ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Εφαρμογής Web σε Σύγχρονο Περιβάλλον

Γιώργος Παλαιολόγος

Επιβλέπων καθηγητής: Τζήμας Γιάννης

Πάτρα – Νοέμβριος 2024

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή
2. Ονοματεπώνυμο, Υπογραφή
3. Ονοματεπώνυμο, Υπογραφή

Αφιέρωση

Στην οικογένεια μου.

Ευχαριστίες

Ευχαριστώ τον καθηγητή μου, κύριο Γιάννη Τζήμα για την καθοδήγηση και υποστήριξη στην εκπόνηση αυτής της εργασίας. Την οικογένεια μου και τα αγαπημένα μου πρόσωπα για την στήριξη όλα αυτά τα χρόνια, χωρίς την οποία δεν θα είχα καταφέρει να ολοκληρώσω τον κύκλο των σπουδών μου. Επίσης, θέλω να ευχαριστήσω τους ανθρώπους που λειτούργησαν ως μέντορες στην πορεία μου ως προγραμματιστής και τους ανθρώπους που μου έδειξαν εμπιστοσύνη δίνοντας μου έτσι την ευκαιρία να αναπτύξω τις δεξιότητες μου.

Περιεχόμενα

Λίστα Σχημάτων

Δεν υπάρχουν καταχωρημένα σχήματα

Λίστα Πινάκων

Λίστα Εικόνων

Πρόλογος

Έχουν περάσει περισσότερα από 30 χρόνια, από τότε που μια ομάδα επιστημόνων του CERN (1989), με πρωτεργάτη τον φυσικό Tim Berners-Lee, δημιούργησαν το πρωτόκολλο επικοινωνίας HTTP (HyperText Transfer Protocol), το οποίο καθιέρωσε τον τρόπο επικοινωνίας μεταξύ ενός διακομιστή (server) και ενός χρήστη (client). Ήταν αυτό το πρώτο βήμα, σε συνδυασμό με τη δημιουργία της HTML (HyperText Markup Language) και των URL (Uniform Resource Locator) που καθιέρωσαν τον τρόπο λειτουργίας του Παγκόσμιου Ιστού, ο οποίος επέτρεψε για πρώτη φορά στον χρήστη ενός τερματικού να αποκτήσει πρόσβαση σε πληροφορίες που έχουν καταχωρηθεί στο Διαδίκτυο. Οι πρώτοι περιηγητές άρχισαν να εμφανίζονται σταδιακά ενσωματώνοντας λειτουργίες που διευκόλυναν τον χρήστη και επίλυαν τεχνικά προβλήματα. Με την ευρύτητα και συνεπώς την άνοδο ενδιαφέροντος που απέκτησε ο Παγκόσμιος Ιστός, μεγάλες εταιρείες στον χώρο της πληροφορικής δυναμικά κέρδισαν μερίδιο σε αυτή την χαρακτηρισμένη ως και "μάχη των περιηγητών" θέτοντας καινούριες βάσεις και πλαίσια για την υποστήριξη των όλο και πιο αναπτυσσόμενων τεχνολογιών. Στο χώρο των διεπαφών χρήστη, ενσωματώθηκαν νέες τεχνολογίες που εισήγαγαν δυναμικότητα (JavaScript) και άλλες που επέτρεπαν την οπτική μορφοποίηση περιεχομένου (Cascading Style Sheets - CSS) στις μέχρι τότε στατικές και απλοϊκές σελίδες του ιστού. Φτάνοντας στο σήμερα, το τοπίο έχει αλλάξει αρκετά, αλλά ο στόχος και οι αρχές που καθόρισαν το Διαδίκτυο παραμένουν ίδια. Η είσοδος των φορητών συσκευών (smartphones) άλλαξαν τον τρόπο που αντιλαμβανόμαστε το Διαδίκτυο αλλά και το πως αλληλεπιδρούμε με αυτό, εισάγοντας νέες έννοιες, τεχνολογίες και κανόνες. Σε έναν κόσμο που αναπτύσσονται ραγδαία τεχνολογίες όπως η Τεχνητή Νοημοσύνη (Artificial Intelligence - AI), το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) και η Επαυξημένη Πραγματικότητα (Augmented Reality - AR), μπορούμε να πούμε με σιγουριά πως βρισκόμαστε μόνο στην αρχή!

Τα τελευταία έτη υπήρξε σημαντική εξέλιξη στον τρόπο που οι χρήστες αλληλεπιδρούν στον διαδικτυακό κόσμο. Το γεγονός αυτό δημιούργησε και την ανάγκη για πιο φιλικές προς τον χρήστη εφαρμογές οι οποίες διακρίνονται από τα χαρακτηριστικά της προσβασιμότητας και της προσαρμοστικότητας στα νέα, πολλαπλά μέσα από τα οποία πλέον ο χρήστης δύναται να αποκτήσει πρόσβαση σε αυτές.

Σημαντικό μέσο για την ικανοποίηση των νέων αναγκών του χρήστη, όσον αφορά την ανάπτυξη διαδραστικών συστημάτων, είναι εκτός των άλλων τα "**Πλαίσια Εργασίας της Διεπαφής Χρήστη**" (**Front-End Frameworks**) τα οποία ενισχύουν τους προγραμματιστές με εξειδικευμένες μεθόδους-συνταγές που επιτρέπουν και διευκολύνουν τη δημιουργία αφαιρετικών επιπέδων για τη δημιουργία σύγχρονων εφαρμογών που πληρούν τις προδιαγραφές τόσο από άποψη οπτικών όσο και τεχνικών δυνατοτήτων. Πλέον, ο προγραμματιστής έχει στη διάθεση του μια γκάμα εργαλείων τα οποία μπορούν να τον υποστηρίξουν στην ενσωμάτωση σύγχρονων, δυναμικών λύσεων οι οποίες παρέχουν εκτός των άλλων ασφάλεια και ευελιξία. Παράλληλα ο χρήστης, όντας τελικός αποδέκτης, αποκτά κεντρικό ενδιαφέρον για τη διαμόρφωση της τελικής λύσης, γεγονός που γεννάει και τον όρο "εμπειρία χρήστη" (User Experience - UX).

Περίληψη

Σκοπός της εργασίας αυτής είναι η ανάπτυξη μίας εφαρμογής διαδικτύου σε ένα από τα δημοφιλέστερα **πλαίσια εργασίας διεπαφής χρήστη (front-end frameworks)**, της **Angular**. Θα ξεκινήσω κάνοντας μία σύντομη αναφορά στις πιο δημοφιλείς βιβλιοθήκες που χρησιμοποιούνται για αυτό το σκοπό, αναλύοντας τη καθεμία ξεχωριστά και αναφέροντας τα ιδιαίτερα τεχνικά χαρακτηριστικά τους. Θα αναφερθώ επίσης, σε βιβλιοθήκες που χρησιμοποιώ και λειτουργούν συμπληρωματικά στην ανάπτυξη ολοκληρωμένων εφαρμογών. Με την ολοκλήρωση της εργασίας, θα έχω αναπτύξει λύσεις που ενσωματώνουν πρακτικές μεθόδους ανάπτυξης σύγχρονων εφαρμογών.

Abstract

The purpose of this thesis i the development of a web application in one of the most popular **front-end frameworks**, **Angular**. I will begin by making a brief reference to the most popular libraries used for this purpose, analyzing each one separately and mentioning their particular technical characteristics. I will also refer to libraries used and which complement the development of integrated applications. Upon completion of the work, I will have developed solutions that incorporate practical methods of developing modern applications.

1 Σύγχρονες Τεχνολογίες

Παρακάτω θα αναφερθώ στις κυριότερες βιβλιοθήκες ανάπτυξης που χρησιμοποιούνται για την ανάπτυξη διεπαφών χρήστη. Για λόγους απλότητας θα εστιάσω μόνο στις τεχνολογίες που υπολογίζουν το περιεχόμενο της σελίδας στο μηχάνημα-πελάτη (client-side rendering) και δεν θα αναφερθώ σε τεχνολογίες που υπολογίζουν το περιεχόμενο στον διακομιστή (server-side rendering).[1]

1.1 React

Η **React** (ή αλλιώς React.js) είναι βιβλιοθήκη ανάπτυξης διεπαφών χρήστη. Εκδόθηκε το 2013 από τη Facebook και πλέον συντηρείται από τη Meta (πρώην Facebook). Θεωρείται ως η πιο ευρέως χρησιμοποιούμενη βιβλιοθήκη για την ανάπτυξη διεπαφών χρήστη κι έχει τη μεγαλύτερη κοινότητα υποστήριξης.[1]

Ως πρότυπο προγραμματισμού χρησιμοποιεί το **Δηλωτικό (declarative programming)**. Αυτό σημαίνει πως ο προγραμματιστής περιγράφει τη τελική μορφή του προγράμματος αντί της ροής του.[2]

Η React ως βιβλιοθήκη, υποστηρίζει μόνο την ανάπτυξη **διαδραστικών στοιχείων** (**components**). Τα στοιχεία (components) χρησιμοποιούν παραμέτρους (props) για τη δημιουργία διόδων επικοινωνίας. Οι δίοδοι που δημιουργούνται μέσω παραμέτρων (props) είναι μονής κατεύθυνσης (γονέας -> παιδί). Εντός κάθε στοιχείου συντηρείται η κατάσταση του (state). Η κατάσταση κάθε στοιχείου (component) είναι προσπελάσιμη μόνο εσωτερικά και κάθε αλλαγή της προκαλεί επανυπολογισμό του μοντέλου της σελίδας (DOM). Είναι σημαντικό εδώ να επισημάνουμε ένα από τα σημαντικότερα πλεονεκτήματα της React το οποίο αφορά το πως ενσωματώνει και διαχειρίζεται τον επανυπολογισμό του περιεχομένου. Μέσω της χρήσης της απεικόνισης **Virtual DOM**, η React ενεργοποιεί κατάλληλους μηχανισμούς ούτως ώστε ο επανυπολογισμός κάθε αλλαγής να γίνεται μόνο εντός των πλαισίων που εκτείνεται η ευθύνη του στοιχείου (component).[3] [4] Αυτό σημαίνει πως κάθε στοιχείο, επανυπολογίζει μόνο το μέρος της σελίδας που προβάλει, και όχι όλη τη σελίδα. Αυτό κάνει τη React αποδοτική, αφού ο επανυπολογισμός ολόκληρης της σελίδας σε κάθε έστω και μικρή αλλαγή, εισάγει σημαντική επιβάρυνση στο σύστημα.

Μία έννοια που προστέθηκε αργότερα στη βιβλιοθήκη React, είναι η λειτουργία των **React Hooks.** Μέσω αυτής της λειτουργίας επιτρέπεται η επαχρησιμοποίηση κύριας λειτουργικότητας της React εντός των στοιχείων (components). Αυτή η λειτουργία κρίνεται απαραίτητη για τα components τα οποία ορίζονται ως μεθόδοι.Σε διαφορετική περίπτωση, η χρήση κλάσεων για τη δήλωση components είναι μονόδρομος. [5] [6] [7] Πιο συγκεκριμένα, μέσω των React Hooks, ο χρήστης της React μπορεί να κάνει χρήση **state** [8], να συσχετίσει λειτουργικότητα του component με **lifecycle hooks** μέσω χρήσης των **effects** [9], να χρησιμοποιήσει δεδομένα που βρίσκονται υψηλότερα στην ιεραρχία των component - αποφεύγοντας έτσι την αλληλοεξάρτηση μέσω props - μέσω **context** [10], και να τρέξει λειτουργία εκτός του υπολογιστικού πλαισίου της React με **refs** [11]. Αυτά είναι τα κυριότερα ενσωματωμένα hooks. Βάσει αυτών ο χρήστης μπορεί να αναπτύξει εξατομικευμένες λύσεις.

Για την ανάπτυξη περαιτέρω λειτουργικότητας όπως για παράδειγμα του δρομολογητή (routing), η οποία απαιτείται από τις σύγχρονες εφαρμογές πρέπει να χρησιμοποιηθούν τρίτα συστήματα. Αυτά τα συστήματα, είτε επεκτείνουν τη βιβλιοθήκη ως προσθήκες είτε προσφέρουν ολοκληρωμένες λύσεις (frameworks), παίρνοντας ως βάση τη React.[12]

Το χαρακτηριστικό που κάνει τη React να ξεχωρίζει είναι το **εικονικό δέντρο απεικόνισης (Virtual DOM)**. Εισάγοντας αυτή την έννοια στην κύρια λειτουργία της, η React αντιμετώπισε το πρόβλημα της επιβάρυνσης συστήματος που προκύπτει από την ανάγκη επανυπολογισμού ολόκληρου του δέντρου απεικόνισης της HTML σε κάθε έστω και μικρή αλλαγή που συμβαίνει σε αυτό. Συγκεκριμένα, αλλαγές που φαινομενικά είναι μικρές, όπως η αλλαγή ενός μικρού κειμένου μόνο σε ένα μέρος της σελίδας του περιηγητή, απαιτούν να υπολογιστεί εκ νέου όλη η σελίδα. Η σελίδα που απεικονίζεται σαν δέντρο (DOM) από την HTML ανήκει στη δικαιοδοσία του browser. Με τη δημιουργία ενός αντίγραφου το οποίο αποθηκεύεται στη μνήμη του προγράμματος, η React καταφέρνει μέσω συγκρίσεων και ανανεώσεων να ενημερώνει μόνο τα σημεία της σελίδας τα οποία επηρεάστηκαν από την αλλαγή. Παρόλο που ο μηχανισμός σύγκρισης και ανανέωσης είναι επίσης απαιτητικός, το εύρος των απαραίτητων αλλαγών είναι πολύ πιο περιορισμένο, γεγονός που τον κάνει πιο αποδοτικό.[13]

Το συντακτικό της React, επωφελείται από τη χρήση της **JSX (Javascript XML)**. Παρόλο που η χρήση της JSX δεν είναι προαπαιτούμενη για την ανάπτυξη σε React, η χρήση της κρίνεται ως η πλέον βέλτιστη. Αυτό που επιτυγχάνεται μέσω αυτής, είναι η μίξη DOM στοιχείων με κώδικα JavaScript. Αυτό κρίνεται απαραίτητο, αφού στο σύγχρονο υπολογιστικό περιβάλλον η παραδοσιακή χρήση της HTML δεν επαρκεί για την ανάπτυξη διαδραστικών και πλούσιων λειτουργικά εφαρμογών.[14]

Η React είναι μια ευέλικτη βιβλιοθήκη ανάπτυξης. Αυτό το επιτυγχάνει κρατώντας μια ουδέτερη στάση απέναντι στον τρόπο επέκτασης και δόμησης της αρχιτεκτονικής της εφαρμογής. Η χρήση της Virtual DOM απεικόνισης την καθιστούν αποδοτική στη χρήση της, ενώ η μεγάλη κοινότητα που έχει αναπτυχθεί γύρω της την τοποθετούν στις πιο διαδεδομένες βιβλιοθήκες. Το μέγεθος της κοινότητας της συνεπάγεται και το βαθμό υποστήριξης για κάποιον που ξεκινάει την ανάπτυξη σε React ή αντιμετωπίζει πιο εξειδικευμένα προβλήματα. Τέλος, η ανάπτυξη και χρήση από τον οργανισμό της Meta της δίνει την επικείμενη ασφάλεια που αναμένει κανείς για την ανάπτυξη μιας εφαρμογής που απευθύνεται σε πραγματικούς χρήστες.

1.2 Angular

Η **Angular** είναι πλατφόρμα ανάπτυξης διεπαφής χρήστη. Αναπτύχθηκε κι εκδόθηκε το 2016 από την Google η οποία και συνεχίζει να τη συντηρεί. Πρόκειται για επανασχεδιασμό κι επανεγγραφή σε **Typescript** της παλαιότερης τεχνολογίας **AngularJS**.[15] [16] [17] Σε σύγκριση με τη React, περιέχει πλούσιες δυνατότητες προσφέροντας έτσι μια ολοκληρωμένη λύση στην ανάπτυξη εφαρμογών. Ενώ αυτή η δυνατότητα μπορεί να της δώσει κάποια προτερήματα, παράλληλα δημιουργεί δυσκολία στην εκμάθηση της, ειδικά σε νέους προγραμματιστές.[18] Προτείνεται η χρήση της Typescript για ανάπτυξη εφαρμογών.

Ενσωματώνει αρχιτεκτονική που βασίζεται στα **υπολογιστικά στοιχεία (components)** για την καλύτερη οργάνωση και διαχείριση του κώδικα. Κάθε στοιχείο (component) δημιουργείται με τη δήλωση μίας κλάσης που μπορεί να ενσωματώνει μεθόδους και παραμέτρους. Η δήλωση του στοιχείου (component) ως τέτοιου, γίνεται με τη χρήση διακοσμητή (decorator)[19], ο οποίος περιέχει όλα τα απαραίτητα δεδομένα (metadata) για την περιγραφή του στοιχείου.[20] Σε κάθε κλάση στοιχείου, ανήκει μία απεικόνιση (template) η οποία περιέχει την απεικονιστική μορφή σε HTML.Κάθε απεικόνιση (template) έχει τη δυνατότητα προβολής **δυναμικών δεδομένων** αλλά και τροποποίησης της κατάστασης στοιχείων μέσω **δυναμικών παραμέτρων**.[21] Επιπλέον, για την οπτική τροποποίηση των προβληθέντων στον χρήστη στοιχείων, υπάρχει η δυνατότητα χρήσης κανόνων CSS. Κάθε στοιχείο (component) εχει το δικό του πλαίσιο μορφοποίησης του στυλ του, το οποίο ορίζεται επίσης στο προσωπικό του αρχείο.

Παρομοίως, τα **στοιχεία συμπεριφοράς (directives)** χρησιμοποιούνται για την προσθήκη συμπεριφοράς σε στοιχεία HTML. Τα directives είναι κλάσεις που χρησιμοποιούνται για τον έλεγχο στοιχείων φόρμας, παραγωγή λίστας, διαμόρφωση του στυλ και έλεγχο δομής μέσω εμφάνισης και απόκρυψης. Υπάρχουν τα ήδη ενσωματωμένα στοιχεία (directives) και τα ειδικά κατασκευασμένα (custom). Τα προκατασκευασμένα directives που είναι ήδη ενσωματωμένα μπορεί να είναι είτε δομές που ρυθμίζουν τη ροή προγράμματος όπως **if** - **switch** - **for**, ορισμό τιμής μέσω **ngModel**, διαμόρφωση στυλ με **ngClass** και **ngStyle.** [22] Ένα ειδικά κατασκευασμένο στοιχείο (directive) θα μπορούσε να στοχεύει στην υπογράμμιση ενός στοιχείου HTML που περιέχει κείμενο.[23]

Συχνά, υπάρχει η ανάγκη διαμοιρασμού λειτουργικότητας ανάμεσα στα components. Αυτό επιτυγχάνεται με τη χρήση **υπηρεσιών (services)**. Οι υπηρεσίες στην Angular εκμεταλλεύονται το **μηχανισμό χορήγησης υπηρεσιών (Dependency Injection - DI)** για τον οποίο θα αναφερθώ παρακάτω. Εκτός της διευκόλυνσης διαμοιρασμού κοινής λειτουργίας ανάμεσα σε λειτουργικά στοιχεία, η χρήση των services ευνοεί τον **διαχωρισμό ευθυνών (separation of concerns)** δίνοντας τη δυνατότητα στο χρήστη να διαχωρίσει τη λειτουργία από τη συμπεριφορά του συστήματος. Έτσι, τα στοιχεία (components) διαχειρίζονται τη συμπεριφορά όπως αποτυπώνεται οπτικά, ενώ οι υπηρεσίες (services) διαχειρίζονται την εσωτερική λειτουργία και την επικοινωνία με άλλα συστήματα.[24] [25]

Ο **μηχανισμός χορήγησης υπηρεσιών (Dependency Injection - DI)** είναι από τους σημαντικότερους μηχανισμούς της Angular αφού επιτρέπει το διαμοιρασμό λειτουργικότητας μεταξύ των κλάσεων μέσω της χορήγησης υπηρεσιών κατά την αρχικοποίηση τους. Ο μηχανισμός αυτός επιτρέπει την επαναχρησιμοποίηση των υπηρεσιών (services) σε διάφορα σημεία του συστήματος.[26] Οι υπηρεσίες που χορηγούνται μπορεί να είναι μοναδικές στο σύστημα (singletons)[27] ή ξεχωριστά αρχικοποιημένες για την κάθε κλάση που τις εισαγάγει.[28]

Οι σύγχρονες εφαρμογές βασίζονται στη μεθοδολογία **SPA (Single Page Application)** για τη σελιδοποίηση και ανακατεύθυνση ανάμεσα στις διάφορες οντότητες του συστήματος. Πρόκειται για έναν τρόπο ανάπτυξης εφαρμογών που βελτιώνει την απόδοση του συστήματος και μειώνει δραστικά τη χρήση πόρων απομακρυσμένων συστημάτων. Αυτό επιτυγχάνεται με τη χρήση μία μόνο σελίδας για τη φόρτωση ολόκληρης της εφαρμογής έναντι πολλαπλών σελίδων που θα προϋπόθεταν την επαναφόρτωση της εφαρμογής σε κάθε ανακατεύθυνση. Στην πράξη, το μηχάνημα-πελάτης ζητάει την κύρια και μοναδική σελίδα της εφαρμογή κατά την εκκίνηση της. Η σελίδα αυτή περιέχει όλο το

περιεχόμενο της εφαρμογής σε κώδικα (JavaScript), δομή (HTML), και κανόνες μορφοποίησης (CSS).[29] Πως μπορεί όμως να επιτευχθεί η πλοήγηση εντός εφαρμογής, χρησιμοποιώντας μία μόνο σελίδα? Παλαιότερα, κάθε μεταβίβαση σε νέα σελίδα μιας εφαρμογής θα απαιτούσε μία αίτηση στον εξυπηρετητή για την αποστολή του αρχείου που περιείχε το περιεχόμενο της σελίδας σε HTML. Η Angular δίνει τη λύση της στο πρόβλημα μέσω του ενσωματωμένου συστήματος **routing**.[30] Ο router της Angular πρόκειται για μια υπηρεσία που τρέχει στο επίπεδο του συστήματος κι επιτρέπει τη μετάβαση σε διάφορες καταστάσεις (views) της εφαρμογής. Κάθε view δηλώνεται ως κατάσταση (state) της εφαρμογής. Με αυτό τον τρόπο, παρόλο που στον χρήστη δίνεται η αίσθηση πως μεταβαίνει σε διάφορες σελίδες, η εφαρμογή τρέχει με τα αρχεία που αιτήθηκε κατά την εκκίνηση της, και όλες οι ανακατευθύνσεις γίνονται προγραμματιστικά εντός της κύριας-ενιαίας εφαρμογής.[31]

Παρόλο που σαν βιβλιοθήκη είναι ανεξάρτητη από την Angular, έχει μεγάλο ενδιαφέρον να αναφερθώ στην **RxJS**. Ο λόγος που το κάνω είναι επειδή η βιβλιοθήκη αυτή είναι σε μεγάλο βαθμό πλέον συνυφασμένη με την Angular, και άρα όποιος θέλει να χρησιμοποιήσει την Angular θα επωφεληθεί σε σημαντικό βαθμό αν ενσωματώσει και την RxJS. Η RxJS είναι μια βιβλιοθήκη που εισάγει το πρότυπο της αναδραστικότητας (reactive [32]) με τη χρήση γεγονότων (event-driven [33]) και διαδικαστικού (functional [34]) προγραμματισμού στα συστήματα που τη χρησιμοποιούν. Αυτό το πετυχαίνει με την εισαγωγή αναδραστικών δομών δεδομένων **Observables** συνδυαστικά με διαδικασίες μεταμόρφωσης της ροής των γεγονότων (**operators**) που αναμεταδίδονται από αυτά και που φτάνουν να καταλήγουν στους ακροατές **subscribers**. Τα Observables είναι δομές δεδομένων που βασίζονται και ενσωματώνουν εσωτερικά το σχεδιαστικό πρότυπο του Observer (Observer Design Pattern [35]) - ενός από τα κύρια σχεδιαστικά πρότυπα που είναι ευρέως γνωστά. Τα Observables δεν είναι τα μοναδικά αντικείμενα που ενσωματώνουν αναδραστικότητα, αλλά για λόγους απλούστευσης εδώ αναφέρομαι μόνο σε αυτά εννοώντας και τα υποσύνολα τους. Οι operators είναι διαδικασίες ενσωματωμένες στη βιβλιοθήκη οι οποίες μπορούν να εισαχθούν στην εφαρμογή και να συνδεθούν αλυσιδωτά πάνω στα Observables για να τροποποιήσουν ή και μεταμορφώσουν τη ροή γεγονότων. Εδώ βρίσκεται και η ουσία του πρότυπου του διαδικαστικού προγραμματισμού. Πρωτίστως μας ενδιαφέρουν οι κύριες διαδικασίες *map*, *filter*, *reduce* και *every* που αποτελούν και τις παραδοσιακές διαδικασίες του διαδικαστικού προγραμματισμού και των δομών λίστας (arrays). [36] Η RxJs είναι μια πλούσια σε λειτουργία βιβλιοθήκη. Τα προγραμματιστικά πρότυπα που εισαγάγει απαιτούν χρόνο και πρακτική εμπειρία για να απορροφηθούν σαν γνώση. Είναι περιττό να εμβαθύνω περισσότερο σε αυτό το σημείο. Αυτή η εισαγωγή είναι ικανή για να καλύψει τις εννοιολογικές απαιτήσεις που μπορούν να λειτουργήσουν ως βάση στην εκμάθηση της βιβλιοθήκης.

1.3 Vue

Η **Vue.js** (κοινώς **Vue**) είναι μια βιβλιοθήκη σχεδιασμένη για να προσφέρει ευελιξία στην ανάπτυξη εφαρμογών διεπαφής χρήστη. Εκδόθηκε τον Φεβρουάριο του 2014 από έναν πρώην προγραμματιστή της Google ο οποίος είχε εξοικειωθεί εις βάθος στη χρήση της AngularJS και με τη δημιουργία της Vue, στόχευε στο να δημιουργήσει μια “ελαφριά” εκδοχή της AngularJS, η οποία όμως θα ενσωματώνει τα πιο ελκυστικά της στοιχεία. Η Vue δεν χρειάζεται απαραίτητα κτίσιμο (build). Μπορεί να χρησιμοποιήθει τόσο σαν πακέτο, δηλαδή ως μια JavaScript βιβλιοθήκη που εισάγεις σε μια απλή εφαρμογή για να χρησιμοποιήσεις κάποιες λειτουργίες της, όσο και σαν διαδικασία ανάπτυξης

πλήρους εφαρμογής που περιλαμβάνει κτίσιμο και φόρτωμα της εφαρμογής με τον ενσωματωμένο builder της βιβλιοθήκης. Για την ενσωμάτωση εξειδικευμένων λειτουργιών που απαιτούνται από τις σύγχρονες εφαρμογές, όπως routing, η Vue υποστηρίζει βιβλιοθήκες ανοιχτού λογισμικού, τις οποίες αναπτύσσει και συντηρεί η ίδια και επεκτείνουν τη βιβλιοθήκη. [37]

Σε περίπτωση που ο χρήστης επιλέξει να χρησιμοποιήσει τον builder σαν διαδικασία για να κτίζει την εφαρμογή, προτείνεται η μέθοδος χρήσης μονού αρχείου component (**Single-File Components - SFC**). Με τη χρήση αυτής της μεθόδου, ο χρήστης έχει τη δυνατότητα να δηλώσει σε ένα μόνο αρχείο τη λογική - δηλαδή JavaScript με χρήση του στοιχείου `<script>`, την παρουσίαση - δηλαδή HTML με χρήση του στοιχείου `<template>` και τη μορφοποίηση - δηλαδή CSS με χρήση του στοιχείου `<style>`. [38]

Για την ανάπτυξη των στοιχείων της εφαρμογής ο χρήστης έχει δύο επιλογές που καθορίζουν το νοητικό μοντέλο με το οποίο θα τα ενσωματώσει. Μπορεί να επιλέξει είτε την επιλογή **Options API** είτε την επιλογή **Composition API**. Η πρώτη επιλογή ενσωματώνει components δομημένα σαν objects, με καθορισμένα κλειδιά όπως *data*, *methods*, και *mounted*. Οι τιμές του object έχουν πρόσβαση στο αντικείμενο `this` του component κι έτσι διαμοιράζονται scope. Η δεύτερη επιλογή είναι πιο δυναμική κι επιτρέπει στον χρήστη μια πιο ελεύθερη δήλωση των μεθόδων και μεταβλητών με τη χρήση της SFC μεθόδου και εισαγωγή μεθόδων της βιβλιοθήκης. Και οι δύο τρόποι μπορούν να καλύψουν τις περισσότερες συνηθισμένες απαιτήσεις που μπορεί να έχει κανείς από μια εφαρμογή. [39]

2 Ανάπτυξη Συστήματος

2.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές περιγράφουν τα λειτουργικά χαρακτηριστικά του συστήματος. Πρόκειται για τη βασική συμπεριφορά που απαιτείται από το σύστημα να ενσωματώσει. [40]

2.1.1 Πρόσβαση στο δελτίο καιρού για δεδομένη περιοχή

Ο χρήστης έχει πρόσβαση σε πληροφορίες που περιγράφουν τον καιρό για μια δεδομένη περιοχή.

2.1.2 Τρέχουσες καιρικές συνθήκες (καιρός τώρα)

Πρόκειται για κάρτα η οποία περιέχει πληροφορίες σχετικές με τις τρέχουσες συνθήκες καιρού. Επιπλέον, αναφέρονται πληροφορίες σχετικές με την ανατολή και δύση του ηλίου.

Λεπτομερώς εμφανίζει:

a. Τοποθεσία και γενική περιγραφή καιρού
b. Εικονίδιο που περιγράφει την επικρατούσα συνθήκη
c. Θερμοκρασία σε βαθμούς Κελσίου
d. Ατμοσφαιρική πίεση (hPa)
e. Υγρασία σε ποσοστό τοις εκατό
f. Ταχύτητα (m/s) και κατεύθυνση ανέμου
g. Δείκτης υπεριώδους ακτινοβολίας (UVI)

h. Ορατότητα (km)
i. Μετρητής βροχόπτωσης (mm)
j. Μετρητής χιονόπτωσης (mm)
k. Δείκτης ποιότητας αέρα (AQI). Χρησιμοποιείται δείκτης 1-10+ όπου όσο ανεβαίνει η τιμή, τόσο μειώνεται η ποιότητα αέρα. Ο χρήστης μπορεί να δει λεπτομέρειες μέσω διάδρασης με το στοιχείο που απεικονίζει δείκτη.

2.1.3 Δελτίο καιρού ανά ώρα

Πρόκειται για κάρτα που εμφανίζει μια γενική περιγραφή του καιρού για την εκάστοτε ώρα. Περιέχει δεδομένα για τις επόμενες 5 ημέρες με βήμα ανά 3 ώρες, όπως προδιαγράφεται από την απομακρυσμένη διεπαφή Open Weather (5 day weather forecast - OpenWeatherMap). Η λίστα με τα δελτία καιρού ξεκινάει από την αμέσως επόμενη, και καταλήγει στην τελευταία.

Λεπτομερώς εμφανίζονται:

a. Γενική περιγραφή του καιρού σε εικονίδιο
b. Μέγιστη κι ελάχιστη θερμοκρασίες σε βαθμούς κελσίου
c. Χρονικό σημείο

2.1.4 Περιορισμός απομακρυσμένων κλήσεων

Το σύστημα πρέπει να κάνει εξοικονόμηση απομακρυσμένων πόρων κρατώντας στη μνήμη δεδομένα τα οποία είναι έγκυρα για τα επόμενα δεκαπέντε (15) λεπτά. Κατά τη διάρκεια των 15 λεπτών δεν θα γίνεται καμία κλήση στην απομακρυσμένη διεπαφή για δεδομένα καιρού. Τα δεδομένα είναι έγκυρα για 15 λεπτά από τη στιγμή που διακομίστηκαν. Με το πέρας των 15 λεπτών, κι εφόσον ο χρήστης αιτηθεί νέα δεδομένα, τα αποθηκευμένα δεδομένα ακυρώνονται και γίνεται κλήση στην απομακρυσμένη διεπαφή.

Με αυτό τον τρόπο, μειώνεται η επικοινωνία μεταξύ πελάτη και διακομιστή.

2.1.5 Αυθεντικοποίηση κλήσης

Η απομακρυσμένη διεπαφή απαιτεί τη χρήση κλειδιού για να επιτρέψει την πρόσβαση. Κάθε κλήση φέρει κλειδάριθμο πιστοποίησης. Ο κλειδάριθμος είναι αποθηκευμένος τοπικά στο μηχάνημα του χρήστη, ούτως ώστε να παραμένει κρυφός. Κατά τη λειτουργία του συστήματος, απαιτείται κεντρικό σημείο διαχείρισης των κλήσεων, για την ενσωμάτωση του κλειδιού σε κάθε απομακρυσμένη κλήση.

2.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις περιγράφουν την ποιοτική συμπεριφορά του συστήματος.

2.2.1 Σύγχρονη αισθητική (modern)

Η εφαρμογή πρέπει να ανταπεξέρχεται στα σύγχρονα σχεδιαστικά δεδομένα. Η διεπαφή χρήστη πρέπει να είναι φιλική και καθαρή.

2.2.2 Ευκολία χρήσης (intuitive)

Όλη η πληροφορία περιέχεται σε μία μόνο σελίδα. Ο χρήστης έχει πρόσβαση σε όλα τα απαραίτητα δεδομένα χωρίς ανάγκη για ψάξιμο και περαιτέρω πλοήγηση στην εφαρμογή.

2.2.3 Προσαρμοστικότητα (responsive)

Η εφαρμογή προσαρμόζεται σε όλα τα μεγέθη οθόνης. Ανταποκρίνεται ομαλά στις αλλαγές μεγέθους του πλαισίου απεικόνισης.

2.3 Βιβλιοθήκες Ανάπτυξης

Για τους σκοπούς αυτής της εργασίας θα χρησιμοποιήσω **Angular**. Ο κύριος λόγος που επιλέγω αυτή τη βιβλιοθήκη είναι η πληρότητα της σε λειτουργία σε συνδυασμό με τη συγκριτική οικειοποίηση που έχω μαζί της. Σε πραγματικές συνθήκες, η χρήση τεχνολογιών με γνώμονα την οικειότητα μας με αυτές αντενδείκνυται, αλλά στα πλαίσια της εργασίας θα μου επιτρέψει να παρουσιάσω μία βιβλιοθήκη που πληροί λειτουργικότητας και ευνοεί την ενσωμάτωση λειτουργίας μέσω έγκυρων πρακτικών ανάπτυξης δοκιμασμένων σε απαιτητικές εφαρμογές. Επιπλέον, η εμπειρία μου στην ενσωμάτωση εφαρμογών με Angular θα μου επιτρέψει να αναπτύξω έννοιες και λειτουργικότητα με ευχέρεια, επιτρέποντας την εμβάθυνση σε πιο εξεζητημένες πρακτικές. Παράλληλα, θα παρουσιάσω στο μέτρο που προσφέρεται από τις απαιτήσεις του συστήματος, τις δυνατότητες που δίνει ένα πλήρες framework ανάπτυξης διεπαφής χρήστη.

Η εφαρμογή περιορίζεται σε μία μόνο σελίδα, συνεπώς χαρακτηριστικά όπως το σύστημα δρομολόγησης (routing) της Angular δεν θα χρειαστεί. Σε περίπτωση επέκτασης της εφαρμογής σε περισσότερες σελίδες, τότε μπορεί να προστεθεί σαν μηχανισμός.

Θα χρησιμοποιήσω επίσης την **RxJS** και την **Typescript**, εκμεταλλευόμενος πλήρως τις δυνατότητες της σύγχρονης μεθοδολογίας ανάπτυξης λογισμικού σε Angular, και την ασφάλεια του συστήματος τύπων.

Για τη στυλιστική μορφοποίηση της εφαρμογής θα ενσωματώσω στην εφαρμογή το CSS framework **Bootstrap**. [41] Θα εκμεταλλευτώ έτσι τις κλάσεις που ενσωματώνει για τη δημιουργία πλέγματος (Grid Layout) [42] και άλλες βοηθητικές κλάσεις. Με τη χρήση των παραπάνω θα εξασφαλίσω όλες τις μη λειτουργικές απαιτήσεις της εφαρμογής. Επιπλέον, θα χρησιμοποιήσω το **ng-bootstrap** [43] για την ενσωμάτωση εξωτερικών components.

2.4 Αρχιτεκτονική

Η εφαρμογή βασίζεται σε τρία σύνολα λειτουργικότητας όπως αποτυπώνονται στο σύστημα φακέλων. Η κατηγοριοποίηση των **features** έχει ως βάση τη σχεδιαστική προσέγγιση Domain Driven Design.[44] Αυτό σημαίνει πως κάθε feature είναι εν δυνάμει ένα ξεχωριστό domain. Μελλοντικές προσθήκες στα features της εφαρμογής θα πρέπει επίσης να διαχωρίζουν το domain τους. Στοιχεία τα οποία δεν ανήκουν ουσιαστικά σε κάποιο feature χωρίζονται είτε σε core λειτουργικότητα ή σε shared. Η **core** είναι απαραίτητη για την εφαρμογή ενώ η **shared** είναι επαναχρησιμοποιήσιμη.

Όλα τα στοιχεία που βρίσκονται στα features και απαιτούν επικοινωνία με απομακρυσμένο διακομιστή χρησιμοποιούν την υπηρεσία **ApiService**. Κατά κανόνα, κάθε απομακρυσμένη επικοινωνία θα πρέπει να γίνεται μέσω αυτής της υπηρεσίας. Η υπηρεσία πρέπει να περιγράφει ρητά τη λειτουργικότητα που ενσωματώνει και να περιγράφει πλήρως τους τύπους δεδομένων που διαχειρίζεται.

Στα features, το μοναδικό feature είναι το **forecast**, το οποίο αποτελεί και το μοναδικό feature της εφαρμογής. Η λειτουργικότητα του forecast ενσωματώνεται κατά κανόνα από τον γονέα component

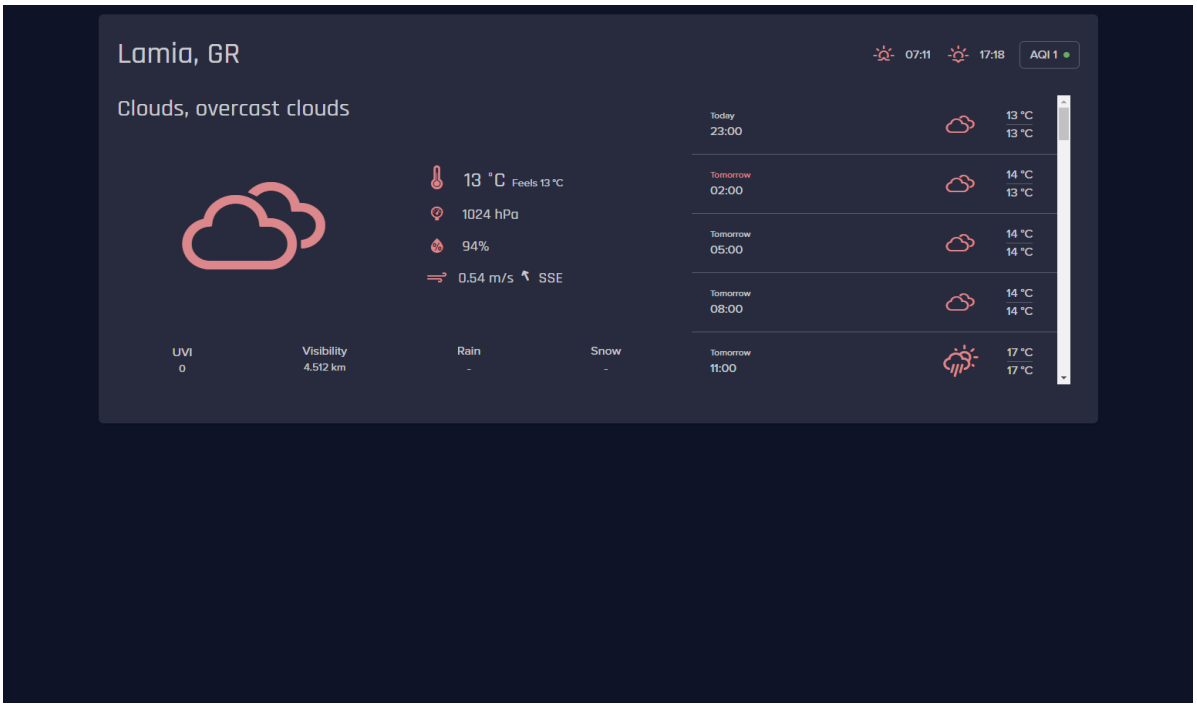
και ακόμη πιο συγκεκριμένα από την υπηρεσία του **ForecastService**. Τα components παιδιά βρίσκονται εμφωλευμένα εντός του feature forecast. Εδώ έχω επιλέξει τον διαμοιρασμό των δεδομένων με χρήση αναδραστικότητας Observable το οποίο βρίσκεται στην υπηρεσία **ForecastService** και προωθεί τα δεδομένα στους ακροατές. Επέλεξα αυτή τη λύση για να επιτρέψω την ευελιξία των παιδιών σε σχέση με τον γονέα component. Ο ρόλος των παιδιών είναι μόνο να προβάλλουν δεδομένα και δεν ενσωματώνουν καθόλου λογική πέραν της απαραίτητης για την ανάκτηση των δεδομένων.

Η εφαρμογή χρησιμοποιεί το **Open Weather API** για τη χρήση δεδομένων καιρού. Για τη λειτουργία του API απαιτείται η χρήση εξατομικευμένου κλειδιού προς την αυθεντικοποίηση της κάθε κλήσης προς αυτό. Το κλειδί είναι προσωπικό και πρέπει να βρίσκεται σε μη προσβασιμό από εξωτερικούς παράγοντες σημείο. Για τη χρήση του στο σύστημα χρησιμοποιείται το πακέτο `dotenv-webpack` κι έχει δημιουργηθεί ειδικό `.env` αρχείο που περιέχει το κλειδί. Αυτό το κλειδί μετά εισάγεται στα `environment` αρχεία της Angular τα οποία διαχειρίζεται το framework μέσα από τη διαδικασία του build. Τέλος, μέσω του `environment` αρχείου το κλειδί εισάγεται στον `AuthenticationInterceptor` κι έτσι η κάθε κλήση προς το Open Weather API είναι πιστοποιημένη. Σημαντικό να αναφέρω πως το `.env` αρχείο πρέπει να εξαιρείται από τη λίστα του version control. Ο διαχειριστής του συστήματος είναι υπεύθυνος για την εξασφάλιση του κλειδιού.

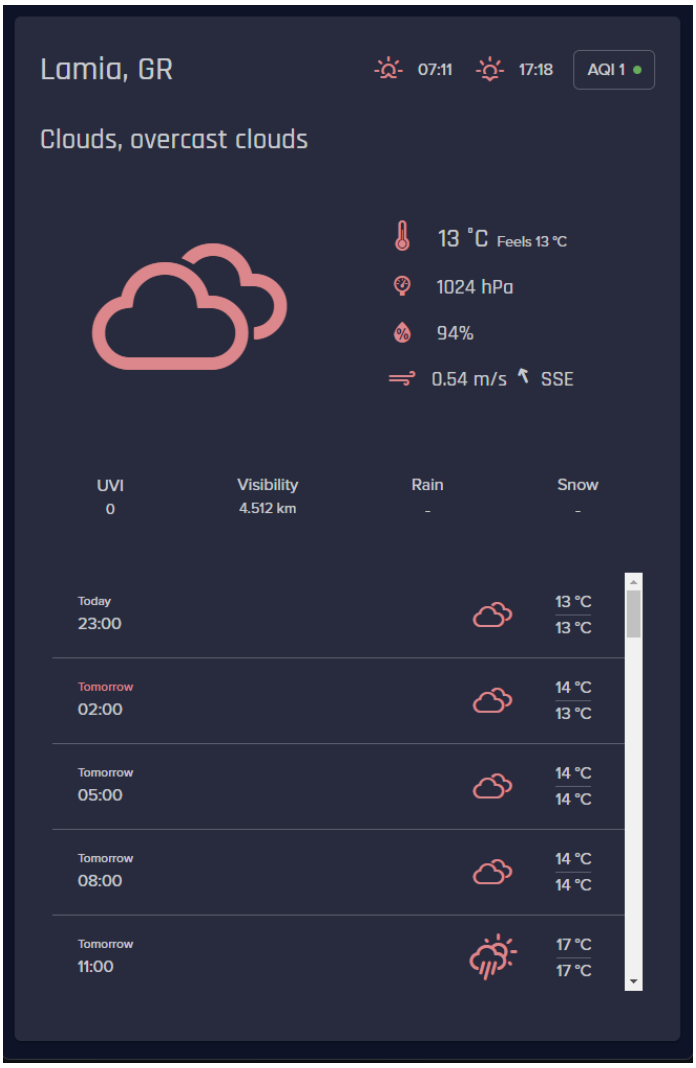
Στην εφαρμογή εφαρμόζεται μέθοδος Proxy για την επικοινωνία με τα απομακρυσμένα συστήματα. Αυτό προσφέρει συνέπεια στην εσωτερική χρήση των endpoints και δυνατότητα για βελτιωμένη σήμανση βάσει της λογικής της εφαρμογής. Η proxy δηλώνεται σαν configuration στο αρχείο `proxy.conf.json` και εισάγεται στο αρχείο της Angular `angular.json`.

2.5 Ενσωμάτωση

Για την ενσωμάτωση των **Μη Λειτουργικών Απαιτήσεων** της εφαρμογής χρησιμοποίησα τη βιβλιοθήκη Bootstrap σε συνδυασμό με ανάπτυξη custom design system. Χρησιμοποίησα μοντέρνα χρώματα σε σκουρόχρωμη χρωματική παλέτα *(βλέπε 2.2.1)*. Επιπλέον, εκμεταλλεύτηκα τη χρωματική αντίθεση μέσω των κύριων χρωμάτων για να δημιουργήσω οπτικό ενθουσιασμό στον χρήστη. Με τη χρήση των βοηθητικών στυλιστικών κλάσεων του Boostrap ενσωμάτωσα προσαρμοστικότητα στη διεπαφή χρήστη *(βλέπε 2.2.3)*. Τέλος, έκανα στοχευμένη τοποθέτηση και κατηγοριοποίηση της οπτικής πληροφορίας ούτως ώστε όλα τα δεδομένα να είναι εύκολα και διαισθητικά προσβάσιμα στον χρήστη *(βλέπε 2.2.2)*.



Εικόνα 2.5-1 Εφαρμογή σε πλήρες μέγεθος οθόνης



Εικόνα 2.5-2 Εφαρμογή σε μικρό μέγεθος οθόνης

Για την περαιτέρω παρουσίαση της ενσωμάτωσης της εφαρμογής και των **Λειτουργικών Απαιτήσεων** θα χρησιμοποιήσω ένα σύνολο επεξηγήσεων, σχεδιαγραμμάτων και κώδικα για την κάθε οντότητα του συστήματος. Για λόγους λογικής συνοχής θα ξεκινήσω την παρουσίαση από τις ζωτικές οντότητες, δηλαδή τις υπηρεσίες που βρίσκονται στον φάκελο ***core*** της εφαρμογής. Στη συνέχεια θα παρουσιάσω κάποια κοινά λειτουργικά στοιχεία που θα περιληφθούν σε διάφορα σημεία της κύριας λειτουργίας. Αυτά τα στοιχεία βρίσκονται στον φάκελο ***shared*** της εφαρμογής. Τέλος, θα παρουσιάσω την κύρια ενσωμάτωση των χαρακτηριστικών της εφαρμογής δηλαδή των features, τα οποία βρίσκονται στον ομώνυμο φάκελο ***features***. Για λόγους σήμανσης, το κύριο και μοναδικό feature της εφαρμογής βρίσκεται στον φάκελο ***forecast*** εντός των *features*. Το forecast feature περιλαμβάνει τα στοιχεία που το συνθέτουν κι έχουν άμεση σχέση με τη λειτουργία του. Τέλος, περιλαμβάνω το σύστημα τύπων του συστήματος και τις σταθερές του.

2.5.1 Core

Ο φάκελος Core περιλαμβάνεις τις κεντρικές για το σύστημα λειτουργίες χωρίς τις οποίες δεν μπορεί να υπάρξει συνεπής ή και καθόλου λειτουργία. Αποτελεί τις πιο βασικές οντότητες του συστήματος οι οποίες χρησιμοποιούνται σε όλο το μήκος και το πλάτος του.

2.5.1.1 API Service

Τοποθεσία: core/api/api.service.ts

Όνομα Κλάσης: ApiService

Σημείο Δήλωσης: *root*

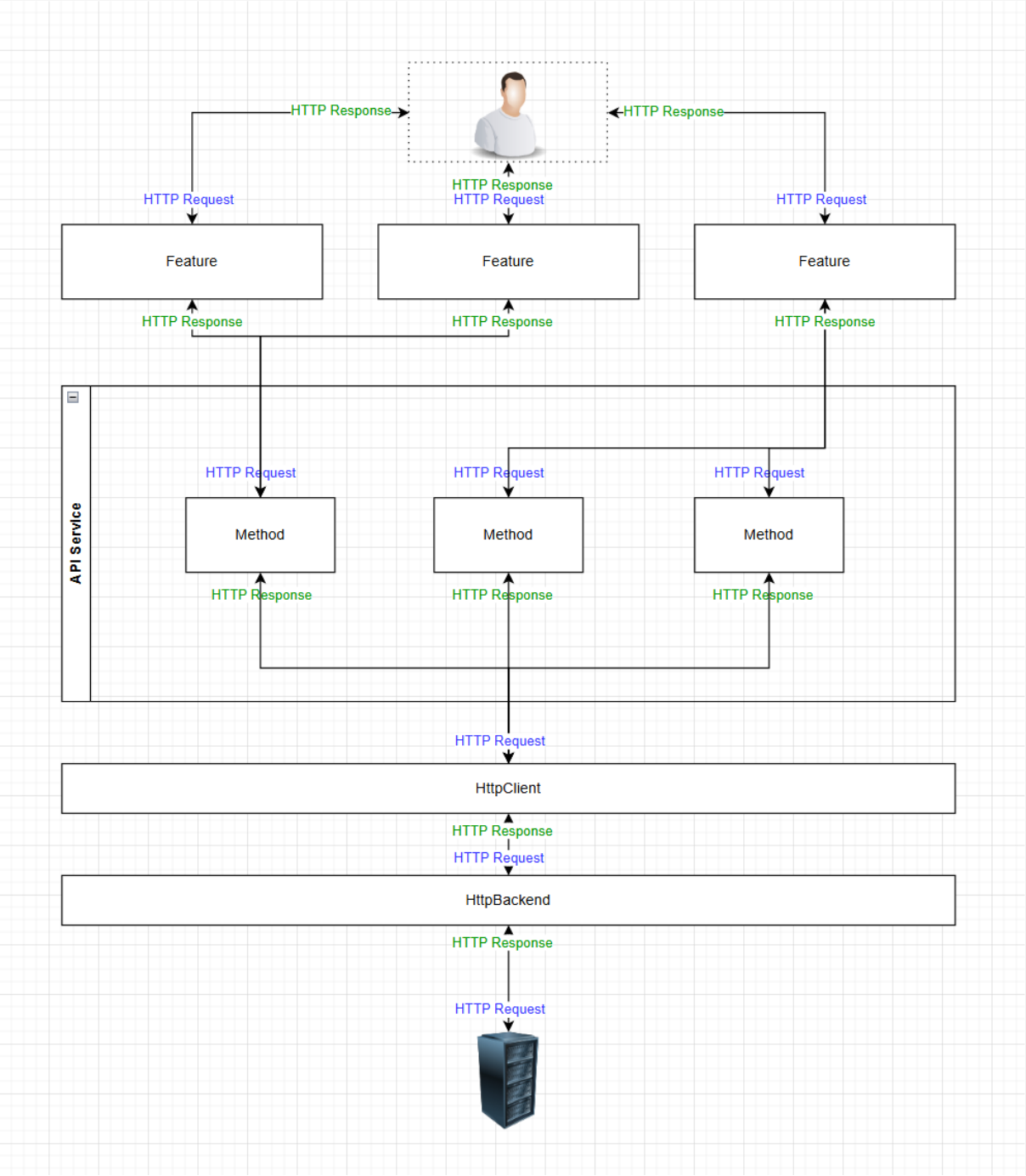
Dependencies: HttpClient[45]

Implements: -

Περιγραφή

Πρόκειται για υπηρεσία η οποία έχει σκοπό τη διασύνδεση πελάτη-διακομιστή, μέσω διεπαφής. Περιέχει μεθόδους οι οποίες χρησιμοποιούνται στο σύστημα για την αποκόμιση απομακρυσμένων δεδομένων. Κάθε μέθοδος έχει ορισμένες παραμέτρους και αντίστοιχο τύπο δεδομένων επιστροφής. Όλες οι μέθοδοι ενσωματώνουν τον HttpClient για την αποκόμιση των δεδομένων. Οι αιτήσεις που απαιτούν αποθήκευση της απάντησης τους στη μνήμη cache περιέχουν στην τιμή του `HttpContext` τους το κλειδί `AppConstants.HttpContextTokens.CACHE_RESPONSE` ορισμένο σε τιμή `true`.

Διάγραμμα



Εικόνα 2.5-3 Διάγραμμα API Service

Κώδικας

```
import { HttpClient, HttpContext, HttpParams } from
'@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable, forkJoin, map } from 'rxjs';
import { API } from './api';
import { AppConstants } from 'src/app/app.constants';

/**
 * Υπηρεσία διεπαφής απομακρυσμένων δεδομένων.
 */
@Injectable({
  providedIn: 'root'
})
export class ApiService {

  /**
   * Εισάγει τις απαραίτητες υπηρεσίες.
   *
   * @param http επιτρέπει κλήσεις σε απομακρυσμένες διεπαφές
   */
  constructor(private http: HttpClient) { }


  /**
   * Καλεί τα `/weather/airPollution/current` και
`/weather/airPollution/forecast` ακρότατα.
   *
   * @param location πληροφορίες τοποθεσίας
   *
   * @returns ομαδοποιημένες προβλέψεις ποιότητας αέρα
   */
  getAirPollutionIndicator(location: API.Location):
Observable<API.AirPollution.ResponseGrouped> {
    const options = {
      observe: 'body',
      context: new HttpContext()
        .set(AppConstants.HttpContextTokens.CACHE_RESPONSE, true),
      params: new HttpParams()
        .set('lat', location.coord.lat)
        .set('lon', location.coord.lon)
    } as const;

    const airPollutionRequests = {
      current:
this.http.get<API.AirPollution.Response>('/api/openWeather/airPolution',
options),
      forecast:
```

```
this.http.get<API.AirPollution.Response>('/api/openWeather/airPolution/forecast'
, options)
    }

    return forkJoin(airPollutionRequests)
      .pipe(map(({ current, forecast }) => {
        const groupByDatetime = (metrics: API.AirPollution.Metric[]):
API.AirPollution.MetricsByDate => metrics
          .reduce((datetimeIndex: { [k: number]: API.AirPollution.Metric },
metric) => {
            datetimeIndex[metric.dt] = metric;

            return datetimeIndex;
          }, {});

        return {
          current: current.list[0],
          forecast: groupByDatetime(forecast.list)
        };
      }));
  }

  /**
   * Καλεί το `/weather/forecastAnalysis` ακρότατο.
   *
   * @param location πληροφορίες τοποθεσίας
   *
   * @returns συλλογή πληροφοριών καιρού για τις επόμενες 5 μέρες / ανά 3 ώρες
   */
  getForecastAnalysis(location: API.Location):
Observable<API.ForecastAnalysis.Response> {
    const options = {
      observe: 'body',
      context: new HttpContext()
        .set(AppConstants.HttpContextTokens.CACHE_RESPONSE, true),
      params: new HttpParams()
        .set('lat', location.coord.lat)
        .set('lon', location.coord.lon)
        .set('units', 'metric')
    } as const;

    return
this.http.get<API.ForecastAnalysis.Response>('/api/openWeather/forecastAnalysis'
, options);
  }

  /**
```

```
   * Ανακτά τις διαθέσιμες τοποθεσίες.
   *
   * @returns διαθέσιμες τοποθεσίες
   */
  getLocations(): Observable<API.Location[]> {
    return this.http.get<API.Location[]>('assets/city.list.min.json')
  }

  /**
   * Καλεί το `/weather/all` ακρότατο.
   *
   * @see https://openweathermap.org/api/one-call-3
   *
   * @param location πληροφορίες τοποθεσίας
   *
   * @returns συλλογή πληροφοριών καιρού
   */
  getWeather(location: API.Location): Observable<API.OneCall.Response> {
    const options = {
      observe: 'body',
      context: new HttpContext()
        .set(AppConstants.HttpContextTokens.CACHE_RESPONSE, true),
      params: new HttpParams()
        .set('lat', location.coord.lat)
        .set('lon', location.coord.lon)
        .set('units', 'metric')
    } as const;

    return this.http.get<API.OneCall.Response>('/api/openWeather/onecall',
options)
  }
}
```

Πίνακας 1 Κώδικας API Service

2.5.1.2 *Authentication Interceptor*

Τοποθεσία: core/authentication/authentication.interceptor.ts

Όνομα Κλάσης: AuthenticationInterceptor

Σημείο Δήλωσης: app.config.ts

Dependencies: -

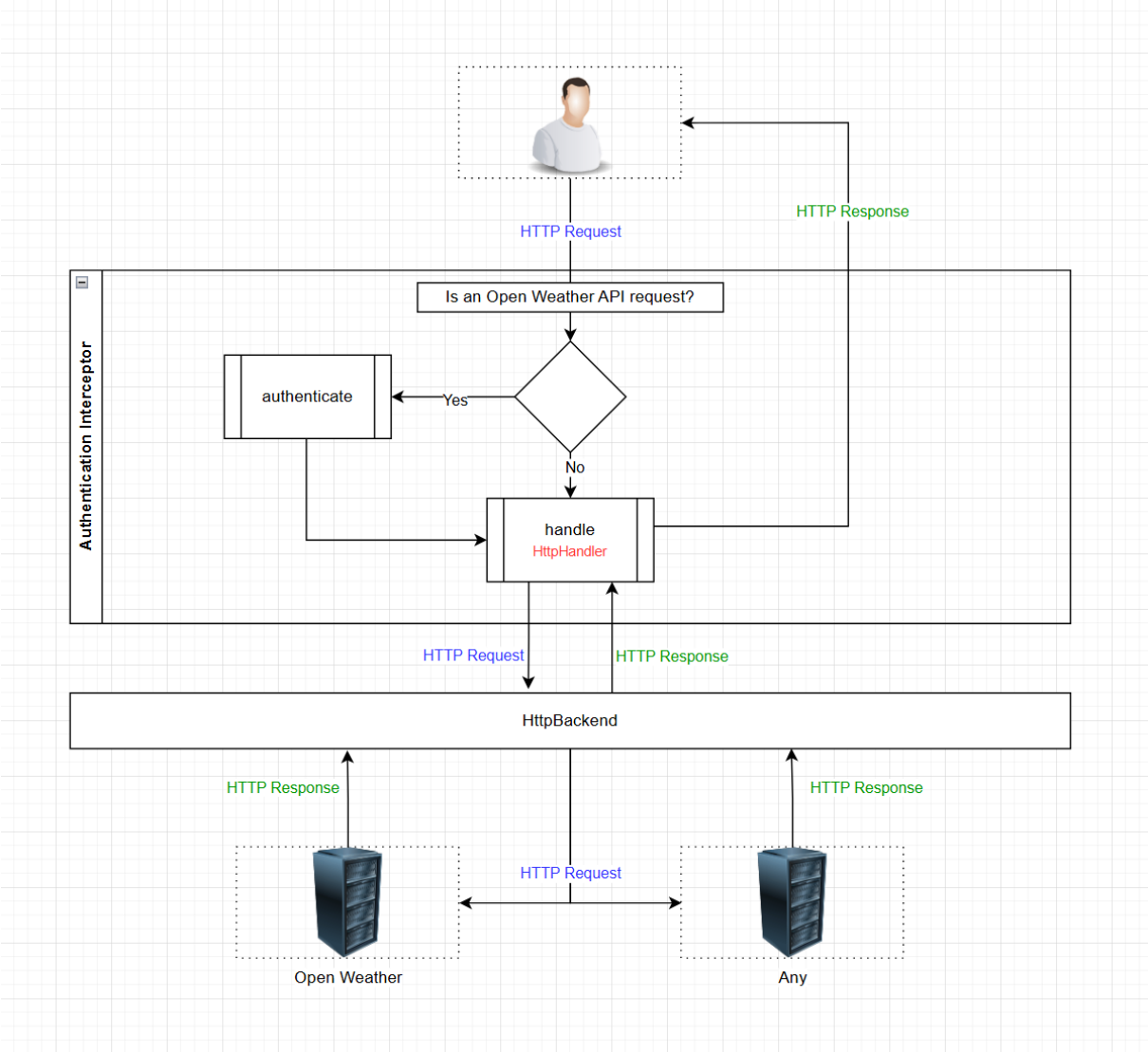
Implements: HttpInterceptor[1]

Περιγραφή

Μεσολαβητής απομακρυσμένων κλήσεων. Λειτουργεί ως σημείο τομής ανάμεσα στον πελάτη και τον διακομιστή για κάθε κλήση από και προς αυτόν. Επεξεργάζεται κάθε αίτηση του πελάτη προς τον διακομιστή και προωθεί την απάντηση του πελάτη στους ακροατές του συστήματος από τους οποίους προέκυψε η κλήση. Στην εφαρμογή του, ενσωματώνει τον κλειδάριθμο που αυθεντικοποιεί τον χρήστη της διεπαφής Open Weather *(βλέπε* 2.1.5*)*. Αφορά μόνο τα ακρότατα που περιέχουν το μονοπάτι `/api/openWeather`. Η απάντηση στην αίτηση του συστήματος προς τον διακομιστή προωθείται στον ακροατή.

Διάγραμμα

[1] Angular - HttpInterceptor



Εικόνα 2.5-4 Διάγραμμα Authentication Interceptor

Κώδικας

```
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor
} from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

/**
 * Μεσολαβητής απομακρυσμένων κλήσεων HTTP.
 *
 * Προσθέτει τα απαραίτητα στοιχεία αυθεντικοποίησης χρήστη σε συγκεκριμένες
 * απομακρυσμένες κλήσεις.
 */
@Injectable()
export class AuthenticationInterceptor implements HttpInterceptor {

  /**
   * Κύρια μέθοδος του μεσολαβητή.
   *
   * Προσθέτει στοιχεία αυθεντικοποίησης χρήστη σε ορισμένα ακρότατα.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @param next αναμεταδότης συμβάντος HTTP
   *
   * @returns απάντηση απομακρυσμένου διαμεσολαβητή
   */
  intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
    if (request.url.includes('/openWeather') &&
!request.params.has('appid')) {
      request = request.clone({
        setParams: {
          'appid': <string>environment.OPEN_WEATHER_API_KEY
        }
      });
    }

    return next.handle(request);
  }
}
```

Πίνακας 2 Κώδικας Authentication Interceptor

2.5.1.3 Cache Interceptor

Τοποθεσία: core/cache/cache.interceptor.ts

Όνομα Κλάσης: CacheInterceptor

Σημείο Δήλωσης: app.config.ts

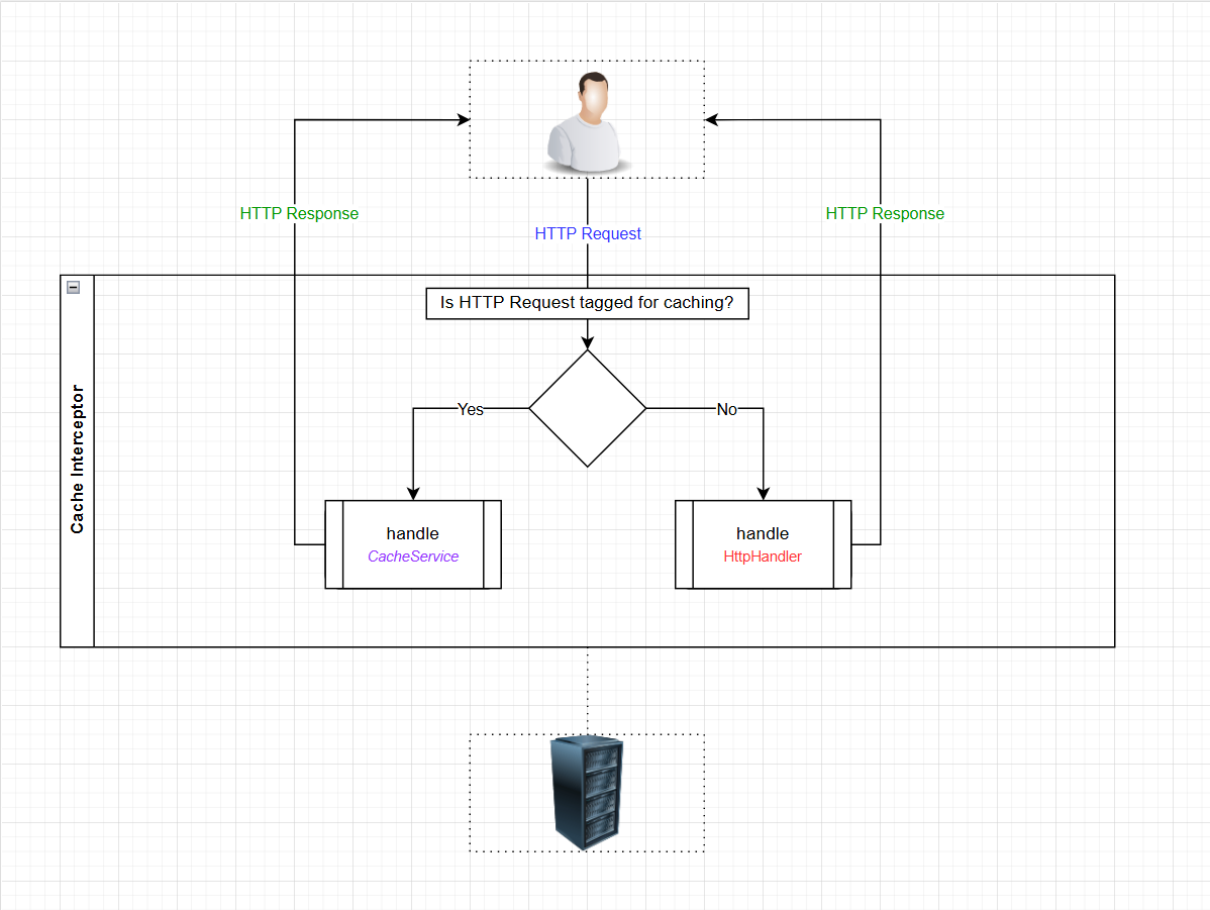
Dependencies: CacheService

Implements: HttpInterceptor

Περιγραφή

Μεσολαβητής απομακρυσμένων κλήσεων. Περιορίζει τις κλήσεις προς τον διακομιστή αποθηκεύοντας τις απαντήσεις του για ορισμένο χρονικό πλαίσιο *(βλέπε* 2.1.4*)*. Λειτουργεί ως σημείο τομής ανάμεσα στον πελάτη και τον διακομιστή για κάθε κλήση από και προς αυτόν. Επεξεργάζεται κάθε αίτηση του πελάτη προς τον διακομιστή και προωθεί την απάντηση του πελάτη στους ακροατές του συστήματος από τους οποίους προέκυψε η κλήση. Στην εφαρμογή του, καλεί τη βοηθητική μέθοδο `isCacheable` της `CacheService` για να ελέγξει αν η αίτηση απαιτεί την αποθήκευση της απάντησης στη μνήμη cache του συστήματος. Αν η αίτηση δεν απαιτεί αποθήκευση στην cache, τότε η απάντηση προωθείται στους ακροατές χωρίς τροποποίηση. Σε διαφορετική περίπτωση η απάντηση διαχειρίζεται από τη μέθοδο `handle` της `CacheService` η οποία επιστρέφεται. Η μέθοδος `handle` δέχεται ως όρισμα το αντικείμενο `request` που περιέχει την αίτηση και το αντικείμενο `next` που διαχειρίζεται την αναμετάδοση της απάντησης στους ακροατές.

Διάγραμμα



Εικόνα 2.5-5 Διάγραμμα Cache Interceptor

Κώδικας

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from
'@angular/common/http';
import { Injectable } from '@angular/core';
import { CacheService } from './cache.service';
import { Observable } from 'rxjs';


/**
 * Μεσολαβητής απομακρυσμένων κλήσεων HTTP.
 *
 * Ενσωματώνει και διαχειρίζεται τη μνήμη cache για απομακρυσμένες κλήσεις.
 *
 * Μόνο οι κλήσεις οι οποίες έχουν ενεργοποιήσει την επιλογή
 * θα ανακτηθούν/αποθηκευθούν από τη μνήμη.
 */
@Injectable()
export class CacheInterceptor implements HttpInterceptor {

  /**
   * @param cache υπηρεσία διαχείρισης μνήμης cache
   */
  constructor(private cache: CacheService) {}

  /**
   * Κύρια μέθοδος του μεσολαβητή.
   *
   * Εκτελεί το μηχανισμό μνήμης cache εφόσον είναι ενεργοποιημένος για
   * την τρέχουσα κλήση.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @param next αναμεταδότης συμβάντος HTTP
   *
   * @returns δεδομένα που έχουν ανακτηθεί σε μορφή συμβάντος ενεργούς
ανταπόκρισης
   */
  intercept(request: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
    if (!this.cache.isCacheable(request)) {
      return next.handle(request);
    }

    return this.cache.handle(request, next);
  }
```

```
}
```

Πίνακας 3 Κώδικας Cache Interceptor

2.5.1.4 Cache Service

Τοποθεσία: core/cache/cache.service.ts

Όνομα Κλάσης: CacheService

Σημείο Δήλωσης: *root*

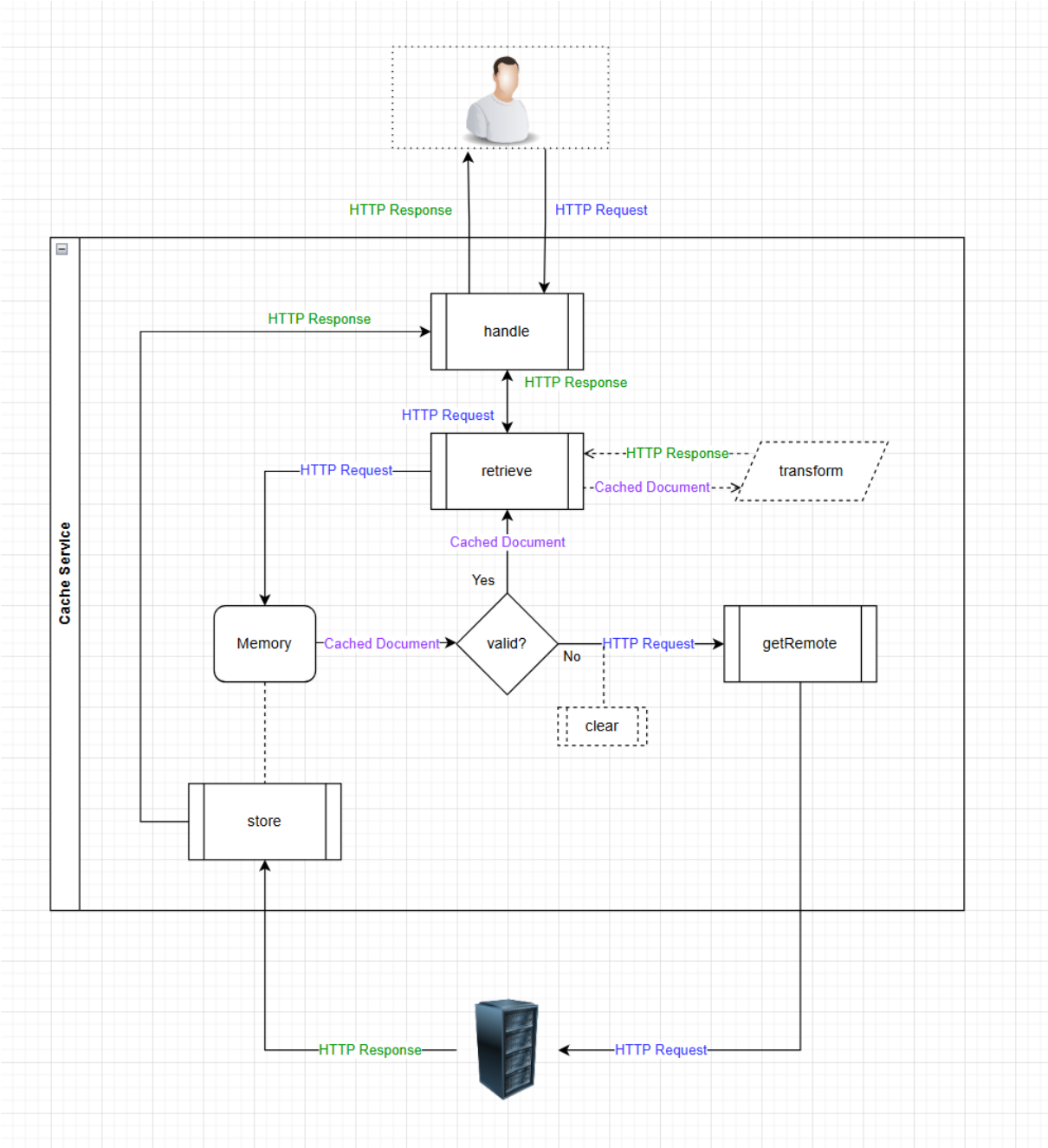
Dependencies: -

Implements: -

Περιγραφή

Υπηρεσία διαχείρισης της μνήμης cache. Χρησιμοποιείται από τον `CacheInterceptor` για την ενσωμάτωση του caching μηχανισμού. Σημείο εισόδου στη λογική του μηχανισμού cache είναι η μέθοδος `handle`. Εντός της μεθόδου γίνεται απόπειρα στο να ανακτηθούν δεδομένα από τη μνήμη μέσω της μεθόδου `retrieve`. Η μέθοδος `retrieve` θα επιστρέψει `Observable` που περιέχει την αποθηκευμένη στην cache τιμή, και σε περίπτωση που δεν βρεθεί τιμή ή δεν είναι έγκυρη, θα επιστρέψει τιμή σφάλματος. Στην πρώτη περίπτωση, η τιμή επιστρέφεται ως έχει, αφού ανακατασκευαστεί σε ένα `HttpResponse` αντικείμενο. Στην περίπτωση σφάλματος, ενεργοποιείται ο χειριστής `catchError`, ο οποίος καλεί τη `getRemote` για την ανάκτηση απομακρυσμένων δεδομένων. Αν τα απομακρυσμένα δεδομένα επιστραφούν επιτυχώς αποθηκεύονται στη μνήμη cache μέσω της `store`. Τα δεδομένα αποθηκεύονται στη μεταβλητή `memory` αλλά και στον τοπικό αποθηκευτικό χώρο Local Storage του περιηγητή. Ο τοπικός αποθηκευτικός χώρος ενεργοποιεί τη χρήση επιμενόντων δεδομένων (persistent). Έτσι τα δεδομένα είναι διαθέσιμα ανάμεσα στις επανεκκινήσεις της εφαρμογής. Λόγω της φύσης της εφαρμογής, η μεταβλητή `memory` δεν προσπελαύνεται ποτέ, αφού η εφαρμογή είναι μονής σελίδας και μόνο με ανανέωση σελίδας γίνεται αίτηση για νέα δεδομένα. Τα δεδομένα της `memory` είναι παροδικά, συνεπώς κάθε φορά που η εφαρμογή κλείνει τα δεδομένα χάνονται. Στην πράξη η `memory` κρατάει τα δεδομένα της μέχρι την επόμενη ανανέωση σελίδας, και άρα καμία αίτηση δεδομένων δεν παίρνει δεδομένα από αυτή, αλλά από τα επιμένοντα δεδομένα που έχουν αποθηκευτεί στον τοπικό αποθηκευτικό χώρο. Για λόγους πληρότητας αποφάσισα να ενσωματώσω και παροδική μνήμη για τα δεδομένα. Σε μελλοντικές προσθήκες της εφαρμογής όπως για παράδειγμα επιπλέων σελίδων ο μηχανισμός cache θα είναι σε θέση να συνεχίζει να λειτουργεί όπως αναμένεται. Όσον αφορά τα δεδομένα, η εγκυρότητα τους ελέγχεται από τη μέθοδο `isValid` η οποία δέχεται μια τιμή τύπου `App.CacheDocument` κι επιστρέφει το συμπέρασμα. Η μέθοδος ελέγχει αν η παράμετρος είναι υπαρκτή κι εφόσον είναι προχωράει τον έλεγχο της ημερομηνίας της, συγκρίνοντας αθροιστικά την ημερομηνία δημιουργίας της και τον χρόνος ζωής της μνήμης cache με την τρέχουσα ημερομηνία τη στιγμή του ελέγχου. Εφόσον το άθροισμα δεν υπερβαίνει την τρέχουσα ημερομηνία, τα δεδομένα είναι έγκυρα και η μέθοδος επιστρέφει αληθής τιμή. Δεδομένα τα οποία δεν είναι έγκυρα διαγράφονται μέσω της μεθόδου `clear`.

Διάγραμμα



Εικόνα 2.5-6 Διάγραμμα Cache Service

Κώδικας

```ts
import { HttpEvent, HttpHandler, HttpRequest, HttpResponse } from
'@angular/common/http';
import { Injectable } from '@angular/core';
import { catchError, Observable, of, tap, throwError } from 'rxjs';
import { AppConstants } from 'src/app/app.constants';

/**
 * Υπηρεσία διαχείρισης μνήμης cache.
 *
 * Αποθηκεύει και ανακτά δεδομένα απόκρισης της απομακρυσμένης διεπαφής.
 */
@Injectable({
  providedIn: 'root'
})
export class CacheService {

  /**
   * Διάρκεια εγκυρότητας αποθηκευμένων στη μνήμη δεδομένων
   * σε λεπτά.
   */
  readonly TIME_TO_LIVE: number = 15;

  /**
   * Ιδιωτική τιμή της cache για αποθήκευση δεδομένων στην εικονική μνήμη.
   *
   * Προσπελαύνεται πριν την τοπική μνήμη.
   */
  private memory = new Map<string, App.CacheDocument>();

  /**
   * Κλειδί ενεργοποίησης του caching μηχανισμού.
   * Περιλαμβάνεται στις αιτήσεις προς τον διακομιστή που αιτούνται
   * να συμπεριληφθούν στο μηχανισμό caching.
   */
  readonly CacheToken = AppConstants.HttpContextTokens.CACHE_RESPONSE;

  /**
   * Βοηθητική μέθοδος για την ανάκτηση του κλειδιού
   * του αντικειμένου για την αίτηση που αντιστοιχεί
   * στη μνήμη cache. Χρησιμοποιείται για λόγους πιστότητας.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @returns κλειδί αντικειμένου cache
   */
```

```
  cacheKey(req: HttpRequest<any>): string {
    return req.url;
  }

  /**
   * Αφαιρεί από τη μνήμη την τιμή που σχετίζεται
   * με την παράμετρο αίτησης.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   */
  clear(req: HttpRequest<any>): void {
    this.memory.delete(this.cacheKey(req));
    localStorage.removeItem(req.url)
  }

  /**
   * Ανακτά από την τοπική μνήμη (Local Storage) τα αποθηκευμένα δεδομένα
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @returns δεδομένα αποθηκευμένα στον τοπικό χώρο αποθήκευσης (local
storage)
   */
  getLocal(req: HttpRequest<any>): App.CacheDocument | undefined {
    const storedValue: string = localStorage.getItem(this.cacheKey(req));

    if (!storedValue) {
      return undefined;
    }

    const { createdOn, value } = JSON.parse(storedValue);
    const documentReconstruct: App.CacheDocument = {
      createdOn: new Date(createdOn),
      value: JSON.parse(value)
    }

    return documentReconstruct;
  }

  /**
   * Αναμεταδίδει την αίτηση δεδομένων στην απομακρυσμένη διεπαφή για να
επιστρέψει
   * τα τελευταία δεδομένα. Κατά την επιστροφή αποθηκεύει την απάντηση στη
   * μνήμη cache.
   *
```

```
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @param next αναμεταδότης συμβάντος HTTP
   *
   * @returns απομακρυσμένα δεδομένα σε μορφή συμβάντος ενεργούς
ανταπόκρισης
   */
  getRemote(req: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
    return next.handle(req)
      .pipe(
        tap((value) => {
          if (value instanceof HttpResponse) {
            this.store(req, value);
          }
        })
      );
  }

  /**
   * Αποφασίζει αν θα επιστραφούν δεδομένα από την cache ή αν
   * θα γίνει αίτηση για απομακρυσμένα δεδομένα. Αν η retrieve απαντήσει
   * αρνητικά στην ανάκτηση δεδομένων, τότε θα γίνει κλήση για
απομακρυσμένα.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @param next αναμεταδότης συμβάντος HTTP
   *
   * @returns δεδομένα που έχουν ανακτηθεί απομακρυσμένα ή τοπικά,
   * σε μορφή συμβάντος ενεργούς ανταπόκρισης
   */
  handle(req: HttpRequest<any>, next: HttpHandler): Observable<any> {
    return this.retrieve(req)
      .pipe(
        catchError(() => this.getRemote(req, next))
      )
  }

  /**
   * Ελέγχει αν η αίτηση πληροί τις προϋποθέσεις διαχείρισης της
   * από τη μνήμη cache.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
```

```
   * @returns αποτέλεσμα πρότασης
   */
  isCacheable(req: HttpRequest<any>): boolean {
    return req.context.has(this.CacheToken) &&
req.context.get(this.CacheToken);
  }

  /**
   * Ελέγχει την εγκυρότητα των αποθηκευμένων δεδομένων.
   *
   * Η εγκυρότητα ελέγχεται βάσει ημερομηνίας λήξεως των δεδομένων.
   *
   * Η ημερομηνία λήξεως των δεδομένων υπολογίζεται βάσει της στατικής
   * τιμής της `TIME_TO_LIVE` παραμέτρου.
   *
   * Αν η παράμετρος είναι null ή undefined, επιστρέφεται αρνητική απάντηση.
   *
   * @param cacheDocument αντικείμενο της μνήμης cache
   *
   * @returns τιμή αληθείας της προκείμενης
   */
  isValid(cacheDocument: App.CacheDocument | undefined): boolean {
    if (!cacheDocument) return false;

    const { createdOn } = cacheDocument;
    const expiresInMinutes = createdOn.getMinutes() + this.TIME_TO_LIVE;
    const timestamp = createdOn.setMinutes(expiresInMinutes)
    const expirationDate = new Date(timestamp);
    const isValid = expirationDate > new Date();

    return isValid;
  }

  /**
   * Ανακτά τα δεδομένα που έχουν αποθηκευτεί στη μνήμη cache.
   * Αν τα δεδομένα δεν είναι έγκυρα θα επιστρέψει τιμή σφάλματος.
   * Εφόσον τα δεδομένα είναι έγκυρα, επαναδημιουργεί αντικείμενο
`HttpResponse`.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @returns απόκριση δεδομένων που έχουν ανακτηθεί από τη μνήμη
   * ή τιμή λάθους αν τα δεδομένα δεν είναι έγκυρα
   */
  retrieve(req: HttpRequest<any>): Observable<HttpResponse<any>> {
```

```
    const storedValue: App.CacheDocument | undefined =
this.memory.has(this.cacheKey(req))
      ? this.memory.get(this.cacheKey(req))
      : this.getLocal(req);

    const isValid: boolean = this.isValid(storedValue);

    // τα δεδομένα δεν είναι έγκυρα, ή δεν υπάρχουν
    if (!isValid) {
      this.clear(req);

      return throwError(() => new Error('Cache miss'))
    }

    const response = new HttpResponse({ body: storedValue.value.body });

    return of(response);
  }

  /**
   * Αποθηκεύει την απάντηση της αίτησης στη μνήμη.
   *
   * @param req αίτηση απομακρυσμένων δεδομένων
   *
   * @param response απομακρυσμένη απάντηση
   */
  store(req: HttpRequest<any>, response: HttpResponse<any>): void {
    const document: App.CacheDocument = {
      createdOn: new Date(),
      value: JSON.stringify(response)
    };
    const cacheKey = this.cacheKey(req);

    this.memory.set(cacheKey, document);
    localStorage.setItem(cacheKey, JSON.stringify(document))
  }

}
```

Πίνακας 4 Κώδικας Cache Service

2.5.2 Shared

Ο φάκελος shared περιέχει κοινά στην εφαρμογή components και pipes.

2.5.2.1 AQI Component

Τοποθεσία: shared/aqi/aqi.component.ts

Όνομα Κλάσης: AqiComponent

Dependencies: -

Περιγραφή

Στοιχείο το οποίο εμφανίζει τιμή για τον δείκτη ατμοσφαιρικής ρύπανσης (Air Quality Index) και αντίστοιχο δείκτη της κατάστασης η οποία περιγράφεται με χρωματική ένδειξη. Χρησιμοποιεί το Bootstrap popover component [2] το οποίο εισαγάγει μέσω του `NgbPopoverModule`. Με αυτό τον τρόπο ενσωματώνει δυνατότητα προβολής δυναμικού περιεχομένου σε αιωρούμενο στοιχείο (overlay). Το περιεχόμενο του αιωρούμενου στοιχείου εμφανίζει πίνακα με ανάλυση των στοιχείων και των τιμών τους που χαρακτηρίζουν τα ποσοστά ρύπανσης της ατμόσφαιρας. Τέλος περιγράφει λεκτικά την ποιότητα του αέρα. Η αντιστοίχιση των στοιχείων της ατμόσφαιρας και των περιγραφών τους, της χρωματικής παλέτας με τη σοβαρότητα του δείκτη ρύπανσης, και του λεκτικού με τον δείκτη, γίνεται με τη βοήθεια σταθερών που έχουν δηλωθεί ρητά στο σύστημα.

Κώδικας

[2] Angular powered Bootstrap

```
import { Component, Input } from '@angular/core';
import { API } from '../../core/api/api';
import { NgbPopoverModule } from '@ng-bootstrap/ng-bootstrap';
import { CommonModule } from '@angular/common';
import { AppConstants } from 'src/app/app.constants';


/**
 * Εμφανίζει Δείκτη Ατμοσφαιρικής Ρύπανσης (Air Quality Index)
 * με επεκτάσιμη διεπαφή εμφάνισης ειδικών λεπτομερειών.
 */
@Component({
  selector: 'wui-aqi',
  standalone: true,
  template: `
    <!-- aqi button -->
    <button
      class="wui-btn btn btn-outline-secondary"
      popoverClass="aqi-popover"
      triggers="click"
      type="button"
      [ngbPopover]="popContent"
    >
      <div class="d-flex align-items-center">
        <div>AQI {{ value.main.aqi }}</div>
        <div class="ms-2 aqi-status-indicator" [ngStyle]="{
          'background-color': severityColor
        }"></div>
      </div>
    </button>
    <!-- end aqi button -->

    <!-- popover content -->
    <ng-template #popContent>
      <h4>Air Pollution</h4>
      <table class="table table-bordered table-sm">
        <thead>
          <tr>
            <th>Gas</th>
            <th>Concentration (μg/m<sup>3</sup>)</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let component of airComponents">
            <td class="mx-3">{{ component.name }}</td>
```

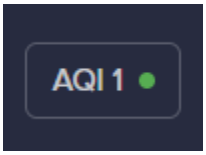
```
            <td class="mx-3">{{ component.value }}</td>
          </tr>
        </tbody>
        <caption>
          Air quality is
          <span [ngStyle]="{ 'color': severityColor }">
            {{ severity }}
          </span>
        </caption>
      </table>
    </ng-template>
    <!-- end popover content -->
  `,
  styles: [`
    :host ::ng-deep
      .aqi-popover
        min-width: 400px !important

      .aqi-status-indicator
        width: .5rem
        display: inline-flex
        height: .5rem
        border-radius: 50%
  `],
  imports: [
    CommonModule,
    NgbPopoverModule
  ]
})
export class AqiComponent {

  /**
   * Τιμή του component.
   * Περιέχει όλες τις απαραίτητες πληροφορίες.
   */
  @Input() value: API.AirPollution.Metric;

  /**
   * Λίστα με επιβλαβή ατμοσφαιρικά στοιχεία και τις αντίστοιχες τιμές
μέτρησης.
   */
  airComponents: { name: App.AirComponent; value: number; }[] = [];

  /**
   * Χρωματικός δείκτης της ποιότητας αέρα.
```

```
   */
  severityColor: App.AirQualitySeverityColor;

  /**
   * Λεκτικός δείκτης της ποιότητας αέρα.
   */
  severity: App.AirQualitySeverityRating;

  /**
   * Μέθοδος της Angular η οποία τρέχει κατά την εκκίνηση του component.
   *
   * Αρχικοποεί τις ιδιότητες που περιγράφουν την κατάσταση της ποιότητας
του αέρα.
   */
  ngOnInit(): void {
    this.generateComponentList();
    this.rateAirQuality();
  }

  /**
   * Δημιουργεί λίστα με ατμοσφαιρικά στοιχεία και τις τιμές τους.
   */
  generateComponentList(): void {
    Object.keys(this.value.components)
      .forEach((component) => {
        const componentName = AppConstants.AirComponentsInfo[component];

        this.airComponents.push({
          name: componentName,
          value: this.value.components[component]
        })
      })
  }

  /**
   * Χαρακτηρίζει την ποιότητα αέρα θέτοντας τιμή
   * χρωματικής και λεκτικής ένδειξης.
   */
  rateAirQuality(): void {
    const index = this.value.main.aqi - 1;

    this.severityColor = AppConstants.AirQualitySeverityColorRange[index];
    this.severity = AppConstants.AirQualitySeverityRatingRange[index];
  }
```

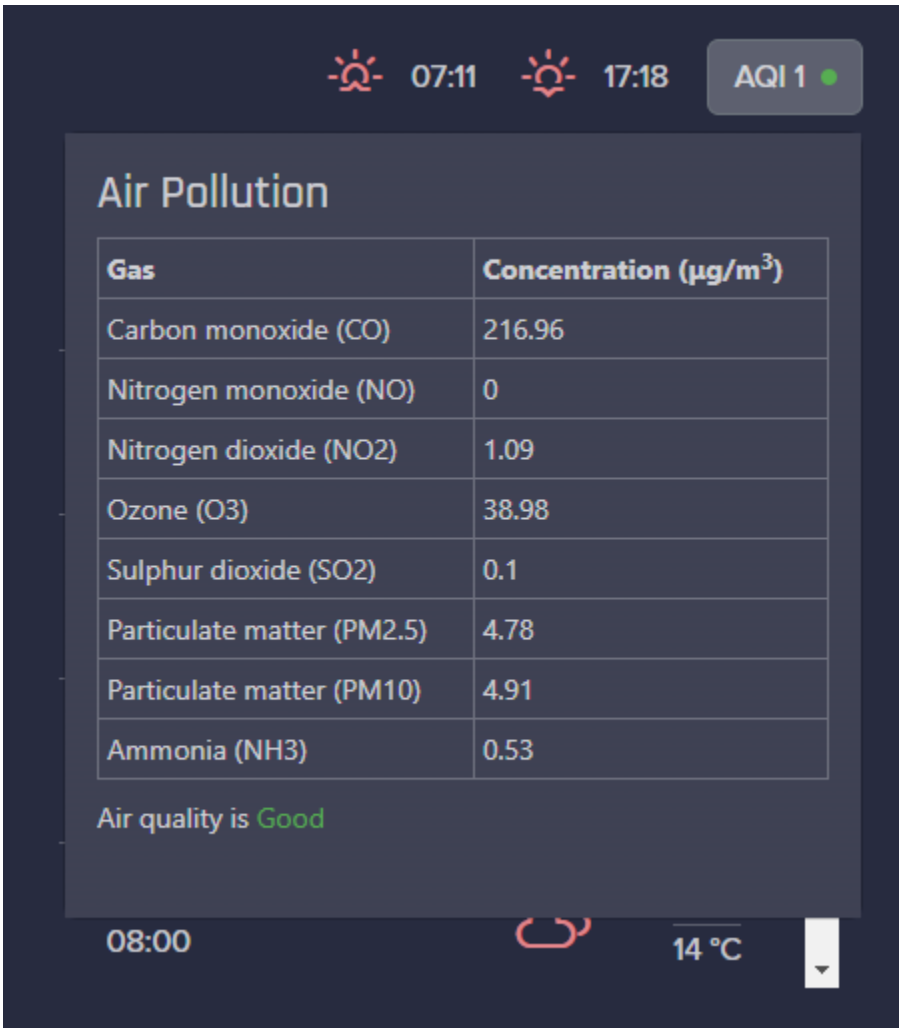
```
}
```

Πίνακας 5 Κώδικας AQI Component

Αποτέλεσμα



Εικόνα 2.5-7 AQI Component σε αδρανή κατάσταση



Εικόνα 2.5-8 AQI Component ενεργοποιημένο από τον χρήστη

2.5.2.2 ToFixed Pipe

Τοποθεσία: shared/converters/toFixed/to-fixed.pipe.ts

Όνομα Κλάσης: ToFixedPipe

Dependencies: -

Περιγραφή

Pipe [3] που χρησιμοποιείται εντός των templates στο σύστημα για τη μετατροπή δεκαδικών αριθμών σε φυσικούς. Η χρήση pipes ενδείκνυται για χρήση μετατροπής δεδομένων στα templates λόγω του ότι είναι διαμορφωμένα από την Angular για την καλύτερη απόδοση τους εντός του Angular Lifecycle [4]

Κώδικας

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'toFixed',
  standalone: true
})
export class ToFixedPipe implements PipeTransform {

  /**
   * Μετατρέπει δεκαδικούς αριθμούς σε φυσικούς.
   *
   * Χρησιμοποιείται στο template έναντι της ενσωματωμένης στο αντικείμενο
αριθμού
   * μεθόδου μετατροπής (toFixed()) για καλύτερο έλεγχος της λειτουργίας
   * στον κύκλο ζωής της Angular (Angular lifecycle).
   *
   * @param value δεκαδικός αριθμός προς μετατροπή
   *
   * @returns φυσικός αριθμός σε τύπο string
   */
  transform(value: number): string {
    return value.toFixed();
  }

}
```

Πίνακας 6 Κώδικας ToFixed Pipe

[3] Pipes • Angular

[4] What are Pipes in Angular and Why Should We Use Them? | Bits and Pieces

2.5.2.3 ToJSDate Pipe

Τοποθεσία: shared/converters/toJSDate/to-jsdate.pipe.ts

Όνομα Κλάσης: ToJSDatePipe

Dependencies: -

Περιγραφή

Μετατρέπει ημερομηνίες UNIX του Open Weather σε ημερομηνίες συμβατές με JavaScript.

Κώδικας

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'toJSDate',
  standalone: true
})
export class ToJSDatePipe implements PipeTransform {

  /**
   * Μετατρέπει χρόνο μορφής UNIX σε μορφή συμβατή με τη JavaScript.
   *
   * Το OpenWeather API χρησιμοποιεί χρόνο μορφή UNIX, ο οποίος
   * μετράται σε δευτερόλεπτα από την ημερολογιακή αρχή (Epoch),
   * ενώ η JavaScript μετράει σε χιλιοστά του δευτερολέπτου.
   *
   * @param unixDt χρόνος σε μορφή UNIX
   *
   * @returns χρόνος σε μορφή συμβατή με JavaScript
   */
  transform(unixDt: number): number {
    return unixDt * 1000;
  }

}
```

Πίνακας 7 Κώδικας ToJSDate Pipe

2.5.2.4 Icon Component

Τοποθεσία: shared/icon/icon.component.ts

Όνομα Κλάσης: IconComponent

Dependencies: -

Περιγραφή

Κοινό component για την ενσωμάτωση εικονιδίων. Φροντίζει για τη σωστή στοιχειοθέτηση του εικονιδίου σε σχέση με τα υπόλοιπα στοιχεία και προσφέρει κοινή διεπαφή για όλα τα εικονίδια του συστήματος.

Κώδικας

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AppConstants } from 'src/app/app.constants';

/**
 * Προβάλλει εικονίδιο καιρικής συνθήκης.
 */
@Component({
  selector: 'wui-icon',
  standalone: true,
  imports: [
    CommonModule
  ],
  template: `
    <div class="d-inline-flex p-2">
      <div class="align-items-center d-flex icon-placeholder justify-
content-center">
        <i [ngStyle]="{ 'font-size.em': size }" [class]="styleClass"></i>
      </div>
    </div>
  `,
  styles: `
    $icon-color: var(--wui-color-secondary-light)

    :host
      &.full-width .icon-placeholder
        width: 100%

      .wi
        color: $icon-color

    .icon-placeholder
      width: 40px
  `
})
export class IconComponent {

  /**
   * Μέγεθος εικονιδίου σε μονάδες em.
   */
  @Input() size: number;

  /**
   * Κωδικός εικονιδίου συσχετιζόμενος με τη
   * συνθήκη καιρού.
```

```
   */
  @Input() value: App.SystemIcon;

  /**
   * Τιμή εικονιδίου. Πρόκειται για τη CSS κλάση.
   */
  styleClass: string;

  /**
   * Αρχικοποιεί το component.
   */
  ngOnInit(): void {
    this.setStyleClass();
  }

  /**
   * Ορίζει τη CSS κλάση που περιέχει το εικονίδιο.
   */
  setStyleClass(): void {
    this.styleClass = AppConstants.SystemIconMap[this.value];
  }

}
```

Πίνακας 8 Κώδικας Icon Component

2.5.3 Features

Ο φάκελος features περιέχει όλα τα χαρακτηριστικά στοιχεία της εφαρμογής. Στα πλαίσια αυτής της εργασίας το μόνο χαρακτηριστικό στην εφαρμογή είναι το **forecast**. Λαμβάνοντας υπόψιν τα παραπάνω, για την παρουσίαση των features και συγκεκριμένα του forecast θα χρησιμοποιήσω επίπεδη κατηγοριοποίηση και θα παρουσιάσω το κάθε στοιχείο και υπό-στοιχείο που απαρτίζουν το forecast στο ίδιο επίπεδο. Για διευκόλυνση στην απεικόνιση των σχέσεων θα περιλάβω στα χαρακτηριστικά τίτλου του κάθε component το πεδίο “Συσχέτιση” με πιθανές τιμές “Γονέας” και “Παιδί”. Έτσι, το κεντρικό component που ορίζει το forecast και περιλαμβάνει τα παιδιά components θα ονομάζεται “Γονέας” ενώ τα παιδιά του “Παιδί”. Το forecast περιλαμβάνει όλη τη λογική του στην υπηρεσία `ForecastService`. Όλα τα components έχουν μόνο ευθύνη για να λάβουν και να παρουσιάσουν δεδομένα. Με αυτό τον τρόπο εξασφάλισα κεντρικοποιημένη διαχείριση της λογικής του feature. Για την παρουσίαση του forecast θα ξεκινήσω από την υπηρεσία του.

2.5.3.1 Forecast Service

Τοποθεσία: features/forecast/forecast.service.ts

Όνομα Κλάσης: ForecastService

Σημείο Δήλωσης: *root*

Dependencies:ApiService

Περιγραφή

Η υπηρεσία περιλαμβάνει μεθόδους για τη διαχείριση τοποθεσιών και δελτίων καιρού. Ο διαμοιρασμός του δελτίου γίνεται με την παράμετρο `event`. Προαπαιτούμενο για να μεταδώσει η `event` δεδομένα είναι να γίνει η κλήση της `getForecast`. Η `getForecast` ανακτά τα απομακρυσμένα δεδομένα και τα αναμεταδίδει μέσω της event. Επίσης τα δεδομένα επιστρέφονται από τη `getForecast` στους ενδιαφερόμενους που την κάλεσαν. Η `search` είναι η εισαγωγική μέθοδος για ανάκτηση δελτίου καιρού με ελεύθερης μορφής όνομα τοποθεσίας. Η μέθοδος αυτή θα ψάξει να βρει την τοποθεσία και θα την προωθήσει στην `getForecast`. Αν κάποια από τις δύο αυτές ενέργειες αποτύχει θα επιστρέψει σφάλμα.

Κώδικας

```
import { Injectable } from '@angular/core';
import { API } from '@core/api/api';
import { ApiService } from '@core/api/api.service';
import { BehaviorSubject, Observable, forkJoin, map, mergeMap, of,
switchMap, tap, throwError } from 'rxjs';


/**
 * Κύρια υπηρεσία ανάκτησης και διαχείρισης της πρόβλεψης καιρού
 */
@Injectable({
  providedIn: 'root'
})
export class ForecastService {

  /**
   * Συμβάν που περιέχει τις απαραίτητες πληροφορίες καιρού.
   */
  event: BehaviorSubject<App.ForecastEvent> = new BehaviorSubject({} as
App.ForecastEvent);

  /**
   * @param api υπηρεσία διεπαφής απομακρυσμένων δεδομένων.
   */
  constructor(private api: ApiService) {}

  /**
   * Εμπλουτίζει την τιμή με δείκτες που χρησιμεύουν στο να υποδείξουν
   * το περιεχόμενο του κάθε στοιχείου λίστας. Συγκεκριμένα προσπελαύνεται
   * η ημερομηνία του κάθε στοιχείου καιρού και συγκριτικά με άλλες τιμές
καθορίζονται
   * κάποια μεταδεδομένα. Τα μεταδεδομένα τα οποία περιλαμβάνονται εν τέλει
στο κάθε στοιχείο
   * είναι τα:
   *
   * - `_$relativeDayName`: Υποδεικνύει αν το δελτίο καιρού αφορά τη
σημερινή ή την αυριανή ημέρα.
   * Τα στοιχεία που έχουν αυτή την τιμή την εμφανίζουν έναντι της
πραγματικής ημέρα.
   *
   * - `_$isFirstHour`: Υποδεικνύει την πρώτη ώρα κάθε ημέρας για να
υπογραμμιστεί το στοιχείο.
   *
   * @param value τιμή λίστας για το δελτίο καιρού ανά ώρα
   *
```

```
   * @returns εμπλουτισμένη τιμή με δείκτες της ποιότητας των δεδομένων
   */
  enrichHourForecast(value: API.ForecastAnalysis.HourForecast[]):
API.HourlyForecast[] {
    return value.map((hour, index) => {
      const forecastDate = new Date(hour.dt * 1000);
      const currentDate = new Date();

      // Το δελτίο καιρού αφορά τη σημερινή ημέρα.
      if (forecastDate.toDateString() === currentDate.toDateString()) {
        (<API.HourlyForecast>hour)._$relativeDayName = 'Today';
      }

      // Το δελτίο καιρού αφορά την αυριανή ημέρα.
      const tomorrow = new Date(currentDate);
      tomorrow.setDate(currentDate.getDate() + 1);
      if (forecastDate.toDateString() === tomorrow.toDateString()) {
        (<API.HourlyForecast>hour)._$relativeDayName = 'Tomorrow';
      }

      // Ελέγχει αν το δελτίο καιρού είναι το πρώτο μιας ημέρας.
      if (index > 0) {
        const previousHour = new Date(value[index - 1].dt * 1000);

        (<API.HourlyForecast>hour)._$isFirstHour = forecastDate.getDate()
!== previousHour.getDate();
      } else {
        (<API.HourlyForecast>hour)._$isFirstHour = false;
      }

      return hour;
    }) as API.HourlyForecast[];
  }

  /**
   * Αναζητά την τοποθεσία στη λίστα με τις τοποθεσίες.
   * Αν δεν τη βρει θα επιστρέψει σφάλμα.
   *
   * @param name όνομα τοποθεσίας σαν όρος αναζήτης
   *
   * @returns τοποθεσία αν τη βρει αλλιώς περιγραφικό σφάλμα
   */
  findLocation(name: string): Observable<API.Location> {
    return this.api.getLocations()
      .pipe(
```

```
        switchMap((value) => {
          const found: API.Location = this.matchLocation(value, name);

          if (!found) {
            return throwError(() => new Error('Location is not found'));
          }

          return of(found);
        }),
      )
  }

  /**
   * Αιτείται την πρόβλεψη του καιρού για τη δοθείσα τοποθεσία.
   *
   * Η αίτηση συνδιάζει δεδομένα διαφόρων πηγών οι οποίες κατασκευάζονται
   * σε ένα τελικό αντικείμενο τύπου `ForecastEvent`. Το αντικείμενο
αποστέλεται
   * στους ακροατές μέσω του αναμεταδότη `event`..
   *
   * @param location η τοποθεσία για την οποία ανακτάται η πρόβλεψη
   *
   * @returns πρόβλεψη καιρού για τη δοθείσα τοποθεσία
   */
  getForecast(location: API.Location): Observable<App.ForecastEvent> {
    const requests = {
      weather: this.api.getWeather(location),
      analysis: this.api.getForecastAnalysis(location),
      airQuality: this.api.getAirPollutionIndicator(location)
    };

    return forkJoin(requests)
      .pipe(
        map((value) => {
          const event: App.ForecastEvent = { ...value, location }

          return event;
        }),
        tap((value) => {
          this.event.next(value);
        })
      )
  }

  /**
```

```
   * Αντιστοιχεί το όνομα της τοποθεσίας σε μια λίστα με τοποθεσίες.
   *
   * @param value λίστα με τις διαθέσιμες τοποθεσίες
   *
   * @param name όνομα τοποθεσίας για την αντιστοίχιση
   *
   * @returns την τοποθεσία που αντιστοίχισε
   */
  matchLocation(value: API.Location[], name: string): API.Location {
    return value.find((location) => {
      return location.name.toLocaleLowerCase() === name.toLocaleLowerCase()
    })
  }

  /**
   * Αναζήτηση δελτίου καιρού βάσει τοποθεσίας.
   * Εντοπίζει την τοποθεσία και την προωθεί για αναζήτηση δελτίου καιρού.
   *
   * @param name ονομα τοποθεσίας σε ελεύθερη μορφή
   *
   * @returns συμβάν που περιέχει το δελτίο καιρού ή σφάλμα αν δεν γίνει
ανάκτηση
   */
  search(name: string): Observable<App.ForecastEvent> {
    return this.findLocation(name)
      .pipe(
        mergeMap((location) => this.getForecast(location))
      )
  }

}
```

Πίνακας 9 Κώδικας Forecast Service

2.5.3.2 *Forecast Component*

Τοποθεσία: features/forecast/forecast.component.ts

Όνομα Κλάσης: ForecastComponent

Σημείο Χρήσης: *app.component.html*

Dependencies: ForecastService

Συσχέτιση: Γονέας

Περιγραφή

Κεντρικό στοιχείο-γονέας για το Forecast feature. Περιλαμβάνει υπό-στοιχεία που το συνθέτουν και προβάλει το δελτίο καιρού. Καλεί τη μέθοδο που ανακτά τα δεδομένα καιρού. Η τοποθεσία που αφορά το δελτίο καιρού είναι εκ των προτέρων δηλωμένη κι εμφανίζεται σαν τίτλος στοιχείου *(βλέπε 2.1)*.
Όσο τα απομακρυσμένα δεδομένα ανακτώνται, εμφανίζει δείκτη φόρτωσης δεδομένων. Αν η ανάκτηση δεδομένων αποτύχει, εμφανίζει οπτικό μήνυμα σφάλματος Σε γενικές γραμμές, λειτουργεί ως διαχειριστής και καθορίζει το πλαίσιο υπό το οποίο όλα τα παιδιά στοιχεία προβάλλονται.

Κώδικας

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ForecastService } from './forecast.service';
import { ForecastCurrentComponent } from './forecast-current/forecast-
current.component';
import { ForecastNextComponent } from './forecast-next/forecast-
next.component';
import { ForecastHeaderComponent } from './forecast-header/forecast-
header.component';

/**
 * Γονέας στοιχείων forecast.
 *
 * Διαχειρίζεται υψηλού επιπέδου δεδομένα και λειτουργεί ως
 * σημείο για τον ορισμό υποστοιχείων.
 */
@Component({
  selector: 'wui-forecast',
  standalone: true,
  imports: [
    CommonModule,
    ForecastCurrentComponent,
    ForecastHeaderComponent,
    ForecastNextComponent
  ],
  template: `
    @if (loading) {
      <div class="d-flex w-100 p-5 justify-content-center">
        <div class="spinner-border" role="status"></div>
      </div>
    }
    @else {
      <ng-container *ngIf="!hasError; else errorMessage">
        <div class="row">
          <div class="col">
            <div class="card wui-card">
              <div class="card-body">
                <div class="p-3">
                  <div class="row">
                    <div class="col">
                      <wui-forecast-header></wui-forecast-header>
                    </div>
                  </div>
                  <div class="row mt-3">
                    <div class="col-12 col-lg-7">
```

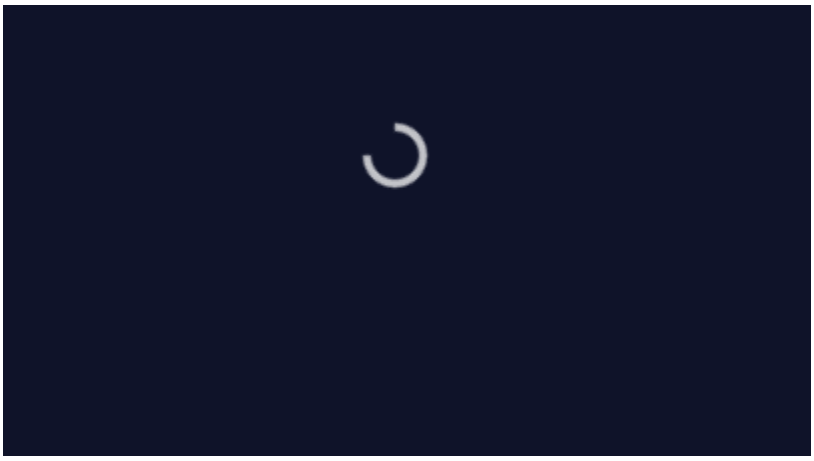
```
                    <wui-forecast-current></wui-forecast-current>
                  </div>
                  <div class="col-12 col-lg-5">
                    <wui-forecast-next></wui-forecast-next>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </ng-container>
    }

    <ng-template #errorMessage>
      <div class="row d-flex justify-content-center">
        <div class="col-6">
          <div class="alert alert-danger d-flex align-items-center"
role="alert">
            <div>
              The forecast could not be loaded.
            </div>
          </div>
        </div>
      </div>
    </ng-template>
  `
})
export class ForecastComponent {

  /**
   * Προεπιλεγμένη τοποθεσία.
   */
  private readonly DEFAULT_LOCATION = 'Lamia';

  /**
   * Υποδηλώνει σφάλγμα στην ανάκτηση δεδομένων.
   */
  hasError: boolean;

  /**
   * Δείκτης κατάστασης της ανάκτησης δεδομένων.
   * Παίρνει τιμή `true` όσο γίνεται η κλήση για να ενημερώσει
   * τον χρήστη με αντίστοιχο στοιχείο φόρτωσης.
   */
```

```
  loading: boolean;

  /**
   * @param service κύρια υπηρεσία ανάκτησης και διαχείρισης της πρόβλεψης
καιρού
   */
  constructor(private service: ForecastService) {}

  /**
   * Μέθοδος της Angular η οποία τρέχει κατά την εκκίνηση του component.
   */
  ngOnInit(): void {
    this.getForecast();
  }

  /**
   * Ανακτά την πρόβλεψη καιρού για ορισμένη τοποθεσία.
   */
  getForecast(): void {
    this.loading = true;

    this.service.search(this.DEFAULT_LOCATION)
      .subscribe({
        error: (err) => {
          console.error(err);

          this.loading = false;
          this.hasError = true;
        },
        complete: () => {
          this.loading = false;
        }
      })
  }

}
```

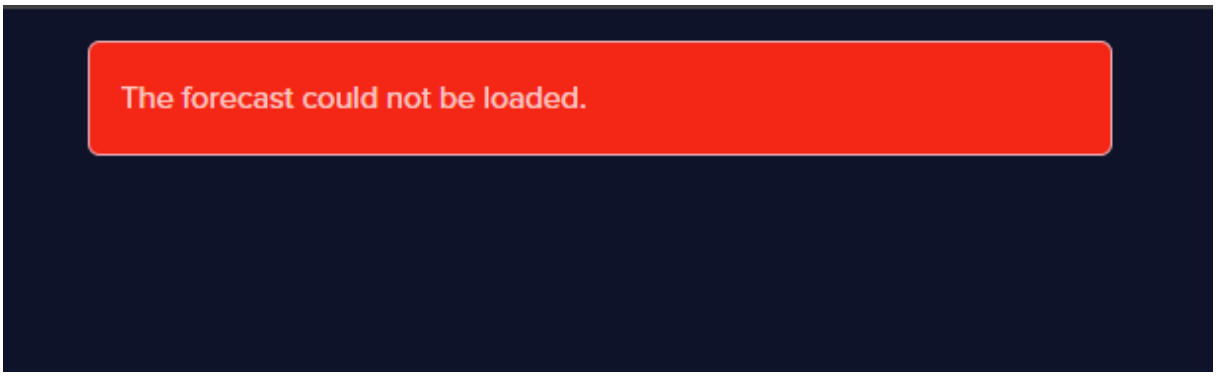
Πίνακας 10 Κώδικας Forecast Component

Αποτέλεσμα

Εικόνα 2.5-9 Forecast Component με δεδομένα



Εικόνα 2.5-10 Forecast Component κατά τη διάρκεια φόρτωσης δεδομένων



Εικόνα 2.5-11 Forecast Component με μήνυμα λάθους

2.5.3.3 Forecast Current Component

Τοποθεσία: features/forecast/forecast-current/forecast-current.component.ts

Όνομα Κλάσης: ForecastCurrentComponent

Σημείο Χρήσης: *forecast.component.html*

Dependencies: ForecastService

Συσχέτιση: Παιδί

Περιγραφή

Παρουσιάζει δεδομένα που έχουν να κάνουν με το τρέχον δελτίο καιρού. Προβάλλει περιγραφή συνθήκης καιρού και αντίστοιχο εικονίδιο. Επίσης, προβάλλει βασικές παραμέτρους καιρού όπως θερμοκρασία *(βλέπε 2.1.2)*

Κώδικας

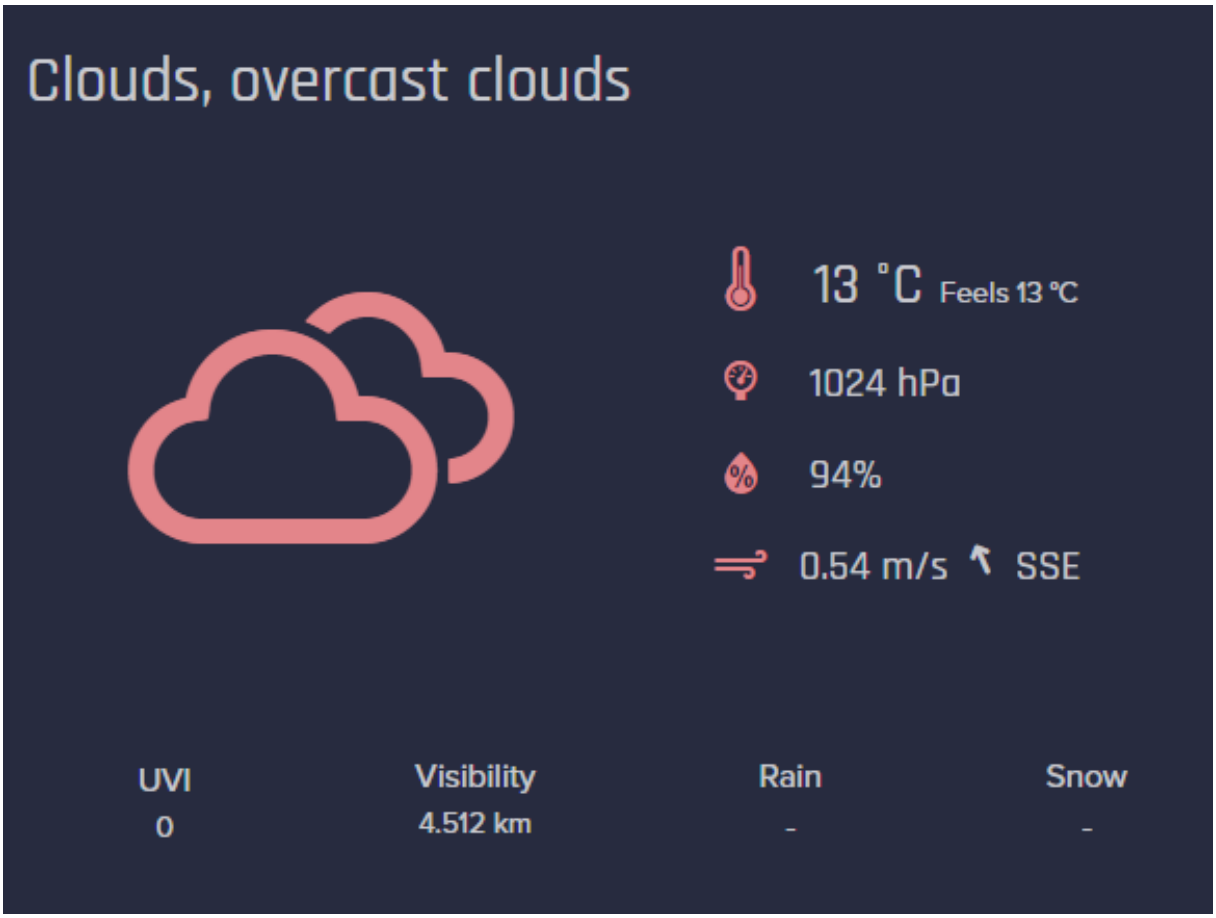
```
import { Component } from '@angular/core';
import { ForecastService } from '../forecast.service';
import { CommonModule } from '@angular/common';
import { WindIndicatorComponent } from '@shared/wind-indicator/wind-
indicator.component';
import { IconComponent } from '@shared/icon/icon.component';
import { API } from '@core/api/api';
import { ToFixedPipe } from '@shared/converters/toFixed/to-fixed.pipe';
import { map, Observable } from 'rxjs';


/**
 * Πρόβλεψη καιρού τώρα.
 *
 * Εμφανίζει την τοποθεσία και τις μετρήσεις που περιγράφουν
 * τις τρέχουσες καιρικές συνθήκες.
 */
@Component({
  selector: 'wui-forecast-current',
  standalone: true,
  imports: [
    CommonModule,
    IconComponent,
    ToFixedPipe,
    WindIndicatorComponent
  ],
  template: `
    <!-- Βασικές πληροφορίες -->
    <div class="row py-3">
      <div class="col">
        <h2 class="wui-text-low-priority">
          {{ (value$ | async).weather[0].main }}, {{ (value$ |
async).weather[0].description }}
        </h2>
      </div>
    </div>
    <div class="row py-3">
      <div class="col">
        <!-- Κύριο εικονίδιο καιρού -->
        <div class="p-3 d-flex justify-content-center">
          <wui-icon class="full-width" [value]="(value$ |
async).weather[0].icon" [size]="10"></wui-icon>
        </div>
        <!-- END Κύριο εικονίδιο καιρού -->
      </div>
```

```
      <div class="col">
        <div class="p-3">
          <div class="align-items-baseline d-inline-flex">
            <h3>
              <wui-icon [value]="'temperature'"></wui-icon> {{ (value$ |
async).temp  | toFixed }} °C
            </h3>
            <small class="ms-2">Feels {{ (value$ | async).feels_like |
toFixed }} °C</small>
          </div>
          <h5>
            <wui-icon [value]="'pressure'"></wui-icon> {{ (value$ |
async).pressure }} hPa
          </h5>
          <h5>
            <wui-icon [value]="'humidity'"></wui-icon> {{ (value$ |
async).humidity }}%
          </h5>
          <h5>
            <wui-icon [value]="'wind'"></wui-icon>
            <wui-wind-indicator
              [degree]="(value$ | async).wind_deg"
              [speed]="(value$ | async).wind_speed"
            ></wui-wind-indicator>
          </h5>
        </div>
      </div>
    </div>
    <!-- END Βασικές πληροφορίες -->

    <!-- Γενικές πληροφορίες -->
    <div class="row py-3">
      <div class="col my-3">
        <div class="d-flex justify-content-around w-100">
          <div class="d-flex flex-column justify-content-center text-
center">
            <label>UVI</label>
            <small>{{ (value$ | async).uvi }}</small>
          </div>
          <div class="d-flex flex-column text-center">
            <label>Visibility</label>
            <small>{{ (value$ | async).visibility / 1000 }} km</small>
          </div>
          <div class="d-flex flex-column justify-content-center text-
center">
```

```
            <label>Rain</label>
            @if ((value$ | async).rain) {
              <small>{{ (value$ | async).rain['1h'] }} mm</small>
            }
            @else {
              -
            }
          </div>
          <div class="d-flex flex-column text-center">
            <label>Snow</label>
            @if ((value$ | async).snow) {
              <small>{{ (value$ | async).snow['1h'] }} mm</small>
            }
            @else {
              -
            }
          </div>
        </div>
      </div>
    </div>
    <!-- END Γενικές πληροφορίες -->
  `
})
export class ForecastCurrentComponent {

  /**
   * Περιέχει τον τρέχοντα καιρό.
   */
  value$: Observable<API.OneCall.HourForecast> = this.forecast.event
    .pipe(map(({ weather }) => weather.current));

  /**
   * @param forecast υπηρεσία διαχείρισης συμβάντων καιρού
   */
  constructor(private forecast: ForecastService) {}

}
```

Πίνακας 11 Κώδικας Forecast Current Component

Αποτέλεσμα



Εικόνα 2.5-12 Forecast Current Component

2.5.3.4 Forecast Header Component

Τοποθεσία: features/forecast/forecast-header/forecast-header.component.ts

Όνομα Κλάσης: ForecastHeaderComponent

Σημείο Χρήσης: *forecast.component.html*

Dependencies: ForecastService

Συσχέτιση: Παιδί

Περιγραφή

Περιλαμβάνει γενικές πληροφορίες τοποθεσίας όπως όνομα τοποθεσίας ως τίτλο, πληροφορίες για να την ανατολή και δύση του ηλίου, και διεπαφή που εμφανίζει πληροφορίες για την ποιότητα του αέρα.

Κώδικας

```
import { Component } from '@angular/core';
import { map, Observable } from 'rxjs';
import { ForecastService } from '../forecast.service';
import { CommonModule } from '@angular/common';
import { IconComponent } from '@shared/icon/icon.component';
import { ToJSDatePipe } from '@shared/converters/toJSDate/to-jsdate.pipe';
import { API } from '@core/api/api';
import { AqiComponent } from '@shared/aqi/aqi.component';
import { NgbTooltip } from '@ng-bootstrap/ng-bootstrap';

@Component({
  selector: 'wui-forecast-header',
  standalone: true,
  imports: [
    CommonModule,
    IconComponent,
    ToJSDatePipe,
    AqiComponent,
    NgbTooltip
  ],
  template: `
    <div class="row">
      <div class="col">
        <div class="d-flex align-items-center justify-content-between">
          <div>
            <h1 class="mb-0">{{ locationTitle$ | async }}</h1>
          </div>
          <div class="d-flex flex-row align-items-center">
            <div class="d-flex px-3">
              <div class="d-flex align-items-center px-1"
ngbTooltip="Sunrise">
                <wui-icon [value]="'sunrise'" [size]="1.3"></wui-icon>
                <div>{{ (dayCycle$ | async).sunrise | toJSDate | date:
'HH:mm' }}</div>
              </div>
              <div class="d-flex align-items-center px-1"
ngbTooltip="Sunset">
                <wui-icon [value]="'sunset'" [size]="1.3"></wui-icon>
                <div>{{ (dayCycle$ | async).sunset | toJSDate | date:
'HH:mm' }}</div>
              </div>
            </div>
            <wui-aqi [value]="aqi$ | async"></wui-aqi>
          </div>
        </div>
```

```
      </div>
    </div>
  `
})
export class ForecastHeaderComponent {

  /**
   * Τιμή του component.
   * Περιέχει πρόβλεψη καιρού ανά 3 ώρες.
   */
  locationTitle$: Observable<string> = this.forecast.event
    .pipe(map(({ location }) => `${location.name}, ${location.country}`));

  /**
   * Περιέχει δεδομένα για την ανατολή και δύση του ηλίου.
   */
  dayCycle$: Observable<{ sunrise: number, sunset: number }> =
this.forecast.event
    .pipe(map(({ weather }) => ({
      sunrise: weather.current.sunrise, sunset: weather.current.sunset
    })));

  /**
   * Τιμή του Δείκτη Ποιότητας Αέρα.
   */
  aqi$: Observable<API.AirPollution.Metric> = this.forecast.event
    .pipe(map(({ airQuality }) => airQuality.current));

  /**
   * @param forecast υπηρεσία διαχείρισης συμβάντων καιρού
   */
  constructor(private forecast: ForecastService) {}

}
```

Πίνακας 12 Κώδικας Forecast Header Component

Αποτέλεσμα

Εικόνα 2.5-13 Forecast Header Component

2.5.3.5 Forecast Next Component

Τοποθεσία: features/forecast/forecast-next/forecast-next.component.ts

Όνομα Κλάσης: ForecastNextComponent

Σημείο Χρήσης: *forecast.component.html*

Dependencies: ForecastService

Συσχέτιση: Παιδί

Περιγραφή

Εμφανίζει λίστα δελτίων καιρού για τις επόμενες ώρες και ημέρες (*βλέπε* 2.1.3*)*. Σε κάθε στοιχείο λίστα διακρίνεται η ημέρα, η ώρα, η συνθήκη καιρού σε εικονίδιο και οι μέγιστη-ελάχιστη θερμοκρασίες. Η σημερινή ημέρα εμφανίζεται με ένδειξη “Today” και η αυριανή ως “Tomorrow” για τη δημιουργία φιλικής προς τον χρήστη πληροφορίας. Η πρώτη ώρα κάθε επόμενης ημέρας εμφανίζεται με διακριτικό χρώμα για καλύτερη σήμανση της πληροφορίας.

Κώδικας

```
import { Component } from '@angular/core';
import { ForecastService } from '../forecast.service';
import { API } from '@core/api/api';
import { CommonModule } from '@angular/common';
import { IconComponent } from '@shared/icon/icon.component';
import { ToJSDatePipe } from '@shared/converters/toJSDate/to-jsdate.pipe';
import { map, Observable } from 'rxjs';


@Component({
  selector: 'wui-forecast-next',
  standalone: true,
  imports: [
    CommonModule,
    IconComponent,
    ToJSDatePipe
  ],
  template: `
    <div class="px-2 py-3">
      <ul class="wui-forecast-list">
        @for (hour of value$ | async; track $index) {
          <li
            class="px-3 py-2 wui-forecast-next-item"
          >
            <div class="d-flex align-items-center py-2 justify-content-
between">
              <div class="px-2">
                <div class="wui-text-sm"
                [ngStyle]="hour._$isFirstHour
                  ? { 'color': 'var(--wui-color-secondary-light)' }
                  : undefined"
                >
                    {{ hour._$relativeDayName ?? (hour.dt | toJSDate | date:
'EEEE') }}
                </div>
                <div>{{ hour.dt | toJSDate | date: 'HH:mm' }}</div>
              </div>
              <div class="d-flex">
                <div class="px-3">
                  <wui-icon [size]="2" [value]="hour.weather[0].icon"></wui-
icon>
                </div>
                <div class="px-3">
                  <div title="Maximum Temperature">
                    {{ hour.main.temp_max.toFixed() }} °C
```

```
                </div>
                <div class="divider"></div>
                <div title="Minimum Temperature">
                {{ hour.main.temp_min.toFixed() }} °C
                </div>
              </div>
            </div>
          </div>
        </li>
      }
    </ul>
  </div>
`,
styles: `
  .wui-forecast-list
    white-space: nowrap
    overflow: auto
    height: 400px

    .wui-forecast-next-item
      list-style: none

    .wui-forecast-next-item + .wui-forecast-next-item
      border-top: 1px solid var(--wui-divider-color)
`
})
export class ForecastNextComponent {

  /**
   * Τιμή του component.
   * Περιέχει πρόβλεψη καιρού ανά 3 ώρες.
   */
  value$: Observable<API.HourlyForecast[]> = this.forecast.event
    .pipe(map(({ analysis }) =>
this.forecast.enrichHourForecast(analysis.list)));

  /**
   * @param forecast υπηρεσία διαχείρισης συμβάντων καιρού
   */
  constructor(private forecast: ForecastService) {}

}
```

Πίνακας 13 Κώδικας Forecast Next Component

Αποτέλεσμα

Εικόνα 2.5-14 Forecast Next Component για τις αμέσως επόμενες ημέρες

Εικόνα 2.5-15 Forecast Next Component για μελλοντικές ημέρες

2.5.4 Σύστημα Τύπων

Το σύστημα τύπων εξασφαλίζει πως τα δεδομένα έχουν την κατάλληλη μορφή. Έτσι μειώνονται τα σφάλματα που προέρχονται από λάθος τύπους δεδομένων.

2.5.4.1 API

Τοποθεσία: core/api/api.types.d.ts

Περιγραφή

Περιέχει τύπους για την περιγραφή των απομακρυσμένων δεδομένων και των μεταμορφώσεων δεδομένων που προκύπτουν από το `ApiService`.

```
import { SelectOption } from "../shared/components/navbar/navbar"
import { Coordinations, DayPhase, TemperatureRange } from "./metrics"

/**
 * Τυπολόγιο σχετικό με το Open Weather API.
 */
export namespace API {
```

```
interface HourlyForecast extends API.ForecastAnalysis.HourForecast {
  _$isFirstHour: boolean;
  _$isToday: boolean;
  _$relativeDayName: 'Today' | 'Tomorrow';
}

/**
 * Τυπολόγιο κλήσης Δείκτη Ποιότητας Αέρα
 */
namespace AirPollution {

  /**
   * Στοιχεία ρύπανσης αέρα.
   */
  type Components = {
    co: number;
    nh3: number;
    no: number;
    no2: number;
    o3: number;
    pm2_5: number;
    pm10: number;
    so2: number;
  }

  /**
   * Λεξικό μετρήσεων ρύπανσης ανά ημερομηνία.
   */
  type MetricsByDate = {
    [dt: number]: API.AirPollution.Metric
  }

  /**
   * Αντικείμενο μετρήσεων.
   */
  type Metric = {
    components: Components;
    dt: number;
    main: {
      aqi: number;
    }
  }

  /**
   * Απάντηση API.
```

```
     */
    type Response = {
      coord: Coordinations;
      list: Metric[];
    }

    /**
     * Ομαδοποιημένη απάντηση API μετά από μεταμόρφωση.
     */
    type ResponseGrouped = {
      current: Metric;
      forecast: MetricsByDate;
    }

}

/**
 * Τυπολόγιο κλήσης γενικής πρόβλεψης καιρού.
 */
namespace OneCall {

    /**
     * Ειδοποίηση έντονων καιρικών φαινομένων σε μορφή alert.
     */
    type Alert = {
      description: string;
      end: number;
      event: string;
      sender_name: string;
      start: number;
    }

    /**
     * Ωριαία πρόβλεψη ανά ώρα για τις επόμενες 48 ώρες.
     */
    type HourForecast = {
      clouds: number;
      dew_point: number;
      dt: number;
      feels_like: number;
      humidity: number;
      pressure: number;
      rain?: TimeRange;
      snow?: TimeRange;
      sunrise: number;
```

```
    sunset: number;
    temp: number;
    uvi: number;
    visibility: number;
    weather: Weather[];
    wind_deg: number;
    wind_speed: number;
  };

  /**
   * Επέκταση ωριαίας πρόβλεψης.
   */
  type HourlyForecastExtension = {
    airPollution: API.AirPollution.Metric;
  }

  /**
   * Απάντηση API.
   */
  type Response = {
    alerts: Alert[];
    current: HourForecast;
    daily: DailyForecast[];
    hourly: HourForecast[];
    lat: number;
    lon: number;
    timezone: string;
    timezone_offset: number;
  }

}

/**
 * Τυπολόγιο κλήσης πρόβλεψης 5 ημερών / 3 ώρες
 */
namespace ForecastAnalysis {

  /**
   * Ωριαία πρόβλεψη για τις επόμενες 5 ημέρες
   * ανά διάστημα 3 ωρών.
   */
  type HourForecast = {
    dt: number;
    main: WeatherMainMetrics,
    weather: Weather[],
```

```
      clouds: Clouds,
      wind: Wind,
      visibility: number;
      pop: number;
      sys: {
        pod: string;
      },
      dt_txt: string;
    }

    /**
     * Απάντηση API.
     */
    type Response = {
      cod: string;
      message: number;
      cnt: number;
      list: HourForecast[],
      city: City;
    }

  }

  /**
   * Κωδικοί εικονιδίου συνθήκης καιρού.
   */
  namespace ConditionIcons {
    type Day = '01d' | '02d' | '03d' | '04d' | '09d' | '10d' | '11d' | '13d'
| '50d';
    type Night = '01n' | '02n' | '03n' | '04n' | '09n' | '10n' | '11n' |
'13n' | '50n';
  }

  /**
   * Τοποθεσία πρόβλεψης καιρού.
   */
  type City = {
    id: number;
    name: string;
    coord: Coordinations;
    country: string;
    timezone: number;
    sunrise: number;
    sunset: number;
  }
```

```
/**
 * Αντικείμενο πληροφοριών σύννεφων.
 */
type Clouds = {
  all: number;
}

/**
 * Αντικείμενο θερμοκρασίας ημερήσιας πρόβλεψης.
 */
type DailyTemp = DailyTempFeelsLike & {
  [ K in TemperatureRange ]: number;
}

/**
 * Αντικείμενο αίσθησης θερμοκρασίας ημερήσιας πρόβλεψης.
 */
type DailyTempFeelsLike = {
  [ K in DayPhase ]: number;
}

/**
 * Διάνυσμα ωριαίας πρόβλεψης.
 */
type TimeRange = {
  '1h': number;
}

/**
 * Αντικείμενο των πιο βασικών στοιχείων καιρού.
 */
type Weather = {
  id: number
  main: string;
  description: string;
  icon: ConditionIcons.Day | ConditionIcons.Night;
}

/**
 * Επέκταση κλάσης Weather.
 */
type WeatherExtended = OneCall.Response & {
  airPollution: AirPollution.ResponseGrouped;
  forecastAnalysis: ForecastAnalysis.Response;
```

```
  }

  /**
   * Βασικές μετρήσεις πρόβλεψης καιρού.
   */
  type WeatherMainMetrics = {
    temp: number;
    feels_like: number;
    temp_min: number;
    temp_max: number;
    pressure: number;
    sea_level: number;
    grnd_level: number;
    humidity: number;
    temp_kf: number;
  }

  /**
   * Στοιχεία μετρήσεως ανέμου.
   */
  type Wind = {
    speed: number;
    deg: number;
  }



  /**
   * Τοποθεσία συσχετισμένη με δελτίο καιρού
   * σε εκτεταμένη μορφή select option.
   */
  type Location = {
    coord: Coordinations;
    country: string;
    id: number;
    name: string;
    _id: string;
  }

}
```

Πίνακας 14 Τυπολόγιο API

2.5.4.2 App

Τοποθεσία: app.types.d.ts

Περιγραφή

Τυπολόγιο γενικής χρήσης. Περιγράφει τους τύπους οντοτήτων σε επίπεδο εφαρμογής.

```ts
/**
 * Περιέχει τους τύπους που δηλώνει η εφαρμογή.
 */
namespace App {

  /**
   * Αντικείμενο που αποθηκεύεται στη μνήμη cache.
   */
  interface CacheDocument {
    /**
     * Ημερομηνία δημιουργίας αντικειμένου.
     */
    createdOn: Date;
    /**
     * Τιμή των αποθηκευμένω δεδομένων.
     */
    value: any;
  }

  /**
   * Στοιχεία ατμόσφαιρας.
   */
  export type AirComponent = 'co'
    | 'no'
    | 'no2'
    | 'o3'
    | 'so2'
    | 'nh3'
    | 'pm2_5'
    | 'pm10'

  /**
   * Τύπος χρωμάτων που υποδηλώνει την κατάσταση της ποιότητας αέρα.
   */
  export type AirQualitySeverityColor = '#4CAF50'
    | '#8BC34A'
    | '#FFEB3B'
    | '#FF9800'
    | '#F44336'

  /**
   * Τύπος λεκτικών που υποδηλώνει την κατάσταση της ποιότητας αέρα.
   */
  export type AirQualitySeverityRating = 'Good'
    | 'Fair'
```

```
    | 'Moderate'
    | 'Poor'
    | 'Very Poor'

  /**
   * Συμβάν πρόβλεψης καιρού.
   *
   * Αποστέλεται στους ακροατές του συμβάντος.
   */
  export type ForecastEvent = {
    location: API.Location;
    weather: API.OneCall.Response;
    analysis: API.ForecastAnalysis.Response;
    airQuality: API.AirPollution.ResponseGrouped;
  }

  /**
   * Γενικοί τύποι εικονιδίων καιρού.
   */
  export type GenericIcons = 'wind' | 'humidity' | 'temperature' |
'pressure' | 'sunrise' | 'sunset';

  /**
   * Κωδικοί εικονιδίων συστήματος
   */
  export type SystemIcon = API.ConditionIcons.Day | API.ConditionIcons.Night
| GenericIcons;

  /**
   * Συσχέτιση κωδικού εικονιδίου με την
   * αντίστοιχη κλάση βιβλιοθήκης.
   */
  export type SystemIconMap = {
    [K in SystemIcon]: string;
  }
}
```

Πίνακας 15 Τυπολόγιο App

2.5.5 Σταθερές

Οι σταθερές εξασφαλίζουν συνέπεια μέσω της επαναχρησιμοποίησης. Παράλληλα, ευνοούν την κεντρική διαχείριση του συστήματος.

2.5.5.1 App Constants

Τοποθεσία: app.constants.ts

Περιγραφή

Περιέχει γενικού τύπου σταθερές που χρησιμοποιούνται στο σύστημα.

```ts
import { HttpContextToken } from "@angular/common/http";

/**
 * Περιέχει τις σταθερές της εφαρμογής.
 */
export class AppConstants {

  /**
   * Tokens που χρησιμοποιούνται σε HttpRequests.
   */
  static HttpContextTokens = {
    /**
     * Ενεργοποιεί το μηχανισμό Cache για τις αιτήσεις που
     * το συμπεριλαμβάνουν σαν token με αληθή τιμή.
     */
    CACHE_RESPONSE: new HttpContextToken<boolean>(() => true)
  }

  /**
   * Λεξικό ατμοσφαιρικών στοιχείων.
   */
  static AirComponentsInfo: { [k in App.AirComponent]: string } = {
    co: 'Carbon monoxide (CO)',
    no: 'Nitrogen monoxide (NO)',
    no2: 'Nitrogen dioxide (NO2)',
    o3: 'Ozone (O3)',
    so2: 'Sulphur dioxide (SO2)',
    nh3: 'Ammonia (NH3)',
    pm2_5: 'Particulate matter (PM2.5)',
    pm10: 'Particulate matter (PM10)'
  };

  /**
   * Εύρος τιμών ποιότητας αέρα (Πράσινο -> Κόκκινο).
   *
   * Οι κάτω τιμές υποδεικνύουν κακή κατάσταση αέρα.
   */
  static AirQualitySeverityColorRange: App.AirQualitySeverityColor[] = [
    '#4CAF50',
    '#8BC34A',
    '#FFEB3B',
    '#FF9800',
    '#F44336'
  ];
```

```
  /**
   * Λεκτικός χαρακτηρισμός ποιότητας αέρα.
   */
  static AirQualitySeverityRatingRange: App.AirQualitySeverityRating[] = [
    'Good',
    'Fair',
    'Moderate',
    'Poor',
    'Very Poor'
  ];

  /**
   * Αντιστοίχιση κωδικού εικονιδίου του Open Weather με CSS κλάση
εικονιδίου καιρού.
   */
  static SystemIconMap: App.SystemIconMap = {
    '01d': 'wi wi-day-sunny',
    '02d': 'wi wi-day-cloudy',
    '03d': 'wi wi-cloud',
    '04d': 'wi wi-cloudy',
    '09d': 'wi wi-rain',
    '10d': 'wi wi-day-rain',
    '11d': 'wi wi-thunderstorm',
    '13d': 'wi wi-snow',
    '50d': 'wi wi-day-fog',
    '01n': 'wi wi-night-clear',
    '02n': 'wi wi-night-cloudy',
    '03n': 'wi wi-cloud',
    '04n': 'wi wi-cloudy',
    '09n': 'wi wi-rain',
    '10n': 'wi wi-night-rain',
    '11n': 'wi wi-thunderstorm',
    '13n': 'wi wi-snow',
    '50n': 'wi wi-night-fog',
    'wind': 'wi wi-strong-wind',
    'humidity': 'wi wi-humidity',
    'temperature': 'wi wi-thermometer',
    'pressure': 'wi wi-barometer',
    'sunrise': 'wi wi-sunrise',
    'sunset': 'wi wi-sunset',
  }

}
```

Πίνακας 16 Σταθερές App Constants

3 Επίλογος

Σε αυτή την εργασία παρουσίασα την ιστορία και την τρέχουσα κατάσταση των διεπαφών χρήστη. Έκανα μια επισκόπηση των κυριότερων βιβλιοθηκών διεπαφής χρήστη και ανέπτυξα μια εφαρμογή σε Angular που ενσωματώνει σύγχρονες τεχνικές και μεθόδους. Μέσω της ανάπτυξης της εφαρμογής προσέγγισα τεχνικές, μεθοδολογίες, και έννοιες της επιστήμης λογισμικού. Το σύστημα είναι πλήρως λειτουργικό και επεκτάσιμο. Αν ήθελα να προσθέσω κάτι για να επεκτείνω την εφαρμογή θα ανέπτυσσα κάποιο στοιχείο εισόδου κειμένου για αναζήτηση τοποθεσίας, ούτως ώστε ο χρήστης να έχει τη δυνατότητα να αναζητά την τοποθεσία που τον ενδιαφέρει. Από τεχνικής πλευράς, και σε επίπεδο κώδικα, θα εξέταζα το ενδεχόμενο δημιουργίας κλάσεων για την αντιστοίχιση των δεδομένων. Μέσω τεχνικής data mapping, θα μπορούσα ίσως να αποκομίσω οφέλη από ένα πιο καθαρά ορισμένο interface. Τέλος, θα ενσωμάτωνα συμπληρωματικά μια ανοιχτόχρωμη παλέτα χρωμάτων και θα έδινα στον χρήστη τη δυνατότητα εναλλαγής θέματος. Ως προεπιλογή επέλεξα τη σκουρόχρωμη παλέτα για να αποδώσω μια πιο μοντέρνα αισθητική αλλά η υποδομή είναι σε θέση να υποστηρίξει πολλαπλά θέματα. Αν ήθελα να κάνω την εφαρμογή πιο διασκεδαστική μπορεί να ενσωμάτωνα πολλαπλές δίχρωμες χρωματικές παλέτες, τόσο σκουρόχρωμες όσο και ανοιχτόχρωμες.

4 Βιβλιογραφία

[1] React (JavaScript library) - Wikipedia

[2] React is Declarative - What Does it Mean? | Alex Sidorenko

[3] ReactJS State vs Props - GeeksforGeeks

[4] React.js for Beginners — Props and State Explained (freecodecamp.org)

[5] React (JavaScript library) - Wikipedia

[6] Built-in React Hooks – React

[7] React Hooks (w3schools.com)

[8] Managing State – React

[9] Synchronizing with Effects – React

[10] useContext – React

[11] useRef – React

[12] Why is React considered a library and not a framework? - GeeksforGeeks

[13] ReactJS Virtual DOM - GeeksforGeeks

[14] Introducing JSX – React (reactjs.org)

[15] Angular (web framework) - Wikipedia

[16] Angular - What is Angular?

[17] Getting started with Angular - Learn web development | MDN (mozilla.org)

[18] Angular's Evolution: Embracing Change in the Web Development Landscape | newline

[19] What are decorators in Angular? - GeeksforGeeks

[20] Getting started with Angular - Learn web development | MDN (mozilla.org)

[21] Angular - What is Angular?

[22] Directives • Overview • Angular

[23] Angular - What is Angular?

[24] Angular - What is Angular?

[25] Angular - Introduction to services and dependency injection

[26] Angular - Understanding dependency injection

[27] Singleton pattern - Wikipedia

[28] Angular - Singleton services

[29] What is Single Page Application? - GeeksforGeeks

[30] Angular - Using Angular routes in a single-page application

[31] Angular Routing Tutorial with Example - TekTutorialsHub

[32] Reactive programming - Wikipedia

[33] Event-driven programming - Wikipedia

[34] Functional programming - Wikipedia

[35] Observer pattern - Wikipedia

[36] RxJS - Introduction

[37] Vue.js - Wikipedia

[38] Single-File Components | Vue.js

[39] Introduction | Vue.js

[40] Functional vs. Non Functional Requirements - GeeksforGeeks

[41] Bootstrap · The most popular HTML, CSS, and JS library in the world.

[42] Grid system · Bootstrap v5.3

[43] Angular powered Bootstrap

[44] Domain-driven design - Wikipedia

[45] HTTP Client • Overview • Angular