



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ  
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας	<b>Ανάπτυξη διαδικτυακής εφαρμογής ενημέρωσης κλιματικών και καιρικών φαινομένων με τη χρήση της τεχνολογίας Spring MVC</b>
Ονοματεπώνυμο Φοιτητή	<b>Κωνσταντίνα Ευαγγελία Μπινιάρη</b>
Αριθμός Μητρώου	<b>Π15181</b>
Επιβλέπων Καθηγητής	<b>Ευθύμιος Αλέπης</b>

Πειραιάς, Ιούλιος 2020

## ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία στοχεύει στην δημιουργία και παρουσίαση της διαδικτυακής εφαρμογής MyWeather, που αναπτύχθηκε, προκειμένου να αναδειχθεί η σημαντικότητα των διαδικτυακών εφαρμογών στην καθημερινότητα των χρηστών.

Στο πρώτο κομμάτι της εργασίας, θα εξηγηθούν κάποιες βασικές έννοιες και τεχνολογίες πάνω στις οποίες βασίστηκε η εφαρμογή μας.

Στο δεύτερο κομμάτι, θα παρουσιαστούν αναλυτικά όλες οι λειτουργίες της εφαρμογής μας, μέσω κειμένου και εικόνων

Στο τρίτο κομμάτι, θα αναλυθούν πρωτίστως όλες οι τεχνολογίες και τα εργαλεία που χρησιμοποιήσαμε για την ανάπτυξη της εφαρμογής, καθώς και η αρχιτεκτονική που ακολουθήσαμε, προκειμένου να δημιουργήσουμε το τελικό αποτέλεσμα της διαδικτυακής εφαρμογής.

Στο τέταρτο και τελευταίο κομμάτι, θα αναφερθούν τα συμπεράσματα που αποκομίστηκαν καθώς και οι πιθανές μελλοντικές επεκτάσεις που θα μπορούσαν να εφαρμοστούν.

Τέλος, αναφέρεται η βιβλιογραφία που χρησιμοποιήθηκε.

## ΠΕΡΙΕΧΟΜΕΝΑ

<b>ΠΕΡΙΛΗΨΗ</b>	<b>1</b>
<b>ΠΕΡΙΕΧΟΜΕΝΑ</b>	<b>2</b>
<b>ΕΙΣΑΓΩΓΗ</b>	<b>4</b>
<b>ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ</b>	<b>5</b>
Τι είναι η διαδικτυακή εφαρμογή	5
Πλεονεκτήματα διαδικτυακών εφαρμογών	5
<b>ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ</b>	<b>6</b>
OpenWeatherMap	6
<b>ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΧΡΗΣΗ ΕΦΑΡΜΟΓΗΣ (User's Manual)</b>	<b>7</b>
<b>Γενικές Λειτουργίες</b>	<b>8</b>
Πλοήγηση	8
Αρχική Σελίδα	8
Σελίδα About	10
Σελίδα Current Weather	10
Σελίδα Forecast Weather	12
Σελίδα Weather History	12
Σελίδα Air Quality	13
Σελίδα Weather News	16
<b>Λειτουργίες Εγγεγραμμένου Χρήστη</b>	<b>16</b>
Εγγραφή	16
Σύνδεση	17
Αρχική Σελίδα	19
Αγαπημένες πόλεις	21
Σελίδα Saved Cities	22
Σελίδα News	24
Σελίδα My Profile	25
<b>ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ</b>	<b>27</b>
<b>Μοντέλο αρχιτεκτονικής Model-View-Controller</b>	<b>27</b>
<b>Εργαλεία</b>	<b>29</b>
IntelliJ Idea	29
PgAdmin III	29
<b>Τεχνολογίες που χρησιμοποιήθηκαν</b>	<b>29</b>
Αρχιτεκτονική REST	29
Java	29
Spring Framework	30

---

Jpa	30
HTML / HTML5	31
Thymeleaf	31
CSS	31
Javascript	31
Ajax	32
Bootstrap	32
PostgreSQL	32
Apache Tomcat	33
Υπηρεσίες API	33
<b>Ανάλυση Κώδικα της εφαρμογής</b>	<b>33</b>
Λειτουργίες Σύνδεση / Εγγραφή χρήστη	34
Το προφίλ μου (My Profile)	36
Τρέχων καιρός (Current Weather)	38
Πενθήμερη πρόβλεψη καιρού / 3 ώρες (Forecast Weather)	41
Ιστορικό θερμοκρασιών ( History Weather)	42
Ποιότητα αέρα (Air Quality)	44
Tags	45
Κλιματικά νέα (Climate News)	46
Προσθήκη / Διαγραφή πόλης (Add City / Delete City)	48
Εμφάνιση αποτελεσμάτων στον χρήστη	50
<b>ΣΥΜΠΕΡΑΣΜΑΤΑ</b>	<b>51</b>
<b>Μελλοντικές Επεκτάσεις</b>	<b>52</b>
<b>ΒΙΒΛΙΟΓΡΑΦΙΑ</b>	<b>52</b>
<b>Ηλεκτρονική Βιβλιογραφία</b>	<b>52</b>

## ΕΙΣΑΓΩΓΗ

Το διαδίκτυο αποτελεί την κύρια πηγή ενημέρωσης των περισσότερων πλέον ανθρώπων. Καθημερινά χρησιμοποιείται μια πληθώρα διαδικτυακών εφαρμογών, που στοχεύουν στην άμεση και έγκυρη ενημέρωση του χρήστη.

Ο κύριος σκοπός της παρούσας εργασίας είναι η δημιουργία μιας διαδικτυακής εφαρμογής όπου ενημερώνει καθημερινά τον χρήστη για την πρόγνωση των καιρικών και κλιματολογικών φαινομένων. Η χρήση των διαδικτυακών εφαρμογών έχει γίνει ευρέως διαδεδομένη, λόγω της εύκολης και άμεσης προσβασιμότητας που προσφέρει στους χρήστες. Η ενημέρωση για τα καιρικά και κλιματολογικά φαινόμενα, είναι απαιτούμενη στην καθημερινότητα μας, κυρίως τα τελευταία έτη, όπου τα φαινόμενα της κλιματικής αλλαγής επηρεάζουν ολοένα περισσότερο τον πλανήτη. Επομένως, η δημιουργία διαδικτυακής εφαρμογής με σκοπό την ενημέρωση, αποτελεί μια καλή μέθοδο για την προσέλκυση περισσότερων χρηστών.

Η εφαρμογή που έχει δημιουργηθεί, είναι φιλική και εύκολα προσβάσιμη προς το χρήστη. Κάθε χρήστης μπορεί να κάνει εγγραφή, και να παρακολουθεί την πρόγνωση για τις πόλεις που έχει επιλέξει καθώς και να ενημερωθεί μέσω άρθρων, για διάφορα κλιματολογικά θέματα της επιλογής του. Επίσης, παρέχεται ενημέρωση, για όποια πόλη επιθυμεί, σχετικά με την ποιότητα του αέρα καθώς και για το ιστορικό των θερμοκρασιών.

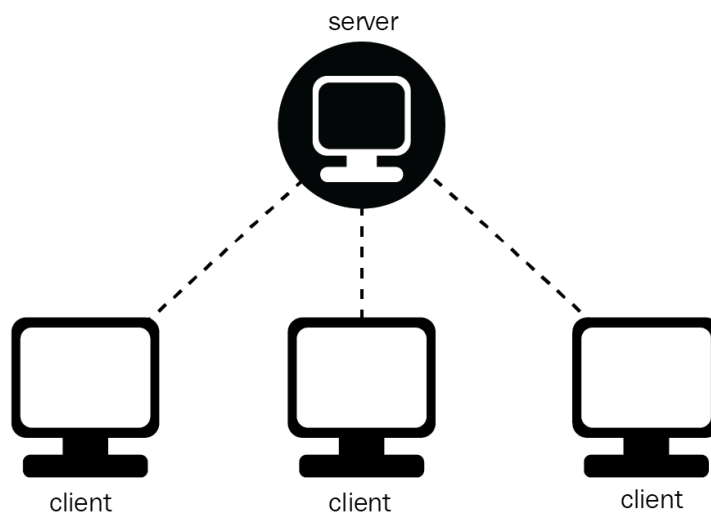
Η παρούσα εφαρμογή υλοποιεί τις παραπάνω δυνατότητες αντλώντας δεδομένα από έγκυρες πηγές, μέσω APIs τα οποία παρέχουν οι παραπάνω πηγές. Έχει γίνει χρήση πολλών διαφορετικών APIs, προκειμένου η διαδικτυακή εφαρμογή να παρέχει περισσότερες πληροφορίες στον χρήστη. Η ανάπτυξη της εφαρμογής έγινε με τη χρήση της τεχνολογίας MVC (Model - View - Controller) και του Java Spring Framework, όπου αποτελεί ένα δυνατό εργαλείο για την ανάπτυξη αντίστοιχων εφαρμογών.

## 1. ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ

### 1.1. Τι είναι η διαδικτυακή εφαρμογή

Η **διαδικτυακή εφαρμογή** (web application ή web app) είναι ένα πρόγραμμα το οποίο βρίσκεται σε έναν απομακρυσμένο διακομιστή (server) και είναι διαθέσιμη στους χρήστες της μέσω του διαδικτύου (Internet). Ο χρήστης το μόνο που χρειάζεται για να έχει πρόσβαση σε αυτή είναι ένα πρόγραμμα περιήγησης (browser) το οποίο χρησιμοποιείται ως client και η διεύθυνση της ιστοσελίδας (url).

Σε ένα περιβάλλον client - server, ο client, όπου στην περίπτωση μας είναι το πρόγραμμα περιήγησης, μπορεί να ζητά υπηρεσίες από έναν server, ο οποίος προσφέρει πληροφορίες.



Εικόνα 1.1. Αναπαράσταση περιβάλλοντος clients - server

Οι πρώτες εφαρμογές που χρησιμοποιήθηκαν αποτελούσαν τοπικές εφαρμογές όπου για την πρόσβαση σε αυτές, είναι απαραίτητη η εγκατάσταση λογισμικού στον υπολογιστή ή στο κινητό του εκάστοτε χρήστη. Με την πάροδο του χρόνου, άρχισαν να αναπτύσσονται ταχύτατα οι διαδικτυακές εφαρμογές, στις οποίες ο χρήστης μπορεί να έχει πρόσβαση από οπουδήποτε.

### 1.2. Πλεονεκτήματα διαδικτυακών εφαρμογών

- ❑ **Εύκολη προσβασιμότητα:** Οι χρήστες των διαδικτυακών εφαρμογών έχουν άμεση πρόσβαση στο περιεχόμενο της εφαρμογής. Χρειάζεται να υπάρχει πρόσβαση στο

διαδίκτυο και ένας περιηγητής, όπου υπάρχει προεγκατεστημένος σε οποιοδήποτε μέσο. Δεν γίνεται καμία εγκατάσταση λογισμικού.

- ❑ **Συμβατότητα:** Οι διαδικτυακές εφαρμογές είναι συμβατές σε όλα τα λειτουργικά συστήματα από τα οποία έχει πρόσβαση ο χρήστης, καθώς είναι προσβάσιμη μέσω του περιηγητή
- ❑ **Γρήγορη Αναβάθμιση:** Σε περίπτωση όπου είναι απαραίτητη η αναβάθμιση του λογισμικού, πραγματοποιείται στον εξυπηρετητή που φιλοξενεί την ιστοσελίδα και είναι διαθέσιμη άμεσα, σε αντίθεση με κάποια τοπική εφαρμογή, όπου θα έπρεπε να γίνει σε κάθε συσκευή.
- ❑ **Δεν καταλαμβάνουν χώρο:** Εφόσον ο χρήστης δεν αποθηκεύει την εφαρμογή σε κάποιο δικό του περιβάλλον, δεν καταλαμβάνεται αποθηκευτικός χώρος του για την πρόσβαση σε αυτή.

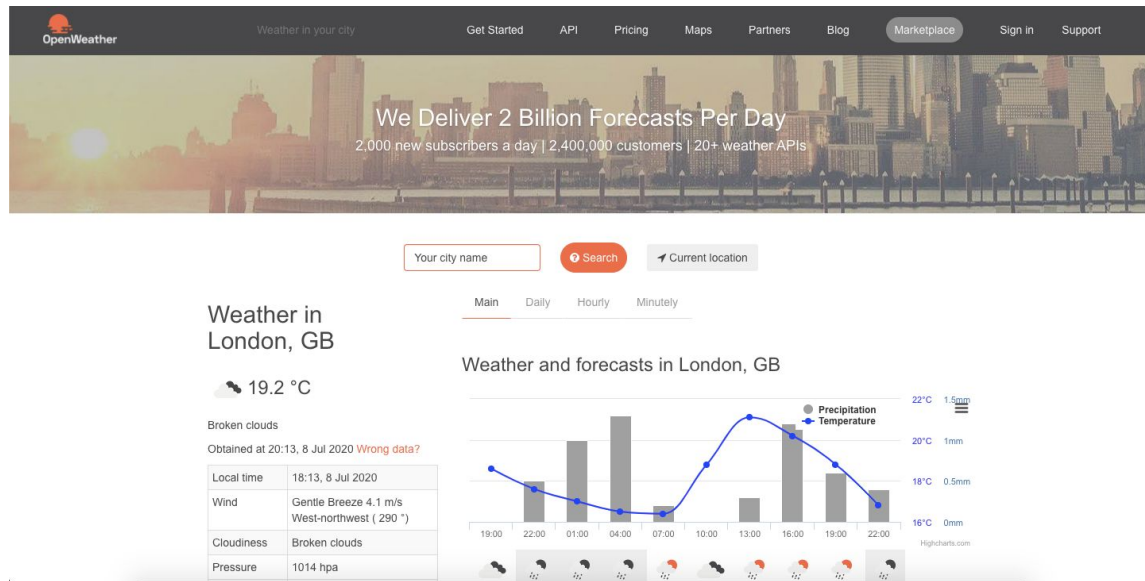
Οι περισσότερες διαδικτυακές εφαρμογές βασίζονται στην αρχιτεκτονική client - server, όπου ο client εισάγει πληροφορίες και ο server αποθηκεύει και ανακτά πληροφορίες.

## 2. ΑΝΑΣΚΟΠΗΣΗ ΠΕΔΙΟΥ

Στην ενότητα αυτή, θα παρουσιάσουμε αντίστοιχες διαδικτυακές εφαρμογές που έχουν αναπτυχθεί για την ενημέρωση των χρηστών σχετικά με την πρόγνωση του καιρού και των κλιματικών φαινομένων

### 2.1. OpenWeatherMap

Η συγκεκριμένη εφαρμογή αποτελεί μία από τις πιο γνωστές διαδικτυακές εφαρμογές για την πρόγνωση καιρού. Ο χρήστης έχει τη δυνατότητα να αναζητήσει τα καιρικά φαινόμενα σε όποια πόλη επιθυμεί ή στην περιοχή που βρίσκεται. Επίσης περιέχει χρήσιμα στατιστικά και γραφήματα καθώς και διαδραστικούς χάρτες οι οποίοι δείχνουν την βροχόπτωση, τα σύννεφα, την πίεση και τον άνεμο γύρω από την τοποθεσία που βρίσκεται.



Εικόνα 2.1. Η σελίδα OpenWeatherMap

### 3. ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΧΡΗΣΗ ΕΦΑΡΜΟΓΗΣ (User's Manual)

Στην ενότητα αυτή, θα γίνει παρουσίαση της διαδικτυακής εφαρμογής και θα αναλυθούν όλες οι λειτουργίες της, παρουσιάζοντας τις με συνδυασμό κειμένου και εικόνων.

Η εφαρμογή μας έχει ως κεντρικό θέμα την ενημέρωση για τον καιρό και τα κλιματολογικά νέα. Σε αυτή έχουν πρόσβαση όλοι οι χρήστες. Ωστόσο, υπάρχει η δυνατότητα εγγραφής, προκειμένου να έχει παραπάνω δυνατότητες. Οι γενικές λειτουργίες στις οποίες όλοι έχουν πρόσβαση είναι:

- ☐ Ημερήσια πρόγνωση καιρού
- ☐ Πενθήμερη πρόβλεψη καιρού
- ☐ 10ήμερο ιστορικό θερμοκρασιών ανά περιοχή
- ☐ Ρύπανση αέρα με βάση τον δείκτη ποιότητας αέρα (AQI)
- ☐ Προβολή άρθρων με λέξεις - κλειδιά με θέμα τα καιρικά φαινόμενα.

Οι χρήστες που κάνουν εγγραφή στην εφαρμογή έχουν τις εξής παραπάνω λειτουργίες:

- ☐ Εγγραφή / Σύνδεση
- ☐ Δημιουργία προφίλ στην εφαρμογή



- ❑ Επιλογή έως 4 λέξεις - κλειδιά (tags) με βάση τα οποία ο χρήστης θα παρακολουθεί τα κλιματολογικά νέα
- ❑ Αποθήκευση οποιαδήποτε πόλης στις “Αγαπημένες πόλεις” (My Cities)
- ❑ Προβολή των πληροφοριών που παρέχονται σε όλες τις αποθηκευμένες πόλεις

### 3.1. Γενικές Λειτουργίες

#### 3.1.1. Πλοήγηση

Η πλοήγηση για την ιστοσελίδα γίνεται από το μενού, το οποίο περιέχει τις εξής σελίδες

- ❑ Αρχική Σελίδα (Home)
- ❑ Σχετικά με εμάς (About page)
- ❑ Καιρός (Weather)
  - ❑ Τρέχων καιρός (Current Weather)
  - ❑ Πρόβλεψη καιρού (Forecast Weather)
  - ❑ Ιστορικό καιρού (Weather history)
  - ❑ Ρύπανση αέρα (Air Quality)
- ❑ Νέα (News)
- ❑ Σύνδεση / Εγγραφή (Log in / Register)



Εικόνα 3.1. Μενού πλοήγησης σελίδας

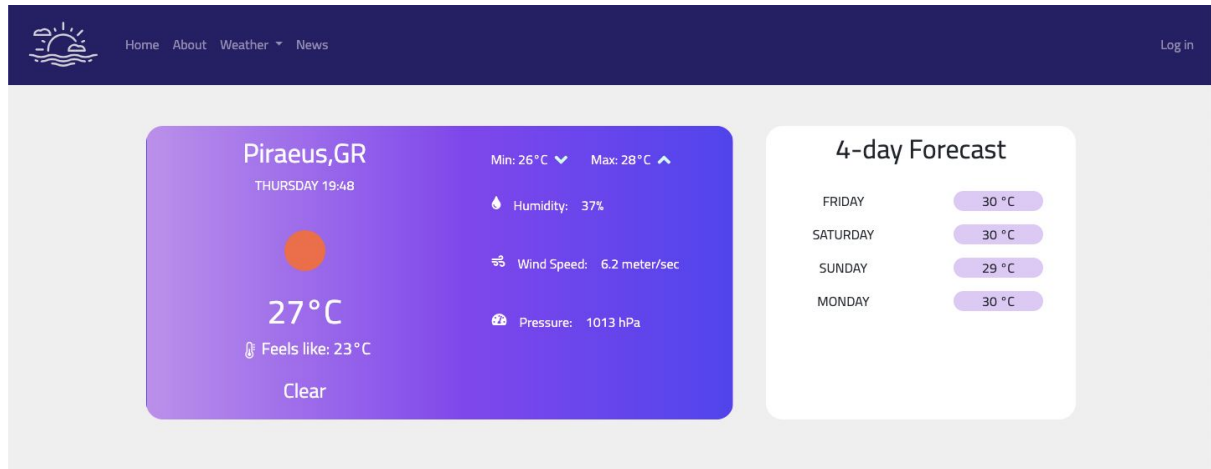
#### 3.1.2. Αρχική Σελίδα

Η αρχική σελίδα που βλέπει ο χρήστης κατά την εισαγωγή του στην εφαρμογή περιέχει μια καρτέλα με πληροφορίες του τρέχοντα καιρού για την τοποθεσία στην οποία βρίσκεται ο χρήστης. Σε αυτή την καρτέλα αναφέρονται οι εξής πληροφορίες:

- ❑ Όνομα πόλης
- ❑ Ημερομηνία / ώρα
- ❑ Θερμοκρασία
- ❑ Περιγραφή καιρού
- ❑ Μέγιστη / ελάχιστη θερμοκρασία εντός της ημέρας
- ❑ Υγρασία

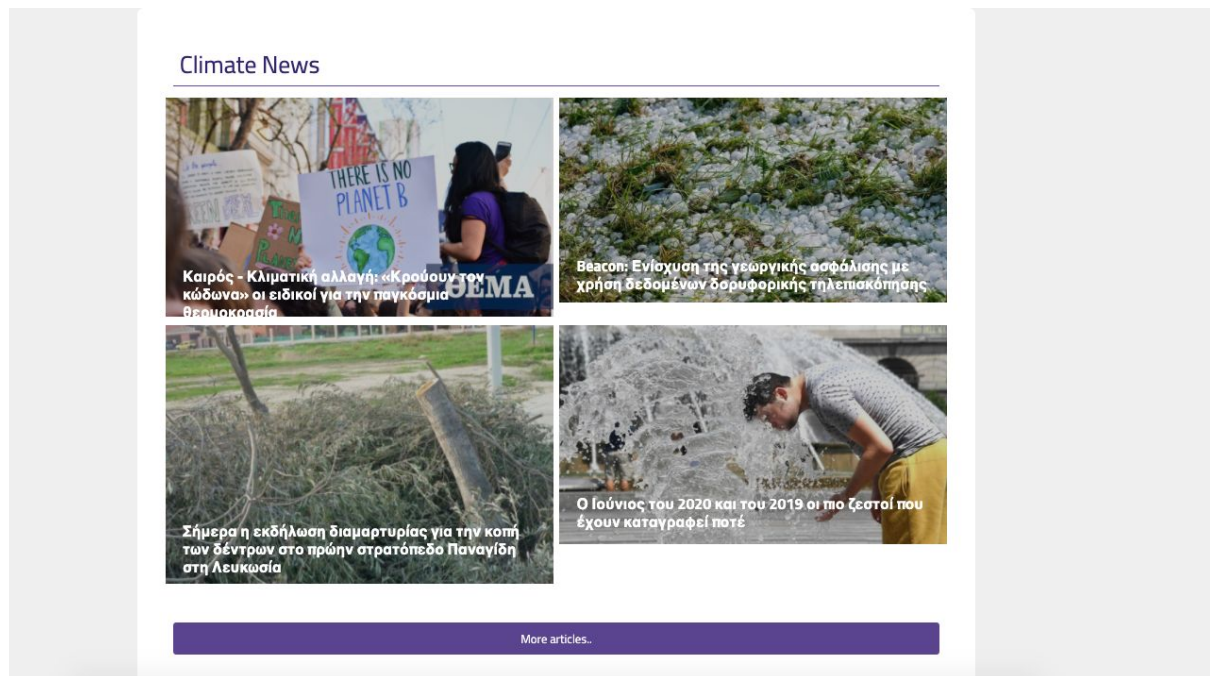
- ❑ Ένταση ανέμου
- ❑ Ατμοσφαιρική Πίεση

Επίσης, εμφανίζεται και μία καρτέλα με την 4ήμερη πρόγνωση καιρού για την τρέχουσα τοποθεσία



Εικόνα 3.2. Καρτέλα ενημέρωσης καιρού

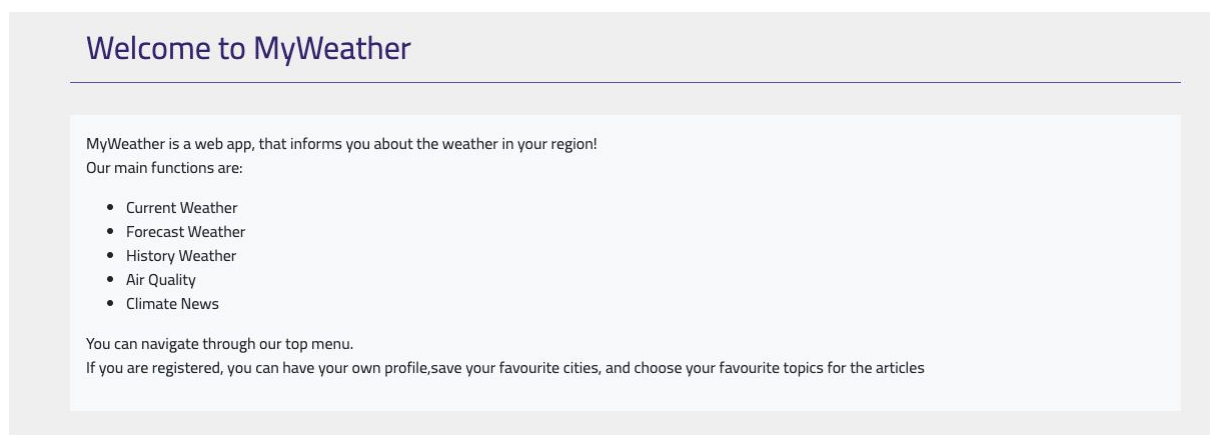
Στη συνέχεια, εμφανίζονται κάποια άρθρα τα οποία περιέχουν σαν λέξη - κλειδί την κλιματική αλλαγή, και ένα κουμπί με το οποίο ο χρήστης μπορεί να ανακατευθυνθεί στην σελίδα "News" με περισσότερα άρθρα. Τα άρθρα περιέχουν, ένα κουμπί "Read More", όπου παραπέμπουν τον χρήστη στην σελίδα του άρθρου.



Εικόνα 3.3. Τα νέα στην αρχική σελίδα

### 3.1.3. Σελίδα About

Πρόκειται για μία απλή σελίδα, όπου ενημερώνει τον χρήστη για τις λειτουργίες της εφαρμογής.



Εικόνα 3.4. Τα νέα στην αρχική σελίδα

### 3.1.4. Σελίδα Current Weather

Στην συγκεκριμένη σελίδα, ο χρήστης έχει τη δυνατότητα αναζήτησης του τρέχοντα καιρού, είτε συμπληρώνοντας την πόλη και τη χώρα για την οποία θέλει να βρει πληροφορίες, είτε κάνοντας κλικ στο “Enable Location”, όπου ενεργοποιεί μια υπηρεσία η οποία εντοπίζει τις συντεταγμένες Latitude , Longitude και εμφανίζει πληροφορίες για την τρέχουσα τοποθεσία.

Current Weather

You can check the current weather in any city, by filling the name and the country of the city. Also, you can save it in your favourite cities

Your City Name Your Country Name

Search OR Enable Location

Εικόνα 3.5. Φόρμα αναζήτησης καιρού

Κάνοντας κλικ στο κουμπί Search, εμφανίζονται τα αποτελέσματα της αναζήτησης, όπου είναι οι καρτέλες που αναφέραμε και στην αρχική σελίδα. Οι πληροφορίες που περιέχουν είναι:

- ❑ Όνομα πόλης
- ❑ Ημερομηνία / ώρα
- ❑ Θερμοκρασία
- ❑ Περιγραφή καιρού
- ❑ Μέγιστη / ελάχιστη θερμοκρασία εντός της ημέρας
- ❑ Υγρασία
- ❑ Ένταση ανέμου
- ❑ Ατμοσφαιρική Πίεση
- ❑ 4ήμερη πρόγνωση καιρού

Athens, GR

THURSDAY 20:07

Min: 25 °C Max: 28 °C

Humidity: 36%

Wind Speed: 6.7 meter/sec

Pressure: 1013 hPa

26 °C

Feels like: 22 °C

Clear

4-day Forecast

FRIDAY 30 °C

SATURDAY 30 °C

SUNDAY 29 °C









MONDAY 29 °C

Εικόνα 3.6 Καρτέλα ενημέρωσης τρέχοντα καιρού

### 3.1.5. Σελίδα Forecast Weather

Σε αυτή την σελίδα, εμφανίζονται πληροφορίες πενθήμερης πρόγνωσης καιρού ανά 3 ώρες. Όπως αναφέραμε και παραπάνω, ο χρήστης έχει τη δυνατότητα αναζήτησης είτε συμπληρώνοντας την πόλη και τη χώρα που επιθυμεί, είτε ενεργοποιώντας την υπηρεσία εύρεσης τοποθεσίας. Συγκεκριμένα βλέπουμε στην οθόνη τις εξής πληροφορίες:

- ☐ Ημερομηνία / ώρα
- ☐ Θερμοκρασία
- ☐ Υγρασία
- ☐ Συννεφιά
- ☐ Ένταση ανέμου
- ☐ Ατμοσφαιρική Πίεση

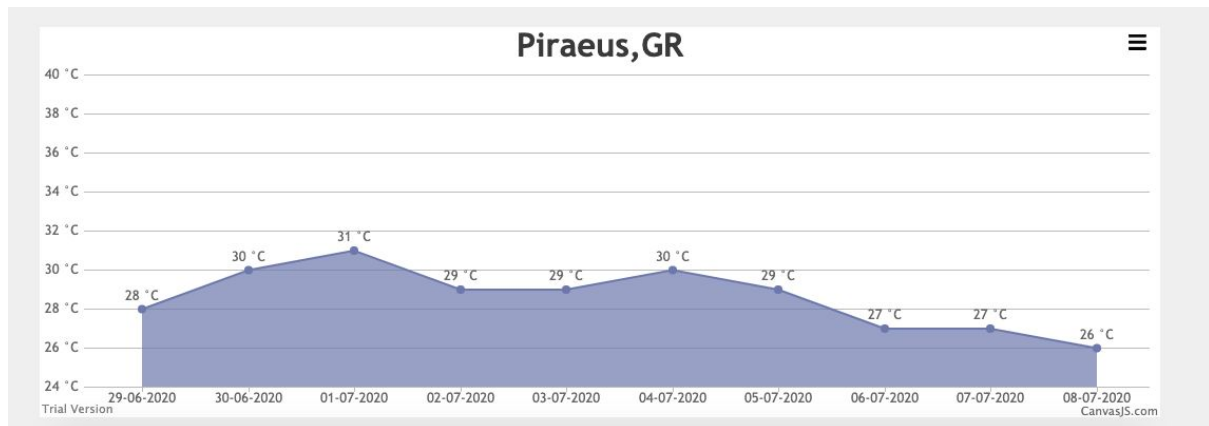
10-07-2020						
Hour	Weather	Feels Like	Humidity	Cloudiness	Wind Speed	Atmospheric Pressure
00:00	23°C 	20°C	62%	0%	6.33 meter/sec	1014 hPa
03:00	22°C 	19°C	61%	0%	6.73 meter/sec	1014 hPa
06:00	25°C 	20°C	47%	0%	8.57 meter/sec	1014 hPa
09:00	28°C 	22°C	40%	0%	9.25 meter/sec	1015 hPa
12:00	28°C 	23°C	39%	0%	9.45 meter/sec	1014 hPa
15:00	28°C 	23°C	42%	0%	8.52 meter/sec	1013 hPa
18:00	26°C 	23°C	48%	0%	5.43 meter/sec	1013 hPa
21:00	24°C 	22°C	54%	0%	4.32 meter/sec	1014 hPa

Εικόνα 3.7 Αποτελέσματα ενημέρωσης 5ήμερης πρόβλεψης καιρού

### 3.1.6. Σελίδα Weather History

Η συγκεκριμένη σελίδα περιέχει πληροφορίες για τις προηγούμενες 10 ημέρες σχετικά με τις θερμοκρασίες της περιοχής. Η αναζήτηση της πόλης γίνεται με τον ίδιο τρόπο που περιγράψαμε στις δύο παραπάνω λειτουργίες. Το αποτέλεσμα της αναζήτησης είναι ένα γράφημα το οποίο

δείχνει την κατανομή των θερμοκρασιών της περιοχής που αναζητήσαμε, για τις προηγούμενες 10 ημέρες, με μέγιστη θερμοκρασία αυτή των 40°C.



Εικόνα 3.8 Αναπαράσταση του ιστορικού θερμοκρασιών σε γράφημα

### 3.1.7. Σελίδα Air Quality

Αυτή η σελίδα ενημερώνει τους χρήστες, για την ατμοσφαιρική ρύπανση του αέρα και τα επίπεδα στα οποία βρίσκεται. Η μέτρηση της ατμοσφαιρικής πίεσης γίνεται σύμφωνα με τον παγκόσμιο δείκτη ποιότητας αέρα **AQI**(Air Quality Index). Η κλίμακα ποιότητας αέρα χωρίζεται στις παρακάτω 5 κατηγορίες:

- ☐ **Good - Καλός ( 0 - 50 )** : Η ποιότητα του αέρα θεωρείται ικανοποιητική και η ατμοσφαιρική ρύπανση παρουσιάζει μικρό ή καθόλου κίνδυνο
- ☐ **Moderate - Μέτριος( 51 - 100 )** : Η ποιότητα του αέρα είναι αποδεκτή. Ωστόσο, για ορισμένους ρύπους μπορεί να υπάρχει μέτρια ανησυχία για την υγεία για ένα πολύ μικρό αριθμό ατόμων που είναι ασυνήθιστα ευαίσθητα στην ατμοσφαιρική ρύπανση.
- ☐ **Unhealthy for sensitive groups - Ανθυγιεινό για ευαίσθητες ομάδες ( 101 - 150 )**: Τα μέλη ευαίσθητων ομάδων ενδέχεται να έχουν επιπτώσεις στην υγεία. Το ευρύ κοινό δεν είναι πιθανό να επηρεαστεί

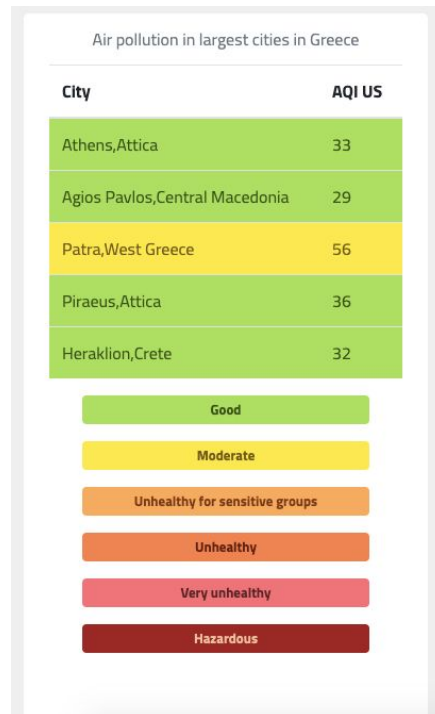
- ❑ **Unhealthy - Ανθυγιεινό ( 151 - 200 )** : Ο καθένας μπορεί να αρχίσει να αντιμετωπίζει τις επιπτώσεις στην υγεία. τα μέλη ευαίσθητων ομάδων ενδέχεται να έχουν πιο σοβαρές επιπτώσεις στην υγεία
- ❑ **Very Unhealthy - Πολύ ανθυγιεινό ( 201 - 300 )** : Προειδοποιήσεις για την υγεία σε συνθήκες έκτακτης ανάγκης. Όλος ο πληθυσμός είναι πολύ πιθανόν να επηρεαστεί.
- ❑ **Hazardous - Επικίνδυνος ( 301 + )** : Προειδοποίηση για την υγεία: Όλοι μπορεί να έχουν πιο σοβαρές επιπτώσεις στην υγεία

Επίσης, τα χρώματα τα οποία εμφανίζονται στο πλαίσιο της πόλης, αναφέρονται στο επίπεδο ρύπανσης και είναι κατανεμημένα ως εξής:

Επίπεδο 0 - 50
Επίπεδο 51 - 100
Επίπεδο 101 - 150
Επίπεδο 151 - 200
Επίπεδο 201 - 300
Επίπεδο 301 +

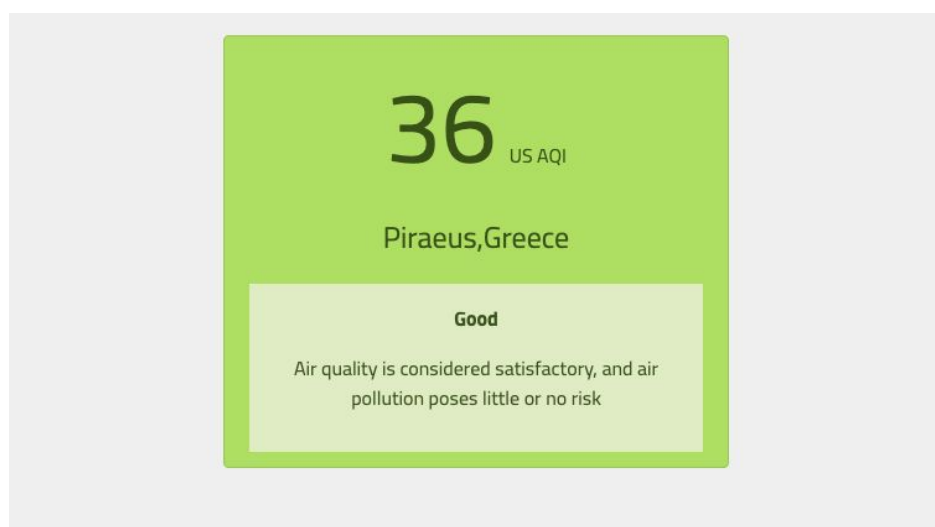
**Πίνακας 1. Χρωματική απεικόνιση των επιπέδων ρύπανσης του αέρα**

Στην σελίδα εμφανίζεται ένας πίνακας με τα ποσοστά ρύπανσης στις δέκα μεγαλύτερες πόλεις της Ελλάδας.



Εικόνα 3.9 Εμφάνιση επιπέδου ρύπανσης του αέρα στις 10 μεγαλύτερες πόλεις της Ελλάδας, και επεξήγηση των χρωματικών απεικονίσεων

Ο χρήστης αναζητά την πόλη που επιθυμεί με τον ίδιο τρόπο που γίνεται στις προηγούμενες λειτουργίες. Το αποτέλεσμα της αναζήτησης αυτής, είναι το ποσοστό ρύπανσης του αέρα της περιοχής αυτής με μια μικρή περιγραφή.

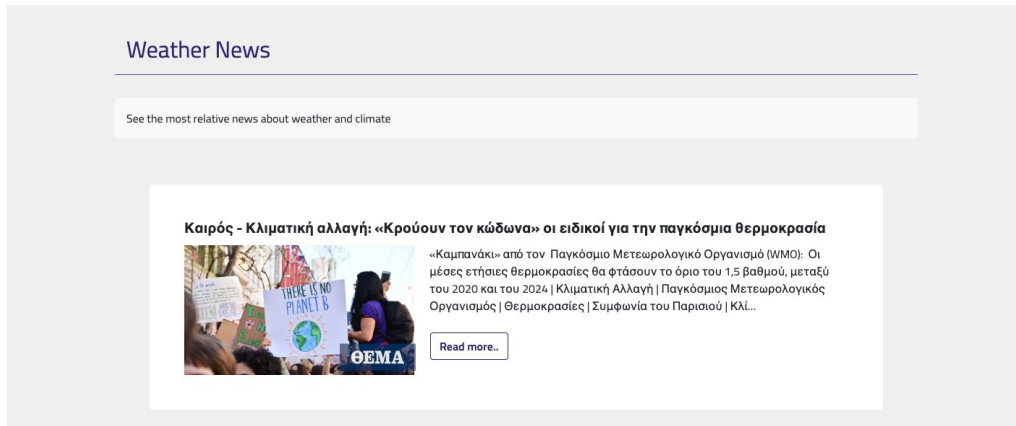


Εικόνα 3.10 Αποτέλεσμα αναζήτησης ρύπανσης του αέρα

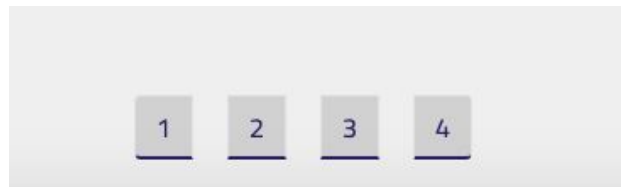


### 3.1.8. Σελίδα Weather News

Σε αυτή την σελίδα, περιέχονται άρθρα με θέμα τα κλιματικά φαινόμενα. Έχουμε τοποθετήσει την λέξη-κλειδί **κλιματική** στην υπηρεσία που μας παρέχει τα άρθρα, προκειμένου να φιλτράρονται τα αποτελέσματα. Υπάρχει αρίθμηση σελίδων στην περίπτωση που τα άρθρα είναι άνω των πέντε.



Εικόνα 3.11 Εμφάνιση άρθρου στην σελίδα News



Εικόνα 3.12. Αλλαγή σελίδας

## 3.2. Λειτουργίες Εγγεγραμμένου Χρήστη

### 3.2.1. Εγγραφή

Η εφαρμογή μας, προσφέρει στον χρήστη τη δυνατότητα να κάνει εγγραφή στην ιστοσελίδα και να μπορεί να συνδεθεί με τους κωδικούς του. Έχουμε δημιουργήσει μια φόρμα εγγραφής, όπου ο χρήστης είναι απαραίτητο να συμπληρώσει τα παρακάτω στοιχεία:

- ☐ Όνομα
- ☐ Επώνυμο
- ☐ Email
- ☐ Password

Επίσης, καλείται να επιλέξει τα tags, με τα οποία θα κατηγοριοποιούνται τα άρθρα του.

Μπορεί να επιλέξει μέχρι τέσσερα tags.

**Register**

Name

Last Name

Email

Password

Please choose your favourite tags

#κλιματική αλλαγή #καιρός #περιβάλλον #δάση

#ατμοσφαιρική #αέρας #πυρκαγιά #ποιότητα αέρα

#καιρικά φαινόμενα #πρόγνωση καιρού #θερμοκρασία

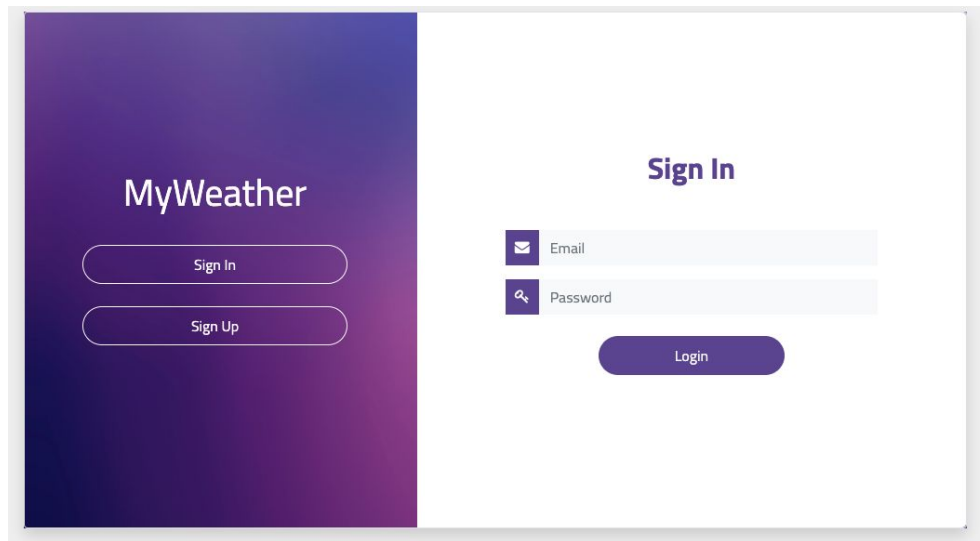
#καύσωνας #παγετώνας #υπερθέρμανση

Sign Up

Εικόνα 3.13 Οθόνη εγγραφής του χρήστη

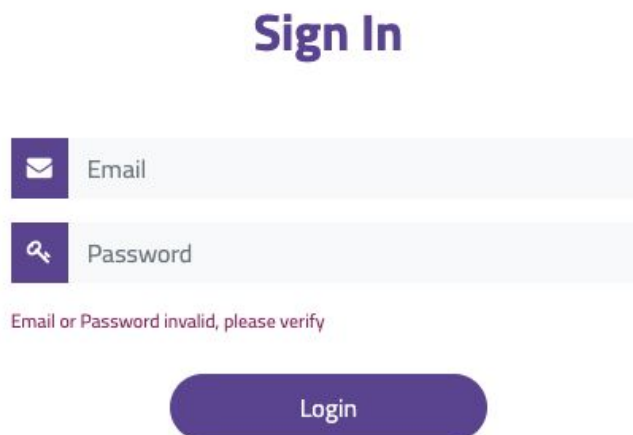
### 3.2.2. Σύνδεση

Για τη σύνδεση του χρήστη, είναι απαραίτητα μόνο το email & password, τα οποία αποθηκεύτηκαν στην βάση δεδομένων κατά την εγγραφή.



Εικόνα 3.14 Φόρμα σύνδεσης

Σε περίπτωση λάθους, εμφανίζεται το ανάλογο μήνυμα στην οθόνη



Εικόνα 3.15 Μήνυμα σφάλματος σύνδεσης

Όταν ο χρήστης συνδέεται, η εφαρμογή τον παραπέμπει στην αρχική σελίδα, όπου έχει κάποιες διαφοροποιήσεις, σε σχέση με αυτή που εμφανίζεται σε όλους τους χρήστες.

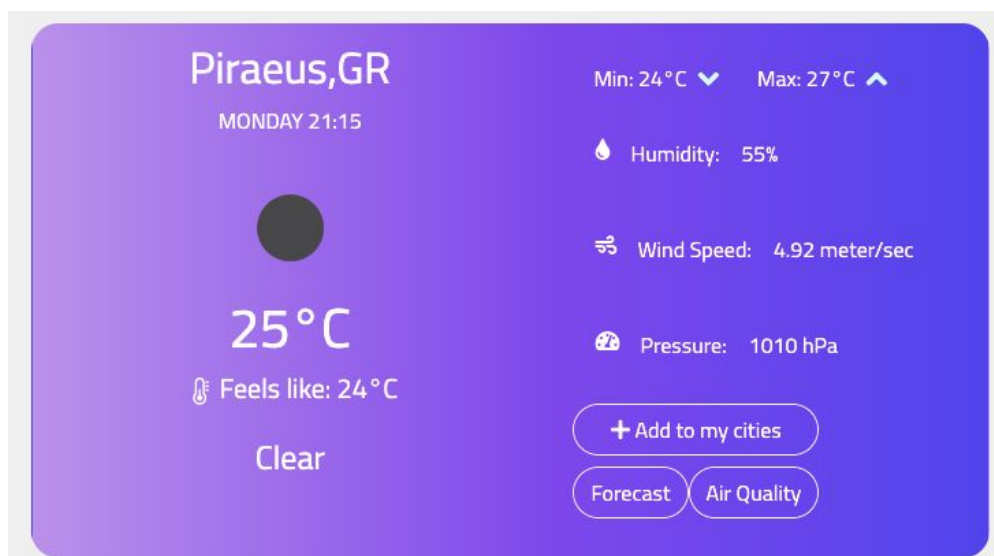
Αρχικά το μενού πλοήγησης περιέχει τις 2 επιπλέον λειτουργίες:

- ❑ Αποθηκευμένες πόλεις ( Saved Cities)
- ❑ Το προφίλ μου (My Profile)

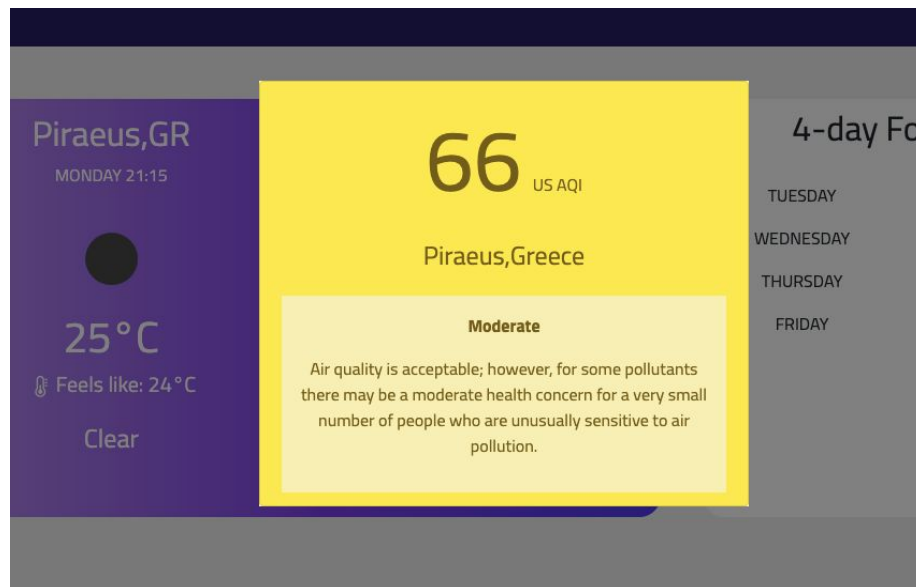
Οι λειτουργίες θα αναλυθούν στη συνέχεια. Επίσης, εφόσον ο χρήστης έχει συνδεθεί, έχει την επιλογή Αποσύνδεση (Logout)

### 3.2.3. Αρχική Σελίδα

Όπως αναφέραμε και παραπάνω, στην αρχική σελίδα εμφανίζεται μια καρτέλα ενημέρωσης σχετικά με τον τρέχοντα καιρό στην ακριβή τοποθεσία που βρισκόμαστε. Ωστόσο, όταν ο χρήστης έχει συνδεθεί, μπορεί να αποθηκεύσει την πόλη στις “Αγαπημένες πόλεις” (Saved Cities), όπου θα αναλυθεί παρακάτω. Επίσης, μπορεί να δει άμεσα τις πληροφορίες για την πενθήμερη πρόβλεψη καιρού, καθώς και για την ρύπανση του αέρα, για την τρέχουσα τοποθεσία. Οι πληροφορίες αυτές, εμφανίζονται σε modal box.

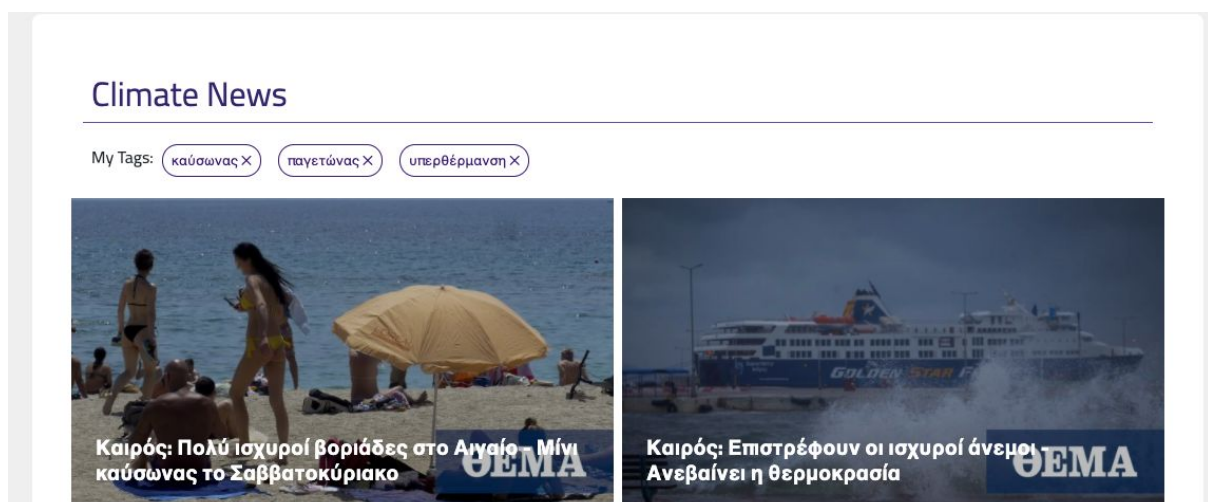


Εικόνα 3.16 Εμφάνιση καρτέλας σε συνδεδεμένο χρήστη



Εικόνα 3.17 Εμφάνιση πληροφοριών σε modal box

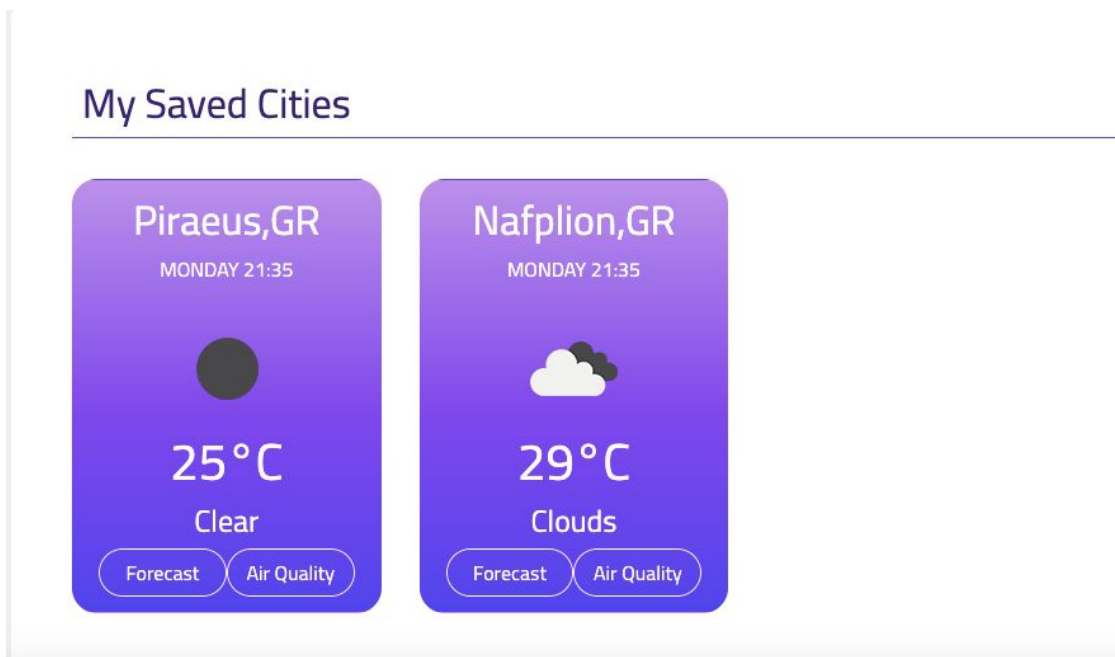
Συνεχίζοντας στην αρχική σελίδα, ένας συνδεδεμένος χρήστης βλέπει άρθρα με βάση τα tags τα οποία έχει επιλέξει κατά την εγγραφή του. Επομένως, μπορεί να επιλέξει ένα από τα αποθηκευμένα του tags, και να δει τα αντίστοιχα άρθρα.



Εικόνα 3.18 Εμφάνιση άρθρων και tags σε συνδεδεμένο χρήστη

### 3.2.4. Αγαπημένες πόλεις

Ο χρήστης έχει τη δυνατότητα να αποθηκεύει πόλεις, και να ανατρέχει στην σελίδα "Saved cities", προκειμένου να ενημερώνεται για τον καιρό σε αυτές. Στην αρχική σελίδα, εμφανίζεται η ενότητα "My Saved Cities", όπου ο χρήστης βλέπει τις πόλεις που έχει αποθηκεύσει. Επίσης, υπάρχει ένα κουμπί "Add City", το οποίο παραπέμπει τον χρήστη στην αναζήτηση πόλης ώστε να αποθηκεύσει αυτή που επιθυμεί, καθώς και "Edit my saved cities", το οποίο κατευθύνει τον χρήστη στην επιλογή αφαίρεσης πόλεων από την λίστα του.



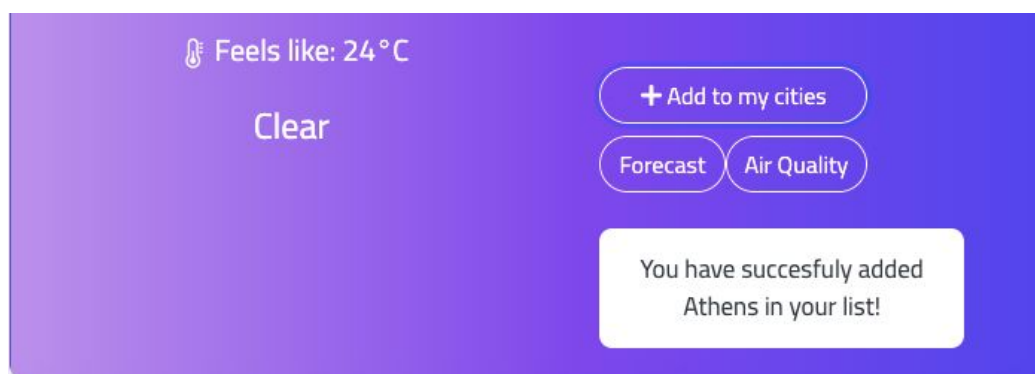
Εικόνα 3.19 Εμφάνιση αποθηκευμένων πόλεων χρήστη



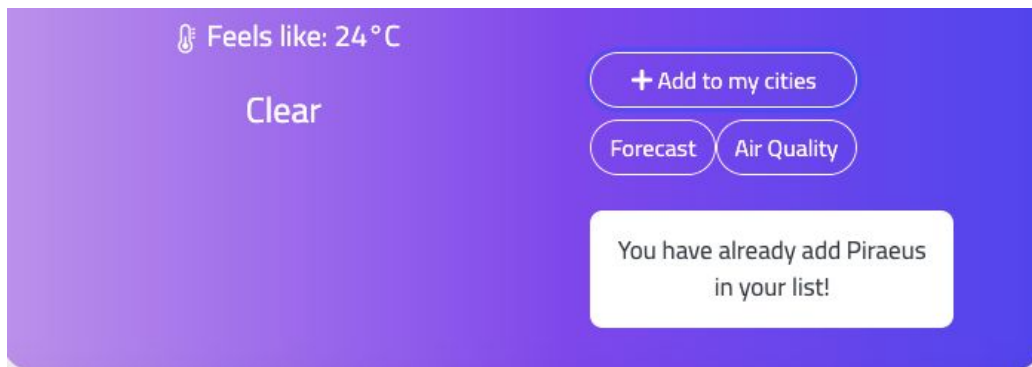
Εικόνα 3.20 Επιλογή προσθήκης νέας πόλης, και επεξεργασίας των αποθηκευμένων πόλεων

### 3.2.5. Σελίδα Saved Cities

Στην σελίδα saved cities (αποθηκευμένες πόλεις), ο χρήστης μπορεί να δει την λίστα με τις πόλεις που έχει αποθηκεύσει. Για να αποθηκεύσει μια πόλη, μπορεί να επισκεφθεί την σελίδα Current Weather, και κάνοντας αναζήτηση για την πόλη που επιθυμεί, να πατήσει αποθήκευση αυτής της πόλης από το κουμπί Add to my cities. Αν η πόλη που επέλεξε να αποθηκεύσει, δεν υπάρχει ήδη στην λίστα του, τότε θα είναι επιτυχής η αποθήκευση. Αν έχει ήδη αποθηκευτεί η πόλη, θα εμφανίσει μήνυμα στον χρήστη.



Εικόνα 3.21 Μήνυμα επιτυχούς αποθήκευσης πόλης



Εικόνα 3.22 Μήνυμα που εμφανίζεται στην προσπάθεια αποθήκευσης της ίδιας πόλης

Από αυτή την λίστα με τις πόλεις που έχει αποθηκεύσει, ο χρήστης έχει την επιλογή να διαγράψει την πόλη που επιθυμεί από την λίστα του. Κάνοντας κλικ στο κουμπί



εμφανίζεται σε κάθε καρτέλα πόλης η επιλογή διαγραφής



της πόλης. Εφόσον το πατήσει ο χρήστης, η σελίδα ανανεώνεται με μήνυμα επιτυχίας της διαγραφής.

✓ You have deleted succesfully Piraeus from your saved cities!

Εικόνα 3.22 Μήνυμα επιτυχούς διαγραφής της πόλης

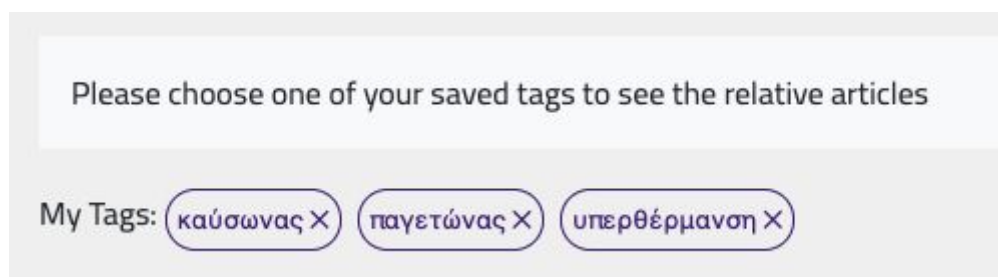




Εικόνα 23. Εμφάνιση καρτέλας με την επιλογή της διαγραφής

### 3.2.6. Σελίδα News

Ο χρήστης που έχει συνδεθεί, πηγαίνοντας στην σελίδα News, θα πρέπει να επιλέξει ένα από τα tags τα οποία έχει αποθηκεύσει.



Εικόνα 3.24 Εμφάνιση ετικετών - tags του χρήστη

Αφού επιλέξει αυτό που επιθυμεί, εμφανίζονται τα άρθρα τα οποία έχουν σαν λέξη κλειδί την αντίστοιχη ετικέτα. Στην περίπτωση που δεν υπάρχει κανένα άρθρο, για το θέμα το οποίο επιλέχθηκε, εμφανίζεται το μήνυμα που φαίνεται και στην παρακάτω εικόνα.



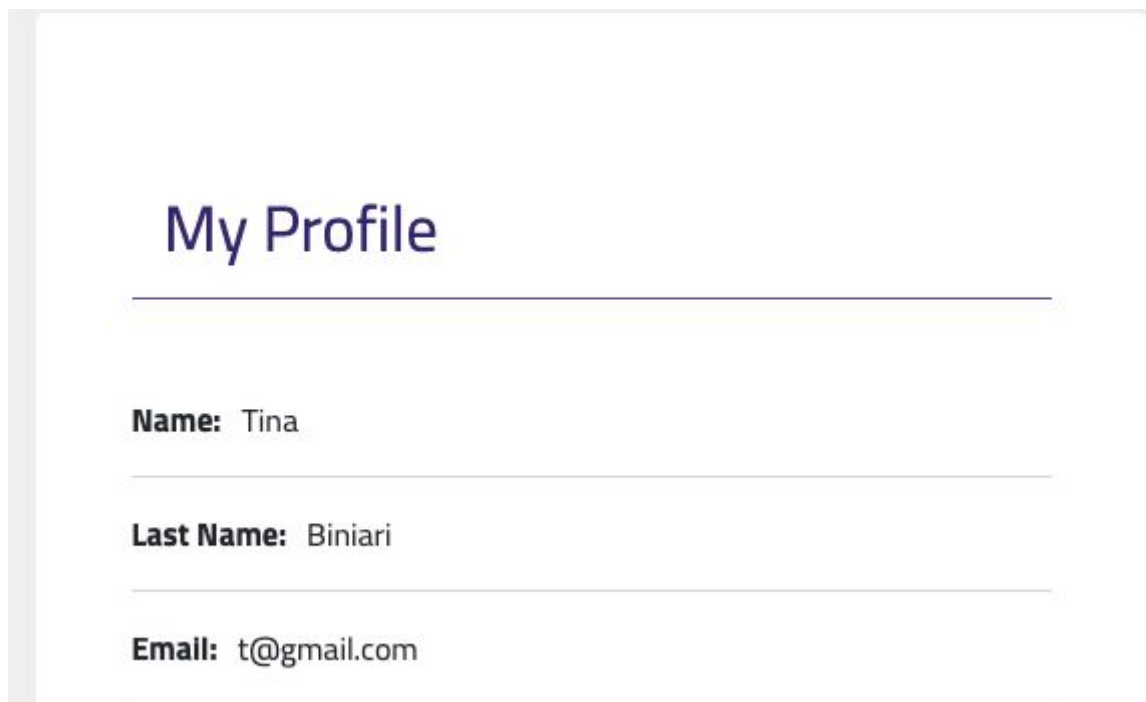
Εικόνα 3.24 Μήνυμα προς τον χρήστη, όταν δεν υπάρχουν άρθρα προς προβολή

### 3.2.7. Σελίδα My Profile

Στην σελίδα αυτή έχουν πρόσβαση μόνο όσοι χρήστες έχουν εγγραφεί στην εφαρμογή, επομένως έχουν δημιουργήσει το προφίλ τους. Ο χρήστης κατά την εγγραφή του, έχει καταχωρήσει τα στοιχεία

- ☐ Όνομα
- ☐ Επώνυμο
- ☐ Email

Τα παραπάνω στοιχεία μπορεί να τα δει μέσω της σελίδας My profile.



Εικόνα 3.25 Παράδειγμα προβολής στοιχείων ενός προφίλ

Δίνεται η δυνατότητα στον χρήστη να επεξεργαστεί τα στοιχεία του, κάνοντας κλικ στο κουμπί

επεξεργασίας

Edit

Ο χρήστης παραπέμπεται, στην σελίδα επεξεργασίας του προφίλ του. Εδώ μπορεί να προσθέσει τις εξής παραπάνω πληροφορίες:

- ☐ Τηλέφωνο
- ☐ Πληροφορίες διεύθυνσης
  - ☐ Διεύθυνση κατοικίας
  - ☐ Πόλη
  - ☐ Ταχυδρομικός κώδικας
  - ☐ Χώρα

Συμπληρώνοντας την φόρμα και κάνοντας αποθήκευση, οι πληροφορίες αυτές έχουν προστεθεί στην σελίδα του προφίλ.

### Address Info

Street Address	Karaoli kai Dimitriou 70	City	Piraeus
Postal Code	18534	Country	Greece

Submit changes

Εικόνα 3.26 Σελίδα επεξεργασίας του προφίλ

**Street Address:** Karaoli kai Dimitriou 70

**City:** Piraeus

**Country:** Greece

**Postal Code:** 18534

Εικόνα 3.27 Προβολή των νέων πληροφοριών στο προφίλ

Ο χρήστης δεν μπορεί να εισάγει μη έγκυρα στοιχεία στην φόρμα επεξεργασίας. Σε περίπτωση μη έγκυρης καταχώρησης εμφανίζεται μήνυμα υπόδειξης προς τον χρήστη.

Postal Code

ihl

Please provide a valid postal code

Telephone

ghh

Please provide a valid telephone

Εικόνα 3.28 Μηνύματα κατά την μη έγκυρη καταχώρηση στοιχείων

## 4. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ

Στο κεφάλαιο αυτό θα παρουσιαστούν αρχιτεκτονική στην οποία βασίστηκε η ανάπτυξη της εφαρμογής και τα δομικά της στοιχεία, δηλαδή τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της, οι γλώσσες προγραμματισμού, η βάση δεδομένων καθώς ο κώδικας που αναπτύχθηκε.

### 4.1. Μοντέλο αρχιτεκτονικής Model-View-Controller

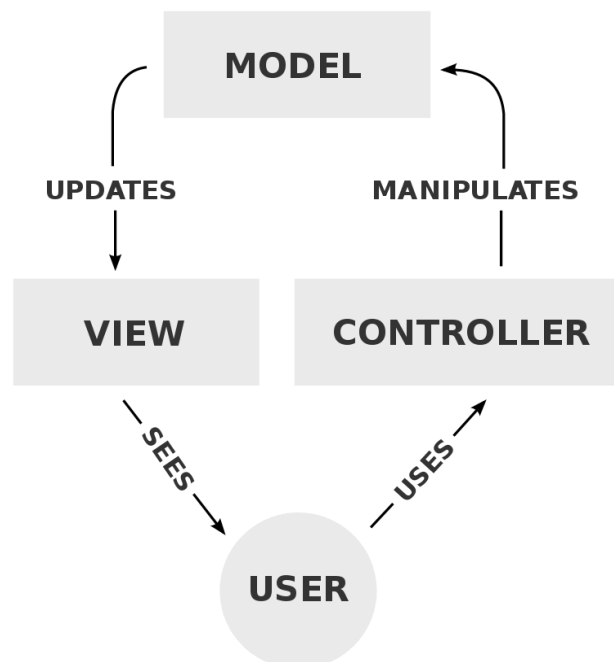
Το Model-view-controller (MVC) είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η

εφαρμογή χωρίζεται σε τρία μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήστη από την μορφή που έχει αποθηκευτεί στο σύστημα.

Το κύριο μέρος του μοντέλου είναι το αντικείμενο Model το οποίο διαχειρίζεται την ανάκτηση/αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο View είναι το αποτέλεσμα που βλέπει χρήστης, για παράδειγμα το γραφικό περιβάλλον. Το τρίτο μέρος είναι ο Controller ο οποίος διαχειρίζεται τα δεδομένα, δέχεται την είσοδο και στέλνει εντολές στο αντικείμενο Model και στο View.

Η αρχιτεκτονική σχεδίαση model-view-controller ορίζει και τις αλληλεπιδράσεις των μοντέλων

- ❑ Ο controller μπορεί να στέλνει εντολές στο μοντέλο και να ενημερώνει την κατάσταση του. Μπορεί επίσης να στέλνει εντολές ώστε να γίνει η αντίστοιχη αναπαράσταση των δεδομένων του μοντέλου μέσω του View.
- ❑ Το model ενημερώνει τις αντίστοιχες αναπαραστάσεις views και τους controllers όταν υπάρχει αλλαγή στα δεδομένα. Αυτή η ενημέρωση επιτρέπει στα views να ενημερώνουν τη γραφική απεικόνιση.
- ❑ Το view αναπαριστά με γραφικό τρόπο την πληροφορία που περιέχει το model δημιουργώντας γραφική παρουσίαση στο χρήστη.



Εικόνα 4.1 Απεικόνιση του μοντέλου Model-View-Controller

## 4.2. Εργαλεία

### 4.2.1. IntelliJ Idea

Πρόκειται για ένα περιβάλλον ανάπτυξης το οποίο είναι προορισμένο για την ανάπτυξη λογισμικού. Κυκλοφόρησε το 2001 και αποτελούσε ένα από τα πρώτα Java IDE με περισσότερες δυνατότητες. Συγκεκριμένα, κάποιες βασικές που παρέχει είναι η συμπλήρωση κώδικα, η πλοήγηση και η ανακατεύθυνση σε κλάσεις και δηλώσεις στον κώδικα, η ανακατασκευή κώδικα, η αποσφαλμάτωση του κώδικα (debugging) .

### 4.2.2. PgAdmin III

Το pgAdmin είναι η πιο δημοφιλής πλατφόρμα διαχείρισης και ανάπτυξης για την PostgreSQL, μία από τις πιο ανεπτυγμένες βάσεις δεδομένων. Η αρχική έκδοση που υλοποιήθηκε, ονομαζόταν pgManager, ήταν διαθέσιμη για την PostgreSQL 6.3.2 το έτος 1998, και υλοποιήθηκε πάλι και εκδόθηκε ως pgAdmin υπό την άδεια GNU General Public License (GPL) μέσα στους επόμενους μήνες

## 4.3. Τεχνολογίες που χρησιμοποιήθηκαν

### 4.3.1. Αρχιτεκτονική REST

Το REST (Representational State Transfer) είναι ένα σύνολο από αρχές σχεδίασης που χρησιμοποιούνται για την δημιουργία μιας διαδικτυακής εφαρμογής. Η λογική που ακολουθείται είναι ότι ο client, δηλαδή ο χρήστης, πλοηγείται στην εφαρμογή και επιλέγει συνδέσμους, προκαλεί δηλαδή μεταβολές στην κατάσταση, με σκοπό να λάβει μια απάντηση.

Η βασική αρχή σχεδίασης του REST είναι η ένα-προς-ένα αντιστοίχιση μεταξύ λειτουργιών CRUD (create, read, update, delete) και HTTP μεθόδων. Σύμφωνα με αυτή την αντιστοίχιση:

- ❑ Για τη δημιουργία ενός πόρου στον server, χρησιμοποιούμε την μέθοδο POST.
- ❑ Για την ανάκτηση ενός πόρου, χρησιμοποιούμε την GET.
- ❑ Για την αλλαγή της κατάστασης ενός πόρου ή την ενημέρωσή του, χρησιμοποιούμε την PUT.
- ❑ Για την διαγραφή ενός πόρου, χρησιμοποιούμε την DELETE.

### 4.3.2. Java

Η java πρόκειται για μία από τις δημοφιλέστερες γλώσσες προγραμματισμού. Είναι μια αντικειμενοστραφής γλώσσα, που σημαίνει ότι προσφέρει ευελιξία στην συγγραφή συντηρίσιμου κώδικα και στη δημιουργία ενός συστήματος, όπου μπορεί μελλοντικά να

τροποποιηθεί εύκολα. Υπάρχει η δυνατότητα χρήσης framework, όπως έγινε και στην δική μας περίπτωση το οποίο θα αναλυθεί στη συνέχεια. Το μεγαλύτερο πλεονέκτημα της Java, είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας. Τα προγράμματα που είναι γραμμένα σε Java τρέχουν ακριβώς το ίδιο σε Windows, Linux, Unix και Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα. Η Java είναι cross platform γλώσσα που τρέχει/ μεταφράζεται σε αντίστοιχη εικονική μηχανή (Java Virtual Machine – JVM).

#### 4.3.3. Spring Framework

Το Spring Framework είναι λογισμικό ανοιχτού κώδικα (open source) που σκοπό έχει να διευκολύνει την ανάπτυξη J2EE λογισμικού σε μεγάλη έκταση και βασίζεται στη γλώσσα προγραμματισμού Java. Η πρώτη έκδοση έγινε τον Οκτώβριο του 2002, μαζί με την δημοσιοποίηση του βιβλίου *Expert One-on-One J2EE Design and Development*. Το Spring Framework περιέχει κάποιες ενότητες οι οποίες παρέχουν μια πληθώρα υπηρεσιών:

- ❑ Spring Core Container: αποτελεί την βασική ενότητα του framework και παρέχει τους spring containers.
- ❑ Έλεγχος ταυτότητας και εξουσιοδότηση: διαμορφώσιμες διεργασίες ασφάλειας οι οποίες υποστηρίζουν μια πληθώρα από πρότυπα, πρωτόκολλα, εργαλεία και πρακτικές μέσω του Spring Security επιμέρους έργο.
- ❑ Πρόσβαση δεδομένων: εργασία που σχετίζεται με τα συστήματα διαχείρισης βάσεων δεδομένων στην πλατφόρμα της Java κάνοντας χρήση του Java Database
- ❑ Connectivity (JDBC). Το JDBC αποτελεί ένα API το οποίο ορίζει την πρόσβαση του client σε μία βάση δεδομένων.
- ❑ Model - View - Controller: ένα framework το οποίο βασίζεται στο πρωτόκολλο HTTP και στην λογική των servlets και παρέχει δέσμες για επέκταση και παραμετροποίηση για τις διαδικτυακές εφαρμογές.

#### 4.3.4. Jpa

Το Java Persistence Api είναι ένα σύνολο κλάσεων και μεθόδων, που στοχεύει στην αποθήκευση μεγάλου όγκου δεδομένων σε μία βάση δεδομένων. Δημιουργήθηκε για να λύσει προβλήματα σχετικά με τη διατήρηση των δεδομένων, και αποτελεί έναν συνδυαστικό κρίκο ανάμεσα στον αντικειμενοστραφή προγραμματισμό και τα σχεσιακά μοντέλα.

#### 4.3.5. HTML / HTML5

Η HTML (Hyper Text Markup Language) είναι γλώσσα σήμανσης, που χρησιμοποιείται για την εμφάνιση των αποτελεσμάτων σε ένα πρόγραμμα περιήγησης. Το πρόγραμμα περιήγησης δέχεται τα HTML αρχεία από έναν διακομιστή και επιστρέφει τα αρχεία ως ιστοσελίδες πολυμέσων. Μια html σελίδα αποτελείται κυρίως από το κείμενο της σελίδας, τις εικόνες και συνδέσμους προς άλλες σελίδες. Εκδόθηκε επίσημα για πρώτη φορά το 1995, ως HTML 2.0, και μέχρι σήμερα έχουν βγεί άλλες 3 εκδόσεις, με την τελευταία να αποτελεί την HTML 5.2.

#### 4.3.6. Thymeleaf

Το Thymeleaf αποτελεί ένα Java XML/XHTML/HTML5 σύστημα προτύπων - βιβλιοθήκη η οποία μπορεί να λειτουργήσει τόσο σε διαδικτυακά όσο και σε μη διαδικτυακά περιβάλλοντα. Είναι κατάλληλο για την ενσωμάτωση σε σελίδες XHTML/HTML5 ως View σε MVC εφαρμογές και παρέχει πλήρη ενοποίηση στο Spring Framework. Στις διαδικτυακές εφαρμογές, αποτελεί αντικαταστάτη των σελίδων JSP (JavaServer Pages) και εφαρμόζει την έννοια των φυσικών προτύπων: αρχεία προτύπων που μπορούν να ανοιχτούν απευθείας σε προγράμματα περιήγησης και εξακολουθούν να εμφανίζονται σωστά ως ιστοσελίδες. Αποτελεί λογισμικό ανοιχτού κώδικα και λειτουργεί υπό την άδεια Apache License 2.0.

#### 4.3.7. CSS

Η CSS (Cascading Style Sheets) αποτελεί μια γλώσσα, όπου χρησιμοποιείται για την διαμόρφωση και της εμφάνισης μιας ιστοσελίδας HTML. Οι σελίδες που μπορεί να διαμορφώσει είναι HTML. Αποσκοπεί στην καλύτερη οπτικοποίηση και στο σχεδιασμό ώστε να αποδίδονται τα αποτελέσματα με τρόπο κατανοητό στον χρήστη. Η αρχική έκδοση κυκλοφόρησε τον Δεκέμβριο του 1996.

#### 4.3.8. Javascript

Η Javascript είναι μια γλώσσα προγραμματισμού η οποία χρησιμοποιείται στους web browsers. Χρησιμοποιείται για την καλύτερη και πιο γρήγορη αλληλεπίδραση του χρήστη με το σύστημα, προσφέρει ασύγχρονη επικοινωνία με τον server και πρόκειται για μια δυναμική γλώσσα. Μαζί με τα HTML και CSS, αποτελούν τις τεχνολογίες που κυριαρχούν στο διαδίκτυο. Δημιουργήθηκε αρχικά με την ονομασία Mocha. Αργότερα, η Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξη της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. Η αρχική της χρήση ήταν ο προγραμματισμός από τον web browser, και πήρε τον χαρακτηρισμό client-side γλώσσα προγραμματισμού. Αυτό σημαίνει



ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο server, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML.

#### 4.3.9. Ajax

Το Ajax, δεν αποτελεί μια γλώσσα προγραμματισμού, αλλά πρόκειται για μια τεχνική με την οποία δίνεται η δυνατότητα δημιουργίας ασύγχρονων Web εφαρμογών. Αυτό σημαίνει ότι, η ανάκτηση και η αποστολή δεδομένων από έναν server, τρέχουν στο παρασκήνιο χωρίς να κάνουν παρέμβαση στην εμφάνιση της σελίδας. Το περιεχόμενο φορτώνεται δυναμικά, και δεν χρειάζεται εκ νέου ανανέωση της σελίδας.

#### 4.3.10. Bootstrap

Το Bootstrap αποτελεί ένα σύνολο εργαλείων ανοιχτού κώδικα με στόχο τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει πρότυπα σχεδιασμού βασισμένα σε CSS και Javascript για βασικά στοιχεία διεπαφής όπως είναι φόρμες, κουμπιά, στοιχεία πλοήγησης. Δημιουργήθηκε τον Αύγουστο του 2011, και από τότε έχουν κυκλοφορήσει 5 εκδόσεις. Το Bootstrap εστιάζει στην απλοποίηση της ανάπτυξης διαδικτυακών ιστοσελίδων και εφαρμογών και ο βασικός λόγος προσθήκης του σε ένα διαδικτυακό project είναι η εφαρμογή των διαφόρων επιλογών σε χρώματα, μεγέθη, γραμματοσειρές και πρότυπα τα οποία προσφέρει.

#### 4.3.11. PostgreSQL

Η PostgreSQL ή Postgres, αποτελεί ένα σύστημα διαχείρισης βάσεων δεδομένων, δωρεάν και ανοιχτού κώδικα το οποίο δίνει έμφαση στην επεκτασιμότητα. Ξεκίνησε να σχεδιάζεται και να εφαρμόζεται ως POSTGRES το 1986, αργότερα συνέχισε με την έκδοση Postgres95 και τελικά μετονομάστηκε σε PostgreSQL, βγάζοντας νέες εκδόσεις μέχρι σήμερα. Τα βασικά της χαρακτηριστικά είναι:

- ❑ Πλήρως συμβατή με το σύνολο ιδιοτήτων ACID (atomicity (ατομικότητα) , consistency(συνέπεια) , isolation(απομόνωση) , durability(μονιμότητα)) το οποίο εγγυάται ότι οι συναλλαγές στην βάση δεδομένων λειτουργούν αξιόπιστα.
- ❑ Υψηλότερη ασφάλεια
- ❑ Διαθέτει πολλούς τύπους δεδομένων καθώς προσφέρει και τη δυνατότητα δημιουργίας νέων τύπων δεδομένων από τους χρήστες
- ❑ Υποστήριξη των διαφόρων συναρτήσεων όπως COUNT, SUM, AVG, MIN, MAX, STDDEV and VARIANCE

- ❑ Περιβάλλον ανάπτυξης γλωσσών προγραμματισμού όπως Perl, Python, Zope, PHP, TCL/TK, ODBC, JDBC, C/C++, Embedded SQL, Delphi/Kylix/Pascal, VB, ASP, Java.
- ❑ Βιβλιοθήκη συναρτήσεων και τελεστών με ορισμένες προεγκατεστημένες συναρτήσεις όπως math, date/time, string, geometric, formatting κ.α.

#### 4.3.12. Apache Tomcat

Το εργαλείο Apache Tomcat είναι μια ανοιχτού κώδικα υλοποίηση της τεχνολογίας των σελίδων διακομιστών Java (Servlet και JSP). Παρέχει ένα περιβάλλον HTTP διαδικτυακού server στο οποίο μπορεί να εκτελεστεί Java. Τα συστατικά από τα οποία αποτελείται είναι:

- ❑ Catalina (servlet container) το οποίο παρέχει κάποιες προδιαγραφές για τους servlet και τις σελίδες JSP
- ❑ Coyote, που αποτελεί ένα συνδετικό συστατικό για τον Tomcat και υποστηρίζει το πρωτόκολλο HTTP 1.1 ως διαδικτυακό server. Αυτό είναι και το στοιχείο το οποίο επιτρέπει στο Catalina, να λειτουργεί ως web server.

#### 4.3.13. Υπηρεσίες API

Οι υπηρεσίες API (Application Programming Interface) είναι ένα ενδιάμεσο λογισμικό που επιτρέπει την επικοινωνία μεταξύ δύο εφαρμογών. Αποτελεί δηλαδή, τον φορέα που επιτρέπει την μεταφορά δεδομένων από ένα σύστημα σε ένα άλλο σύστημα δημιουργώντας συνδεσιμότητα. Το κύριο πλεονέκτημα που προσφέρουν τα API είναι η άντληση πληροφοριών μεταξύ των συστημάτων. Επίσης προσφέρει μεγάλη ταχύτητα στη διεκπεραίωση διαφόρων λειτουργιών. Η άντληση των δεδομένων γίνεται μέσω των endpoints, τα οποία περιλαμβάνουν ένα URL μιας υπηρεσίας ή ενός διακομιστή και είναι η τοποθεσία από την οποία θα λάβουμε τους πόρους που χρειαζόμαστε.

### 4.4. Ανάλυση Κώδικα της εφαρμογής

Στην παρακάτω ενότητα θα αναλυθεί η υλοποίηση των βασικών λειτουργιών της εφαρμογής μας παραθέτοντας και τα αντίστοιχα κομμάτια κώδικα.

Η λογική στην οποία βασιστήκαμε για την ανάπτυξη της εφαρμογής ήταν ο συνδυασμός πολλών υπηρεσιών API με σκοπό την άντληση πληροφοριών και ενημέρωση του χρήστη με αυτές.

Χρησιμοποιήσαμε, λοιπόν, υπηρεσίες API από τις παρακάτω πηγές:

- ❑ **OpenWeatherMap:** είναι ένας από τους κορυφαίους παρόχους πληροφοριών για τον καιρό. Διαθέτει κάποια βασικά πακέτα δωρεάν, τα οποία είναι και αυτά τα οποία χρησιμοποιήσαμε. Συγκεκριμένα, κάναμε χρήση του πακέτου “Current Weather Data”

(Τρέχοντα δεδομένα καιρού) και “5 day / 3 hour Forecast” (Πενθήμερη / ανά 3 ώρες πρόβλεψη)

- ❑ **Stormglass:** Το API Storm Glass παρέχει προβλέψεις υψηλής ανάλυσης έως και 10 ημέρες μπροστά, καθώς και ιστορικά δεδομένα. Χρησιμοποιούμε το πακέτο “Historical Data” (Ιστορικά δεδομένα) για το ιστορικό θερμοκρασιών τις τελευταίες 10 ημέρες.
- ❑ **NewsApi:** Το News API είναι ένα απλό REST API για αναζήτηση και ανάκτηση άρθρων από όλο το διαδίκτυο. Χρειαστήκαμε το NewsApi, για την ανάκτηση άρθρων με συγκεκριμένες λέξεις κλειδιά.
- ❑ **AirVisual Api:** Η AirVisual συλλέγει δεδομένα από το μεγαλύτερο δίκτυο αισθητήρων εδάφους παγκοσμίως, παρέχοντας τις πιο ακριβείς και αξιόπιστες πληροφορίες για την ποιότητα του αέρα. Το πακέτο που χρησιμοποιήσαμε μας παρέχει πληροφορίες για την ποιότητα του αέρα σε πραγματικό χρόνο.

Η δομή που ακολουθήσαμε για την ανάπτυξη της εφαρμογής είναι:

- ❑ Δημιουργία του configuration, όπου περιέχει τις κλάσεις SecurityConfiguration & WebMvcConfiguration. Οι κλάσεις παρέχουν ασφάλεια στην εφαρμογή μας, σε αυτές δηλώνουμε τις διαδρομές (paths) που περιέχει η εφαρμογή μας και μέσω αυτών δηλώνουμε αν οι σελίδες θέλουμε να φαίνονται σε εγγεγραμμένους χρήστες.
- ❑ Δημιουργία των models, και η αναπαράσταση τους στην βάση δεδομένων
- ❑ Δημιουργία των views, δηλαδή των σελίδων μας
- ❑ Δημιουργία των Controller, για την διεκπεραίωση των λειτουργιών
- ❑ Δημιουργία των Repositories, για την επικοινωνία με την βάση δεδομένων
- ❑ Δημιουργία των Services, για την διαχείριση των δεδομένων από την βάση ή από τις υπηρεσίες API
- ❑ Δημιουργία του φακέλου Integration, όπου περιέχει τις κλάσεις μέσω των οποίων έχουμε πρόσβαση στα δεδομένα των υπηρεσιών API

#### 4.4.1. Λειτουργίες Σύνδεση / Εγγραφή χρήστη

Αρχικά δημιουργήσαμε το Model User, δηλαδή την κλάση που αντιστοιχεί στον χρήστη και αυτόματα, με την δημιουργία του model, δημιουργείται ο πίνακας “**user\_details**”, στην βάση δεδομένων. Η κλάση αποδίδει τα εξής χαρακτηριστικά στον χρήστη:

- ❑ `userId`

- ☐ email
- ☐ κωδικός πρόσβασης
- ☐ Όνομα
- ☐ Επίθετο
- ☐ Ενεργός / ανενεργός
- ☐ Ρόλος
- ☐ Προφίλ
- ☐ tags

Ύστερα, έγινε δημιουργία της μεθόδου **saveUser()**, στον Registration Controller, όπου κατά την υποβολή της φόρμας από την σελίδα **register**, κάνει έναν έλεγχο εγκυρότητας των στοιχείων και αποθηκεύει τον χρήστη στον πίνακα της βάσης.

```
@PostMapping
ModelAndView saveUser(@Valid @ModelAttribute("user") User user, BindingResult bindingResult,
ModelAndView modelAndView, @RequestParam("checkedTags") List<Long> checkedTags,
HttpServletRequest request) {
    if (userService.findByEmail(user.getEmail()) != null) {
        bindingResult.rejectValue("email", "error.user", "This email already exists");
    }
    if (bindingResult.hasErrors()) {
        modelAndView.addObject("show", "show active");
        return modelAndView;
    }
    User regUser = userService.saveUser(user);
    regUser.setTags(checkedTags);
    userRepository.save(user);
    modelAndView.setViewName("register");
    modelAndView.addObject("headerMessage", "Please choose your
favourite topics");
    modelAndView.addObject("description", "Pick your tags in order to see news about these topics.
You can choose up to 4 tags");
    modelAndView.addObject("buttonText", "Submit");
    modelAndView.addObject("successMessage", "User " + regUser.getEmail() + " has being
successfully registered");
    modelAndView.addObject("alert", "alert-success");
    modelAndView.addObject("show", "show active");
    modelAndView.addObject("tags", tagsService.getTags());
    return modelAndView;
}
```

Εικόνα 4.1 Κλάση saveUser() στον controller

```
public User saveUser(User user) {  
    user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));  
    user.setActive(1);  
    user.setRole(user.getRole());  
    user.setProfile(new Profile());  
    return userRepository.save(user);  
}
```

Εικόνα 4.2 Κλάση saveUser() στο UserService

Για την σύνδεση στην εφαρμογή, ο χρήστης συμπληρώνει τα στοιχεία του στην φόρμα σύνδεσης και ο έλεγχος εγκυρότητας γίνεται μέσω της κλάσης **SecurityConfiguration**. Αν η σύνδεση είναι επιτυχής γίνεται ανακατεύθυνση στην αρχική σελίδα index, αλλιώς γίνεται εμφάνιση των error που έχουμε θέσει στην σελίδα register

Η σελίδα **register** περιέχει τις φόρμες σύνδεσης και εγγραφής, και η εναλλαγή τους γίνεται με τη δημιουργία tabs μέσω bootstrap

```
.loginPage("/login").failureUrl("/login?error=true")  
.defaultSuccessUrl("/index")
```

Εικόνα 4.3: Κώδικας ανακατεύθυνσης μετά την σύνδεση, στην κλάση SecurityConfiguration

```
<div align="center" th:if="{param.error}">  
    <p class="d-flex text-12 text-non-valid">Email or Password invalid, please verify</p>  
</div>
```

Εικόνα 4.4: Παράδειγμα μηνύματος σφάλματος στην φόρμα σύνδεσης

#### 4.4.2. Το προφίλ μου (My Profile)

Αφού γίνει η εγγραφή του χρήστη, γίνεται αυτόματα και η δημιουργία ενός προφίλ του χρήστη. Αρχικά φτιάξαμε το model **Profile** με τα χαρακτηριστικά:

- ☐ profileId
- ☐ userId
- ☐ telephone
- ☐ streetAddress
- ☐ city
- ☐ postalCode
- ☐ country

- ❑ birthDate
- ❑ image

Δημιουργείται ένας πίνακας στη βάση, **profile**, με τις παραπάνω στήλες, και επίσης δημιουργείται και πίνακας **user\_profile**, ο οποίος ενώνει το προφίλ με τον χρήστη μέσω των **id**.

Ο χρήστης μπορεί με την πρόσβαση του σε αυτό, να προσθέσει/επεξεργαστεί τις πληροφορίες. Η επεξεργασία εκτελείται μέσω των μεθόδων

- ❑ **editProfile** (GET method)
- ❑ **submitMethod**(POST method)

```
@GetMapping("/editProfile")
ModelAndView editProfile(@ModelAttribute("profile") Profile profile, Principal principal, ModelAndView
modelAndView){
    User user = userService.findByEmail(principal.getName());
    modelAndView.setViewName("editProfile");
    modelAndView.addObject("profile", user.getProfile());
    if(user.getProfile().getBirthDate() != null) {
        String birthdate = user.getProfile().getBirthDate().toString();
        modelAndView.addObject("birthDay", getBirthdate(birthdate));
    }
    modelAndView.addObject("user", user);
    modelAndView.addObject("tags", tagsService.getTags());
    modelAndView.addObject("myTags", tagsService.getTagsByUser(user.getUserId()));
    return modelAndView;
}
```

Εικόνα 4.5: Η μέθοδος editProfile, στον userController

Η **editProfile()** μας εμφανίζει την σελίδα με την φόρμα επεξεργασίας.

Η **submitMethod()** στέλνει το αίτημα υποβολής των στοιχείων στον server. Τα στοιχεία υποβάλλονται μέσω της συνάρτησης **updateProfile()**, που υπάρχει στην κλάση **UserService**.

```
@PostMapping("/editProfile")
ModelAndView submitProfile(@Valid @ModelAttribute("profile") Profile profile, BindingResult result,
HttpServletRequest request, Principal principal, ModelAndView modelAndView){
    User user = userService.findByEmail(principal.getName());
    user.setFirstName(request.getParameter("firstName"));
    user.setLastName(request.getParameter("lastName"));
    modelAndView.setViewName("editProfile");
    if (result.hasErrors()) {
        modelAndView.addObject("user", user);
        modelAndView.addObject("birthDay", request.getParameter("birthDate"));
        return modelAndView;
    }
    Profile savedProfile = userService.updateProfile(user, profile);
    User updatedUser = userService.updateUser(user);

    modelAndView.addObject("profile", savedProfile);
    modelAndView.addObject("user", updatedUser);
    modelAndView.addObject("success", user.getFirstName() + "success");
    return getProfile(principal, modelAndView);
}
```

Εικόνα 4.6: Η μέθοδος submitProfile, στον userController

```
public Profile updateProfile(User user, Profile profile){
    if(profileRepository.countByUserId(user.getUserId()) > 0) {
        Profile oldProfile = profileRepository.findById(user.getUserId());
        profile.setProfileId(oldProfile.getProfileId());
    }
    profile.setProfileId(user.getProfile().getProfileId());
    profile.setUserId(user.getUserId());
    return profileRepository.save(profile);
}
```

Εικόνα 4.7: Η μέθοδος ενημέρωσης του προφίλ, στον userController

#### 4.4.3. Τρέχων καιρός (Current Weather)

Τα δεδομένα που αντλούμε για τον τρέχοντα καιρό προέρχονται από το OpenWeatherMap.

Δημιουργήσαμε μια κλάση **OWMApi**, όπου δημιουργεί αντικείμενα (objects) για τα δεδομένα που θέλουμε να αντλήσουμε με βάση τη μορφή του JSON που έχει το API.

Στη συνέχεια, δημιουργήσαμε την κλάση **CurrentService** όπου περιέχει τα endpoints του API από το οποίο θέλουμε να αντλήσουμε πληροφορίες. Τα endpoints περιέχουν παραμέτρους, τις οποίες τις λαμβάνουν καλώντας τις παρακάτω μεθόδους:

- ❑ **getCurrentByCity**(String city,String country): επιστρέφει αποτελέσματα με βάση την πόλη και την χώρα
- ❑ **getCurrentByCoord**(Double lon, Double lat): επιστρέφει αποτελέσματα με βάση τις συντεταγμένες latitude, longitude.

```
public ArrayList getCurrentByCity(String city,String country) {  
    ArrayList<OWMApi> citySummary = new ArrayList<>();  
    OWMApi owmApi=this.getCurrentUrlByCity(city, country);  
    citySummary.add(owmApi);  
    return citySummary;  
}
```

Εικόνα 4.8: Μέθοδος επιστροφής αποτελεσμάτων τρέχοντα καιρού με βάση την πόλη και την χώρα

```
public ArrayList getCurrentByCoord(Double lon, Double lat) {  
    ArrayList<OWMApi> citySummary = new ArrayList<>();  
    OWMApi owmApi=this.getCurrentUrlByCoord(lon, lat);  
    citySummary.add(owmApi);  
    return citySummary;  
}
```

Εικόνα 4.9: Μέθοδος επιστροφής αποτελεσμάτων τρέχοντα καιρού με βάση την πόλη και την χώρα

Δημιουργήθηκε ο **CurrentController** ο οποίος περιέχει τις μεθόδους :

- ❑ **getCurrentCity()**
- ❑ **getCurrentLocation()**

και η φόρμα αναζήτησης στην οποία ο χρήστης είτε μπορεί να αναζητήσει μέσω πόλης και χώρας, είτε με την ακριβή τοποθεσία.

Στην αναζήτηση μέσω πόλης καλείται η συνάρτηση **getCurrentCity()** όπου παίρνει ως παράμετρο την πόλη και την χώρα, τα οποία τα έχει συμπληρώσει ο χρήστης. Η **getCurrentCity()** καλεί, την **getCurrentByCity()**, από την κλάση **CurrentService**, η οποία μας φέρνει τα αποτελέσματα από την υπηρεσία API.

```
@GetMapping("/current/city")
```



```

public ModelAndView getCurrentCity(@Valid @ModelAttribute("city") City city, BindingResult
bindingResult, @ModelAttribute("coord") Coord coord, ModelAndView modelAndView,
@RequestParam("city") String name, @RequestParam("country") String country) {
    modelAndView.setViewName("currentWeatherCard");
    if((name == "") || (country == "")){
        bindingResult.rejectValue("city", "error.city", "Please fill in both fields");
    }
    if (bindingResult.hasErrors()) {
        return modelAndView;
    }
    try {
        OWMApi owmApi = (OWMApi) currentService.getCurrentByCity(name, country).get(0);
        String iconId = owmApi.weather.get(0).getIcon();
        modelAndView.addObject("cityList", currentService.getCurrentByCity(name, country));
        modelAndView.addObject("currentDate",
currentService.getTime());
        ArrayList<Double> coordByCity = forecastService.getCoordByCity(name, country);
        modelAndView.addObject("fourDay", forecastService.getFourDayForecast(coordByCity.get(0),
coordByCity.get(1)));
        modelAndView.addObject("iconLink", currentService.getIconLink(iconId));
    } catch (HttpClientErrorException ex){
        throw new ResponseStatusException(
            HttpStatus.NOT_FOUND, "Not found", ex
        );
    }
    ArrayList<String> namesOfDays = new ArrayList<>();
    for (int i = 1; i < 5; i++) {
        namesOfDays.add(LocalDate.now().getDayOfWeek().plus(i).toString());
    }
    modelAndView.addObject("namesOfDays", namesOfDays);
    return modelAndView;
}

```

Εικόνα 4.10: Μέθοδος επεξεργασίας αποτελεσμάτων αναζήτησης καιρού με βάση πόλη/χώρα

Στην αναζήτηση μέσω συντεταγμένων, καλείται η συνάρτηση Javascript **showLocation()**, η οποία με τη σειρά της καλεί μια υπηρεσία εύρεσης της τοποθεσίας της συσκευής. Η υπηρεσία ονομάζεται `Navigator.geolocation`, και επιστρέφει τις συντεταγμένες της περιοχής που βρισκόμαστε. Στη συνέχεια, καλείται η συνάρτηση **getCurrentLocation()** όπου παίρνει ως παράμετρο τις συντεταγμένες. Η **getCurrentLocation()** καλεί, την **getCurrentByCoord()**, από την κλάση **CurrentService**, η οποία μας φέρνει τα αποτελέσματα από την υπηρεσία API.

```

@GetMapping("/current/location")
public ModelAndView getCurrentLocation(@Valid @ModelAttribute("coord") Coord coord,
@ModelAttribute("city") City city, ModelAndView modelAndView,Principal principal,
@RequestParam("lon") Double lon, @RequestParam("lat") Double lat) {

modelAndView.setViewName("currentWeatherCard");
OWMApi owmApi = (OWMApi) currentService.getCurrentByCoord(lon, lat).get(0);
String iconId = owmApi.weather.get(0).getIcon();
modelAndView.addObject("cityList", currentService.getCurrentByCoord(lon,lat));
modelAndView.addObject("currentDate", currentService.getTime());

modelAndView.addObject("fourDay",forecastService.getFourDayForecast(coord.getLat(),coord.getLon())
);
modelAndView.addObject("iconLink", currentService.getIconLink(iconId));
ArrayList<String> namesOfDays = new ArrayList<>();
for (int i = 1; i < 5 ; i++) {
namesOfDays.add(LocalDate.now().getDayOfWeek().plus(i).toString());
}
modelAndView.addObject("namesOfDays",namesOfDays);
String currentDate = LocalDate.now().toString();
String daysAgo = LocalDate.now().minusDays(10).toString();
modelAndView.addObject("charts", historyService.getHistoryChart(lat, lon, daysAgo, currentDate));
return modelAndView;
}

```

Εικόνα 4.11: Μέθοδος επεξεργασίας αποτελεσμάτων αναζήτησης καιρού με βάση τις συντεταγμένες

#### 4.4.4. Πενθήμερη πρόβλεψη καιρού / 3 ώρες (Forecast Weather)

Η αναζήτηση της πενθήμερης πρόγνωσης καιρού γίνεται με τον ίδιο τρόπο που περιγράψαμε στην προηγούμενη λειτουργία. Αντίστοιχα, δημιουργήσαμε την κλάση OWMForecast για την δημιουργία των objects, την κλάση ForecastService όπου περιέχει τις μεθόδους getForecastByCity(), getForecastByCoord().Ακουλουθώντας, την ίδια διαδικασία από τον ForecastController, οι κλάσεις getForecastCity() & getForecastLocation(), μας επιστρέφουν πληροφορίες για την πενθήμερη ανά 3 ώρες,πρόγνωση.

Επιπλέον, δημιουργήσαμε μια συνάρτηση η οποία υπολογίζει την μέση θερμοκρασία της ημέρας σε βάθος 4 ημερών, λαμβάνοντας τις ημερήσιες θερμοκρασίες από την παραπάνω υπηρεσία.

```

public LinkedHashMap<String,Object> getFourDayForecast(Double lat, Double lon){
OWMForecast forecast = this.getForecastUrlByCoord(lat, lon);
double temp = 0;
String str = " ";
LinkedHashMap<String,Object> avgtemp = new LinkedHashMap<>();

```

```

for (int i = 0; i < forecast.getList().size() - 1 ; i++) {
    if(!(str.equals(forecast.getList().get(i).getDt_txt().substring(0,10)))){
        str = forecast.getList().get(i).getDt_txt().substring(0,10);
        temp = 0;
        temp += forecast.getList().get(i).getMain().getTemp();
        avgtemp.put(historyService.fixDate(str),Math.round((temp/8) * 10) / 10.0);
    }
    else{
        temp += forecast.getList().get(i).getMain().getTemp();
        avgtemp.put(historyService.fixDate(str),Math.round((temp/8) * 10) / 10.0);
    }
}
Set<Map.Entry<String, Object>> entries = avgtemp.entrySet();
Iterator<Map.Entry<String, Object>> iterator = entries.iterator();
Map.Entry<String, Object> entry = null, firstEntry = null, lastEntry = null;

while(iterator.hasNext()){
    entry = iterator.next();
    if(firstEntry == null)
        firstEntry = entry;
    lastEntry = entry;
}
avgtemp.remove(firstEntry.getKey());
avgtemp.remove(lastEntry.getKey());
return avgtemp;
}

```

Εικόνα 4.12: Συνάρτηση υπολογισμού μέσης θερμοκρασίας ανά ημέρα

#### 4.4.5. Ιστορικό θερμοκρασιών ( History Weather)

Βασίζόμενοι στην ίδια λογική, δημιουργήσαμε τις κλάσεις History, Hours για την δημιουργία των objects, και την HistoryServices, όπου φέρνει τα δεδομένα από το API της Stormglass. Το API προσφέρει ιστορικό θερμοκρασιών μιας τοποθεσίας μέσω των συντεταγμένων latitude, longitude, για κάθε ώρα εντός του 24ώρου. Ορίσαμε το χρονικό διάστημα για το οποίο θα ενημερώνεται ο χρήστης να είναι έως 10 ημέρες πριν. Ωστόσο, επειδή θέλαμε την μέση θερμοκρασία ανά ημέρα, δημιουργήσαμε την συνάρτηση **getAverageDailyWeather()**, όπου υπολογίζει την μέση τιμή της θερμοκρασίας. Επίσης, κάναμε χρήση της βιβλιοθήκης CanvasJS, για την οπτικοποίηση των αποτελεσμάτων σε γράφημα

```

public LinkedHashMap<String,Double> getAverageDailyWeather(Double lat, Double lon,String startDate,
String endDate){
    History history = this.getHistoryUrl(lat, lon, startDate, endDate);
}

```

```

double temp = 0;
String str = " ";
LinkedHashMap<String,Double> avgtemp = new LinkedHashMap<>();
for (int i = 0; i < history.getHours().size() - 1; i++) {
    if(!(str.equals(history.getHours().get(i).getTime().substring(0,10)))){
        str = history.getHours().get(i).getTime().substring(0,10);
        temp = 0;
        temp += history.getHours().get(i).getAirTemperature().get(1).getValue();
        avgtemp.put(fixDate(str),Math.round((temp/24) * 10) / 10.0);
    }
    else{
        temp += history.getHours().get(i).getAirTemperature().get(1).getValue();
        avgtemp.put(fixDate(str),Math.round((temp/24) * 10) / 10.0);
    }
}
return avgtemp;
}

```

**Εικόνα 4.13: Συνάρτηση υπολογισμού μέσης θερμοκρασίας ανά ημέρα**

```

@PostMapping("/history/location")
public ModelAndView getHistoryByLocation(@ModelAttribute("history") History history,
@ModelAttribute("city") City city,
    @ModelAttribute("coord") Coord coord, ModelAndView modelAndView, @RequestParam("lon")
Double lon,
    @RequestParam("lat") Double lat) {
    modelAndView.setViewName("history");
    try {
        String currentDate = LocalDate.now().toString();
        String daysAgo = LocalDate.now().minusDays(10).toString();
        String cityName = forecastService.getCityByCoord(lon, lat);
        HashMap<String, Double> averageDailyWeather = historyService.getAverageDailyWeather(lat, lon,
daysAgo,
        currentDate);
        modelAndView.addObject("avgWeather", averageDailyWeather);
        modelAndView.addObject("charts", historyService.getHistoryChart(lat, lon, daysAgo, currentDate));
        modelAndView.addObject("cityName", cityName);
    }
    catch (HttpClientErrorException e){
        modelAndView.addObject("errorField", "Sorry, the city or country you are searching for, is not
found");
        modelAndView.addObject("class", "my-4 p-2 rounded");
        modelAndView.addObject("style", "color:#680e0e;font-size:16px;background:#c1172f36");
    }
    return modelAndView;
}

```

**Εικόνα 4.14: Μέθοδος POST για την επιστροφή αποτελεσμάτων**

#### 4.4.6. Ποιότητα αέρα (Air Quality)

Ομοίως, για την παροχή πληροφοριών για την ποιότητα του αέρα, κάναμε χρήση του AirVisualAPI ακολουθώντας ακριβώς την ίδια λογική, δηλαδή αναζήτηση με βάση την πόλη/χώρα ή την ακριβή μας τοποθεσία. Επιπλέον στην σελίδα, εμφανίζεται ένας πίνακας με την ποιότητα του αέρα για τις δέκα μεγαλύτερες πόλεις της Ελλάδας. Για αυτή την υλοποίηση, δημιουργήσαμε μία κλάση - model LargestCities και καταχωρήσαμε στον πίνακα που δημιουργήθηκε από το model, τις πόλεις αυτές με τις συντεταγμένες τους.

Κατασκευάσαμε, μια συνάρτηση η οποία επιστρέφει τα αποτελέσματα με την ποιότητα του αέρα για αυτή την λίστα.

Επίσης, λόγω των χρωματικών απεικονίσεων από τις οποίες χαρακτηρίζεται η μόλυνση του αέρα, το χρώμα του background στο αποτέλεσμα, επιστρέφεται δυναμικά, ανάλογα με το εύρος στο οποίο βρίσκεται ο αριθμός AQI που ορίζει την ποιότητα αέρα. Τέλος ανάλογα με τον αριθμό AQI, καθορίζονται και οι περιγραφές των επιπέδων της ποιότητας.

```
if(this.aqius > 0 && this.aqius <= 50){
    this.color = "bg-green";
    this.title = "Good";
    this.description = "Air quality is considered satisfactory, and air pollution poses little or no risk";
}
else if(this.aqius > 50 && this.aqius <= 100){
    this.color = "bg-yellow";
    this.title = "Moderate";
    this.description = "Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.";
}
else if(this.aqius > 100 && this.aqius <= 150){
    this.color = "bg-orange";
    this.title = "Unhealthy for sensitive groups";
    this.description = "Members of sensitive groups may experience health effects. The general public is not likely to be affected.";
}
else if (this.aqius > 150 && this.aqius <= 200) {
    this.color = "bg-deep-orange";
    this.title = "Unhealthy";
    this.description = "Everyone may begin to experience health effects; members of sensitive groups may experience more serious health effects";
}
```

Εικόνα 4.15 : Εφαρμογή μηνυμάτων και στυλ, ανάλογα με το επίπεδο μόλυνσης AQI

#### 4.4.7. Tags

Κάθε χρήστης κατά την εγγραφή του, επιλέγει από 1 έως 4 tags, για τα οποία θα βλέπει άρθρα με λέξη-κλειδί το συγκεκριμένο tag. Δημιουργήσαμε το model Tags, στο οποίο δηλώνουμε :

- ❑ id
- ❑ tagName

Από το model, δημιουργείται ένας πίνακας στην βάση, στον οποίο έχουμε καταχωρήσει τα tags που θέλαμε. Ο χρήστης επιλέγει μέχρι τέσσερα tags, και μέσω συνάρτησης Javascript, δεν έχει τη δυνατότητα να επιλέξει παραπάνω.

```
$(document).ready(function() {  
    $('#tag-checkbox').on('change', function() {  
        if($('#tag-checkbox:checked').length > 4) {  
            this.checked = false;  
        }  
    });  
});
```

Εικόνα 4.16 : Κώδικας JQuery, για τον περιορισμό έως 4 tags

Με την εγγραφή, γίνεται αυτόματα και αποθήκευση των tags στον χρήστη, στην μέθοδο **saveUser()** του **RegistrationController**. Κατασκευάστηκε η κλάση **TagsService** και το repository **TagsRepository**, δημιουργώντας εκεί τα queries και τις κλάσεις με τις οποίες μπορούμε να φέρουμε από την βάση, τα tags του συνδεδεμένου χρήστη.

```
public interface TagsRepository extends JpaRepository<Tags,Long> {  
    List<Tags> findAll();  
    @Query(nativeQuery = true,value="SELECT id,tag_name FROM tags INNER JOIN user_tags ON  
user_tags.tags=tags.id where user_user_id=?")  
    ArrayList<Tags> findByIdTags(Long userId);  
}
```

Εικόνα 4.17 : Εκτέλεση query μέσω JPA

```
@Service  
public class TagsService {  
    public List<Tags> getTags() {  
        List<Tags> tags = tagsRepository.findAll();  
        return tags;  
    }  
    public ArrayList<Tags> getTagsByUser(Long userId) {  
        ArrayList<Tags> tags = tagsRepository.findByIdTags(userId);  
    }  
}
```

```
    return tags;
  }
}
```

Εικόνα 4.18 : Κλάση TagsService

#### 4.4.8. Κλιματικά νέα (Climate News)

Για την προβολή των άρθρων χρησιμοποιήσαμε την υπηρεσία NewsAPI, η οποία φέρνει αποτελέσματα άρθρων με τα tags που θέτουμε εμείς στην εφαρμογή μας. Ομοίως, δημιουργήθηκε η κλάση NewsApi και η NewsService, για την δημιουργία των αντικειμένων και την εξόρυξη των πληροφοριών

```
public NewsApi getNews(LocalDate date, String keyword) {
    URI url = new UriTemplate(NEWS_SERVICE_URL).expand(date, this.newsApiKey, keyword);
    return newsService.invoke(url, NewsApi.class);
}

public List<Article> getArticlesList(LocalDate date, String keyword) {
    List<Article> articles = getNews(date, keyword).getArticles();
    return articles;
}
```

Εικόνα 4.19 : Μέθοδοι κλήσης του API και επιστροφής των άρθρων

Επίσης, φτιάξαμε μια συνάρτηση, η οποία σελιδοποιεί τα άρθρα, ώστε να είναι πιο φιλική προς τον χρήστη η σελίδα.

```
public Page<Article> paginateArticles(LocalDate date, String keyword, Pageable pageable){
    int pageSize = pageable.getPageSize();
    int currentPage = pageable.getPageNumber();
    int startItem = currentPage * pageSize;
    List<Article> articlesList;
    if (getArticlesList(date, keyword).size() < startItem) {
        articlesList = Collections.emptyList();
    } else {
        int toIndex = Math.min(startItem + pageSize, getArticlesList(date, keyword).size());
        articlesList = getArticlesList(date, keyword).subList(startItem, toIndex);
    }
    Page<Article> bookPage
        = new PageImpl<Article>(articlesList, PageRequest.of(currentPage, pageSize),
        getArticlesList(date, keyword).size());
    return bookPage;
}
```

Εικόνα 4.20 : Δημιουργία pagination για την σελιδοποίηση των άρθρων

Στη συνέχεια, δημιουργήσαμε τον **ArticlesController**, ο οποίος περιέχει τις μεθόδους:

- ❑ **getArticles()**
- ❑ **getArticlesByTag()**

Η πρώτη συνάρτηση χρησιμοποιείται στην αρχική σελίδα, όπου υπάρχει διαφοροποίηση με την σύνδεση ή μη ενός χρήστη. Επίσης, σε αυτή την περίπτωση φορτώνονται στην σελίδα 4 άρθρα από κάθε κατηγορία tag, και η εναλλαγή τους με βάση το tag που έχει επιλέξει ο χρήστης γίνεται με συνάρτηση της Javascript, την οποία δημιουργήσαμε.

```
@GetMapping("/articles")
ModelAndView getArticles(ModelAndView modelAndView, @RequestParam("page") Optional<Integer>
page,
    @RequestParam("size") Optional<Integer> size) {
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    User user = userService.findByEmail(auth.getName());
    modelAndView.setViewName("articles");
    if(user!=null) {
        ArrayList<Tags> tagMap = tagsService.getTagsByUser(user.getUserId());
        modelAndView.addObject("tags", tagMap);
        modelAndView.addObject("title", "Weather News");
        modelAndView.addObject("description", "Please choose one of your saved tags to see the relative
articles");
    }
    else {
        LocalDate lastMonthNews = LocalDate.now().minusMonths(1);
        modelAndView.addObject("news", newsService.getNews(lastMonthNews, "
κλιματική").getArticles());
        int currentPage = page.orElse(1);
        int pageSize = size.orElse(5);
        Page<Article> articlePage = newsService.paginateArticles(lastMonthNews, "κλιματική",
PageRequest.of(currentPage - 1, pageSize));
        modelAndView.addObject("articlePage", articlePage);
        int totalPages = articlePage.getTotalPages();
        if (totalPages > 0) {
            List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages)
                .boxed()
                .collect(Collectors.toList());
            modelAndView.addObject("pageNumbers", pageNumbers);
        }
        modelAndView.addObject("title", "Weather News");
        modelAndView.addObject("description", "See the most relative news about weather and climate");
    }
    return modelAndView;
}
```



Εικόνα 4.20 : Μέθοδος `getArticles()`

Η δεύτερη συνάρτηση, χρησιμοποιείται στην σελίδα Climate News, όπου αρχικά ο χρήστης επιλέγει το tag που θέλει, και στέλνεται στον controller, το id του tag ως παράμετρος στην συνάρτηση `getArticlesByTag()`, όπου με τη σειρά του, μας επιστρέφει τα άρθρα με το ανάλογο keyword.

```
@GetMapping("/articles/{tag}")
ModelAndView getArticlesByTag(ModelAndView modelAndView, @RequestParam("page")
Optional<Integer> page,
    @RequestParam("size") Optional<Integer> size, @PathVariable("tag") String tag) {
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    User user = userService.findByEmail(auth.getName());
    modelAndView.setViewName("articles");
    modelAndView.addObject("title", "Weather News");
    modelAndView.addObject("description", "Check out the latest news");
    ArrayList<Tags> tagMap = tagsService.getTagsByUser(user.getId());
    modelAndView.addObject("tags", tagMap);
    LocalDate lastMonthNews = LocalDate.now().minusMonths(1);
    modelAndView.addObject("news", newsService.getNews(lastMonthNews, tag).getArticles());
    int currentPage = page.orElse(1);
    int pageSize = size.orElse(5);
    Page<Article> articlePage = newsService.paginateArticles(lastMonthNews, tag,
    PageRequest.of(currentPage - 1, pageSize));
    modelAndView.addObject("articlePage", articlePage);

    int totalPages = articlePage.getTotalPages();
    if (totalPages > 0) {
        List<Integer> pageNumbers = IntStream.rangeClosed(1, totalPages)
            .boxed()
            .collect(Collectors.toList());
        modelAndView.addObject("pageNumbers", pageNumbers);
    }
    if (articlePage.getTotalElements() == 0) {
        modelAndView.addObject("errorField", "Unfortunately, there are no articles about: "+tag);
        modelAndView.addObject("class", "my-4 p-2 rounded");
        modelAndView.addObject("style", "color:#680e0e;font-size:16px;background:#c1172f36");
    }
    return modelAndView;
}
```

Εικόνα 4.21 : Μέθοδος `getArticlesByTag()`

#### 4.4.9. Προσθήκη / Διαγραφή πόλης (Add City / Delete City)

Για την δημιουργία της λειτουργίας Αποθηκευμένες πόλεις του χρήστη, αρχικά φτιάξαμε το model **SavedCities** με τα χαρακτηριστικά

- ❑ id
- ❑ userId
- ❑ lat
- ❑ lon

Ύστερα, κατασκευάσαμε το **SavedCitiesRepository**, για την εκτέλεση των queries στον πίνακα που δημιουργήθηκε από το model, και το **SavedCitiesService**, όπου εκεί υπάρχουν οι μέθοδοι προσθήκης/διαγραφής μιας πόλης από τον χρήστη.

Στη συνέχεια δημιουργήσαμε τον **SavedCitiesController** όπου περιέχει τις μεθόδους που διαχειρίζονται τα αιτήματα προσθήκης και διαγραφής μιας πόλης.

```
@PostMapping(value = "/current/save/{lat}/{lon}")
@ResponseBody
public String saveCity(@ModelAttribute("city") City city, @ModelAttribute("coord") Coord coord,
Principal principal, HttpServletRequest request, @PathVariable String lat, @PathVariable String lon) {
    User user = userService.findByEmail(principal.getName());
    String name = forecastService.getCityByCoord(Double.valueOf(lon), Double.valueOf(lat));
    String cityName= name.substring(0,name.length()-3);
    SavedCities savedCities = new SavedCities();
    savedCities.setUserId(user.getUserId());
    savedCities.setLat(lat);
    savedCities.setLon(lon);
    String message = citiesService.check(savedCities,lat,lon,user.getUserId(),
StringUtils.capitalize(cityName));
    return message;
}
```

Εικόνα 4.22 : Κώδικας αποθήκευσης πόλης

```
@PostMapping(value = "/deleteCity/{cityId}")
@ResponseBody
public RedirectView deleteCity(@ModelAttribute("savedCities")SavedCities savedCities, Principal
principal, HttpServletRequest request, ModelAndView modelAndView, @PathVariable String cityId,
RedirectAttributes attributes){
    String cityName=request.getParameter("cityName");

    Long city_Id=Long.valueOf(cityId);
    savedCitiesService.deleteCity(city_Id);
    attributes.addFlashAttribute("succesMessage","You have deleted succesfully "+cityName+" from your
saved cities!");
    attributes.addFlashAttribute("stylesuccesMessage","succesMessage");
    attributes.addFlashAttribute("visible","d-block ");
    return new RedirectView("/savedCities");
}
```

Εικόνα 4.23 : Κώδικας διαγραφής πόλης

#### 4.4.10. Εμφάνιση αποτελεσμάτων στον χρήστη

Για την εμφάνιση των αποτελεσμάτων στο χρήστη, επιλέξαμε να χρησιμοποιήσουμε την τεχνολογία των AJAX Requests, όπου κατά την υποβολή αιτημάτων, η σελίδα δεν ανανεώνεται αλλά εμφανίζει άμεσα τα αποτελέσματα.

Δημιουργήσαμε μία συνάρτηση, όπου χρησιμοποιείται από τις τέσσερις βασικές λειτουργίες της εφαρμογής μας:

- ❑ **Τρέχων καιρός (Current Weather)**
- ❑ **Πενθήμερη πρόβλεψη καιρού / 3 ώρες (Forecast Weather)**
- ❑ **Ιστορικό θερμοκρασιών (History Weather)**
- ❑ **Ποιότητα αέρα (Air Quality)**

Η συνάρτηση αυτή παίρνει τους εξής παραμέτρους

- ❑ **searchBy**: αν η αναζήτηση γίνει μέσω πόλης/χώρας ή συντεταγμένων τοποθεσίας
- ❑ **param1,param2**: ανάλογα με τον τρόπο αναζήτησης, αλλάζουν και τα params σε
  - ❑ **city,country**, αν πρόκειται για τον πρώτο τρόπο
  - ❑ **lat,lon** αν πρόκειται για τον δεύτερο

Επίσης, η συνάρτηση έχει την μεταβλητή **predictionType**, η οποία αλλάζει ανάλογα με την υπηρεσία για την οποία την καλούμε.

Για παράδειγμα, καλούμε τη συνάρτηση για να αναζητήσουμε τον τρέχοντα καιρό στην πόλη Αθήνα, Ελλάδα.

Κατά την υποβολή της αναζήτησης, καλείται η παραπάνω συνάρτηση, όπου σε αυτή στέλνονται τα απαραίτητα δεδομένα, τα οποία υπάρχουν στο View μέσω του Controller.

Θα σταλούν δηλαδή τα παρακάτω:

- ❑ **predictionType** : current
- ❑ **searchBy** : city
- ❑ **param1** : city
- ❑ **param2** : country
- ❑ **city** : Αθήνα
- ❑ **country** : Ελλάδα

Η συνάρτηση,λοιπόν, στέλνει το αίτημα GET που επιθυμούμε με AJAX Request, εκτελώντας ουσιαστικά την μέθοδο **getCurrentCity()** στον **CurrentController**.

```
if(searchBy == 'city'){
    if((city) || (country)){
        xhttp.open("GET", "/" + predictionType + "/" + searchBy + "?" + param1 + "=" + city + "&" + param2 +
            "=" + country, true);
        console.log("/" + predictionType + "/" + searchBy + "?" + param1 + "=" + city + "&" + param2 + "=" +
            country);
    }else{
        document.getElementById("errorField").style.display="block";
        document.getElementById("errorField").innerHTML = "Please fill in both fields";
    }
}
if(searchBy == 'location'){
    document.getElementById("errorField").style.display="none";
    xhttp.open("GET", "/" + predictionType + "/" + searchBy + "?" + param1 + "=" + lat + "&" + param2 + "=" + lon,
true);
}
xhttp.send();
```

Εικόνα 4.23 : Κώδικας εκτέλεσης AJAX Request, για την ανάκτηση του καιρού

## 5. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην παρούσα εργασία, στόχος μας ήταν η παρουσίαση μιας εφαρμογής φιλική προς τον χρήστη, και η ανάδειξη της χρησιμότητας των web application στην καθημερινότητα μας. Περιγράψαμε τα εργαλεία με τα οποία δουλέψαμε και τις τεχνολογίες στις οποίες βασίστηκαμε για την υλοποίηση αυτή. Παρουσιάσαμε, εκτενώς, τις λειτουργίες της εφαρμογής μέσω κειμένου αλλά και εικόνων, προκειμένου να υπάρχει μια καλύτερη αντίληψη για την εφαρμογή. Επιπλέον, αναλύσαμε την λογική και τους τρόπους με τους οποίους αναπτύχθηκε η ιδέα αυτή ως εφαρμογή.

Έχοντας ολοκληρώσει την εφαρμογή μας, μπορούμε να συμπεράνουμε η τεχνολογία των διαδικτυακών εφαρμογών, αποτελεί μία από τις καλύτερες επιλογές στην για την ανάπτυξη μιας εφαρμογής, στην εποχή μας, καθώς είναι πολύ εύκολα προσβάσιμη και προσιτή σε όλες τις συσκευές. Επιπλέον, η χρήση των APIs και ιδιαίτερα ο συνδυασμός τους, μπορούν να μας βοηθήσουν να δημιουργήσουμε μια ολοκληρωμένη εφαρμογή ενημέρωσης, παρέχοντας όλες τις σωστές πληροφορίες προς τον χρήστη. Φυσικά, ιδιαίτερο ρόλο παίζει ο τρόπος με τον οποίο παρουσιάζεται η πληροφορία προς τον χρήστη, όπου δόθηκε ιδιαίτερη έμφαση ώστε η εφαρμογή να μην κουράζει στην χρήση της και να είναι όσο το δυνατόν γρηγορότερη.

## 5.1. Μελλοντικές Επεκτάσεις

Θα επιθυμούσαμε μελλοντικά, η εφαρμογή να είναι πιο προσωποποιημένη προς τον χρήστη, επεκτείνοντας την με λειτουργίες οι οποίες θα βασίζονται στα προσωπικά του στοιχεία. Επίσης, θα μπορούσε επεκταθεί δημιουργώντας λειτουργία για ειδοποιήσεις, εφόσον το επιθυμεί ένας εγγεγραμμένος χρήστης, ώστε να ενημερώνεται καθημερινά από την εφαρμογή μας.

## ΒΙΒΛΙΟΓΡΑΦΙΑ

### Ηλεκτρονική Βιβλιογραφία

- [1] Margaret Rousse, “**Client-server model (client-server architecture)**”, 30 Oct 2008  
<https://searchnetworking.techtarget.com/definition/client-server>
- [2] Wikipedia contributors, “**Client-server model**”, Wikipedia, The Free Encyclopedia, 24 Jul. 2020  
[https://en.wikipedia.org/wiki/Client%E2%80%93server\\_model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model)
- [3] Wikipedia contributors, “**Web application**”, Wikipedia, The Free Encyclopedia, 15 Jul. 2020.  
[https://en.wikipedia.org/wiki/Web\\_application](https://en.wikipedia.org/wiki/Web_application)
- [4] Emily Stevens, “**What Is The Difference Between A Mobile App And A Web App?**”, Careerfoundry, 3 Apr. 2018  
<https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/>
- [5] Daniel Nations, “**What Is a Web Application?**”, Lifewire, 25 June 2020  
<https://www.lifewire.com/what-is-a-web-application-3486637>
- [6] Wikipedia contributors, “**Model-view-controller**”, Wikipedia, The Free Encyclopedia, 13 Jul. 2020  
<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>
- [7] Wikipedia contributors, “**IntelliJ IDEA**”, Wikipedia, The Free Encyclopedia, 8 Jul. 2020.  
[https://en.wikipedia.org/wiki/IntelliJ\\_IDEA](https://en.wikipedia.org/wiki/IntelliJ_IDEA)
- [8] Wikipedia contributors, “**PostgreSQL**” Wikipedia, The Free Encyclopedia, 18 Jul. 2020  
<https://en.wikipedia.org/wiki/PostgreSQL>
- [9] Γουγουόσης Αλέξανδρος, “**Εισαγωγή στην αρχιτεκτονική δικτυακών υπηρεσιών REST**”, Σχολή Μηχανικών Παραγωγής και Διοίκησης  
<http://www.users.dpem.tuc.gr/gougousis/rest/>
- [10] Sagar Mane, “**Understanding REST (Representational State Transfer)**”, Medium, 9 Jun. 2017  
<https://medium.com/@sagar.mane006/understanding-rest-representational-state-transfer-85256b9424aa>
- [11] Baeldung, “**The Guide to RestTemplate**”, Baeldung, 7 Jul. 2020  
<https://www.baeldung.com/rest-template>
- [12] Wikipedia contributors. “**Representational state transfer**”, Wikipedia, The Free Encyclopedia, 22 Jul. 2020

- [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)
- [13] Wikipedia contributors. "**Java (programming language)**", Wikipedia, The Free Encyclopedia, 20 Jul. 2020  
[https://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
- [14] Source: <https://stackoverflow.com/>, "**Γιατί Java**", Developers  
<https://devs.kariera.gr/coding-school-java/why-java-coding/>
- [15] Wikipedia contributors, "**Spring Framework**", Wikipedia, The Free Encyclopedia, 25 July 2020  
[https://en.wikipedia.org/wiki/Spring\\_Framework](https://en.wikipedia.org/wiki/Spring_Framework)
- [16] Manoj Debnath, "**What Is JPA Technology?**", Developer.com, 9 Jul. 2018  
<https://www.developer.com/java/data/what-is-jpa-technology.html>
- [17] "**JPA vs. Hibernate**", Javatpoint  
<https://www.javatpoint.com/jpa-vs-hibernate>
- [18] "**HTML Introduction**", W3Schools  
[https://www.w3schools.com/html/html\\_intro.asp](https://www.w3schools.com/html/html_intro.asp)
- [19] Wikipedia contributors, "**HTML**", Wikipedia, The Free Encyclopedia, 17 May 2018  
<https://el.wikipedia.org/wiki/HTML>
- [20] Wikipedia contributors, "**Thymeleaf**", Wikipedia, The Free Encyclopedia, 11 May 2020  
<https://en.wikipedia.org/wiki/Thymeleaf>
- [21] Wikipedia contributors, "**Cascading Style Sheets**", Wikipedia, The Free Encyclopedia, 22 July 2020  
[https://en.wikipedia.org/wiki/Cascading\\_Style\\_Sheets](https://en.wikipedia.org/wiki/Cascading_Style_Sheets)
- [22] Wikipedia contributors, "**JavaScript**", Wikipedia, The Free Encyclopedia, 24 July 2020  
<https://en.wikipedia.org/wiki/JavaScript>
- [23] "**AJAX Introduction**", W3Schools  
[https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)
- [24] Wikipedia contributors, "**Ajax (programming)**", Wikipedia, The Free Encyclopedia, 20 July 2020  
[https://en.wikipedia.org/wiki/Ajax\\_\(programming\)](https://en.wikipedia.org/wiki/Ajax_(programming))
- [25] Wikipedia contributors, "**Bootstrap (front-end framework)**", Wikipedia, The Free Encyclopedia, 24 July 2020  
[https://en.wikipedia.org/wiki/Bootstrap\\_\(front-end\\_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))
- [26] Wikipedia contributors, "**PostgreSQL**", Wikipedia, The Free Encyclopedia, 18 July 2020,  
<https://en.wikipedia.org/wiki/PostgreSQL>
- [27] Wikipedia contributors, "**Apache Tomcat**", Wikipedia, The Free Encyclopedia, 21 July 2020  
[https://en.wikipedia.org/wiki/Apache\\_Tomcat](https://en.wikipedia.org/wiki/Apache_Tomcat)
- [28] "**API Endpoints - What Are They? Why Do They Matter?**", SmartBear  
<https://smartbear.com/learn/performance-monitoring/api-endpoints/>
- [29] Tyler Elliot Bettilyon, "**What Is an API and Why Should I Use One?**", Medium, 11 Jan. 2018  
<https://medium.com/@TebbaVonMathenstien/what-is-an-api-and-why-should-i-use-one-863c3365726b>
- [30] Stormglass.io, **Introduction - Storm Glass | API Documentation**, <https://docs.stormglass.io/#/>
- [31] Newsapi.org, **Documentation**, <https://newsapi.org/docs>
- [32] IQAir, **AirVisual API**, <https://www.iqair.com/air-pollution-data-api>

[33] AirVisual.com, **Documentation,Api,Air Pollution - Live Air Quality Index (AQI) - AirVisual**  
<https://airvisual.com/api/documentation>