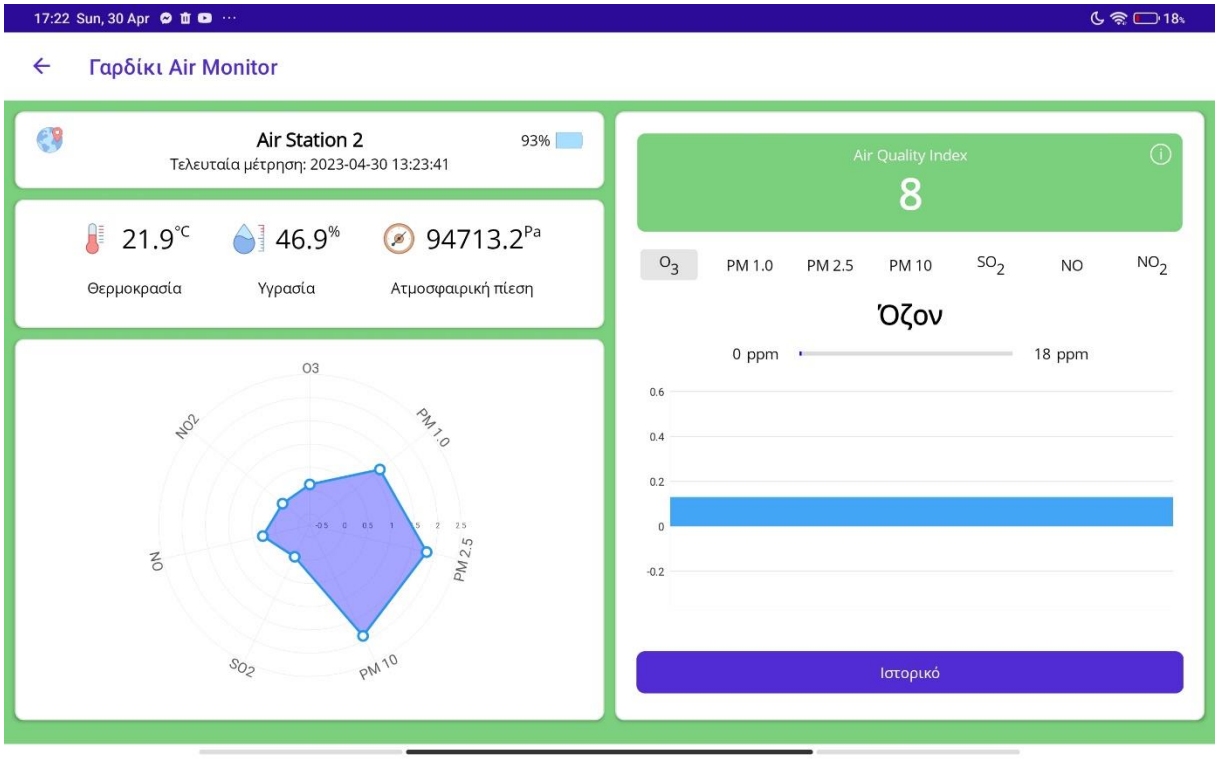


Εικόνα 17 – Project Page (Nodes)

6.4 Node Overview Page

Η συγκεκριμένη σελίδα αποτελεί και την κύρια σελίδα της εφαρμογής. Είναι η σελίδα που αναλαμβάνει να αξιοποιήσει τις μετρήσεις και τις πληροφορίες του επιλεγμένου σταθμού και να τις αναδείξει. Κατά την φόρτωση αυτής της σελίδας, μεταφέρεται το επιλεγμένο Node από το AppShell και, στη συνέχεια, η σελίδα ζητάει από το GAIAService να φέρει από το back-end τις πληροφορίες του. Σε αυτό το βήμα, ενεργοποιούνται στοιχεία σε όλα τα πεδία της σελίδας, με σκοπό να αναδειχθεί στο χρήστη, ότι η εφαρμογή είναι σε κατάσταση αναμονής, ως προς την απάντηση που περιμένει από το web service. Το τελευταίο, υλοποιήθηκε καθώς η συγκεκριμένη κλήση, μπορεί να διαρκέσει από κάποια χιλιοστά του δευτερολέπτου, έως και λίγα δευτερόλεπτα. Η σελίδα είναι γεμάτη με ενδείξεις, διαγράμματα και πληροφορίες. Βάσει των πληροφοριών του σταθμού, υπολογίζεται και το Air Quality Index, σύμφωνα με το οποίο, αποφασίζεται, τόσο το χρώμα του background όσο και του πεδίου που φαίνεται το AQI. Για το χρώμα, να αναφέρουμε συνοπτικά, πως έχουν οριστεί 6 συγκεκριμένες κλίμακες βάσει και του βαθμού επικινδυνότητας και του επιπέδου μόλυνσης του αέρα, όπως έχει οριστεί από την *Airnow.gov*. Δίνεται, επίσης, δυνατότητα να

εντοπίσουμε τον σταθμό (node) βάσει των συντεταγμένων που μας έχει δώσει το back-end. Πατώντας, επομένως, στο εικονίδιο με την υδρόγειο (), ο χρήστης ανακατευθύνεται σε άλλη αντίστοιχη διαθέσιμη εφαρμογή χαρτών για τον εντοπισμό. Δίνεται, επίσης, στο χρήστη, η δυνατότητα προβολής γραφημάτων των δεικτών ρύπων συνολικά στο κάτω και αριστερά μέρος της σελίδας, ενώ, δεξιά, μπορούμε να δούμε σε λεπτομέρεια μία περιορισμένη ιστορικότητα, ανά επιλεγμένο δείκτη. Υπάρχουν, επιπρόσθετα, διαθέσιμες και πιο συγκεκριμένες πληροφορίες για το AQI στην σελίδα Info Popup Page την οποία μπορούμε να καλέσουμε πατώντας στο κουμπί . Τέλος το κουμπί «Ιστορικό» (Ιστορικό), μας κατευθύνει στην οθόνη με όλα τα γραφήματα των μετρήσεων, περνώντας τις μετρήσεις που ήδη έχει ανακτήσει η εφαρμογή από το back-end.

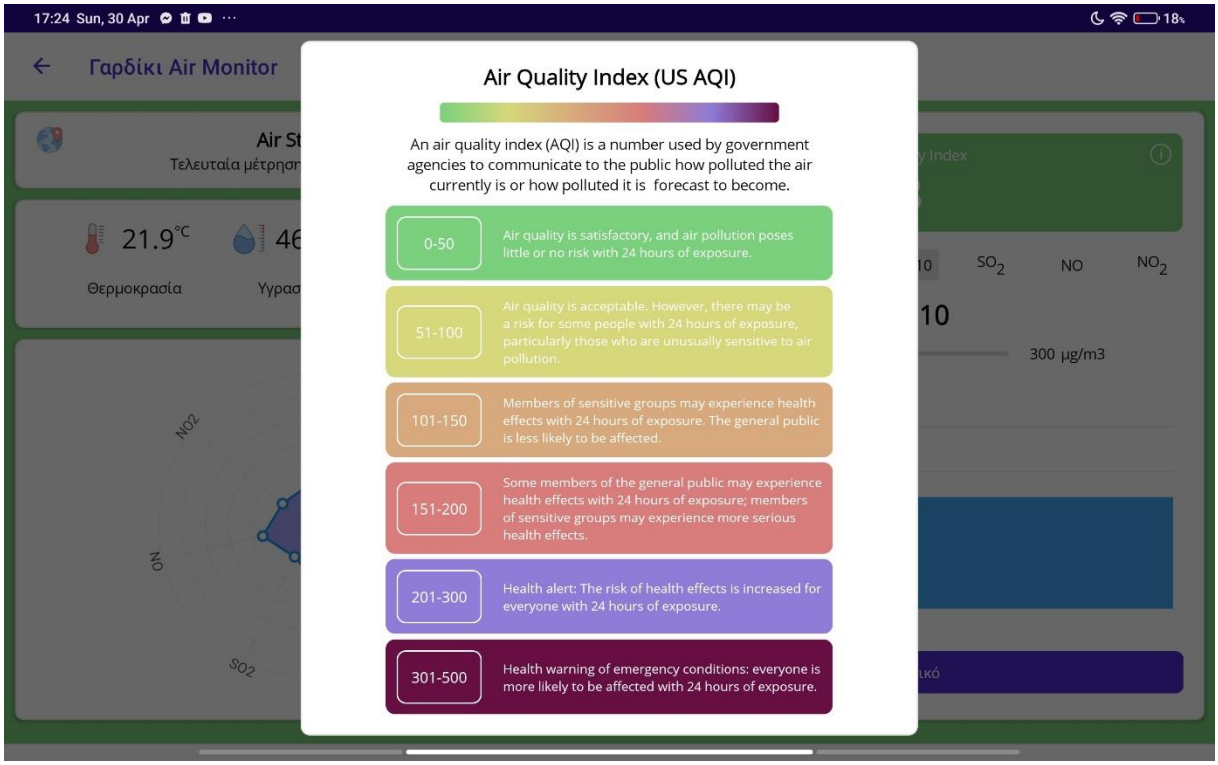


Εικόνα 18 – Node Overview Page

6.5 Info Popup Page

Το Info Popup Page, είναι μία ιδιαίτερη σελίδα καθώς δεν χρειάζεται ViewModel, αφού δεν σχετίζεται με business logic ή με ανταλλαγή δεδομένων, αλλά αποσκοπεί καθαρά στην εμφάνιση μιας στατικής σελίδας. Σκοπός της είναι, η εύκολη ενημέρωση προς τον χρήστη

σχετικά με τα επίπεδα του AQI και τις επιπτώσεις του στην υγεία, ανά εύρος. Η σελίδα αυτή, εμφανίζεται πάνω από την ήδη υπάρχουσα σελίδα, με ένα σκούρο ημιδιάφανο background για να επιδεικνύει στον χρήστη, ότι δεν έχει αποχωρήσει από την κύρια σελίδα, αλλά βλέπει κάποιες πληροφορίες πάνω σε αυτή ως overlay. Πατώντας το back hardware πλήκτρο της συσκευής ή οπουδήποτε στο background, εξαφανίζεται το popup και έρχεται στο προσκήνιο πάλι η σελίδα με τις ενδείξεις.

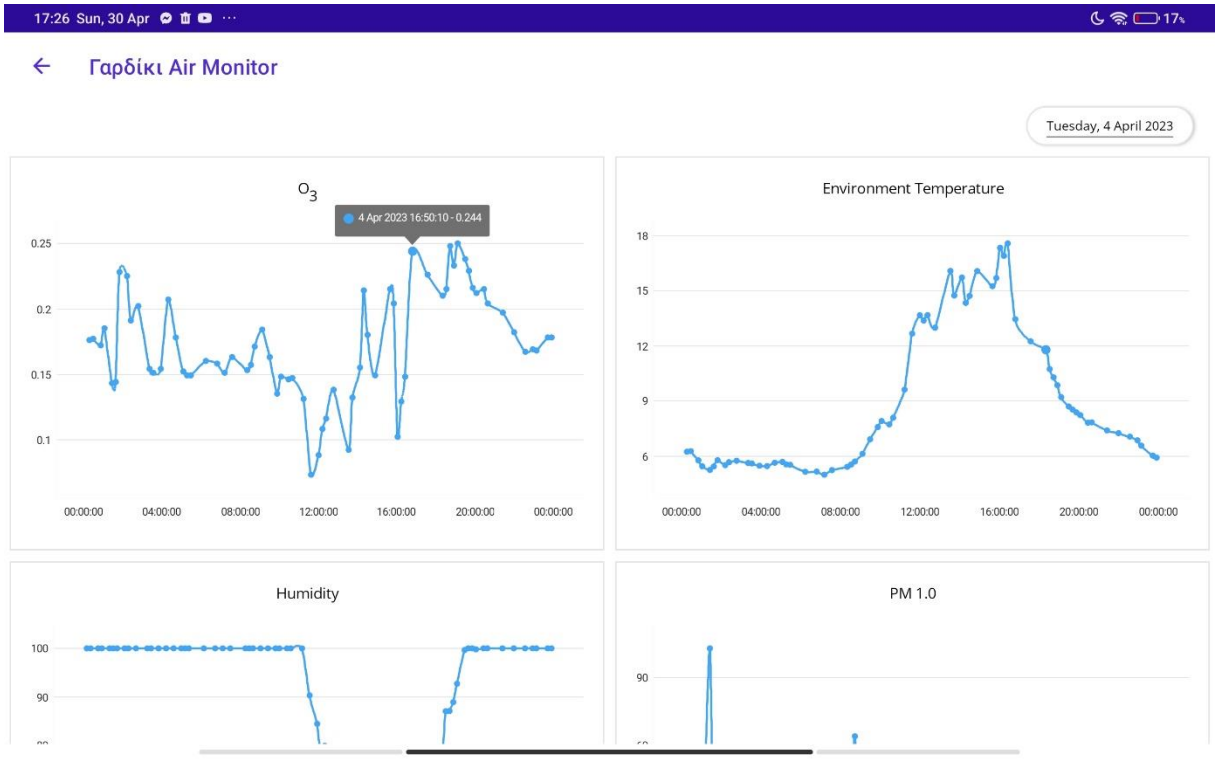


Εικόνα 19 – Info Popup Page (Air Quality Index info)

6.6 Node Page

Στην Node Page υπάρχει μία λίστα που αποδίδει γραφήματα για όλες τις μετρήσεις που έχει ο σταθμός, με ιστορικότητα της επιλεγμένης ημερομηνίας για όλη την ημέρα. Υπάρχει το πεδίο της ημερομηνίας στο οποίο πατώντας κανείς, ενεργοποιείται η επιλογή ημερομηνίας από το native date picker της εκάστοτε πλατφόρμας και, κατά τον ορισμό, η εφαρμογή ζητάει από το GAIAService τις μετρήσεις για την ημερομηνία αυτή. Επίσης, πατώντας πάνω σε κάποια κουκίδα ενός γραφήματος μπορούμε να δούμε την ακριβή ώρα

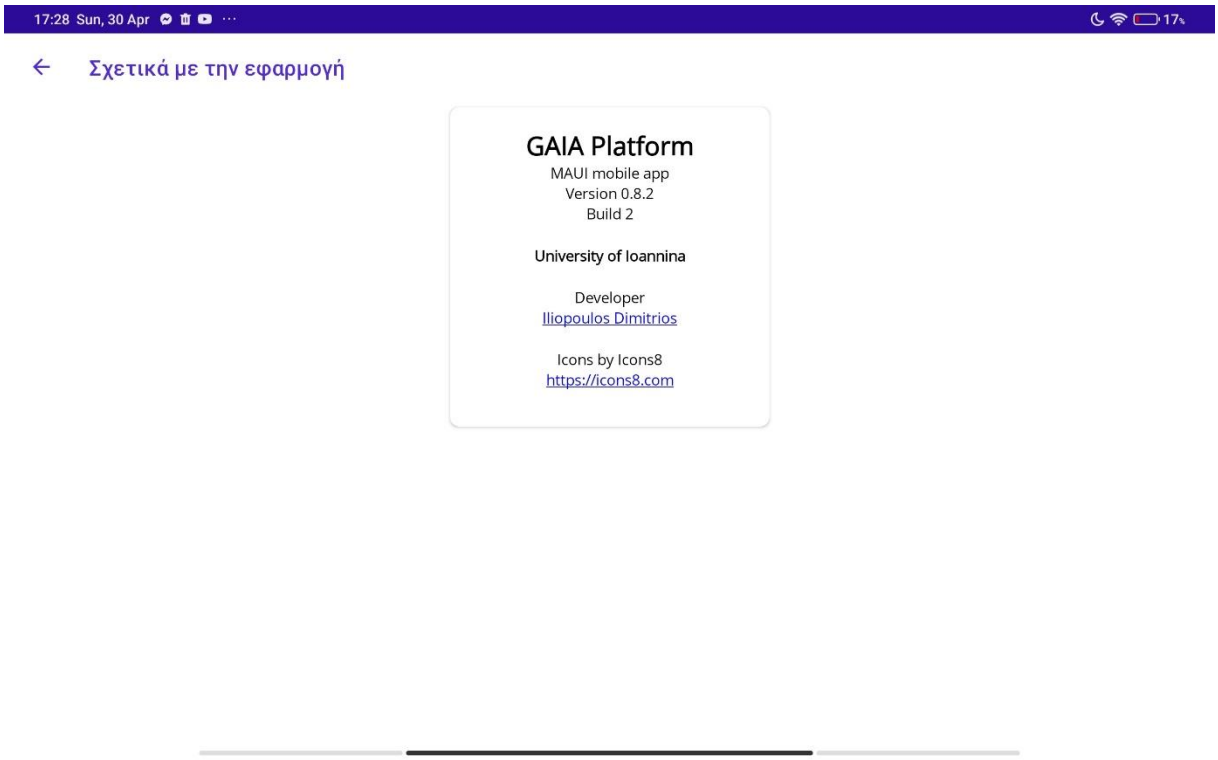
και τον δείκτη μέτρησης με την αντίστοιχη τιμή του, από τον σταθμό, για το στοιχείο που απεικονίζεται στο συγκεκριμένο γράφημα.



Εικόνα 20 – Node Page (Αναλυτικά γραφήματα μετρήσεων)

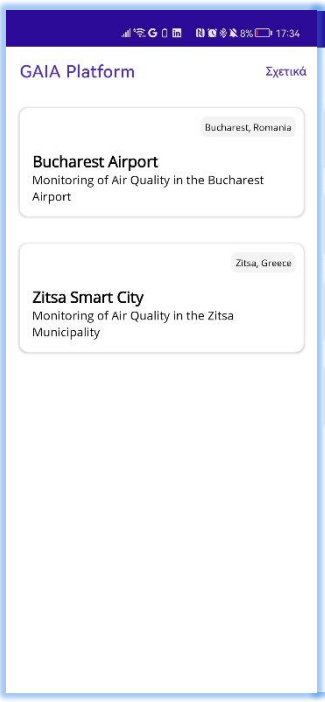
6.7 About Page

Τελευταία σελίδα της εφαρμογής είναι το About Page, στο οποίο μπορούμε να πλοηγηθούμε από την Main Page πατώντας στην επιλογή «Σχετικά» (Σχετικά) στο πάνω και δεξιά μέρος της οθόνης. Αυτή η σελίδα, περιέχει πληροφορίες σχετικά με την έκδοση της εφαρμογής, το ίδρυμα στο οποίο διεκπεραιώνεται η πτυχιακή εργασία που αφορά την εφαρμογή, το όνομα του developer, με link στο προφίλ του, καθώς και αναφορά στο Icons8, καθώς αυτό απαιτούνταν, λόγω των όρων χρήσης της δωρεάν συλλογής τους.

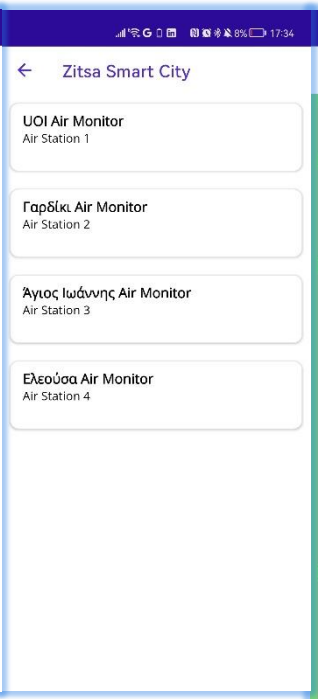


Εικόνα 21 – About Page

6.8 Εικόνες της εφαρμογής από smartphone σε κάθετη οθόνη



Εικόνα 25 – Main Page (Smartphone Vertical)



Εικόνα 23 – Project Page (Smartphone Vertical)

Εικόνα 24 – Node Overview Page (Smartphone Vertical Scrollshot)

Εικόνα 22 – Node Page (Smartphone Vertical)

Βιβλιογραφία

[1] Global market share held by operating systems for desktop PCs, from January 2013 to January 2023
https://www.statista.com/statistics/218089/global-market-share-of-windows-7/

[2] Native Apps vs. Web Apps
https://www.lifewire.com/native-apps-vs-web-apps-2373133

[3] Υπουργείο Περιβάλλοντος και Ενέργειας – Αιθαλομίχλη
https://ypen.gov.gr/perivallon/poiotita-tis-atmosfairas/aithalomichli/

[4] What is Air Quality Index (AQI) & How Is It Calculated?
https://www.pranaair.com/blog/what-is-air-quality-index-aqi-and-its-calculation/

[5] European Air Quality Index
https://airindex.eea.europa.eu/Map/AQI/Viewer/

[6] Airnow.gov
https://www.airnow.gov/aqi/aqi-basics/

[7] .NET MAUI Documentation
https://learn.microsoft.com/en-gb/dotnet/maui/?WT.mc_id=dotnet-35129-website&view=net-maui-7.0

[8] Microsoft Learn - .NET MAUI Navigation
https://learn.microsoft.com/en-us/dotnet/architecture/maui/navigation

[9] C# | Modern, open-source programming language for .NET
https://dotnet.microsoft.com/en-us/languages/csharp

[10] Microsoft Learn - .NET MAUI XAML
https://learn.microsoft.com/en-us/dotnet/maui/xaml/

[11] Microsoft Learn - .NET MAUI Data bindings
https://learn.microsoft.com/en-us/dotnet/maui/fundamentals/data-binding/basic-bindings

[12] Microsoft Learn - Model-View-ViewModel (MVVM)
https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

[13] Presentation Model
https://martinfowler.com/eaaDev/PresentationModel.html

[14] Microsoft Learn - .NET MAUI Get started with XAML - Anatomy of a XAML file
https://learn.microsoft.com/en-us/dotnet/maui/xaml/fundamentals/get-started#anatomy-of-a-xaml-file

[15] Microsoft Learn - .NET MAUI Dependency Injection
https://learn.microsoft.com/en-us/dotnet/architecture/maui/dependency-injection

[16] Codecademy - What Is Back End?
https://www.codecademy.com/resources/blog/what-is-back-end/

[17] W3C - Web Services Overview
https://www.w3.org/TR/ws-arch/

[18] W3Schools - JavaScript JSON
https://www.w3schools.com/js/js_json.asp

[19] Microsoft Learn - Understanding HTTP Authentication
https://learn.microsoft.com/en-us/dotnet/framework/wcf/feature-details/understanding-http-authentication

[20] Microsoft Visual Studio 2022
https://visualstudio.microsoft.com/vs/

[21] .NET Community Toolkit Introduction
https://learn.microsoft.com/en-us/dotnet/communitytoolkit/introduction

[22] Επίσημη ιστοσελίδα της DevExpress
https://www.devexpress.com/

[23] Άρθρο της Wikipedia για την DevExpress
https://en.wikipedia.org/wiki/DevExpress

[24] Επίσημη ιστοσελίδα Livecharts2
https://www.lvcharts.com/

[25] Επίσημη ιστοσελίδα Json.NET (Newtonsoft.Json)
https://www.newtonsoft.com/json

[26] What's new in .NET 7

https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-7