



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη εφαρμογής web για ημιαυτόματη χαρτογράφηση  
πλημμυρών με χρήση συνθετικού ανοίγματος ραντάρ  
τηλεπισκοπικών απεικονίσεων

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αδαμάντιος Σιγιώργης

Επιβλέπων : Όνομα, Αρχικό Πατρώνυμου, Επίθετο  
Ιδιότητα Επιβλέποντα

Αθήνα, Μήνας Έτος





ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ  
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ  
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ  
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη εφαρμογής web για ημιαυτόματη χαρτογράφηση  
πλημμυρών με χρήση συνθετικού ανοίγματος ραντάρ  
τηλεπισκοπικών απεικονίσεων

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αδαμάντιος Σιγιώργης

Επιβλέπων : Όνομα, Αρχικό Πατρώνυμου, Επίθετο  
Ιδιότητα Επιβλέποντα

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31<sup>η</sup> Μήνα Έτος.

.....  
Ον/μο Μέλος Δ.Ε.Π  
Ιδιότητα Μέλους Δ.Ε.Π

.....  
Ον/μο Μέλος Δ.Ε.Π  
Ιδιότητα Μέλους Δ.Ε.Π

.....  
Ον/μο Μέλος Δ.Ε.Π  
Ιδιότητα Μέλους Δ.Ε.Π

Αθήνα, Μήνας Έτος

Αδαμάντιος Σιγιώργης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Αδαμάντιος, Σιγιώργης, 2025.

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

# Περίληψη

Οι εικόνες συνθετικού ανοίγματος ραντάρ (SAR) είναι ιδιαίτερα χρήσιμες για τη χαρτογράφηση πλημμυρών, καθώς παρέχουν πληροφορίες για την επιφάνεια της Γης ακόμα και υπό δυσμενείς καιρικές συνθήκες, όπως νεφώσεις ή έντονες βροχοπτώσεις. Το πρόγραμμα Sentinel-1 του Ευρωπαϊκού Οργανισμού Διαστήματος διαθέτει εικόνες SAR, οι οποίες αξιοποιούνται από τη βιβλιοθήκη FLOODPY για τη δημιουργία χαρτογραφήσεων πλημμυρών με εντυπωσιακά αποτελέσματα. Ωστόσο, το FLOODPY απευθύνεται σε χρήστες με τεχνικές γνώσεις και δεν περιλαμβάνει λειτουργίες για την συστηματική αποθήκευση και τον διαμοιρασμό των χαρτογραφήσεων που παράγονται.

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας ολοκληρωμένης web εφαρμογής που ενσωματώνει τις δυνατότητες του FLOODPY για τη χαρτογράφηση πλημμυρών με τη χρήση εικόνων SAR. Η εφαρμογή επιτρέπει τη δημιουργία, την αποθήκευση και την προβολή χαρτογραφήσεων εντός διαδραστικού χάρτη στο περιβάλλον του φυλλομετρητή.

Κατά την ανάπτυξη του frontend της εφαρμογής χρησιμοποιήθηκαν σύγχρονες τεχνολογίες, όπως η React και το MUI για την υλοποίηση μιας ευχρηστης και λειτουργικής διεπαφής χρήστη. Η διεπαφή καθοδηγεί τους χρήστες για την δημιουργία νέων χαρτογραφήσεων και παρουσιάζει με λεπτομέρεια τις απεικονίσεις των πλημμυρισμένων περιοχών που παράγονται. Επιπλέον, για την ταυτοποίηση των χρηστών και την προστασία υπολογιστικά κοστοβόρων υπηρεσιών αναπτύχθηκαν σελίδες δημιουργίας λογαριασμού και αυθεντικοποίησης.

Το backend σχεδιάστηκε ώστε να υποστηρίζει όλες τις απαιτούμενες υπολογιστικές λειτουργίες, όπως η ασύγχρονη εκτέλεση του FLOODPY, η αποθήκευση χαρτογραφήσεων και η διαχείριση των δεδομένων που αποστέλλονται στο frontend. Για τον σκοπό αυτό, χρησιμοποιήθηκαν τεχνολογίες όπως το Django, ο GeoServer, το Redis και η PostgreSQL. Παράλληλα, υλοποιήθηκαν οι απαραίτητες υπηρεσίες για την ασφαλή εγγραφή και την αυθεντικοποίηση χρηστών.

Τέλος, για την αυτοματοποίηση της εγκατάστασης της εφαρμογής χρησιμοποιήθηκε το Docker Compose. Δημιουργήθηκαν κατάλληλα Docker Images και αρχεία παραμετροποίηση που διασφαλίζουν ότι η εγκατάσταση της εφαρμογής είναι απλή στο περιβάλλον ανάπτυξη και αξιόπιστη στο περιβάλλον παραγωγής.

**Λέξεις Κλειδιά:** Χαρτογράφηση Πλημμυρών, Ανάπτυξη Web Εφαρμογής, FLOODPY, Διαδραστική Διεπαφή Χρήστη, Ασύγχρονη Εκτέλεση Εργασιών, Τεχνολογίες Frontend (React, Leaflet, MUI, Vite, Yup, NPM), Τεχνολογίες Backend (Django, Celery, Channel, GeoServer, PostgreSQL, Redis, Nginx), Εικόνες Συνθετικού Ανοίγματος Ραντάρ, Sentinel-1, WebSockets, GeoTIFF

# Abstract

Synthetic Aperture Radar (SAR) images are particularly useful for flood mapping, as they provide information about the Earth's surface even under adverse weather conditions such as clouds or heavy rainfall. The European Space Agency's Sentinel-1 program provides SAR images, which are utilized by the FLOODPY library to generate flood maps with impressive results. However, FLOODPY is designed for users with technical expertise and lacks functionalities for the systematic storage and sharing of the generated maps.

The objective of this thesis is to develop a comprehensive web application that integrates the capabilities of FLOODPY in order to map floods using SAR images. The application allows the creation, storage and visualization of flood maps within an interactive map in the browser environment.

During the development of the application's frontend, modern technologies such as React and MUI were used to implement a user-friendly and functional interface. The interface guides users in creating new maps and presents detailed visualizations of the flooded areas generated. Additionally, pages for account creation and authentication were developed to identify users and protect computationally expensive services.

The backend was designed to support all necessary computational functions, such as the asynchronous execution of FLOODPY, the storage of maps, and the management of data sent to the frontend. For this purpose, technologies such as Django, GeoServer, Redis, and PostgreSQL were used. Furthermore, necessary services for secure registration and user authentication were implemented.

Finally, Docker Compose was used to automate the application's deployment. Suitable Docker Images and configuration files were created to ensure that the application's installation is simple in development environments and reliable in production environments.

**Key Words:** Flood Mapping, Developing Web Application, FLOODPY, Interactive User Interface, Asynchronous Job Execution, Frontend Technologies (React, Leaflet, MUI, Vite, Yup, NPM), Backend Technologies (Django, Celery, Channel, GeoServer, PostgreSQL, Redis, Nginx), Synthetic Aperture Radar (SAR), Sentinel-1, WebSockets, GeoTIFF

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένειά μου για την αγάπη και τη στήριξή τους καθ' όλη την διάρκεια των σπουδών μου. Επίσης, ευχαριστώ τον κ. Δημήτρη Συκά που μου έδωσε την ευκαιρία να εκπονήσω την παρούσα διπλωματική εργασία, καθώς και όλους τους καθηγητές της σχολής για τα πολύτιμα μαθήματα και τις γνώσεις που μου παρείχαν.

# Περιεχόμενα

<b>Περίληψη.....</b>	<b>5</b>
<b>Abstract.....</b>	<b>7</b>
<b>Ευχαριστίες.....</b>	<b>8</b>
<b>1. Εισαγωγή.....</b>	<b>11</b>
<b>2. Πρόγραμμα FLOODPY.....</b>	<b>13</b>
2.1 Βήματα Επεξεργασίας.....	13
2.2 Αποτελέσματα.....	16
2.3 Χρήση και Περιορισμοί.....	17
<b>3. Στόχοι και Χαρακτηριστικά της Web Εφαρμογής.....</b>	<b>19</b>
<b>4. Τεχνολογίες του Frontend.....</b>	<b>21</b>
4.1 React.....	21
4.2 React-Router.....	22
4.3 MUI.....	23
4.4 NPM, Rollup, Vite.....	24
<b>5. Λειτουργίες και Ανάπτυξη του Frontend.....</b>	<b>26</b>
5.1 Δημιουργία Νέας Χαρτογράφησης.....	26
5.2 Παρουσίαση Περιεχομένου Χαρτογράφησης.....	32
5.2.1 Ενημερώσεις Κατάστασης Εργασίας.....	35
5.2.2 Απεικόνιση Προϊόντων.....	38
5.3 Εμφάνιση Συνόλου Αποθηκευμένων Χαρτογραφήσεων.....	39
5.4 Εγγραφή και Αυθεντικοποίηση.....	43
5.4.1 Δημιουργία Λογαριασμού.....	44
5.4.2 Αυθεντικοποίηση.....	46
5.4.3 Διαχείριση Λογαριασμού.....	49
5.5 Εμφάνιση Χαρτογραφήσεων Χρήστη.....	49

5.6 Έλεγχος Εργασιών FLOODPY .....	52
<b>6. Επιβεβαίωση του Frontend.....</b>	<b>55</b>
<b>7. Λειτουργίες και Αρχιτεκτονική του Backend.....</b>	<b>57</b>
7.1 Τεχνικές Προδιαγραφές.....	57
7.2 Αρχιτεκτονική.....	58
<b>8. Τεχνολογίες του Backend.....</b>	<b>61</b>
8.1 Django.....	61
8.1.1 Ενσωμάτωση.....	62
8.1.2 Βασικές Δυνατότητες.....	63
8.2 Celery.....	65
8.3 Channels.....	66
8.4 GeoServer.....	67
<b>9. Υπηρεσίες του Backend.....</b>	<b>69</b>
9.1 Διαχείριση Χαρτογραφήσεων.....	69
9.1.1 Δημιουργία Χαρτογράφησης.....	70
9.1.2 Λήψη Χαρτογράφησης.....	74
9.1.3 Λήψη Συνόλου Χαρτογραφήσεων.....	75
9.2 Εκτέλεση Εργασίας FLOODPY .....	77
9.3 Αποστολή Ενημερώσεων.....	81
9.4 Δημιουργία και Διαχείριση Λογαριασμού.....	82
9.5 Αυθεντικοποίηση.....	84
<b>10. Επιβεβαίωση του Backend.....</b>	<b>86</b>
<b>11. Εγκατάσταση της Web Εφαρμογής.....</b>	<b>87</b>
<b>12. Σύνοψη και Μελλοντικές Επεκτάσεις.....</b>	<b>90</b>
<b>Βιβλιογραφία.....</b>	<b>91</b>

# 1. Εισαγωγή

Οι πλημμύρες αποτελούν μία από τις πιο συχνές και θανατηφόρες φυσικές καταστροφές [1]. Συνήθως προκαλούν εκτεταμένες ζημιές σε υποδομές, κατοικίες και αγροτικές εκτάσεις, ενώ σε ορισμένες περιπτώσεις επιφέρουν και ανθρώπινες απώλειες. Η επιδείνωση της κλιματικής αλλαγής αναμένεται να εντείνει περαιτέρω τις επιπτώσεις των πλημμυρών [2], καθώς η συχνότητα και η σφοδρότητα των ακραίων καιρικών φαινομένων που τις προκαλούν αυξάνονται. Επομένως, είναι επιτακτική ανάγκη να ληφθούν άμεσα ουσιαστικές πρωτοβουλίες, ώστε να περιοριστούν οι πιθανότητες καταστροφικών συνεπειών στο μέλλον.

Βασικός παράγοντας για την αντιμετώπιση των επιπτώσεων των πλημμυρών είναι η δυνατότητα χαρτογράφησης των πλημμυρισμένων περιοχών. Πράγματι, η χαρτογράφηση των εκτάσεων που επικαλύπτονται από ύδατα κατά τη διάρκεια πλημμυρικών φαινομένων επιτρέπει τον προσδιορισμό των πληγέντων περιοχών, διευκολύνοντας την εφαρμογή κατάλληλων μέτρων από τους αρμόδιους φορείς. Επιπλέον, η διατήρηση ενός ιστορικού δεδομένων με προηγούμενες χαρτογραφήσεις πλημμυρών προσφέρει πολύτιμες πληροφορίες, οι οποίες καθιστούν δυνατή την αναγνώριση τοποθεσιών που είναι ευάλωτες σε πλημμυρικά φαινόμενα. Με αυτόν τον τρόπο, καθίσταται δυνατός ο αποτελεσματικός σχεδιασμός προληπτικών δράσεων, μειώνοντας τον κίνδυνο και τις συνέπειες μελλοντικών φαινομένων.

Στο πλαίσιο αυτό, αναδεικνύεται η ανάγκη για την ανάπτυξη μιας εύχρηστης και αξιόπιστης εφαρμογής χαρτογράφησης πλημμυρών. Μια τέτοια εφαρμογή πρέπει να επιτρέπει την έγκαιρη και ακριβή χαρτογράφηση πλημμυρικών υδάτων, ώστε να υποστηρίζει την άμεση διαχείρισης των καταστροφών. Παράλληλα, πρέπει να αποθηκεύει τα δεδομένα προηγούμενων χαρτογραφήσεων, προκειμένου να διευκολύνει την παρακολούθηση τάσεων και την υλοποίηση κατάλληλων αντιπλημμυρικών έργων στο μέλλον. Για την αξιοποίηση των εν λόγω λειτουργιών από τους αρμόδιους φορείς, η εφαρμογή πρέπει να είναι προσιτή, να μην απαιτεί εξειδικευμένες τεχνικές ή επιστημονικές γνώσεις από τους χρήστες, και να είναι προσβάσιμη τόσο από κινητές όσο και από σταθερές συσκευές.

Στόχος της παρούσας διπλωματικής εργασία είναι η ανάπτυξη μιας web εφαρμογής για τη χαρτογράφηση πλημμυρών, η οποία βασίζεται στο λογισμικό FLOODPY και ικανοποιεί τις απαιτήσεις λειτουργικότητας, αξιοπιστίας και προσβασιμότητας που αναφέρθηκαν. Η εφαρμογή παρέχει στους χρήστες δυνατότητες δημιουργίας, αποθήκευσης και προβολής χαρτογραφήσεων πλημμυρών μέσω μιας σύγχρονης και διαδραστικής διεπαφής, προσβάσιμης από το περιβάλλον

του φυλλομετρητή. Για την παραγωγή των χαρτογραφήσεων, η εφαρμογή χρησιμοποιεί το λογισμικό FLOODPY, μέσω του οποίου πραγματοποιείται αποτελεσματική αποτύπωση πλημμυρισμένων περιοχών. Κατά την υλοποίηση δόθηκε έμφαση στην κλιμακωσιμότητα και στην αποδοτικότητα της εφαρμογής, επιτρέποντας ταυτόχρονα την μελλοντική αναβάθμιση των λειτουργιών της και την ενσωμάτωση άλλων λογισμικών χαρτογράφησης επιπροσθέτως του FLOODPY. Ως εκ τούτου, η εφαρμογή μπορεί να επεκταθεί και να παραμένει επίκαιρη με την πρόοδο της έρευνας στο χώρο της χαρτογράφησης φυσικών καταστροφών.

## 2. Πρόγραμμα FLOODPY

Το FLOODPY (FLOOD PYthon toolbox) είναι λογισμικό ανοικτού κώδικα, το οποίο έχει αναπτυχθεί σε Python και επιτρέπει την χαρτογράφηση γεωγραφικών περιοχών που έχουν πλημμυρίσει έπειτα από φυσικές καταστροφές [3]. Ο εντοπισμός των πλημμυρισμένων περιοχών πραγματοποιείται μέσω λήψης εικόνων συνθετικού ανοίγματος ραντάρ (SAR) και κατηγοριοποίησης των εικονοστοιχείων τους σε πλημμυρισμένα και μη πλημμυρισμένα. Η διαδικασία κατηγοριοποίησης βασίζεται στην ανάλυση των στατιστικών διαφορών που παρατηρούνται στις εικόνες πριν και μετά από την φυσική καταστροφή. Η εν λόγω διαδικασία παρουσιάζεται με λεπτομέρεια στο επόμενο τμήμα.

### 2.1 Βήματα Επεξεργασίας

Για την δημιουργία της χαρτογράφησης μιας πλημμύρας, το FLOODPY αρχικά συλλέγει εικόνες SAR. Οι εικόνες αυτές λαμβάνονται από κεραίες εγκατεστημένες σε δορυφόρους ή σε αεροσκάφη μέσω επαναλαμβανόμενης εκπομπής ηλεκτρομαγνητικών παλμών, συνήθως μεταξύ 300MHz και 12GHz [4]. Με την λήψη των αντίστοιχων ανακλώμενων παλμών, την επεξεργασία και το συνδυασμό των αποτελεσμάτων, δημιουργούνται εικόνες υψηλές ανάλυσης, οι οποίες είναι ανεξάρτητες των καιρικών φαινομένων, αλλά ευαίσθητες στις τραχείς και στις υδάτινες επιφάνειες ανάκλασης [5]. Τα χαρακτηριστικά αυτά καθιστούν εικόνες τέτοιου τύπου ιδιαίτερα χρήσιμες για την χαρτογράφηση πλημμυρών.

Το FLOODPY χρησιμοποιεί εικόνες SAR, οι οποίες έχουν ληφθεί στο πλαίσιο του προγράμματος Sentinel-1 του Ευρωπαϊκού Οργανισμού Διαστήματος [6]. Η επιλογή των προς επεξεργασία λήψεων γίνεται βάσει της ορισμένης από το χρήστη περιοχής ενδιαφέροντος και της χρονικής στιγμής που έγινε η πλημμύρα. Αρχικά, επιλέγεται μια εικόνα της περιοχής ενδιαφέροντος που λήφθηκε αμέσως μετά την πλημμύρα. Στη συνέχεια, επιλέγονται εικόνες της ίδια περιοχής που λήφθηκαν κατά την διάρκεια μιας προκαθορισμένης περιόδου πριν την πλημμύρα, υπό την προϋπόθεση ότι την στιγμή της εκάστοτε λήψης η βροχόπτωση ήταν περιορισμένη. Τα όρια βροχόπτωσης και η διάρκεια της χρονικής περιόδου που καθορίζουν την επιλογή των λήψεων πριν την πλημμύρα αποσκοπούν στο να μην συμπεριληφθούν εικόνες που περιέχουν εξωγενείς αλλαγές, δηλαδή αλλαγές που δεν οφείλονται στην προς εξέταση

πλημμύρα. Οι βέλτιστες τιμές των παραμέτρων αυτών διαφέρουν ανά περίπτωση και μπορούν να ρυθμιστούν από το χρήστη.

Αφότου έχουν συλλεχθεί εικόνες SAR από το πρόγραμμα Sentinel-1, εφαρμόζεται σε αυτές ένα σύνολο βήματων προεπεξεργασίας. Τα βήματα αυτά παρουσιάζονται περιληπτικά στο [3] και ο στόχος τους είναι να προετοιμάσουν τις εικόνες για τη στατιστική χρονική ανάλυση που επακολουθεί. Για το σκοπό αυτό, πραγματοποιούνται τυπικές διορθώσεις και δημιουργείται μια co-registered στοίβα που περιέχει τις εικόνες πριν και μετά την πλημμύρα. Στην co-registered στοίβα, οι εικόνες ευθυγραμμίζονται ώστε οι αντίστοιχες περιοχές τους να αναφέρονται στην ίδια γεωγραφική τοποθεσία, αλλά σε διαφορετικές χρονικές στιγμές.

Στο επόμενο βήμα, εφαρμόζεται στην co-registered στοίβα μια στατιστική χρονική ανάλυση. Πιο συγκεκριμένα, η στοίβα χρησιμοποιείται για την δημιουργία ενός χάρτη που καταγράφει τις στατιστικές αποκλίσεις των τιμών των εικονοστοιχείων μετά την πλημμύρα. Η μέτρηση των στατιστικών αποκλίσεων γίνεται με τον υπολογισμό του t-score. Αυτός ο υπολογισμός βασίζεται στην υπόθεση ότι το γινόμενο των συντελεστών οπισθοσκέδασης (backscatter coefficients) των πολώσεων VH και VV ακολουθούν την κατανομή T, η οποία είναι ένα είδος κανονικής κατανομής που χρησιμοποιείται σε περιπτώσεις μικρών δειγμάτων [3]. Οι πολώσεις VH (Κατακόρυφο Οριζόντιο) και VV (Κατακόρυφο Κατακόρυφο) αναφέρονται στον προσανατολισμό των εκπεμπόμενων και λαμβανόμενων ηλεκτρομαγνητικών κυμάτων σε σχέση με την επιφάνεια της Γης και η έντασή τους διαφέρει ανάλογα με τα χαρακτηριστικά του εδάφους όπου γίνεται η ανάκλαση [7].

Η διαδικασία ολοκληρώνεται με την κατηγοριοποίηση του χάρτη των t-score του προηγούμενου βήματος σε πλημμυρισμένες και μη περιοχές. Η μέθοδος που εφαρμόζεται χρησιμοποιεί κυρίως αλγορίθμους κατωφλίωσης και αποτελείται από τις ακόλουθες διαδικασίες:

1. Υπολογίζεται μια μάσκα βάσει του συντελεστή διτροπικότητας (bimodality). Μέσω αυτής, επισημαίνονται οι περιοχές του χάρτη όπου η κατανομή της τιμής του t-score παρουσιάζει κατά προσέγγιση δύο κορυφές. Η μάσκα διτροπικότητας, όπως αναφέρεται στο [3], βελτιώνει σημαντικά την απόδοση των αλγορίθμων κατωφλίωσης.
2. Εκτελείται ο αλγόριθμος κατωφλίωσης Kittler-Illingworth επί του αποτελέσματος της εφαρμογής της μάσκας διτροπικότητας στο χάρτη των t-score. Αυτός ο αλγόριθμος στηρίζεται στην υπόθεση ότι η κατανομή των τιμών των εικονοστοιχείων είναι ένα μίγμα δύο κανονικών κατανομών οι οποίες αντιστοιχούν στις περιοχές προς διαχωρισμό. Όπως

περιγράφεται στο [8], βάσει της μέσης τιμής και της διακύμανσης των τιμών των εικονοστοιχείων στις περιοχές που προκύπτουν από την επιλογή ενός κατωφλιού μπορούν να οριστούν δύο κανονικές κατανομές. Η επικάλυψη των κατανομών αυτών υποδεικνύει την πιθανότητα λανθασμένης κατηγοριοποίησης που σχετίζεται με το επιλεγμένο κατώφλι. Το κατώφλι που ελαχιστοποιεί την επικάλυψη αυτή, ελαχιστοποιεί ταυτόχρονα και την πιθανότητα λανθασμένης κατηγοριοποίησης. Επομένως, ο αλγόριθμος Kittler-Illingworth, επιλύοντας το αντίστοιχο πρόβλημα ελαχιστοποίησης, βρίσκει το βέλτιστο κατώφλι που χρησιμοποιείται για να διαχωριστεί ο χάρτης των t-score σε πλημμυρισμένες και μη πλημμυρισμένες περιοχές.

3. Η διάκριση των εικονοστοιχείων σε πλημμυρισμένα και μη βελτιώνεται περεταίρω μέσω της εφαρμογής μιας δεύτερης, τοπικής, μεθόδου κατωφλίωσης. Η μέθοδος βασίζεται στον αλγόριθμο Otsu και εφαρμόζεται στα εικονοστοιχεία της πλημμυρισμένης περιοχής που προκύπτει στο προηγούμενο βήμα. Ειδικότερα, γύρω από κάθε εικονοστοιχείο που έχει χαρακτηριστεί προηγουμένως ως πλημμυρισμένο, επιλέγεται το τμήμα της εικόνας με το μέγιστο συντελεστή διτροπικότητας. Αν η τιμή του συντελεστή είναι επαρκώς υψηλή, τότε εκτελείται ο αλγόριθμος Otsu στην αντίστοιχη περιοχή. Ο αλγόριθμος Otsu υπολογίζει το κατώφλι για το οποίο οι περιοχές διαχωρισμού έχουν την μέγιστη μεταξύ τους διακύμανση (between-class variance) [9]. Στη συνέχεια, αυτές οι περιοχές συγκρίνονται με τις αντίστοιχες του προηγούμενου βήματος και, ανάλογα με τα αποτελέσματα, η κατηγοριοποίηση τροποποιείται προκειμένου να επιλυθούν τυχόν διαφορές. Η διαδικασία επίλυσης των διαφορών αυτών, δηλαδή των διαφορών μεταξύ των κατηγοριοποιήσεων που προκύπτουν από τους αλγόριθμους Otsu και Kittler-Illingworth, βασίζεται σε ένα δένδρο αποφάσεων το οποίο παρουσιάζεται στο [3].
4. Οι μέθοδοι κατωφλίωσης που έχουν εφαρμοστεί μέχρι το σημείο αυτό βασίζονται στην κατανομή των τιμών των εικονοστοιχείων και δεν επηρεάζονται από την σχετική θέση αυτών. Προκειμένου, να συμπεριληφθεί στην κατηγοριοποίηση η πληροφορία αναφορικά με τη σχετική θέση των εικονοστοιχείων με υψηλό t-score, εφαρμόζεται μια μέθοδος επέκτασης περιοχής (region growing). Η περιοχή εκκίνησης της μεθόδου αποτελείται από το σύνολο των πλημμυρισμένων εικονοστοιχείων του προηγούμενου βήματος. Η μέθοδος επεκτείνει επαναληπτικά την πλημμυρισμένη περιοχή με την προσθήκη εικονοστοιχείων που συνορεύουν με πλημμυρισμένα εικονοστοιχεία και έχουν παρόμοια τιμή t-score με αυτά.

Σύμφωνα με το [3], η μέθοδος αυτή μειώνεται το πλήθος των πλημμυρισμένων εικονοστοιχείων που δεν έχουν εντοπιστεί.

5. Το τελευταίο βήμα του αλγορίθμου χαρτογράφησης που υλοποιεί το FLOODPY είναι με μία διαδικασία βελτίωσης. Κατά την διαδικασία αυτή, από το σύνολο των πλημμυρισμένων εικονοστοιχείων αφαιρούνται οι περιοχές, οι οποίες, σύμφωνα με τα διαθέσιμα ψηφιακά μοντέλα υψομέτρου (Digital Elevation Models), έχουν κλίση μεγαλύτερη των 15 μοιρών [3]. Επιπλέον, γίνονται διορθώσεις σε λεπτομέρειες που είναι μικρότερες από μια προκαθορισμένη ελάχιστη μονάδα χαρτογράφησης, ώστε να παραχθεί ο τελικός χάρτης των πλημμυρισμένων γεωγραφικών περιοχών.

## 2.2 Αποτελέσματα

Οι χαρτογραφήσεις που παράγει η παραπάνω μέθοδος είναι ιδιαίτερα ακριβείς και αξιόπιστες. Στην ανάλυση που παρουσιάζεται στο [3], οι χάρτες που παράγει το FLOODPY σε ένα πλήθος περιπτώσεων χρήσης συγκρίνονται με τα αντίστοιχα προϊόντα από το ευρωπαϊκό πρόγραμμα Emergency Management Service (EMS) [10]. Βάσει της σύγκρισης αυτής, επί του συνόλου των πλημμυρών που εξετάζονται, το ποσοστό των πλημμυρισμένων εικονοστοιχείων που κατηγοριοποιούνται σωστά και το Kappa score είναι υψηλότερα από 0.95 και 0.77 αντίστοιχα. Επιπρόσθετα, η εν λόγω αξιολόγηση δεν αναδεικνύει πλήρως την ποιότητα των αποτελεσμάτων του FLOODPY, διότι η ακρίβεια των προϊόντων του EMS στα οποία βασίζεται δεν είναι ομοιόμορφη σε όλες τις περιπτώσεις [3].

Ωστόσο, η αξιοπιστία των αποτελεσμάτων του FLOODPY προϋποθέτει ότι η περιοχή ενδιαφέροντος πληρεί ορισμένες συνθήκες. Πιο συγκεκριμένα, η μέθοδος του FLOODPY σχεδιάστηκε για την χαρτογράφηση πλημμυρών επί εδάφους που είτε δεν επικαλύπτεται, είτε επικαλύπτεται με χαμηλή βλάστηση. Στην περίπτωση που στην πλημμυρισμένη περιοχή υπάρχει υψηλή βλάστηση ή αστικός ιστός, τότε η χαρτογράφηση είναι ιδιαίτερα δύσκολη [3].

Περεταίρω, η ακρίβεια της χαρτογράφησης περιορίζεται από την ποιότητα των εικόνων του προγράμματος Sentinel-1 για την περιοχή ενδιαφέροντος κατά την χρονική περίοδο πριν την πλημμύρα. Αν οι εικόνες παρουσιάζουν μεγάλη μεταβλητότητα, όπως για παράδειγμα λόγω συχνών βροχοπτώσεων, η πιθανότητα σφάλματος στην χαρτογράφηση που παράγει το FLOODPY είναι αυξημένη.

## 2.3 Χρήση και Περιορισμοί

Για την εγκατάσταση του FLOODPY και τη χαρτογράφηση πλημμυρών απαιτούνται τεχνικές γνώσεις και επαρκής χρόνος. Πράγματι, ο ενδιαφερόμενος χρήστης πρέπει να κατεβάσει τον κώδικα του FLOODPY από το GitHub [11], να λάβει τις απαραίτητες βιβλιοθήκες Python, και να εγκαταστήσει το λογισμικό ESA-SNAP. Παράλληλα, πρέπει να δημιουργήσει λογαριασμούς για την λήψη κλιματικών δεδομένων από το Climate Data Store και εικόνων SAR από το πρόγραμμα Sentinel-1. Μετά την ολοκλήρωση αυτών των βημάτων, ο χρήστης καλείται να ανοίξει ένα Jupyter Notebook, να εισάγει τις παραμέτρους της πλημμύρας προς χαρτογράφηση και να εκτελέσει τα βήματα επεξεργασίας του FLOODPY. Για πραγματοποίηση της εν λόγω διαδικασίας απαιτείται πρόσβαση σε ένα περιβάλλον Linux και γνώσεις Python. Επιπρόσθετα, το σύστημα στο οποίο εκτελείται το FLOODPY πρέπει να διαθέτει επαρκείς πόρους, διότι η εργασία του FLOODPY μπορεί να δεσμεύσει σημαντική ποσότητα μνήμης.

Η παραπάνω διαδικασία καθιστά την πρόσβαση στις δυνατότητες του FLOODPY ιδιαίτερα δύσκολη για χρήστες που δεν διαθέτουν τεχνικές γνώσεις. Η ανάγκη για εγκατάσταση εξειδικευμένου λογισμικού, η κατανόηση εργαλείων όπως τα Jupyter Notebooks και η αλληλεπίδραση με τη γραμμή εντολών του Linux περιορίζουν η χρήση του FLOODPY σε άτομα με εμπειρία στον προγραμματισμό και στη διαχείριση υπολογιστών. Ως αποτέλεσμα, αποκλείονται δυνητικοί χρήστες, όπως τοπικές αρχές, ερευνητές χωρίς τεχνική κατάρτιση, ή απλοί πολίτες που ενδιαφέρονται για τη χαρτογράφηση πλημμυρών. Συνεπώς, η πρακτική εφαρμογή του FLOODPY περιορίζεται σημαντικά, καθώς η ευρεία χρήση του είναι εφικτή μόνο από οργανισμούς ή άτομα με υψηλό επίπεδο τεχνικών γνώσεων.

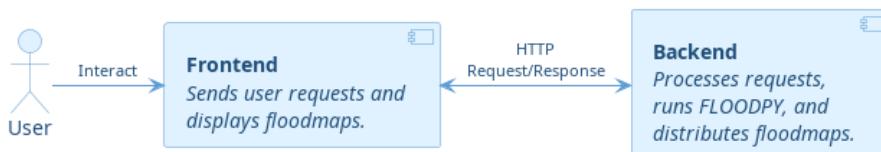
Οστόσο, ακόμα και τεχνικά καταρτισμένοι χρήστες αντιμετωπίζουν προκλήσεις κατά την πλήρη αξιοποίηση των δυνατοτήτων του FLOODPY. Δεδομένου ότι το εργαλείο εκτελείται εντός Jupyter Notebooks, η εισαγωγή παραμέτρων, όπως οι συντεταγμένες της περιοχής ενδιαφέροντος, απαιτεί επεξεργασία κώδικα Python. Αυτή η διαδικασία είναι πιο επιρρεπής σε σφάλματα σε σχέση με τη χρήση μιας απλής και φιλικής προς τους χρήστες διεπαφής. Επιπλέον, εφόσον δεν υπάρχει ενσωματωμένη δυνατότητα διαμοιρασμού των χαρτογραφήσεων που παράγονται, είναι εξαιρετικά πιθανό η ίδια πλημμύρα να χαρτογραφείται πολλές φορές από διαφορετικούς χρήστες. Οι περιορισμοί αυτοί, αν και δεν σχετίζονται με τη βασική

λειτουργικότητα ή τους άμεσα στόχους του FLOODPY, δεν επιτρέπουν την ανάδειξη των εξαιρετικών αποτελεσμάτων του και της συνολικής χρησιμότητά του.

### 3. Στόχοι και Χαρακτηριστικά της Web Εφαρμογής

Η web εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας έχει ως κύριο στόχο να επιτρέπει τη χαρτογράφηση πλημμυρών μέσω μιας εύχρηστης διεπαφής, απευθείας από το περιβάλλον του φυλλομετρητή. Για το σκοπό αυτό, το frontend της εφαρμογής διαθέτει μια φιλική διεπαφή που αποτελείται από στοιχεία, όπως φόρμες και διαδραστικοί χάρτες, μέσω των οποίων οι χρήστες παρέχουν τις απαραίτητες παραμέτρους για την χαρτογράφηση πλημμυρών. Το frontend υποβάλλει αυτές τις παραμέτρους στο backend της εφαρμογής, όπου για κάθε αίτημα δημιουργείται μια χαρτογράφηση. Επακολούθως, για την δημιουργία του περιεχομένου κάθε χαρτογράφησης, δηλαδή της απεικόνισης των πλημμυρισμένων περιοχών, εκτελείται η αντίστοιχη εργασία του FLOODPY. Με τον τρόπο αυτό, η εφαρμογή καθιστά τις δυνατότητες του FLOODPY διαθέσιμες σε όλες τις κατηγορίες χρηστών ανεξαρτήτως τεχνικών γνώσεων.

Εξίσου κρίσιμος στόχος της web εφαρμογής είναι να παρέχει στους χρήστες δυνατότητες προβολής και διαμοιρασμού των χαρτογραφήσεων που δημιουργούν. Μετά την ολοκλήρωση κάθε εργασίας του FLOODPY, το backend αποθηκεύει τα αποτελέσματα που παράγονται και τα διαθέτει στο frontend. Η γραφική διεπαφή του frontend περιλαμβάνει όλες τις απαραίτητες λειτουργίες για την παρουσίαση των αποθηκευμένων χαρτογραφήσεων, ενώ επιτρέπει σε κάθε χρήστη να προβάλλει οποιαδήποτε αποθηκευμένη χαρτογράφηση της εφαρμογής. Συνεπώς, κάθε χρήστης μπορεί να ελέγξει αν η πλημμύρα που τον ενδιαφέρει έχει ήδη χαρτογραφηθεί προτού αιτηθεί την εκτέλεση μιας νέας, κοστοβόρας, εργασίας του FLOODPY.



Εικόνα 3.1: Βασική δομή web εφαρμογής.

Προκειμένου η εφαρμογή να προσφέρει την καλύτερη δυνατή εμπειρία χρήστη, η διεπαφή του frontend σχεδιάστηκε με γνώμονα την απλότητα, την ομοιομορφία και τη δυναμικότητα. Πιο συγκεκριμένα, η διεπαφή παρουσιάζει χαρτογραφήσεις πλημμυρών και σχετικά δεδομένα με σαφή και κατανοητό τρόπο. Τα στοιχεία αλληλεπίδραση, όπως τα κουμπιά, οι φόρμες, και οι σύνδεσμοι της διεπαφής, είναι κομψά και επιτρέπουν την εύκολη και

αποτελεσματική εκτέλεση των επιθυμητών ενεργειών. Η συνολική εμφάνιση είναι συνεπής μεταξύ διαφορετικών σελίδων και προσαρμόζεται δυναμικά στο μέγεθος της οθόνης της συσκευής του χρήστη. Έτσι, εξασφαλίζεται άριστη αισθητική και χρηστικότητα τόσο σε κινητές συσκευές όσο και σε σταθερούς υπολογιστές.

Για την βέλτιστη υποστήριξη των λειτουργιών του frontend, οι υπηρεσίες του backend της εφαρμογής είναι ασφαλείς, αποδοτικές και επεκτάσιμες. Στόχος της υλοποίησης είναι να διασφαλιστεί ότι οι υπηρεσίες του backend έχουν μικρούς χρόνους απόκρισης και είναι ικανές να διαχειριστούν υψηλούς ρυθμούς αιτημάτων, χωρίς δυσανάλογες απαιτήσεις σε υπολογιστικούς πόρους. Επιπλέον, η σχεδίαση εξασφαλίζει την ασφάλεια των υπηρεσιών και των δεδομένων της εφαρμογής με τη χρήση σύγχρονων πρακτικών ασφαλείας. Παράλληλα, η αρχιτεκτονική του backend υποστηρίζει την εύκολη τροποποίηση ή την επέκταση των υπηρεσιών, καθώς και τη μελλοντική ενσωμάτωση αλλαγών στο FLOODPY ή ακόμα και την προσθήκη εναλλακτικών προγραμμάτων χαρτογράφησης.

## 4. Τεχνολογίες του Frontend

Πριν από την έναρξη της συστηματικής υλοποίηση της εφαρμογής, δημιουργήθηκε μια πειραματική έκδοση του frontend με ορισμένες βασικές λειτουργίες. Αρχικά, στην έκδοση αυτή χρησιμοποιήθηκαν μόνο οι γλώσσες HTML, CSS, και JavaScript. Ωστόσο, στην πορεία της ανάπτυξης ενσωματώθηκαν επιπλέον τεχνολογίες βάσει των απαιτήσεων της υλοποίησης. Ο στόχος ήταν η ανάδειξη τεχνικών προκλήσεων που αφορούν το frontend και η επιλογή των κατάλληλων βιβλιοθηκών, εργαλείων, και frameworks για την επίλυσή τους.

Οι δυσκολίες που εντοπίστηκαν κατά τη ανάπτυξη της πειραματικής έκδοσης του frontend είναι οι ακόλουθες:

1. Πολυπλοκότητα στην διαχείριση της κατάστασης της διεπαφής. Ως “κατάσταση” ορίζεται το σύνολο των μεταβλητών οι τιμές των οποίων καθορίζουν το περιεχόμενο και τη λειτουργικότητα της διεπαφής χρήστη. Η διαχείριση αυτών των μεταβλητών αποδείχθηκε ιδιαίτερα δύσκολη λόγω του πλήθους τους και της κεντροποιημένης οργάνωσης τους.
2. Μεγάλος όγκος κώδικα για την εφαρμογή αλλαγών στη διεπαφή χρήστη. Οι αλλαγές αυτές περιλάμβαναν τη λήψη, εμφάνιση, απόκρυψη, και τροποποίηση του περιεχομένου των στοιχείων HTML στις σελίδες του frontend. Παρόλο που ο σχετικός κώδικας αποτελούνταν κυρίως από περιπτώσεις χρήσης του Document Object Model (DOM), ο μεγάλος όγκος του αποτελούσε εμπόδιο για την ομαλή ανάπτυξη νέων λειτουργιών.
3. Δυσκολίες στον διαχωρισμό του κώδικα (HTML, JavaScript) σε μικρά και ανεξάρτητα τμήματα. Αυτό είχε ως αποτέλεσμα μια εν μέρει μονολιθική οργάνωση.
4. Έλλειψη ομοιομορφίας στη διεπαφή χρήστη και ασυνέπεια στην εμπειρία χρήσης.

Προκειμένου να επιλυθούν τα παραπάνω προβλήματα και να βελτιωθεί η ποιότητα της τελικής υλοποίησης, αξιοποιήθηκαν σύγχρονα εργαλεία και βιβλιοθήκες. Στη συνέχεια, περιγράφονται οι συγκεκριμένες τεχνολογίες που επιλέχθηκαν, καθώς και τα προβλήματα που αντιμετωπίστηκαν με τη χρήση κάθε μίας.

### 4.1 React

Τη μεγαλύτερη επιρροή στην ανάπτυξη του frontend της εφαρμογής είχε η χρήση της βιβλιοθήκης React. Η React είναι μια πολύ δημοφιλής βιβλιοθήκη της JavaScript [12] που διευκολύνει τη δημιουργία διαδραστικών διεπαφών χρήστη. Το βασικό χαρακτηριστικό της είναι

ότι ενθαρρύνει την διαίρεση της διεπαφής σε μια ιεραρχία διακριτών components, καθένα από τα οποία αποδίδει ένα συγκεκριμένο τμήμα της οθόνης [13]. Τα component της React είναι συναρτήσεις γραμμένες σε μια επέκταση της JavaScript που ονομάζεται JSX και συμπεριλαμβάνει σήμανση παρόμοια με την HTML [14]. Κάθε component διαθέτει τις δικές του μεταβλητές κατάστασης και επιστρέφει ένα τμήμα του περιεχομένου της διεπαφής χρήστη. Όταν η τιμή μια μεταβλητής κατάστασης αλλάζει, η συνάρτηση του component εκτελείται ξανά και το αντίστοιχο περιεχόμενο της διεπαφής ανανεώνεται αυτόματα, χωρίς την ανάγκη άμεσης πρόσβασης στο DOM.

Με τη χρήση της React επιλύθηκαν τρία από τα τέσσερα προβλήματα που εντοπίστηκαν στην πειραματική υλοποίηση. Το ζήτημα του διαχωρισμού του κώδικα σε ανεξάρτητα τμήματα αντιμετωπίστηκε μέσω της οργάνωση του κώδικα της διεπαφής σε React components. Επιπλέον, το πρόβλημα της έκτασης του κώδικα για την εφαρμογή αλλαγών στη διεπαφή (διαχείριση DOM) επιλύθηκε, καθώς η React ενημερώνει την διεπαφή αυτόματα με το περιεχόμενο που επιστρέφουν οι συναρτήσεις των component χωρίς να απαιτείται χρήση του DOM. Ταυτόχρονα, η πολυπλοκότητα διαχείρισης της κατάστασης της διεπαφής μειώθηκε σημαντικά με τη μεταφορά των μεταβλητών κατάστασης εντός των components που αφορούν. Με αυτόν τον τρόπο, το εύρος επιρροής (scope) των μεταβλητών κατάστασης περιορίστηκε και διευκολύνθηκαν η παρακολούθηση και η ενημέρωσή τους. Συνολικά, οι εν λόγω βελτιώσεις απλοποίησαν ουσιαστικά τη διαδικασία ανάπτυξης του frontend και ενίσχυσαν την επεκτασιμότητα και την συντηρησιμότητα της εφαρμογής.

## 4.2 React-Router

Παρά τα οφέλη που προσφέρει η React για την ανάπτυξη διεπαφών χρήστη, η αποδοτική πλοήγηση μεταξύ διαφορετικών σελίδων της εφαρμογής δεν είναι δυνατή χωρίς τη χρήση πρόσθετων βιβλιοθηκών. Σε απλές εφαρμογές, όταν ο χρήστης επιλέγει ένα σύνδεσμο, ο φυλλομετρητής στέλνει ένα HTTP GET αίτημα στον διακομιστή (web server) για τη λήψη των αρχείων που αντιστοιχούν στο URL του συνδέσμου. Αφού λάβει τα αρχεία, ο φυλλομετρητής εκτελεί τον κώδικα που περιέχουν και εμφανίζει την σελίδα που προκύπτει στην οθόνη. Η διαδικασία αυτή επαναλαμβάνεται κάθε φορά που ο χρήστης επισκέπτεται κάποια σελίδα [15].

Η βιβλιοθήκη React-Router βελτιώνει την διαδικασία πλοήγησης σε εφαρμογές που έχουν υλοποιηθεί με React. Όταν ο χρήστης επιλέγει ένα σύνδεσμο, η React-Router επιτρέπει

στο frontend να αλλάξει το URL και να εμφανίσει στην οθόνη μια σελίδα, χωρίς να στείλει απαραίτητα νέο αίτημα στον διακομιστή [15]. Αν η σελίδα του συνδέσμου είναι αποθηκευμένη στην μνήμη του φυλλομετρητή από προηγούμενη επίσκεψη, το frontend την εμφανίζει αμέσως και δεν αποστέλλει αίτημα στον διακομιστή. Εναλλακτικά, στην περίπτωση που η σελίδα του συνδέσμου απαιτεί δεδομένα που ανανεώνονται διαρκώς ή που δεν είναι άμεσα διαθέσιμα στο φυλλομετρητή, τότε παρουσιάζεται προσωρινά μια υποκατάστατη σελίδα η οποία, για παράδειγμα, υποδεικνύει ότι φορτώνονται πληροφορίες. Ταυτόχρονα, αποστέλλονται αιτήματα στο διακομιστή και όταν ληφθούν τα απαραίτητα δεδομένα εμφανίζεται η σελίδα που ζήτησε ο χρήστης. Εφόσον κατά την πλοήγηση στην εφαρμογή εμφανίζονται άμεσα σελίδες χωρίς να απαιτείται πάντα αναμονή για τη φόρτωση αρχείων και την εκτέλεση του περιεχομένου τους, η εμπειρία πλοήγησης του χρήστη βελτιώνεται αισθητά [15].

Εκτός της διαδικασίας πλοήγησης, η βιβλιοθήκη React-Router διευκολύνει επιπλέον την λήψη και την υποβολή δεδομένων. Μέσω της React-Router μπορούν να οριστούν συναρτήσεις φόρτωσης δεδομένων, οι οποίες εκτελούνται όταν ο χρήστης επισκέπτεται συγκεκριμένες σελίδες του frontend που απαιτούν δεδομένα από τον διακομιστή [16]. Αντίστοιχα, μπορούν να οριστούν συναρτήσεις υποβολής δεδομένων, οι οποίες εκτελούνται όταν ο χρήστης υποβάλει φόρμες από επιλεγμένες σελίδες [17]. Αυτές οι συναρτήσεις μπορούν επίσης να κληθούν δυναμικά από τον κώδικα των React components, επιτρέποντας την ευέλικτη ανταλλαγή δεδομένων με τον διακομιστή.

### 4.3 MUI

Η τελευταία δυσκολία που εντοπίστηκε στην πειραματική υλοποίηση, η οποία δεν έχει αναλυθεί, αφορά τη κατασκευή μιας ομοιόμορφης διεπαφής χρήστη. Η δημιουργία ενός συνόλου καλαίσθητων, ομοιόμορφων και διαδραστικών οπτικών στοιχείων (πχ. κουμπιά, μενού, πλαίσια, φόρμες) για την σύνθεση μιας ελκυστικής διεπαφής απαιτεί μη αμελητέα ποσότητα εργασίας. Ωστόσο, η διαδικασία σχεδίασης τέτοιων στοιχείων δεν αποτελεί τον βασικό στόχο της παρούσας εργασίας. Συνεπώς, για την δημιουργία της διεπαφής χρήστη της εφαρμογής χρησιμοποιήθηκε μια βιβλιοθήκη με προσχεδιαμένα στοιχεία, τα οποία παραμετροποιήθηκαν και προσαρμόστηκαν στις ανάγκες της εφαρμογής.

Η βιβλιοθήκη που χρησιμοποιήθηκε είναι η MUI (Material UI), η οποία προσφέρει μια εκτενή συλλογή προσχεδιασμένων React components βασισμένων στο συστήμα Material Design

της Google [18]. Τα στοιχεία της βιβλιοθήκης MUI είναι υψηλής ποιότητας, ευέλικτα και ιδανικά για την ανάπτυξη διαδραστικών διεπαφών χρήστη, ενώ ταιριάζουν και στο περιβάλλον κινητών συσκευών. Επιπλέον, η MUI ακολουθεί πρακτικές προσβασιμότητας, καθιστώντας τις διεπαφές που δημιουργούνται με αυτήν φιλικές προς όλους τους χρήστες [19].

## 4.4 NPM, Rollup, Vite

Προκειμένου να γίνει λήψη και προσθήκη εργαλείων και βιβλιοθηκών όπως οι React, React-Router και MUI στο frontend της εφαρμογής, χρησιμοποιήθηκε το NPM (Node Package Manager). Το NPM είναι ένα σύστημα διαχείρισης πακέτων της JavaScript. Αποτελείται από έναν iστότοπο, ένα μητρώο, και έναν client για την λήψη και την προσθήκη πακέτων σε εφαρμογές [20]. Όταν γίνεται λήψη μιας βιβλιοθήκης μέσω του NPM εντοπίζονται οι εξαρτήσεις της και διασφαλίζεται ότι οι εκδόσεις των πακέτων που θα ληφθούν είναι μεταξύ τους συμβατές. Ταυτόχρονα, το NPM προειδοποιεί για μη ασφαλείς ή παρωχημένες εκδόσεις πακέτων και προτείνει αναβαθμίσεις σε πιο ασφαλείς εκδόσεις.

Ωστόσο, η λήψη μιας βιβλιοθήκης μέσω του NPM δεν επαρκεί για την χρήση της στο frontend της εφαρμογής. Επί σειρά ετών, η JavaScript δεν διέθετε ενσωματωμένο τρόπο ώστε να συμπεριλαμβάνονται πολλαπλά αρχεία – όπως αυτά που περιέχει μια βιβλιοθήκη – σε μια εφαρμογή. Αυτή η δυνατότητα προστέθηκε με την εισαγωγή των ES modules στην αναθεώρηση ES6 της JavaScript επιτρέποντας τον διαμοιρασμό κώδικα μεταξύ αρχείων και την φόρτωσή τους από τον φυλλομετρητή όταν απαιτούνται [21]. Παρόλα αυτά, τα ES modules δεν είναι επαρκώς αποδοτικά, διότι η χρήση τους απαιτεί την αποστολή ενός ξεχωριστού HTTP αιτήματος στο διακομιστή για κάθε αρχείο που φορτώνεται από τον φυλλομετρητή [22].

Για την αντιμετώπιση του εν λόγω προβλήματος, κατά την ανάπτυξη του frontend χρησιμοποιήθηκε ένας module bundler που ονομάζεται Rollup. Το Rollup (και γενικά οι module bundlers) συγκεντρώνει τον κώδικα του frontend σε λίγα βελτιστοποιημένα αρχεία, συμπεριλαμβάνοντας τις εισαγόμενες βιβλιοθήκες και τις εξαρτήσεις τους, ώστε να μην απαιτούνται πολλαπλά HTTP αιτήματα στον διακομιστή. Από τα αρχεία που παράγονται, το Rollup αφαιρεί κώδικα που δεν είναι προσβάσιμος κατά την εκτέλεση [23]. Επιπρόσθετα, αφαιρεί περιεχόμενο που δεν είναι απαραίτητο για την εκτέλεση όπως σχόλια, περιττά κενά και newlines. Με τον τρόπο αυτό, ελαχιστοποιείται το μέγεθος των αρχείων που πρέπει να σταλούν από τον διακομιστή στον φυλλομετρητή και μειώνεται ο χρόνος φόρτωσης του frontend.

Η χρήση του Rollup, αν και είναι απαραίτητη για τη δημιουργία των τελικών βελτιστοποιημένων αρχείων που αποστέλλονται στους πελάτες από τον διακομιστή, παρουσιάζει ένα μειονέκτημα στο περιβάλλον ανάπτυξης. Καθώς ο όγκος του κώδικα του frontend αυξάνεται, η εκτέλεση του Rollup γίνεται χρονοβόρα. Ως εκ τούτου, ο χρόνος που απαιτείται για να εμφανιστεί στην εφαρμογή το αποτέλεσμα μιας αλλαγής στον κώδικα αυξάνεται [22]. Επομένως, επιβραδύνεται η υλοποίησης και επιβαρύνεται η εμπειρία ανάπτυξης.

Για να αποφευχθεί η καθυστέρηση που συνεπάγεται η επαναλαμβανόμενη εκτέλεση του Rollup κατά την ανάπτυξη, χρησιμοποιήθηκε το Vite. Το Vite αξιοποιεί τα ES modules για την εμφάνιση του frontend στο περιβάλλον ανάπτυξης [24]. Όπως αναφέρθηκε, τα ES modules εισάγουν καθυστερήσεις στην εφαρμογή, διότι απαιτούν την απόστολη ενός HTTP αιτήματος για την λήψη κάθε αρχείου κώδικα στο φυλλομετρητή. Ωστόσο, η ταχύτητα της εφαρμογής δεν έχει ιδιαίτερη σημασία κατά την ανάπτυξη, ο στόχος είναι η ελαχιστοποίηση του χρόνου ανανέωσης που περιεχομένου που παρουσιάζεται τον φυλλομετρητή μετά από μια αλλαγή στον κώδικα. Το Vite επιτυγχάνει αυτόν τον στόχο, φορτώνοντας αυτόματα μόνο τα αρχεία που έχουν τροποποιηθεί, χωρίς να χρειάζεται να μεσολαβήσει εκτέλεση του Rollup. Επιπλέον, για την ανανέωση του περιεχομένου δεν απαιτείται επαναφόρτωση της σελίδας στο φυλλομετρητή, καθώς το Vite χρησιμοποιεί έναν ενσωματωμένο διακομιστή ανάπτυξης που υποστηρίζει HMR (Hot Module Replacement) [25]. Μετά την ολοκλήρωση ενός κύκλου ανάπτυξης, το Vite παράγει τα βελτιστοποιημένα αρχεία για το περιβάλλον παραγωγής εκτελώντας το Rollup [24].

## 5. Λειτουργίες και Ανάπτυξη του Frontend

Δεδομένων των πολλαπλών στόχων της εφαρμογής, η ανάπτυξη του frontend οργανώθηκε σε ένα σύνολο ανεξάρτητων λειτουργιών, οι οποίες σχεδιάστηκαν και υλοποιήθηκαν ξεχωριστά. Με τον τρόπο αυτό, πραγματοποιήθηκε ακριβέστερη ανάλυση των απαιτήσεων κάθε λειτουργίας και έγινε καλύτερη διαχείριση της πολυπλοκότητας της υλοποίησης. Επιπλέον, η ενσωμάτωση των λειτουργιών έγινε σταδιακά διευκολύνοντας τον εντοπισμό σφαλμάτων και την συγγραφή τεστ.

Στο παρόν κεφάλαιο παρουσιάζονται οι λειτουργίες του frontend με τη σειρά που υλοποιήθηκαν. Η παρουσίαση ξεκινά με το user story της προκείμενης λειτουργίας, δηλαδή με μια άτυπη περιγραφή του επιθυμητού αποτελέσματος από την οπτική του χρήστη με έμφαση στην αξία που προσδίδει η λειτουργία στην εφαρμογή [26]. Στη συνέχεια, περιγράφονται το περιεχόμενο της διεπαφής και η αλληλεπίδραση του frontend με το backend. Όπου είναι απαραίτητο συμπεριλαμβάνονται και τεχνικές λεπτομέρειες, ώστε να δοθεί έμφαση σε ορισμένες προκλήσεις που προέκυψαν κατά την υλοποίηση.

### 5.1 Δημιουργία Νέας Χαρτογράφησης

Η βασικότερη λειτουργία της εφαρμογής είναι η δυνατότητα χαρτογράφησης νέων πλημμυρών. Προς αυτή την κατεύθυνση, το frontend παρέχει μια εύχρηστη διεπαφή που επιτρέπει την επιλογή της περιοχής ενδιαφέροντος, την εισαγωγή απαραίτητων παραμέτρων και την υποβολή του αιτήματος που προκύπτει το backend. Παρακάτω, παρουσιάζεται το αντίστοιχο user story, μαζί με τα κριτήρια αποδοχής που αφορούν την ανάπτυξη της λειτουργίας:

*Ως χρήστης της εφαρμογής θέλω να μπορώ να δημιουργώ νέες χαρτογραφήσεις πλημμυρών επιλέγοντας την περιοχή ενδιαφέροντος εντός διαδραστικού χάρτη.*

Κριτήρια Αποδοχής:

- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου εμφανίζεται ένας διαδραστικός παγκόσμιος χάρτης.
- Ο χρήστης μπορεί να επιλέξει την περιοχή ενδιαφέροντος χωρίς να χρειάζεται να εισάγει συντεταγμένες, αλλά έχει και αυτή την επιλογή.
- Ο χρήστης μπορεί να εισάγει όλες τις απαραίτητες παραμέτρους για την εκτέλεση της εργασία του FLOODPY εντός φόρμας.

➤ Η εφαρμογή καθοδηγεί το χρήστη επεξηγώντας τις απαραίτητες ενέργειες σε κάθε βήμα.

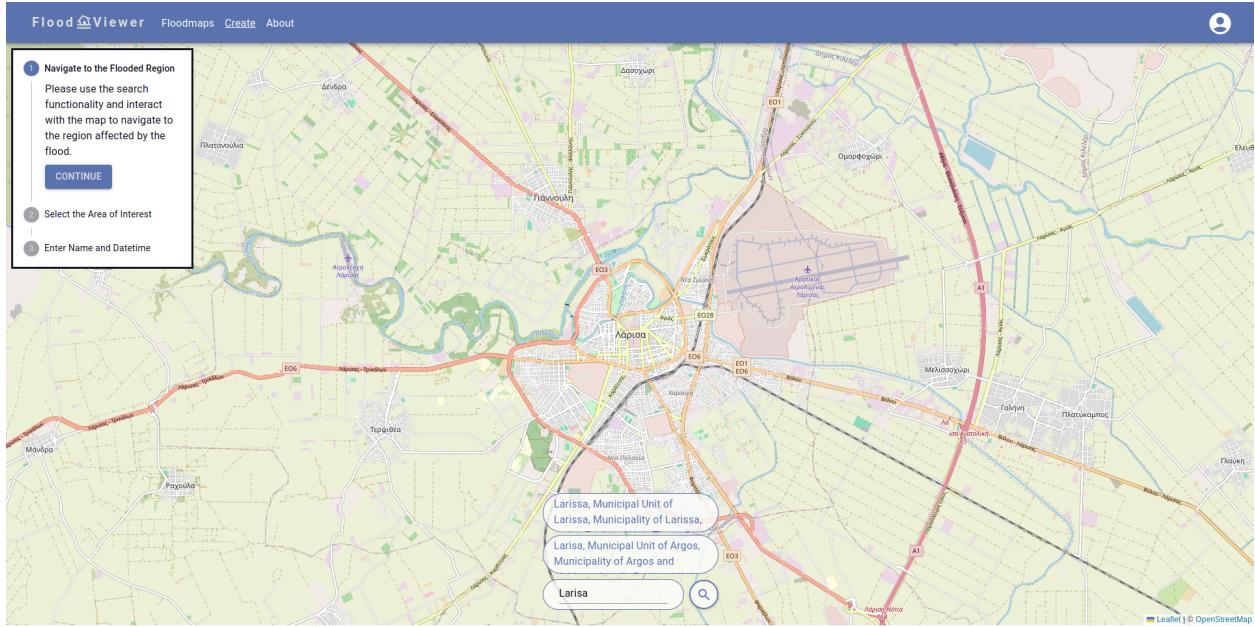
Για την ικανοποίηση των παραπάνω κριτηρίων, η εφαρμογή υλοποιεί στο μονοπάτι `/create` έναν wizard με τρία βήματα. Στη σελίδα του wizard εμφανίζεται ένας διαδραστικός χάρτης, καθώς και ένα πλαίσιο που παρέχει οδηγίες για τον χρήστη. Το πλαίσιο περιλαμβάνει επίσης κουμπιά για την μετάβαση στο επόμενο ή στο προηγούμενο βήμα.

Η προβολή του χάρτη υλοποιήθηκε με χρήση της βιβλιοθήκης React-Leaflet [27].

Ειδικότερα, τα δεδομένα του χάρτη λαμβάνονται με τη μορφή ψηφίδων (tiles) από τον οργανισμό OpenStreetMaps (OSM). Τα δεδομένα είναι ανοικτά, αρκεί να υπάρχει αναφορά ότι λήφθηκαν από τον OSM [28]. Η λήψη των ψηφίδων και ο συνδυασμός τους για την παρουσίαση του ολοκληρωμένου χάρτη πραγματοποιείται δυναμικά από την βιβλιοθήκη Leaflet. Πράγματι, καθώς ο χρήστης περιηγείται στο χάρτη, το Leaflet στέλνει αιτήματα στους διακομιστές του OSM για να λάβει τις απαραίτητες ψηφίδες, από τις οποίες συνθέτει την απεικόνιση της τοποθεσίας που προβάλλεται [29]. Η βιβλιοθήκη React-Leaflet προσαρμόζει το Leaflet για χρήση εντός της React.

Πάνω στον χάρτη ο wizard εμφανίζει ένα πλαίσιο με κουμπιά ελέγχου και οδηγίες για τον χρήστη. Το σχήμα, η θέση και το περιεχόμενο του πλαισίου προσαρμόζονται ανάλογα με το μέγεθος και τον προσανατολισμό της οθόνης στην οποία προβάλλεται η εφαρμογή. Ο στόχος είναι να ελαχιστοποιηθεί το τμήμα του χάρτη που επικαλύπτεται από το πλαίσιο, ώστε να μην εμποδίζεται η περιήγηση του χρήστη. Επιπλέον, το περιεχόμενο του πλαισίου ενημερώνεται ανάλογα με το ποιο από τα τρία βήματα του wizard είναι ενεργό και εμφανίζει κατάλληλες οδηγίες για κάθε φάση της διαδικασίας.

Στο πρώτο βήμα του wizard, ο χρήστης καλείται να περιηγηθεί εντός του χάρτη στην τοποθεσία της πλημμύρας προς χαρτογράφηση (Εικόνα 5.1). Με την φόρτωση της εφαρμογής, ο χάρτης εμφανίζει αρχικά μια προκαθορισμένη περιοχή (π.χ. την Αττική). Ακολουθώντας τις οδηγίες που αναγράφονται στο πλαίσιο, ο χρήστης πρέπει να εντοπίσει και να εστιάσει στην τοποθεσία όπου θα πραγματοποιηθεί η χαρτογράφηση. Για την διευκόλυνση της διαδικασία αυτής, ο wizard προσφέρει την δυνατότητα αναζήτησης περιοχών βάσει ονόματος. Ο χρήστης εισάγει το όνομα της επιθυμητής τοποθεσίας σε ένα πεδίο αναζήτησης και το frontend προτείνει περιοχές με ονόματα που ταιριάζουν σε αυτό. Αφότου έχει εντοπιστεί η επιθυμητή τοποθεσία, ο χρήστης μπορεί να πατήσει το κουμπί *CONTINUE* για να προχωρήσει στο επόμενο βήμα.



**Εικόνα 5.1:** Πρώτο βήμα του wizard.

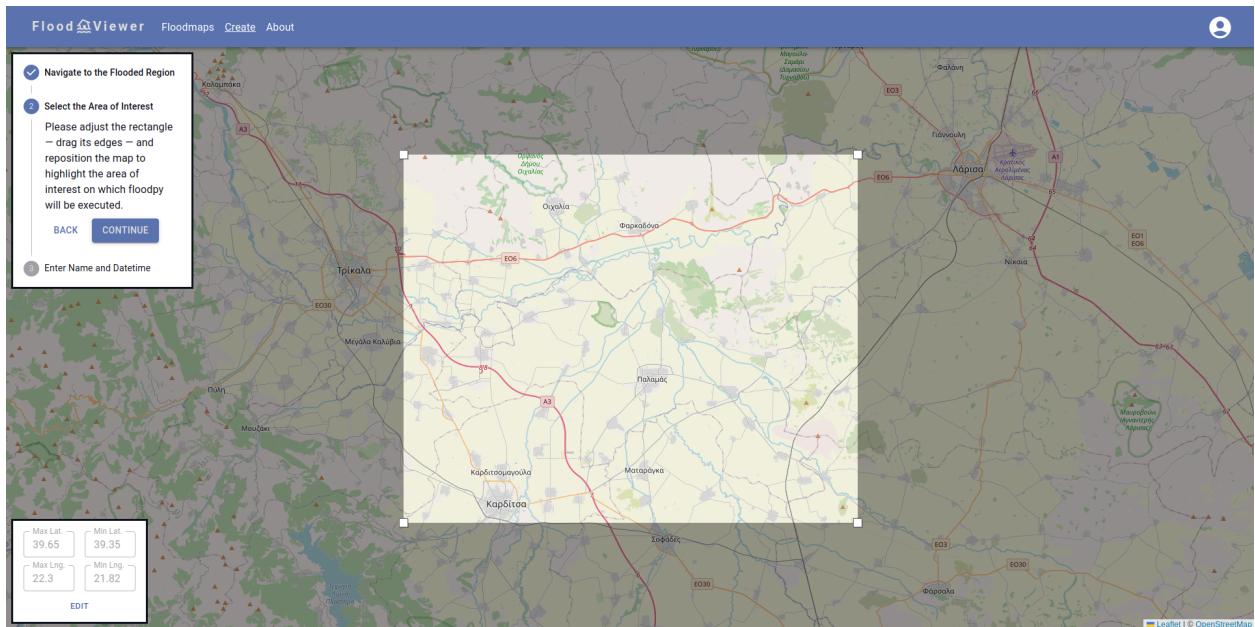
Η λειτουργία αναζήτησης τοποθεσιών υλοποιείται μέσω της υπηρεσίας geocoding του Nominatim [30]. Όταν ο χρήστης υποβάλλει το όνομα της τοποθεσίας προς αναζήτηση, το frontend αποστέλλει ένα HTTP GET αίτημα στην υπηρεσία του Nominatim που επιτρέπει την αναζήτηση τοποθεσιών. Το αίματα αυτό περιλαμβάνει ως παράμετρο το όνομα του εισήγαγε ο χρήστης (Παράδειγμα 5.1). Στην απάντηση, η υπηρεσία του Nominatim επιστρέφει αποτελέσματα σε μορφή JSON, τα οποία περιέχουν πληροφορίες όπως τα ονόματα και οι συντεταγμένες περιοχών που ταιριάζουν με το υποβληθέν όνομα. Τα ονόματα των περιοχών των αποτελεσμάτων εμφανίζονται στην διεπαφή του wizard και εφόσον ο χρήστης επιλέξει κάποιο από αυτά, ο χάρτης του Leaflet εστιάζει στις αντίστοιχες συντεταγμένες.

<https://nominatim.openstreetmap.org/search?format=json&q=Sydney>

**Παράδειγμα 5.1:** URI για την αναζήτηση περιοχών με όνομα “Sydney”.

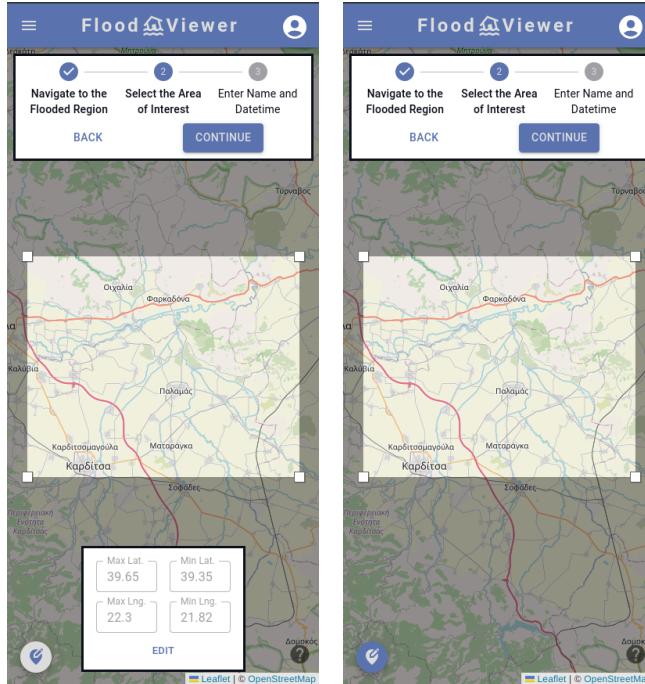
Το δεύτερο βήμα του wizard προτρέπει τον χρήστη να προσδιορίσει ακριβώς την περιοχή ενδιαφέροντος (Εικόνα 5.2). Για το σκοπό αυτό, εμφανίζεται στο κέντρο του χάρτη ένα παραλληλόγραμμο, το οποίο ορίζει ένα bounding box, δηλαδή μια περιοχή που καθορίζεται από δύο γεωγραφικά μήκη και πλάτη [31]. Το τμήμα του χάρτη που βρίσκεται εκτός του bounding box σκοτεινιάζει, ώστε να δοθεί έμφαση στην περιοχή ενδιαφέροντος που περικλείεται. Ο

χρήστης μπορεί εύκολα να επιλέξει την περιοχή ενδιαφέροντος μεταβάλλοντας το μέγεθος του παραλληλογράμμου. Αν η περιοχή που περικλείεται έχει πολύ μεγάλη έκταση, ο χρήστης ενημερώνεται και καλείται να τροποποιήσει το παραλληλόγραμμο. Αξίζει να σημειωθεί ότι η λειτουργία αυτή υλοποιήθηκε με χρήση της βιβλιοθήκης Leaflet AreaSelect, η οποία είναι διαθέσιμη στο GitHub [32].



**Εικόνα 5.2:** Δεύτερο βήμα του wizard.

Ωστόσο, ο προσδιορισμός της περιοχής ενδιαφέροντος με εποπτικό τρόπο δεν είναι πάντα ακριβής. Συνεπώς, στη σελίδα παρουσιάζεται ένα επιπλέον πλαίσιο, στο οποίο φαίνονται οι συντεταγμένες του επιλεγμένου bounding box. Ο χρήστης μπορεί να τροποποιήσει τις συντεταγμένες και το παραλληλόγραμμο που εμφανίζεται στο χάρτη προσαρμόζεται αυτόματα. Σε περίπτωση που η συσκευή του χρήστη έχει μικρή οθόνη (π.χ. κινητό τηλέφωνο), τότε το προαναφερόμενο πλαίσιο δεν φαίνεται παρά μόνο αν ο χρήστης πιέσει το κουμπί που το εμφανίζει (Εικόνα 5.3).



**Εικόνα 5.3:** Δεύτερο βήμα του wizard σε κινητό τηλέφωνο με (αριστερά) και χωρίς (δεξιά) το πλαίσιο επεξεργασίας των συντεταγμένων της περιοχής ενδιαφέροντος.

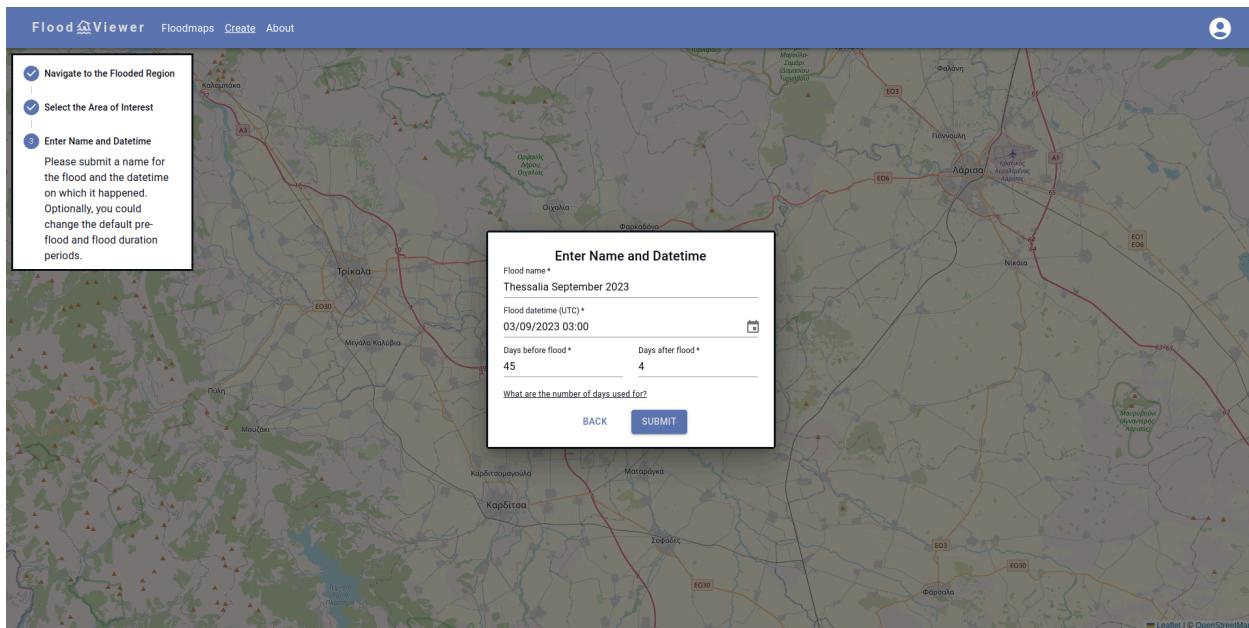
Αφότου η περιοχή ενδιαφέροντος έχει προσδιοριστεί, στο τρίτο και τελευταίο βήμα του wizard ο χρήστης καλείται να εισάγει απαραίτητες πληροφορίες σχετικά με την πλημμύρα (Εικόνα 5.4). Οι πληροφορίες αυτές είναι:

- Το όνομα της πλημμύρας που χρησιμοποιείται ως αναγνωριστικό της χαρτογράφησης.
- Η ημερομηνία της πλημμύρας.
- Οι χρονικές περίοδοι σε πλήθος ημερών εντός των οποίων λαμβάνονται εικόνες από το πρόγραμμα Sentinel-1 για την περιοχή ενδιαφέροντος πριν και μετά την πλημμύρα.

Για την συλλογή των πληροφοριών εμφανίζεται μία φόρμα με τα αντίστοιχα πεδία, η οποία καλύπτει και αδρανοποιεί το χάρτη. Παρόλα αυτά, ο χρήστης έχει τη δυνατότητα να επιστρέψει στα προηγούμενα βήματα και να τροποποιήσει την επιλεγμένη περιοχή ενδιαφέροντος.

Πριν ολοκληρωθεί η διαδικασία του wizard, πραγματοποιείται έλεγχος της ορθότητας των προς υποβολή δεδομένων και ο χρήστης καλείται να διορθώσει τυχόν σφάλματα.

Προφανώς, η ημερομηνία της πλημμύρας δεν μπορεί να είναι στο μέλλον ή πριν οι δορυφόροι του προγράμματος Sentinel τεθούν σε τροχιά. Παρόμοια, υπάρχουν περιορισμοί στην επιλογή του ονόματος της πλημμύρας, καθώς και όρια μέγιστης και ελάχιστης διάρκειας για τις χρονικές περιόδους λήψης των εικόνων της περιοχής ενδιαφέροντος.



**Εικόνα 5.4:** Τρίτο βήμα του wizard.

Οι έλεγχοι των στοιχείων που εισάγει ο χρήστης σε φόρμες όπως αυτή του τρίτου βήματος του wizard, γίνονται μέσω των βιβλιοθηκών Yup και React Hook Form. Η βιβλιοθήκη Yup επιτρέπει τον ορισμό κανόνων και απαιτήσεων που μπορούν να εφαρμοστούν σε δεδομένα [33]. Η React Hook Form απλοποιεί την υλοποίηση αποδοτικών φορμών στο περιβάλλον της React και χρησιμοποιεί τους κανόνες του Yup για την επικύρωση των δεδομένων τα οποία εισάγει ο χρήστης σε κάθε πεδίο [34]. Για παράδειγμα, ο κανόνας του Yup για το όνομα της πλημμύρας επαληθεύει συμβολοσειρές (strings) με μήκος το πολύ 40 χαρακτήρες που περιέχουν τουλάχιστον 3 μη κενούς χαρακτήρες. Ο κανόνας αυτός εφαρμόζεται μέσω της React Hook Form στην είσοδο του χρήστη στο αντίστοιχο πεδίο της φόρμας. Όπως θα φανεί στη συνέχεια, με ανάλογο τρόπο ελέγχονται όλα τα δεδομένα που εισάγει ο χρήστης σε όλες τις φόρμες της διεπαφής.

Στο τέλος του τρίτου βήματος του wizard, αν οι κανόνες ορθότητας του Yup ικανοποιούνται για όλα τα πεδία της φόρμας, το σύνολο των δεδομένων που έχουν συλλεχθεί υποβάλλονται στο backend. Πιο συγκεκριμένα, το frontend στέλνει ένα HTTP POST αίτημα στην υπηρεσία /api/floodmaps/ του backend που είναι υπεύθυνη για την διαχείριση χαρτογραφήσεων. Στο αίτημα αυτό περιέχονται τα δεδομένα που έχουν συλλεχθεί από τον

wizard σε μορφή JSON, όπως φαίνεται στο Παράδειγμα 5.2. Η επεξεργασία που πραγματοποιείται στην υπηρεσία του backend αναλύεται με λεπτομέρεια στο Κεφάλαιο 9.1.1.

```
{  
    "name": "Test Flood 1",  
    "bbox": {  
        "min_lat": 37.923,  
        "min_lng": 23.622,  
        "max_lat": 38.019,  
        "max_lng": 23.842  
    },  
    "flood_date": "2024-10-08T00:00:00Z",  
    "days_before_flood": 30,  
    "days_after_flood": 2,  
}
```

**Παράδειγμα 5.2:** JSON με τα δεδομένα που υποβάλλονται για την δημιουργία μιας νέας χαρτογράφησης.

Αν δεν συμβεί κάποιο σφάλμα, η απάντηση του backend έχει HTTP status κώδικα 201 Created. Στην περίπτωση αυτή, η εφαρμογή ανακατευθύνει το χρήστη σε μία νέα σελίδα όπου παρουσιάζεται το περιεχόμενο της χαρτογράφησης. Η σελίδα αυτή, η λειτουργία της οποίας είναι το αντικείμενο του επόμενου κεφαλαίου, προβάλλει πληροφορίες για την πλημμύρα που χαρτογραφείται, καθώς και για την εξέλιξη της σχετικής εργασίας του FLOODPY. Κάθε απάντηση με κώδικα διάφορο του 201 Created σηματοδοτεί ότι έχει συμβεί κάποιο σφάλμα στο backend. Το σφάλμα συμπεριλαμβάνεται στην απάντηση του backend και εμφανίζεται κάτω από την φόρμα του wizard, ενώ ο χρήστης καλείται να δοκιμάσει ξανά την υποβολή της φόρμας.

## 5.2 Παρουσίαση Περιεχομένου Χαρτογράφησης

Η δημιουργία μιας νέας χαρτογράφησης προσθέτει ελάχιστη αξία στην εφαρμογή αν οι χρήστες δεν έχουν πρόσβαση στο περιεχόμενό της. Επομένως, το frontend παρέχει την δυνατότητα προβολής του περιεχομένου μιας επιλεγμένης χαρτογράφησης, καθώς και της κατάστασης της εργασίας του FLOODPY που δημιουργεί το περιεχόμενο αυτό. Η εν λόγω λειτουργία περιγράφεται καλύτερα από το ακόλουθο user story:

*Ως χρήστης της εφαρμογής θέλω να μπορώ να βλέπω την χαρτογράφηση μιας πλημμύρας εντός διαδραστικού χάρτη. Αν το περιεχόμενο της χαρτογράφηση δεν είναι διαθέσιμο, διότι η εργασία του*

*FLOODPY* που το δημιουργεί δεν έχει ολοκληρωθεί ή έχει αποτύχει, τότε θέλω να λαμβάνω ενημερώσεις για την κατάσταση της εργασίας του *FLOODPY*.

Κριτήρια Αποδοχής:

- Ο χρήστης έχει πρόσβαση σε μια ξεχωριστή σελίδα για κάθε χαρτογράφηση, όπου εμφανίζονται πληροφορίες σχετικά με την πλημμύρα που χαρτογραφείται.
- Αν η εργασία του *FLOODPY* για μια χαρτογράφηση έχει ολοκληρωθεί με επιτυχία, τότε η σελίδα της χαρτογράφησης παρουσιάζει εντός διαδραστικού χάρτη τις πλημμυρισμένες περιοχές και όλα τα υπόλοιπα προϊόντα που έχουν παραχθεί.
- Ο χρήστης μπορεί να επιλέξει ποιά από τα προϊόντα της εργασίας του *FLOODPY* εμφανίζονται στο χάρτη κάθε στιγμή.
- Αν η εργασία του *FLOODPY* δεν έχει ολοκληρωθεί, τότε η σελίδα της χαρτογράφησης παρουσιάζει την πορεία της εργασίας σε πραγματικό χρόνο.
- Αν η εργασία του *FLOODPY* αποτύχει, τότε η σελίδα της χαρτογράφησης αναγράφει τον λόγο που οδήγησε στην αποτυχία της και το βήμα στο οποίο απέτυχε.

Προκειμένου να παρουσιαστεί μια χαρτογράφηση στην διεπαφή της εφαρμογής, πρώτα απαιτείται να γίνει λήψη των δεδομένων της. Όταν ο χρήστης επισκέπτεται (ή ανακατευθύνεται από τον wizard) το μονοπάτι /floodmaps/<id>, όπου <id> ο αναγνωριστικός αριθμός της χαρτογράφησης, το frontend στέλνει ένα HTTP GET αίτημα στην υπηρεσία /api/floodmaps/<id>/ του backend. Αν η χαρτογράφηση με αναγνωριστικό αριθμό <id> υπάρχει, το backend επιστρέφει ένα JSON με τα δεδομένα της. Παρακάτω φαίνεται ένα παράδειγμα του JSON που επιστρέφει το backend (Παράδειγμα 5.3). Το περιεχόμενο του JSON μπορεί να χωριστεί σε τρία τμήματα που περιλαμβάνουν:

- Τις πληροφορίες της πλημμύρας (name, bbox, flood\_date, days\_before\_flood, days\_after\_flood)
- Την κατάσταση της εργασία του *FLOODPY* (job).
- Τα προϊόντα της εργασία του *FLOODPY* (product).

```
{  
    "id": 8,  
    "name": "Test Flood",  
    "bbox": {  
        "min_lat": 37.923,  
        "min_lng": 23.622,
```

```

        "max_lat": 38.019,
        "max_lng": 23.842
    },
    "flood_date": "2024-11-04T00:00:00Z",
    "days_before_flood": 30,
    "days_after_flood": 2,
    "job": {
        "status": "Succeeded",
        "stage": "Completed",
        "posted_at": "2024-11-11T11:34:44.596666Z"
    },
    "product": {
        "built_at": "2024-11-11T11:57:50.075102Z",
        "geoserver_workspace": "floodmap_8",
        "esa_world_cover_layer": "esa_world_cover_layer_8",
        "s1_backscatter_layer": "s1_backscatter_layer_8",
        ...
    }
}

```

**Παράδειγμα 5.3:** JSON με τα δεδομένα μιας χαρτογράφησης.

Αφότου τα δεδομένα έχουν ληφθεί από το backend, το frontend εμφανίζει την σελίδα της χαρτογράφησης στην οποία, σε κάθε περίπτωση, παρουσιάζονται οι πληροφορίες της πλημμύρας. Ειδικότερα, παρουσιάζεται ένα πλαίσιο στο οποίο αναγράφονται το όνομα και η ημερομηνία της πλημμύρας, καθώς επίσης και οι συντεταγμένες του bounding box της περιοχής ενδιαφέροντος. Η θέση και το μέγεθος του πλαισίου εξαρτώνται από το μέγεθος και τον προσανατολισμό της οθόνης της συσκευής του χρήστη, ώστε να μην επικαλύπτονται άλλα στοιχεία της σελίδας.



**Εικόνα 5.5:** Πλαίσιο με πληροφορίες της πλημμύρας.

Εκτός του πλαισίου με τις βασικές πληροφορίες, το υπόλοιπο περιεχόμενο της σελίδας εξαρτάται από την κατάσταση της εργασίας του FLOODPY που δημιουργεί το περιεχόμενο της χαρτογράφησης. Σύμφωνα με τα κριτήρια αποδοχής, αν η εργασία του FLOODPY δεν έχει

ολοκληρωθεί, τότε η σελίδα πρέπει να εμφανίζει την τρέχουσα πρόοδο της. Αντίθετα, αν η εργασία έχει ολοκληρωθεί, η σελίδα πρέπει να παρουσιάζει τα παραγόμενα προϊόντα, επιτρέποντας στο χρήστη να ελέγχει ποια από αυτά εμφανίζονται στο χάρτη.

Για την υλοποίηση της πρώτης περίπτωσης, η εργασία του FLOODPY διαχωρίστηκε σε 9 στάδια. Τα στάδια που προέκυψαν είναι τα ακόλουθα:

1. Pending approval (Εκκρεμεί έγκριση)
2. Waiting in queue (Αναμονή στην ουρά)
3. Downloading precipitation data (Λήψη δεδομένων βροχόπτωσης)
4. Downloading Sentinel-1 images (Λήψη εικόνων Sentinel-1)
5. Preprocessing Sentinel-1 images (Προεπεξεργασία εικόνων Sentinel-1)
6. Performing statistical analysis (Εκτέλεση στατιστικής ανάλυσης)
7. Classifying floodwater (Κατηγοριοποίηση πλημμυρικών υδάτων)
8. Committing results (Καταχώρηση αποτελεσμάτων)
9. Completed (Ολοκληρώθηκε)

Τα στάδια 3 έως 7 αντιστοιχούν στα βήματα του αλγορίθμου του FLOODPY, ενώ η σημασία των σταδίων 1, 2 και 8 παρουσιάζεται σε επόμενα κεφάλαια (βλ. Κεφ. 5.6, 9.2). Γενικά, καθώς η εργασία προοδεύει, η κατάστασή της μεταβαίνει διαδοχικά από το ένα στάδιο στο επόμενο μέχρις ότου να ολοκληρωθεί ή να αποτύχει.

### 5.2.1 Ενημερώσεις Κατάστασης Εργασίας

Προκειμένου να παρουσιάζεται η πρόοδος μιας εργασίας του FLOODPY σε πραγματικό χρόνο, το frontend πρέπει να γνωρίζει κάθε στιγμή το στάδιο στο οποίο βρίσκεται η εν λόγω εργασία. Η λειτουργία αυτή μπορεί να υλοποιηθεί με τρεις τρόπους:

- Polling: το frontend στέλνει HTTP αιτήματα στο backend ανά τακτά χρονικά διαστήματα για να λάβει την πιο πρόσφατη τιμή του σταδίου στο οποίο βρίσκεται η εργασία. Η λύση αυτή απορρίφθηκε ως μη κλιμακώσιμη, καθώς ακόμα και ένας μικρός αριθμός πελατών μπορεί να υπερφορτώσει το backend με αιτήματα.
- Server-Sent Events: το frontend και το backend διατηρούν ανοικτό ένα HTTP αίτημα επιτρέποντας στο backend να στέλνει ενημερώσεις στο frontend οποιαδήποτε στιγμή [35]. Συνεπώς, το backend μπορεί να ενημερώνει το frontend κάθε φορά που η εργασία μεταβαίνει

σε νέο στάδιο. Ωστόσο, η επικοινωνία είναι μονόδρομη, καθώς στο πλαίσιο των server-sent events το frontend μπορεί μόνο να λάβει και όχι να στείλει μηνύματα στο backend.

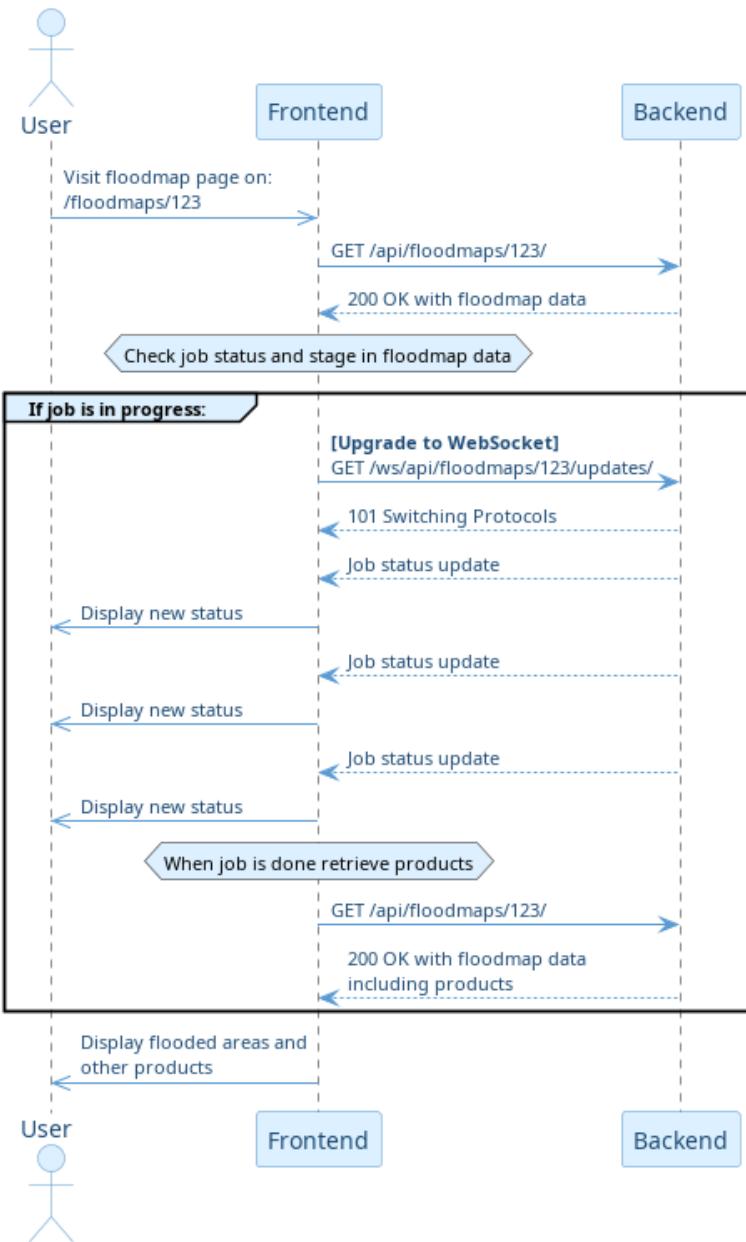
- WebSockets: το frontend και το backend χρησιμοποιούν το πρωτόκολλο WebSocket για να ανοίξουν έναν αμφίδρομο δίαυλο επικοινωνίας μεταξύ τους [36]. Με τον τρόπο αυτό, το backend μπορεί να στείλει στο frontend ενημερώσεις για κάθε αλλαγή του σταδίου της εργασίας.

Ανάμεσα στις λύσεις με Server-Sent Events και WebSockets, επιλέχθηκε η λύση με WebSockets, διότι επιτρέπει αμφίδρομη επικοινωνία μεταξύ frontend και backend. Στην παρούσα υλοποίηση η αμφίδρομη επικοινωνία δεν είναι απαραίτητη, καθώς οι εργασίες του FLOODPY δεν είναι διαδραστικές, δηλαδή δεν απαιτούν είσοδο από το χρήστη αφότου ξεκινήσουν. Όμως, αυτό ίσως αλλάξει στο μέλλον αν το FLOODPY εξελιχθεί ή αντικατασταθεί από κάποιο άλλο σύστημα χαρτογράφησης. Για να προετοιμαστεί η υλοποίηση της web εφαρμογής για την περίπτωση όπου η εργασία χαρτογράφησης απαιτεί είσοδο από το χρήστη κατά τη διάρκεια της εκτέλεσης, χρησιμοποιήθηκε το πρωτόκολλο WebSocket.

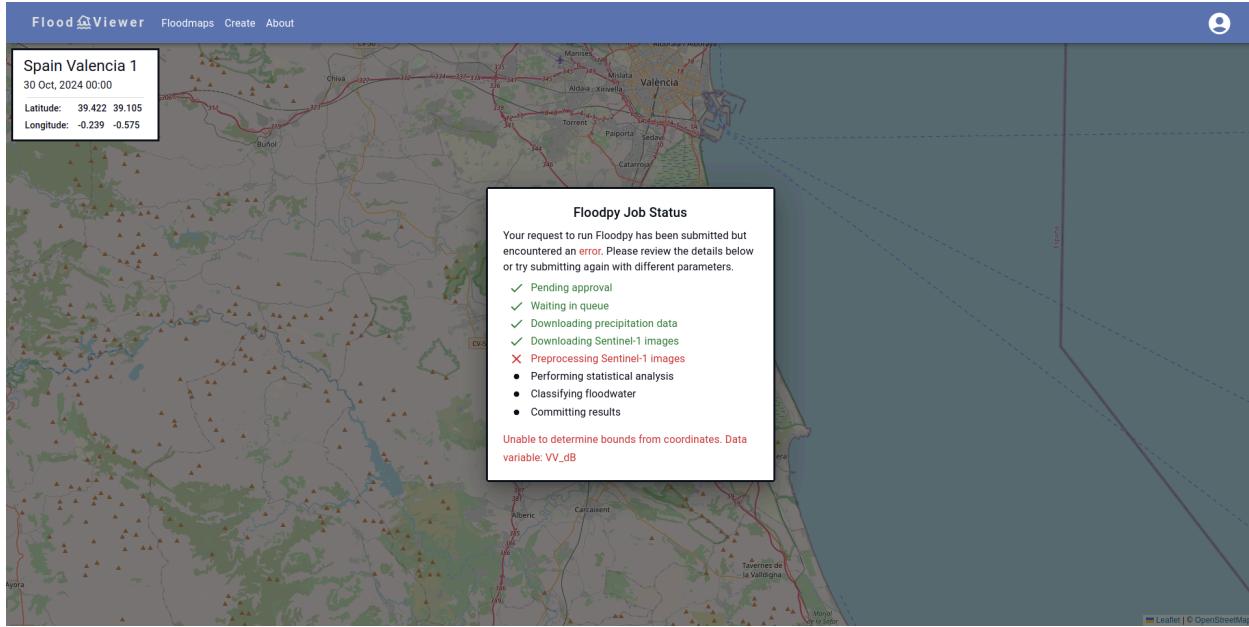
Στο παρακάτω διάγραμμα (Εικόνα 5.5) φαίνεται πως αλληλεπιδρούν ο χρήστης, το frontend και το backend κατά την εμφάνιση μιας αποθηκευμένης χαρτογράφησης. Αρχικά, το frontend λαμβάνει τα δεδομένα της χαρτογράφησης και ελέγχει την κατάσταση της εργασία του FLOODPY. Στην περίπτωση που η εργασία είναι σε εξέλιξη, το frontend ανοίγει μία σύνδεση WebSocket με το backend. Κάθε φορά που η κατάσταση (*Succeeded*, *Failed*, *Progressing*) ή το στάδιο (*Pending approval*, *Waiting in queue*, ...) της εργασίας μεταβάλλονται, το backend στέλνει ένα μήνυμα στο frontend μέσω της σύνδεσης WebSocket. Το περιεχόμενο του μηνύματος είναι ένα JSON με την τρέχουσα κατάσταση και το στάδιο της εργασίας. Όταν η εργασία ολοκληρωθεί ή αποτύχει η σύνδεση WebSocket κλείνει.

Η σελίδα της χαρτογράφησης χρησιμοποιεί τις ενημερώσεις από το backend για να εμφανίζει σε πραγματικό χρόνο το στάδιο και την κατάσταση της εργασίας του FLOODPY. Συγκεκριμένα, η σελίδα εμφανίζει ένα πλαίσιο που παρουσιάζει όλα τα στάδια από τα οποία πρέπει να περάσει η εργασία (Εικόνα 5.6). Κάθε στάδιο είναι χρωματισμένο και επισημειωμένο με ένα εικονίδιο ανάλογα με το αν έχει ολοκληρωθεί, έχει αποτύχει ή είναι σε εξέλιξη. Επιπλέον, το πλαίσιο περιλαμβάνει σχόλια για το τρέχον στάδιο, ενώ σε περίπτωση σφάλματος αναγράφει την αιτία που το προκάλεσε, η οποία αποστέλλεται στο frontend από το backend. Αν η εργασία

ολοκληρωθεί με επιτυχία, το αναφερόμενο πλαίσιο αφαιρείται από την σελίδα, ώστε να απεικονιστούν τα παραχθέντα προϊόντα του FLOODPY.



**Εικόνα 5.5:** Αλληλεπίδραση χρήστη, frontend και backend κατά την παρουσίαση μιας αποθηκευμένης χαρτογράφησης.



**Εικόνα 5.6:** Παρουσίαση χαρτογράφησης για την οποία η εργασία του FLOODPY απέτυχε στο βήμα προεπεξεργασίας των εικόνων του Sentinel-1.

### 5.2.2 Απεικόνιση Προϊόντων

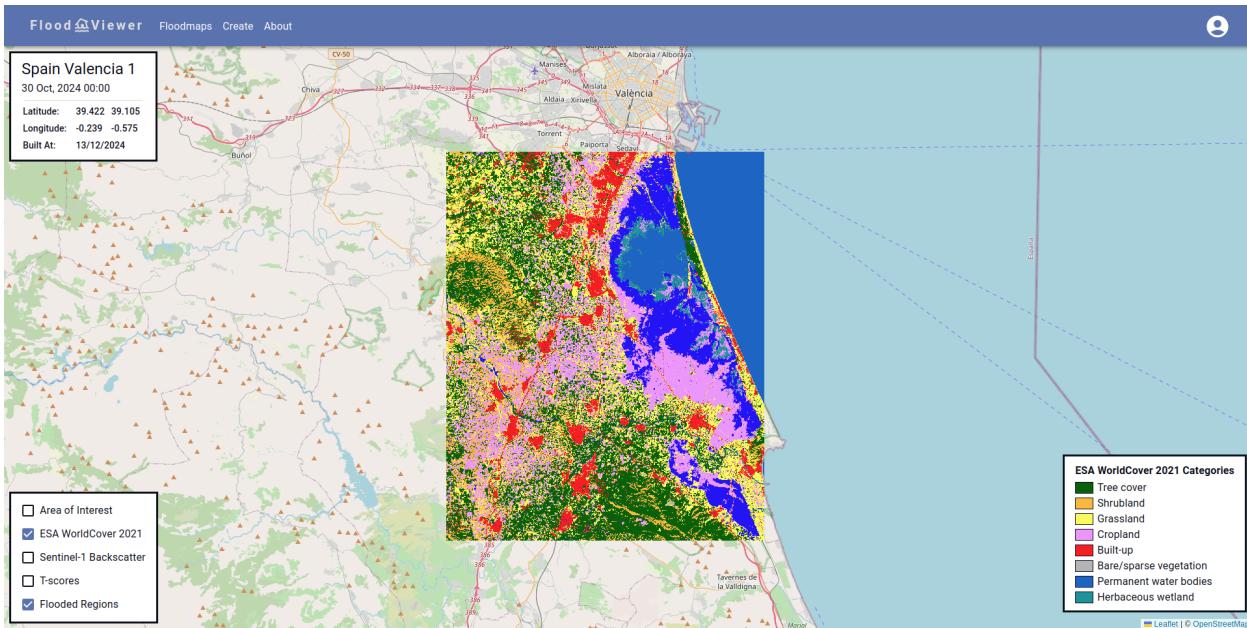
Στη σελίδα μιας χαρτογράφησης, για την οποία η εργασία του FLOODPY έχει ολοκληρωθεί επιτυχώς, παρουσιάζονται τα τελικά αποτελέσματα. Τα αποτελέσματα αυτά συμπεριλαμβάνονται στα δεδομένα της χαρτογράφησης που λαμβάνει το frontend από το backend και αποτελούν τα προϊόντα της εργασίας του FLOODPY. Τα προϊόντα αυτά είναι:

- Η απεικόνιση των περιοχών που έχουν πλημμυρίσει.
- Η απεικόνιση των αλλαγών του t-score.
- Η απεικόνιση των συντελεστών οπισθοσκέδασης (backscattering, VV πόλωση).
- Η απεικόνιση της κάλυψης του εδάφους σύμφωνα με τον χάρτη ESA WorldCover [37].

Όλες οι απεικονίσεις προβάλλονται εντος διαδραστικού χάρτη επί της περιοχής ενδιαφέροντος, ενώ ταυτόχρονα εμφανίζονται υπομνήματα όταν είναι απαραίτητο (Εικόνα 5.7). Η υλοποίηση του χάρτη έγινε με τη βιβλιοθήκη React-Leaflet, όπως περιγράφτηκε στο Κεφάλαιο 5.1.

Δεδομένου ότι οι παραπάνω απεικονίσεις αφορούν την ίδια περιοχή, αν εμφανιστούν όλες ταυτόχρονα αλληλοεπικαλύπτονται. Για το λόγο αυτό, η εφαρμογή επιτρέπει στο χρήστη να επιλέξει ποιες απεικονίσεις εμφανίζονται στο χάρτη κάθε στιγμή. Η σελίδα περιλαμβάνει ένα πλαίσιο ρυθμίσεων το οποίο περιέχει ένα checkbox για την εμφάνιση/απόκρυψη καθεμίας από

τις απεικονίσεις. Το πλαίσιο αυτό προσαρμόζεται (ή αποκρύπτεται) ανάλογα με το μέγεθος της οθόνης της συσκευής του χρήστη, ώστε να μην καλύπτει τις απεικονίσεις των προϊόντων.



**Εικόνα 5.7:** Παρουσίαση χαρτογράφησης για την οποία η εργασία του FLOODPY έχει ολοκληρωθεί με επιτυχία. Απεικονίζονται οι πλημμυρισμένες τοποθεσίες (βαθύ μπλε χρώμα) και η κάλυψη των εδάφων σύμφωνα με το χάρτη ESA WorldCover (υπόλοιπα χρώματα).

### 5.3 Εμφάνιση Συνόλου Αποθηκευμένων Χαρτογραφήσεων

Μια χαρτογράφηση είναι αρκετά πιθανό να ενδιαφέρει πολλούς χρήστες. Ως εκ τούτου, η εφαρμογή δίνει τη δυνατότητα σε όλους τους χρήστες να αναζητούν και να προβάλλουν χαρτογραφήσεις που έχουν δημιουργηθεί με επιτυχία και είναι αποθηκευμένες στην εφαρμογή. Το σχετικό user story έχει ως εξής:

Ως χρήστης της εφαρμογής θέλω να έχω πρόσβαση στο σύνολο των αποθηκευμένων χαρτογραφήσεων που έχουν δημιουργηθεί με επιτυχία είτε από εμένα είτε από άλλους χρήστες. Επιπλέον, θέλω να μπορώ να εκτελώ αναζητήσεις για εύκολη και γρήγορη εύρεση συγκεκριμένων χαρτογραφήσεων.

Κριτήρια Αποδοχής:

- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου εμφανίζονται οι αποθηκευμένες χαρτογραφήσεις της εφαρμογής που έχουν ολοκληρωθεί με επιτυχία. Η σελίδα δεν

περιλαμβάνει χαρτογραφήσεις για τις οποίες η εργασία του FLOODPY έχει αποτύχει ή είναι σε εξέλιξη.

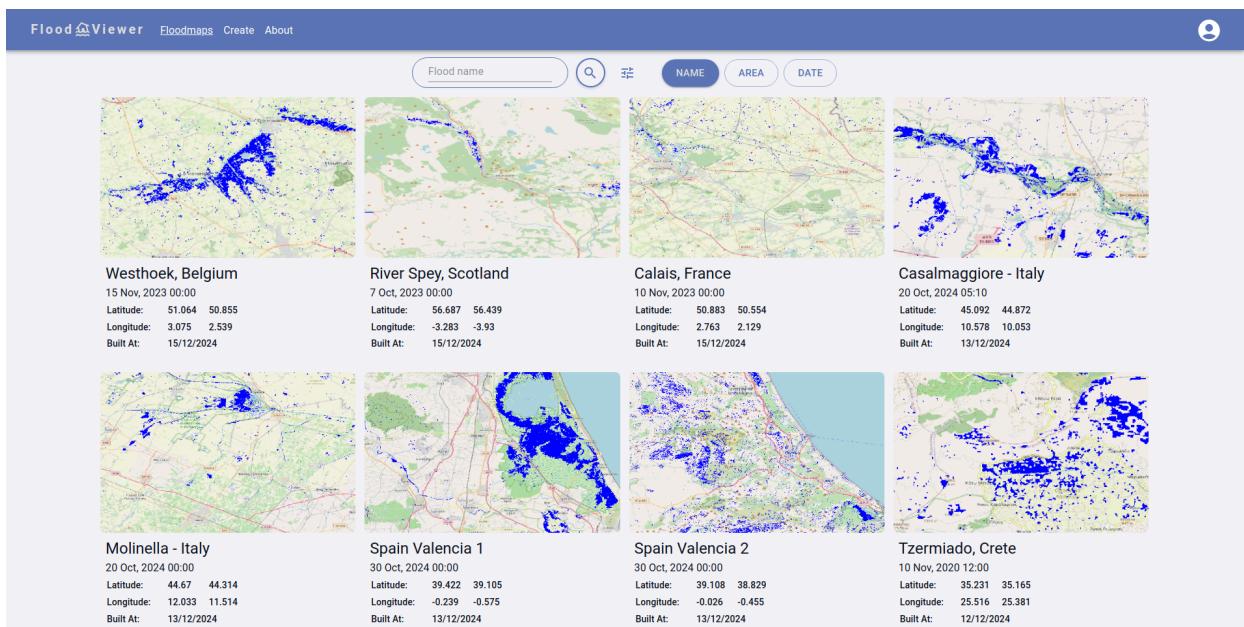
- Η εφαρμογή ταξινομεί τις χαρτογραφήσεις στη σελίδα κατά φθίνουσα σειρά με βάση την ημερομηνία που δημιουργήθηκαν, ώστε οι πιο πρόσφατα δημιουργημένες χαρτογραφήσεις να παρουσιάζονται πρώτες.
- Η εφαρμογή εμφανίζει για κάθε χαρτογράφηση στη σελίδα μια προεπισκόπηση με βασικές πληροφορίες και μια εικόνα των πλημμυρισμένων περιοχών.
- Ο χρήστης μπορεί να επιλέξει την προεπισκόπηση μιας χαρτογράφησης για να επισκεφτεί την σελίδα με την πλήρη παρουσίασή της.
- Ο χρήστης μπορεί να αναζητήσει στη σελίδα χαρτογραφήσεις με βάση το όνομα, τις συντεταγμένες του bounding box και την ημερομηνία της πλημμύρας.

Προκειμένου το frontend να εμφανίσει τις αποθηκευμένες χαρτογραφήσεις πρέπει να λάβει τα αντίστοιχα δεδομένα από το backend. Επομένως, όταν ο χρήστης επισκέπτεται το μονοπάτι `/floodmaps`, το οποίο αντιστοιχεί στη σελίδα που παρουσιάζονται οι αποθηκευμένες χαρτογραφήσεις, το frontend στέλνει ένα HTTP GET αίτημα στην υπηρεσία `/api/floodmaps/` του backend. Η απάντηση περιλαμβάνει ένα JSON με τα δεδομένα των χαρτογραφήσεων σε σελιδοποιημένη μορφή. Όπως φαίνεται στο Παράδειγμα 5.4, τα δεδομένα που λαμβάνει το frontend περιέχουν μια λίστα εμφωλευμένων JSON (πεδίο `results`) καθένα από τα οποία αντιστοιχεί σε μια χαρτογράφηση.

```
{  
    "count": 16,  
    "next": "http://127.0.0.1:8000/api/floodmaps/?page=2",  
    "previous": null,  
    "results": [  
        {  
            "id": 1,  
            "name": "Test Flood 1",  
            ...  
        },  
        {  
            "id": 2,  
            "name": "Test Flood 2",  
            ...  
        },  
        ...  
    ]  
}
```

#### Παράδειγμα 5.4: JSON με σελιδοποιημένα δεδομένα αποθηκευμένων χαρτογραφήσεων.

Από τα δεδομένα που λαμβάνονται δημιουργείται στη σελίδα μια κάρτα για κάθε χαρτογράφηση. Κάθε κάρτα παρέχει μια προεπισκόπηση της αντίστοιχης χαρτογράφησης, παρουσιάζοντας μια εικόνα των πλημμυρισμένων περιοχών, καθώς και βασικές πληροφορίες, όπως το όνομα της χαρτογράφησης, την ημερομηνία της πλημμύρας και τις συντεταγμένες του bounding box (Εικόνα 5.8). Οι κάρτες οργανώνονται σε ένα πλέγμα (grid) κάθε σειρά του οποίου περιέχει από μία έως τέσσερις κάρτες ανάλογα με το μέγεθος της οθόνης της συσκευής του χρήστη. Προκειμένου να περιοριστεί ο χρόνος φόρτωσης της σελίδας, το πλέγμα δεν μπορεί να διαθέτει περισσότερες από 12 κάρτες χαρτογραφήσεων.



Εικόνα 5.8: Εμφάνιση αποθηκευμένων χαρτογραφήσεων ως πλέγμα καρτών.

Αν στην εφαρμογή έχουν αποθηκευτεί περισσότερες από 12 χαρτογραφήσεις, τα αποτελέσματα οργανώνονται σε σελίδες. Ειδικότερα, το JSON που λαμβάνει το frontend περιέχει δεδομένα για έως 12 χαρτογραφήσεις, καθώς και ένα πεδίο που αναφέρει τον συνολικό αριθμό των αποθηκευμένων χαρτογραφήσεων (Παράδειγμα 5.4, πεδίο count). Κατά την σελιδοποίηση των αποτελεσμάτων, παρουσιάζονται στη σελίδα οι κάρτες με τα δεδομένα των 12 χαρτογραφήσεων που έχουν ληφθεί. Παράλληλα, επισημαίνεται ότι τα συνολικά αποτελέσματα περιλαμβάνουν αριθμό σελίδων που ισούται με:

## Γπλήθος αποθηκευμένων χαρτογραφήσεων / 121

Για παράδειγμα, αν η εφαρμογή έχει 45 αποθηκευμένες χαρτογραφήσεις, εμφανίζονται κάρτες για τις πρώτες 12 και αναγράφεται ότι υπάρχουν συνολικά 4 σελίδες αποτελεσμάτων:

$$45 / 121 = 3,75 \approx 4.$$

Ο χρήστης έχει τη δυνατότητα να πλοηγηθεί στις σελίδες των αποτελεσμάτων, επιλέγοντας τον αριθμό της σελίδας που επιθυμεί να επισκεφθεί. Κατά την επιλογή σελίδας, το μονοπάτι στο frontend ενημερώνεται ώστε να περιλαμβάνει τον αριθμό της σελίδας ως παράμετρο. Επιπλέον, το frontend λαμβάνει τα δεδομένα από το backend στέλνοντας νέο HTTP GET αίτημα, το οποίο περιλαμβάνει τον αριθμό της σελίδας ως παράμετρο. Ενδεικτικά, αν ο χρήστης επιλέξει τη σελίδα 2, επισκέπτεται το μονοπάτι `/floodmaps?page=2` και το frontend στέλνει HTTP GET αίτημα στο `/api/floodmaps/?page=2` του backend για να λάβει τα δεδομένα.

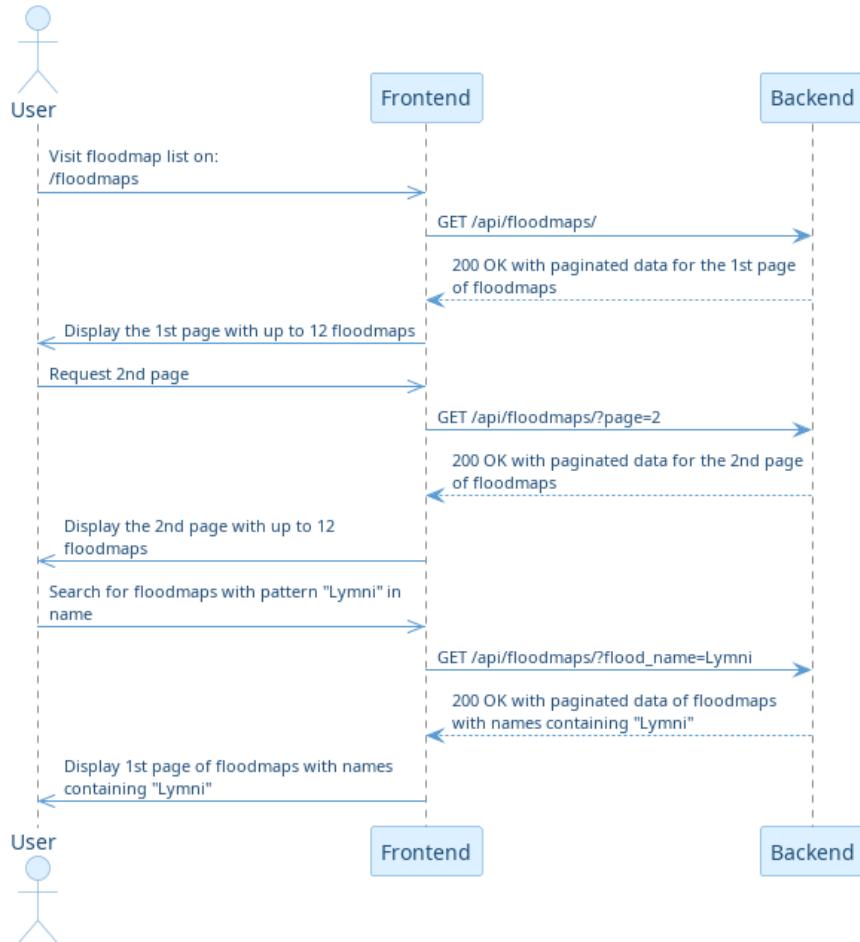
Για την περίπτωση που το πλήθος των αποθηκευμένων χαρτογραφήσεων είναι μεγάλο, η σελίδα διαθέτει μια φόρμα αναζήτησης. Ο χρήστης μπορεί να χρησιμοποιήσει την φόρμα για να περιορίσει τις χαρτογραφήσεις που εμφανίζονται στη σελίδα, εφαρμόζοντας συνδυασμούς των εξής φίλτρων:

- Το όνομα της χαρτογράφησης πρέπει να περιέχει συγκεκριμένη συμβολοσειρά.
- Η περιοχή ενδιαφέροντος πρέπει να είναι εντός καθορισμένων μέγιστων και ελάχιστων συντεταγμένων.
- Η ημερομηνία της πλημμύρας πρέπει να εμπίπτει σε συγκεκριμένη χρονική περίοδο.

Με τον τρόπο αυτό, ο χρήστης μπορεί εύκολα να διαπιστώσει αν η πλημμύρα που τον ενδιαφέρει έχει ήδη χαρτογραφηθεί προτού ξεκινήσει την δημιουργία νέας χαρτογράφησης.

Οι υλοποίηση των φίλτρων στο frontend ακολουθεί παρόμοια λογική με αυτή της επιλογής σελίδας. Πράγματι, όπως ο αριθμός της σελίδας, έτσι και οι τιμές των φίλτρων χρησιμοποιούνται ως παράμετροι τόσο στο μονοπάτι της σελίδας όσο και στο αίτημα του frontend προς το backend. Όπως φαίνεται στο παρακάτω διάγραμμα (Εικόνα 5.9), στην περίπτωση που ο χρήστης αναζητήσει χαρτογραφήσεις με ονόματα που περιέχουν την λέξη “Lymni”, το μονοπάτι της σελίδας γίνεται `/floodmaps?flood_name=Lymni`. Στη συνέχεια, το frontend στέλνει ένα HTTP GET αίτημα στο `/api/floodmaps/?flood_name=Lymni` του backend για να λάβει τα δεδομένα των αντίστοιχων χαρτογραφήσεων. Πολλαπλά φίλτρα

μπορούν να συνδυαστούν επιτρέποντας πιο σύνθετες και ακριβείς αναζητήσεις, όπως φαίνεται στο Παράδειγμα 5.5.



**Εικόνα 5.9:** Άλληλεπίδραση χρήστη, frontend και backend κατά την εμφάνιση αποθηκευμένων χαρτογραφήσεων. Περιλαμβάνει τις διαδικασίες σελιδοποίησης αποτελεσμάτων και εφαρμογής φίλτρων.

/floodmaps?flood\_name=Lymni&max\_lat=30&min\_lat=10&page=2

**Παράδειγμα 5.5:** Παράμετροι αναζήτησης κατά το συνδυασμό φίλτρων.

## 5.4 Εγγραφή και Αυθεντικοποίηση

Δεδομένου ότι οι χρήστες μπορούν να προσθέτουν περιεχόμενο στην εφαρμογή δημιουργώντας χαρτογραφήσεις, είναι σημαντικό η εφαρμογή να επιτρέπει την εγγραφή των χρηστών και την συσχέτισή τους με το περιεχόμενο που δημιουργούν. Μέσω αυτής της συσχέτισης, γίνεται εφικτό κάθε χρήστης να έχει πρόσβαση σε προσωπικό περιεχόμενο. Για το

σκοπό αυτό, η εφαρμογή παρέχει τη δυνατότητα δημιουργίας και διαχείρισης λογαριασμών χρηστών, εξασφαλίζοντας ταυτόχρονα ασφαλή αυθεντικοποίηση και σύνδεση στην εφαρμογή. Το σχετικό user story διατυπώνεται ως εξής:

*Ως χρήστης της εφαρμογής θέλω να μπορώ να δημιουργήσω ένα λογαριασμό με τον οποίο να συνδέομαι στην εφαρμογή. Επιπλέον, θέλω να μπορώ να αλλάξω ή να ανακτήσω τον κωδικό πρόσβασης που έχω ορίσει, καθώς και να διαγράψω οριστικά τον λογαριασμό μου.*

### Κριτήρια Αποδοχής

- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου μπορεί να δημιουργήσει ένα νέο λογαριασμό εισάγωντας διεύθυνση ηλεκτρονικού ταχυδρομείου και κωδικό πρόσβασης.
- Η εφαρμογή επιβεβαιώνει την διεύθυνση ηλεκτρονικού ταχυδρομείου πριν την δημιουργία νέου λογαριασμού.
- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου μπορεί να ανακτήσει τον κωδικό πρόσβασης του λογαριασμού του.
- Ο χρήστης έχει πρόσβαση σε μια σελίδα μέσω της οποίας μπορεί να συνδεθεί στην εφαρμογή εισάγοντας την διεύθυνση ηλεκτρονικού ταχυδρομείου και τον κωδικό πρόσβασης του λογαριασμού του.
- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου μπορεί να αλλάξει τον κωδικό πρόσβασης που έχει ορίσει.
- Ο χρήστης έχει πρόσβαση σε μια σελίδα όπου μπορεί να διαγράψει οριστικά τον λογαριασμό του.

#### 5.4.1 Δημιουργία Λογαριασμού

Η διαδικασία αυθεντικοποίησης ξεκινά με τη δημιουργία λογαριασμού. Η εφαρμογή διαθέτει στο μονοπάτι /sign up μια φόρμα στην οποία ο χρήστης καλείται να εισάγει μία διεύθυνση ηλεκτρονικού ταχυδρομείου (email) και ένα κωδικό πρόσβασης (Εικόνα 5.10, αριστερά). Τα δεδομένα που εισάγει ο χρήστης ελέγχονται προτού υποβληθούν, μέσω των βιβλιοθηκών React Hook Form και Yup, με τον τρόπο που περιγράφτηκε στο Κεφάλαιο 5.1. Το email πρέπει να ακολουθεί τη γνωστή μορφή, ενώ ο κωδικός πρόσβασης, προκειμένου να είναι ασφαλής, πρέπει να αποτελείται από τουλάχιστον 8 χαρακτήρες και να περιλαμβάνει 1 κεφαλαίο γράμμα, 1 μικρό γράμμα και 1 αριθμό. Εφόσον τα δεδομένα είναι έγκυρα, υποβάλλονται στο backend μέσω HTTP POST αιτήματος στην υπηρεσία /auth/users/. Στην περίπτωση που το

backend απαντήσει με σφάλμα (π.χ. HTTP status κώδικα 400) το περιεχόμενο του σφάλματος παρουσιάζεται στο χρήστη (π.χ. "Email already in use"). Αν δεν προκύψει σφάλμα, ο χρήστης καλείται να ελέγξει τα email που του έχουν αποσταλεί.

The image contains three side-by-side screenshots of a web application interface for 'Flood Viewer'.  
 1. **Sign up:** A form with fields for 'Email \*', 'Password \*', and 'Confirm Password \*'. Below the fields is a blue 'SIGN UP' button. At the bottom, it says 'Already have an account? [Log in](#)'.  
 2. **Log in:** A form with fields for 'Email \*' and 'Password \*'. Below the fields is a blue 'LOG IN' button. To the right, it says 'Forgot password?' and 'Don't have an account? [Sign up](#)'.  
 3. **Password reset:** A form with a single field for 'Email \*'. Below the field are two buttons: 'CANCEL' and 'SEND EMAIL'. Above the field, it says 'Please enter your registered email to receive a link to reset your password.'

**Εικόνα 5.10:** Σελίδες δημιουργίας λογαριασμού (αριστερά), αυθεντικοποίησης (κέντρο) και ανάκτησης κωδικού πρόσβασης (δεξιά) σε κινητό τηλέφωνο.

Για την ολοκλήρωση της διαδικασίας εγγραφής, η διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη πρέπει να επαληθευτεί. Μετά την επιτυχή υποβολή των δεδομένων εγγραφής, το backend αποστέλλει στο email του χρήστη έναν σύνδεσμο της μορφής `/activation/<UID>/<token>`, όπου τα `<UID>` και `<token>` είναι αναγνωριστικά του αιτήματος εγγραφής. Όταν ο χρήστης ακολουθήσει τον σύνδεσμο, η σελίδα του frontend που φορτώνεται στέλνει ένα HTTP POST αίτημα στην υπηρεσία `/auth/users/activation/` του backend, συμπεριλαμβάνοντας ένα JSON με το UID και το token του συνδέσμου. Το backend ελέγχει τα αναγνωριστικά UID και token και, εφόσον είναι ορθά, ολοκληρώνει την εγγραφή του χρήστη. Αν η διαδικασία είναι επιτυχής, το backend απαντά με HTTP status κώδικα 204 (No Content) και το frontend τον ανακατευθύνει το χρήστη στη σελίδα αυθεντικοποίησης (Εικόνα 5.10, κέντρο).

#### 5.4.2 Αυθεντικοποίηση

Εφόσον ο χρήστης διαθέτει ένα λογαριασμό, μπορεί να συνδεθεί στην εφαρμογή. Η εφαρμογή παρουσιάζει στο μονοπάτι `/login` μία φόρμα αυθεντικοποίησης, η υποβολή της οποίας απαιτεί την εισαγωγή της διεύθυνσης email και του κωδικού πρόσβασης του χρήστη. Τα στοιχεία αυτά αποστέλλονται από το frontend στην υπηρεσία αυθεντικοποίησης `/auth/users/login/` του backend. Αν τα στοιχεία δεν είναι έγκυρα, το backend απαντά με HTTP status κώδικα 401 (Unauthorized) και το frontend εμφανίζει στην φόρμα ένα μήνυμα που ενημερώνει το χρήστη ότι η αυθεντικοποίηση απέτυχε. Διαφορετικά, ο χρήστης ταυτοποιείται με επιτυχία και το frontend φορτώνει τη σελίδα `/floodmaps`, όπου παρουσιάζονται οι αποθηκευμένες χαρτογραφήσεις.

Οστόσο, στην περίπτωση επιτυχούς αυθεντικοποίησης, η εφαρμογή πρέπει επιπλέον να διατηρήσει την ταυτότητα του χρήστη μέχρι την αποσύνδεσή του. Για το σκοπό αυτό χρησιμοποιούνται JSON Web Tokens (JWTs).

Τα JWTs είναι συμβολοσειρές που ακολουθούν το ομώνυμο πρότυπο, το οποίο ορίζει μια μέθοδο ασφαλούς μετάδοσης πληροφοριών σε μορφή αντικειμένων JSON [38]. Κάθε JWT αποτελείται από τρία μέρη: το Header, το Payload και το Signature, τα οποία κωδικοποιούνται σε μορφή Base64URL και διαχωρίζονται με τελείες (Εικόνα 5.11). Το Header είναι ένα JSON που καθορίζει το είδος του token (JWT) και τον αλγόριθμο υπογραφής που χρησιμοποιείται (HS256). Το Payload είναι επίσης ένα JSON, το οποίο περιέχει τα δεδομένα προς μετάδοση. Στην παρούσα εφαρμογή τα δεδομένα αυτά είναι τα ακόλουθα:

- `token_type`: ο τύπος του token (access ή refresh)
- `exp`: η ημερομηνία λήξης του token
- `iat`: η ημερομηνία έκδοσης του token
- `jti`: ο κωδικός ταυτότητας του token
- `user_id`: ο κωδικός ταυτότητας του χρήστη για τον οποίο εκδόθηκε το token.

Το Signature είναι η ψηφιακή υπογραφή που προκύπτει από την χρήση του αλγορίθμου υπογραφής (HS256) επί των κωδικοποιημένων Header, Payload και ενός μυστικού κλειδιού [38]. Η υπογραφή και το μυστικό κλειδί εγγυώνται ότι το περιεχόμενο του JWT δεν μπορεί να τροποποιηθεί χωρίς να το γνωρίζει ο εκδότης του.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J0b2tlb190eXB1IjoiYWNjZXNzIiwiZXhwIjoxN
zMyMTQxMjY2LCJpYXQiOjE3MzIxMTc3OTUsImp0
aSI6IjMwMjc2YzgzMzg0NzRjZjU4OTlmbHGIOmGU
xZTV1ZjgyIiwidXNlc19pZCI6MX0.oPgA_2jnyF
mkgyew26cVxBxePJmf0pRjx5sz2oWFhI
```

**Εικόνα 5.11:** Παράδειγμα JWT, όπου τα κωδικοποιημένα Header, Payload και Signature διακρίνονται με τα χρώματα κόκκινο, μοβ και μπλε αντίστοιχα.

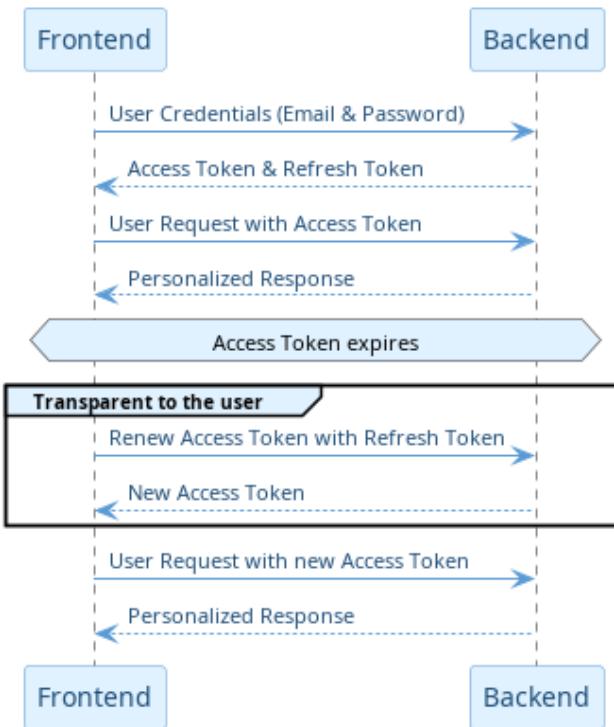
Προκειμένου να διατηρείται η ταυτότητα ενός χρήστη μετά την αυθεντικοποίηση του και να μην απαιτείται η εισαγωγή διαπιστευτηρίων σε κάθε αίτημα, χρησιμοποιούνται JWTs δύο τύπων. Ο πρώτος τύπος είναι το access token, με το οποίο ταυτοποιείται ο χρήστη στο backend και αποκτά πρόσβαση σε προστατευμένες υπηρεσίες [39]. Το access token έχει μικρή διάρκεια ζωής (20 λεπτά), ώστε αν διαρρεύσει να περιοριστεί το χρονικό διάστημα κατά το οποίο μπορεί να χρησιμοποιηθεί από μη εξουσιοδοτημένο κάτοχο. Ο δεύτερος τύπος είναι το refresh token, με το οποίο μπορούν να εκδοθούν νέα access tokens [39]. Το refresh token έχει μεγαλύτερη διάρκεια ζωής (14 ημέρες), ώστε ο χρήστης να μην χρειάζεται να αυθεντικοποιείται συχνά, ενώ παράλληλα είναι ιδιαίτερα προστατευμένο, καθώς ο κώδικας του frontend δεν έχει πρόσβαση σε αυτό. Γενικά, και οι δύο τύποι token απαιτούν προσεκτικό χειρισμό, διότι η έκθεσή τους μπορεί να δημιουργήσει κενά ασφαλείας στην εφαρμογή.

Κατά την αποστολή ενός έγκυρου αιτήματος αυθεντικοποίησης, το frontend λαμβάνει ένα ζεύγος token από το backend και τα χρησιμοποιεί είτε μέχρι τη λήξη του refresh token είτε μέχρι την αποσύνδεση του ταυτοποιημένου χρήστη. Ειδικότερα, η απάντηση του backend σε αιτήματα αυθεντικοποίησης περιλαμβάνει:

- Ένα access token στο σώμα του μηνύματος.
- Ένα refresh token ως HTTP Only cookie, το οποίο δεν είναι προσβάσιμο από τη Javascript του frontend [40].

Μετά τη λήψη της απάντησης, το frontend θέτει στο localStorage του φυλλομετρητή μια μεταβλητή `isLoggedIn` που σηματοδοτεί ότι ο χρήστης είναι αυθεντικοποιημένος. Για λόγους ασφαλείας, το access token αποθηκεύεται στη μνήμη και όχι στο localStorage (βλ. Κεφ. 9.5). Σε όλα τα αιτήματα που ακολουθούν, το frontend συμπεριλαμβάνει το access token στο Authorization HTTP header, ώστε το backend να αναγνωρίζει τον αυθεντικοποιημένο χρήστη. Επιπλέον, πριν από κάθε αίτημα, το frontend ελέγχει αν το access token έχει λήξει και το

ανανεώνει εφόσον είναι απαραίτητο. Η ανανέωση πραγματοποιείται με αποστολή ενός HTTP POST αιτήματος, το οποίο περιλαμβάνει αυτόματα το refresh token ως cookie, στην υπηρεσία /auth/users/refresh/. Όταν ο χρήστης αποσυνδεθεί, το access token αφαιρείται από τη μνήμη και η μεταβλητή `isLoggedIn` διαγράφεται από το localStorage. Η αλληλεπίδρση μεταξύ frontend και backend κατά τη διαδικασία αυθεντικοποίησης συνοψίζεται το ακόλουθο διάγραμμα (Εικόνα 5.12).



**Εικόνα 5.12:** Ανταλλαγή μηνυμάτων μεταξύ frontend και backend κατά την αυθεντικοποίησης του χρήστη μέσω access και refresh tokens.

Η αποθήκευση της μεταβλητής `isLoggedIn` στο localStorage του φυλλομετρητή έχει ιδιαίτερη σημασία για την ασφαλή διατήρηση της ταυτότητας του χρήστη. Πράγματι, αν ο αυθεντικοποιημένος χρήστης ανοίξει την εφαρμογή σε μία νέα καρτέλα του φυλλομετρητή ή κλείσει τον φυλλομετρητή και τον ανοίξει ξανά, τότε το access token δεν είναι διαθέσιμο στη μνήμη. Ωστόσο, η μεταβλητή `isLoggedIn` διατηρείται στο localStorage και το refresh token παραμένει στα cookies. Συνεπώς, στις παραπάνω περιπτώσεις το frontend ελέγχει το localStorage για την τιμή του `isLoggedIn` και αν την βρεί χρησιμοποιεί το refresh token για να

λάβει ένα νέο access token από το backend. Έτσι, ο χρήστης δεν χρειάζεται να επαναλαμβάνει τη διαδικασία αυθεντικοποίησης, εκτός αν αποσυνδεθεί ή αν λήξει το refresh token.

#### 5.4.3 Διαχείριση Λογαριασμού

Ένας από τους βασικούς λόγους που επιλέχθηκε η διεύθυνση email ως διαπιστευτήριο για τη δημιουργία λογαριασμού είναι ότι επιτρέπει την ανάκτηση του κωδικού πρόσβασης. Η εφαρμογή διαθέτει μια φόρμα στην οποία ο χρήστης μπορεί να εισάγει την διεύθυνση ηλεκτρονικού ταχυδρομείου του λογαριασμού του, ώστε να λάβει ένα email ανάκτησης του κωδικού πρόσβασης (Εικόνα 5.10, δεξιά). Το email αυτό, όπως και στην περίπτωση δημιουργίας λογαριασμού, περιέχει ένα σύνδεσμο που αποτελείται από ένα UID και ένα token. Ο σύνδεσμος οδηγεί σε μία σελίδα που ζητά από τον χρήστη να εισάγει ένα νέο κωδικό πρόσβασης. Στη συνέχεια, το frontend υποβάλλει τον νέο κωδικό μαζί με τα UID και token στην υπηρεσία /auth/users/reset\_password\_confirm/ του backend. Εφόσον τα στοιχεία είναι έγκυρα, ο κωδικός πρόσβασης του λογαριασμού ενημερώνεται. Με τον τρόπο αυτό, ο χρήστης μπορεί να ανακτήσει τον κωδικό πρόσβασης του λογαριασμού με ασφάλεια.

Εκτός από τη διαδικασία ανάκτησης του κωδικού πρόσβασης, η εφαρμογή υποστηρίζει επιπλέον λειτουργίες διαχείρισης, όπως η αλλαγή του κωδικού και η διαγραφή του λογαριασμού του χρήστη. Οι λειτουργίες αυτές προϋποθέτουν ότι ο χρήστης έχει συνδεθεί στην εφαρμογή και απαιτούν την εκ νέου εισαγωγή του κωδικού πρόσβασης για λόγους επαλήθευσης. Η μέθοδος υλοποίησης είναι παρόμοια με λειτουργίες υποβολής φόρμας που έχουν περιγραφεί: το frontend ελέγχει τη μορφή των δεδομένων που έχουν εισαχθεί και τα υποβάλλει μέσω HTTP POST/DELETE αιτημάτων στις αντίστοιχες υπηρεσίες του backend. Στην περίπτωση αλλαγής κωδικού, ο χρήστης λαμβάνει ενημερωτικό email και αποσυνδέεται αυτόματα από την εφαρμογή σε όλες τις συσκευές, διότι τα tokens που έχουν εκδοθεί χάνουν την εγκυρότητά τους.

### 5.5 Εμφάνιση Χαρτογραφήσεων Χρήστη

Ο κύριος στόχος των λειτουργιών δημιουργίας λογαριασμού και αυθεντικοποίησης είναι να παρέχουν στους ταυτοποιημένους χρήστες πρόσβαση σε προσωπικό περιεχόμενο. Με αυτόν τον τρόπο, οι χρήστες μπορούν εύκολα να βρίσκουν τις χαρτογραφήσεις που έχουν δημιουργήσει, χωρίς να χρειάζεται να τις αναζητούν μέσα στο σύνολο όλων των αποθηκευμένων χαρτογραφήσεων της εφαρμογής. Επιπλέον, διατηρούν πρόσβαση σε χαρτογραφήσεις που έχουν

αιτηθεί, των οποίων οι εργασίες του FLOODPY έχουν αποτύχει ή είναι σε εξέλιξη και ως εκ τούτου δεν εμφανίζονται στη δημόσια σελίδα παρουσίασης χαρτογραφήσεων. Από την οπτική του χρήστη, η σχετική λειτουργία περιγράφεται με το ακόλουθο user story:

*Ως χρήστης της εφαρμογής, εφόσον έχω συνδεθεί με τον λογαριασμό μου, θέλω να μπορώ να βλέπω όλες τις χαρτογραφήσεις που έχω αιτηθεί, ακόμα και αυτές για τις οποίες η εργασία του FLOODPY δεν έχει ολοκληρωθεί με επιτυχία.*

### Κριτήρια Αποδοχής

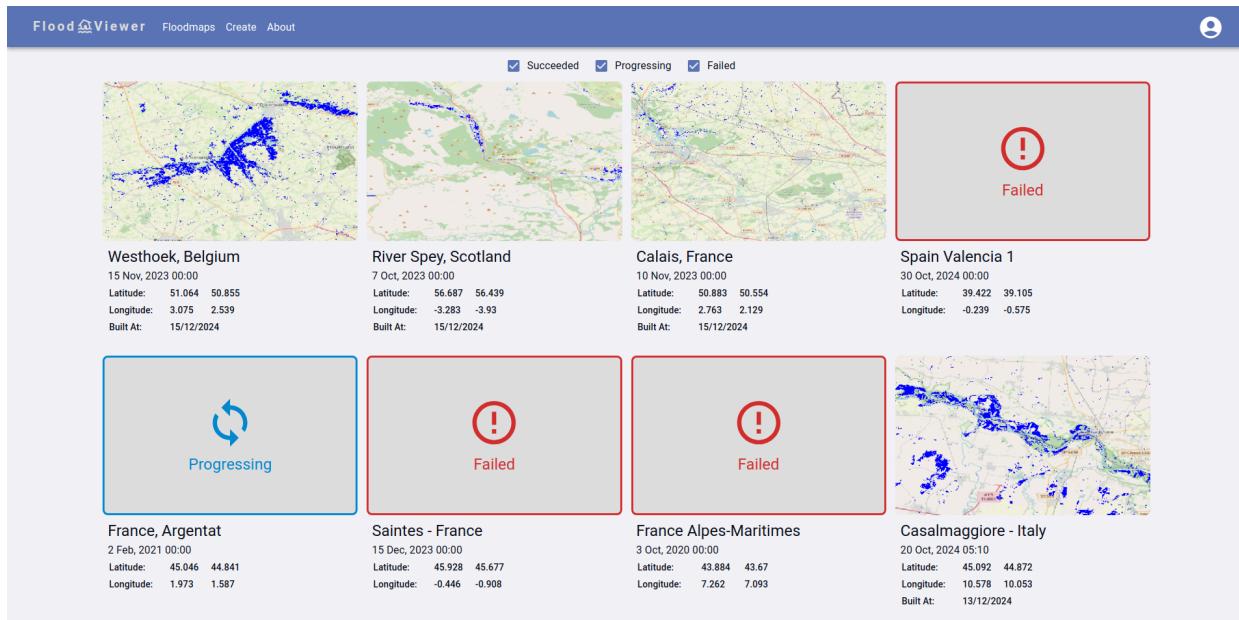
- Ο αυθεντικοποιημένος χρήστης έχει πρόσβαση σε μία σελίδα όπου, από προεπιλογή, παρουσιάζονται όλες οι χαρτογραφήσεις που έχει αιτηθεί.
- Ο αυθεντικοποιημένος χρήστης, στη σελίδα των χαρτογραφήσεών του, έχει τη δυνατότητα εφαρμογής φίλτρων, ώστε να εμφανίζονται μόνο οι χαρτογραφήσεις για τις οποίες οι εργασίες του FLOODPY βρίσκονται σε προεπιλεγμένες καταστάσεις (*Succeeded, Progressing, Failed*).

Η εμφάνιση προσωπικού περιεχομένου προϋποθέτει την συσχέτιση των χρηστών με το περιεχόμενο που δημιουργούν. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, όταν ένας χρήστης αυθεντικοποιείται επιτυχώς λαμβάνει ένα αναγνωριστικό access token, το οποίο συμπεριλαμβάνεται σε κάθε HTTP αίτημα από το frontend προς το backend. Επομένως, όταν ο χρήστης ζητά τη δημιουργία μιας νέας χαρτογράφησης, το αντίστοιχο HTTP αίτημα που αποστέλλεται από το frontend περιέχει το access token του χρήστη. Συνεπώς, το backend μπορεί να συσχετίσει τη νέα χαρτογράφηση με τον λογαριασμό για τον οποίο εκδόθηκε το access token, δηλαδή με τον λογαριασμό του αυθεντικοποιημένου χρήστη.

Για την προβολή των χαρτογραφήσεων που έχει δημιουργήσει ο χρήστης, η εφαρμογή διαθέτει μια σελίδα στο μονοπάτι /jobs. Κατά την φόρτωση της σελίδας αυτής, το frontend στέλνει ένα HTTP GET αίτημα στην υπηρεσία /api/floodmaps/ του backend. Για να ληφθούν μόνο οι χαρτογραφήσεις του αυθεντικοποιημένου χρήστη, χρησιμοποιείται η παράμετρος `owned=true`, με αποτέλεσμα η διεύθυνση του αιτήματος να γίνεται `/api/floodmaps/?owned=true`. Αφού το backend ελέγχει το access token και επιστρέψει τα ζητούμενα δεδομένα, κάθε χαρτογράφηση παρουσιάζεται στη σελίδα μέσω μιας κάρτας προεπισκόπησης, η οποία περιέχει βασικές πληροφορίες και μια εικόνα. Οι κάρτες οργανώνονται σε ένα δυναμικό πλέγμα, όπως περιγράφηκε στο Κεφάλαιο 5.3. Η εικόνα κάθε

κάρτας υποδεικνύει την κατάσταση της σχετικής εργασία του FLOODPY. Αν η εργασία έχει επιτύχει απεικονίζονται οι πλημμυρισμένες περιοχές, αλλιώς εμφανίζεται το όνομα της κατάστασης (*Progressing* ή *Failed*) μαζί με το ανάλογο εικονίδιο (Εικόνα 5.13).

Επιπλέον, η σελίδα επιτρέπει την χρήση φίλτρων για την εμφάνιση χαρτογραφήσεων βάσει της κατάστασης των αντίστοιχων εργασιών του FLOODPY. Για καθεμία από τις τρεις δυνατές καταστάσεις (*Succeeded*, *Progressing*, *Failed*), η σελίδα διαθέτει ένα ξεχωριστό checkbox. Κατά την αρχική φόρτωση της σελίδας όλα τα checkboxes είναι επιλεγμένα, με αποτέλεσμα να παρουσιάζονται όλες οι χαρτογραφήσεις του χρήστη. Αν κάποιο checkbox αποεπιλεγεί, το frontend στέλνει εκ νέου αίτημα στο backend εφαρμόζοντας τις τρεις καταστάσεις ως παραμέτρους με τιμές σύμφωνα με τις τρέχουσες επιλογές των checkboxes. Για παράδειγμα, προκειμένου να ληφθούν οι χαρτογραφήσεις χωρίς να συμπεριλαμβάνονται όσες έχουν επιτυχημένες εργασίες, αποεπιλέγεται το “*Succeeded*” checkbox. Τότε, το frontend πραγματοποιεί ένα HTTP GET αίτημα στο `/api/floodmaps/?succeeded=false&progressing=true&failed=true` του backend, ώστε να λάβει και να παρουσιάσει τα αντίστοιχα δεδομένα.



**Εικόνα 5.13:** Εμφάνιση χαρτογραφήσεων αυθεντικοποιημένου χρήστη.

Αξίζει να σημειωθεί ότι για τους λογαριασμούς με προνόμια διαχειριστή, η σελίδα `/jobs` εμφανίζει όλες τις χαρτογραφήσεις ανεξαρτήτως δημιουργού. Αυτό συμβαίνει διότι οι

διαχειριστές της εφαρμογής έχουν πλήρη δικαιώματα πρόσβασης επί όλων των χαρτογραφήσεων. Περαιτέρω, όπως περιγράφεται στη συνέχεια, οι διαχειριστές είναι υπεύθυνοι για την έγκριση των εργασιών του FLOODPY που αιτούνται οι χρήστες. Επομένως, η δυνατότητα προβολής όλων των χαρτογραφήσεων αποτελεί μία εξαιρετικά χρήσιμη λειτουργία, καθώς επιτρέπει στους διαχειριστές να παρακολουθούν συνολικά την εξέλιξη των αντίστοιχων εργασιών.

## 5.6 Έλεγχος Εργασιών FLOODPY

Η εφαρμογή δεν είναι εφικτό να επιτρέπει την εκτέλεση εργασιών του FLOODPY από οποιονδήποτε χρήστη χωρίς περιορισμούς. Πράγματι, η δημιουργία του περιεχομένου μιας χαρτογράφησης είναι υπολογιστικά ακριβή και χρονοβόρα διαδικασία. Μια εκτέλεση του FLOODPY, ανάλογα με το εύρος της περιοχής ενδιαφέροντος και τους διαθέσιμους υπολογιστικούς πόρους, μπορεί να διαρκέσει περισσότερα από 30 λεπτά. Κατά αυτό το χρονικό διάστημα, γίνεται λήψη δεδομένων που συνήθως υπερβαίνουν τα 25GB και πραγματοποιείται επεξεργασία που δεσμεύει μεγάλη ποσότητα μνήμης. Επιπλέον, το αποτέλεσμα που παράγεται έχει αξία μόνο αν η περιοχή ενδιαφέροντος και η χρονική στιγμή που επιλέχθηκαν αντιστοιχούν σε κάποια πλημμύρα. Επομένως, αν η εφαρμογή επέτρεπε σε κάθε επισκέπτη να εκτελεί εργασίες του FLOODPY, τότε ένα χρήστης θα μπορούσε, είτε κατά λάθος, είτε κακοβούλως, να σπαταλά τους πόρους της εφαρμογής.

Για να αποτραπεί το ενδεχόμενο κατάχρησης της εφαρμογής και να διασφαλιστεί η ποιότητα του περιεχομένου της, η εκτέλεση της εργασίας του FLOODPY για κάθε νέα χαρτογράφηση πραγματοποιείται μόνο εφόσον έχουν εγκριθεί από κάποιο διαχειριστή της εφαρμογής. Η λειτουργία αυτή αντιστοιχεί στο ακόλουθο user story:

*Ως χρήσης της εφαρμογής θέλω οι εργασίες του FLOODPY να εκκινούν μόνο εφόσον έχουν εγκριθεί από κάποιον διαχειριστή της εφαρμογής. Με τον τρόπο αυτό, εξασφαλίζεται η ποιότητα των περιεχομένου της εφαρμογής και αποφεύγεται η σπατάλη υπολογιστικών πόρων.*

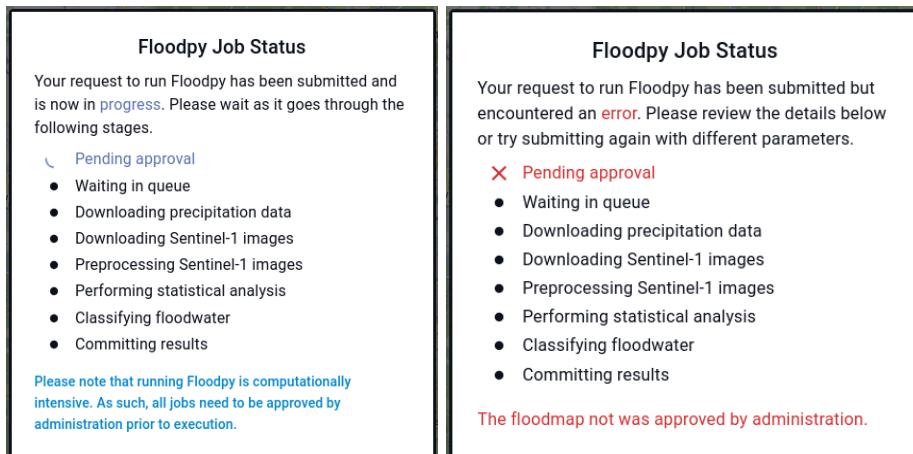
### Κριτήρια Αποδοχής

- Η εφαρμογή δεν εκτελεί αυτόματα εργασίες του FLOODPY για χαρτογραφήσεις που υποβάλλονται από απλούς χρήστες.

- Ο διαχειριστής μπορεί να επισκεφθεί την σελίδα μιας χαρτογράφησης και να εγκρίνει ή να απορρίψει την σχετική εργασία του FLOODPY.
- Ο χρήστης μπορεί να επισκεφθεί την σελίδα μιας χαρτογράφησης και να ελέγξει αν η σχετική εργασία του FLOODPY έχει εγκριθεί ή απορριφθεί.

Προκειμένου οι χρήστες να γνωρίζουν αν μια εργασία έχει εγκριθεί, το πρώτο βήμα κάθε εργασίας υποδεικνύει την κατάσταση έγκρισής της. Όπως αναφέρθηκε στο Κεφάλαιο 5.2, η σελίδα κάθε μη ολοκληρωμένης χαρτογράφησης απεικονίζει την πρόοδο της εργασίας του FLOODPY που δημιουργεί το περιεχόμενό της. Στη σελίδα αυτή, παρουσιάζονται όλα τα στάδια που πρέπει να ολοκληρωθούν, ενώ κάθε στιγμή υποδεικνύεται το τρέχον στάδιο. Το πρώτο στάδιο φέρει τον τίτλο *Pending approval* και η εργασία παραμένει σε αυτό μέχρι να εγκριθεί από έναν διαχειριστή (Εικόνα 5.14, αριστερά). Επίσης, όσο η εργασία βρίσκεται στο εν λόγω στάδιο, στη σελίδα αναγράφεται ο λόγος για τον οποίο οι εργασίες απαιτούν έγκριση.

Σε περίπτωση που ένας διαχειριστής επισκεφτεί τη σελίδα μιας χαρτογράφησης που αναμένει έγκριση έχει τη δυνατότητα να την εγκρίνει ή να την απορρίψει. Για το σκοπό αυτό, όταν ο επισκέπτης της σελίδας είναι διαχειριστής, στα δεδομένα της χαρτογράφησης που λαμβάνει το frontend από το backend περιλαμβάνεται το πεδίο “owner” που υποδεικνύει ότι ο χρήστης έχει αυξημένα δικαιώματα επί της χαρτογράφησης. Τότε, η διεπαφή εμφανίζει ένα πλαίσιο με κουμπιά για την έγκριση και την απόρριψη της εργασίας.



**Εικόνα 5.14:** Στάδια εργασίας χαρτογράφησης που αναμένει έγκριση (αριστερά) και που απορρίφθηκε (δεξιά).

Όταν ο διαχειριστής επιλέξει να εγκρίνει ή να απορρίψει μια εργασία του FLOODPY, το frontend στέλνει ένα HTTP PATCH αίτημα στην υπηρεσία διαχείρισης εργασιών `/api/job/<id>/` του backend, όπου `<id>` ο αναγνωριστικός αριθμός της χαρτογράφησης. Το περιεχόμενο του αιτήματος είναι ένα JSON, στο οποίο αναγράφεται αν η εργασία έγινε αποδεκτή ή όχι. Αν δεν συμβεί κάποιο σφάλμα κατά την επεξεργασία του αιτήματος, το frontend αμέσως μετά θα λάβει ενημερώσεις για την νέα κατάσταση της εργασίας μέσω της σύνδεσης WebSocket που περιγράφηκε στο Κεφάλαιο 5.2.1. Απόρριψη της εργασίας του FLOODPY συνεπάγεται αποτυχία στο στάδιο *Pending approval* και εμφάνιση αντίστοιχου μηνύματος στον χρήστη (Εικόνα 5.14, δεξιά). Έγκριση της εργασίας σημαίνει επιτυχία του βήματος *Pending approval* και μεταπίδηση στο επόμενο στάδιο (*Waiting in queue*).

## 6. Επιβεβαίωση του Frontend

Προκειμένου να επιβεβαιωθεί ότι το frontend λειτουργεί σύμφωνα με τις προκαθορισμένες απαιτήσεις και να μειωθεί η πιθανότητα εμφάνισης σφαλμάτων, εφαρμόστηκε μια στρατηγική υλοποίησης αυτοματοποιημένων τεστ. Γενικά, τα τεστ δεν μπορούν να αποδείξουν ότι το λογισμικό έχει την αναμενόμενη συμπεριφορά σε κάθε περίπτωση [41, p. 227]. Επιπλέον, καθώς οι απαιτήσεις της εφαρμογής εξελίσσονται, η συγγραφή νέων και η αναπροσαρμογή υπαρχόντων τεστ μπορούν να γίνουν ιδιαίτερα χρονοβόρες διαδικασίες. Ωστόσο, η δημιουργία μιας αυτοματοποιημένης διαδικασία για την διασφάλιση της ποιότητας του λογισμικού είναι απαραίτητη, διότι μειώνει τις πιθανότητες εμφάνισης σφαλμάτων και υλοποίησης λειτουργιών που δεν ανταποκρίνονται στις απαιτήσεις.

Η στρατηγική που ακολουθήθηκε κατά την υλοποίηση των τεστ επιτυγχάνει ένα ιδανικό συμβιβασμό μεταξύ διασφάλισης ποιότητας και επένδυσης χρόνου. Συγκεκριμένα, ελέγχονται οι βασικές λειτουργίες του frontend χωρίς να επιδιώκεται απαραίτητη η πλήρης κάλυψη του κώδικα. Τα τεστ προσομοιώνουν την αλληλεπίδραση του χρήστη με την διεπαφή και εξετάζουν αν το περιεχόμενο της διεπαφής και τα μηνύματα του frontend προς το backend έχουν την αναμενόμενη μορφή. Στα τεστ δεν ελέγχονται παράμετροι του frontend που δεν είναι προσβάσιμες από τον χρήστη και αφορούν λεπτομέρειες υλοποίησης, όπως οι μεταβλητές κατάστασης της React.

Για την συγγραφή των τεστ χρησιμοποιήθηκαν οι βιβλιοθήκες React Testing Library και Mocking Service Worker (MSW). Μέσω της βιβλιοθήκης React Testing Library, η σελίδα που πρόκειται να ελεγχθεί φορτώνεται στο περιβάλλον jsdom που υλοποιεί ένα υποσύνολο των λειτουργιών του φυλλομετρητή [42]. Με την ίδια βιβλιοθήκη, γίνεται αναζήτηση και χρήση στοιχείων του DOM βάσει του περιεχομένου που εμφανίζεται στη σελίδα, προσομοιώνοντας τον τρόπο με τον οποίο ένας πραγματικός χρήστης αλληλεπιδρά με την διεπαφή [43]. Τα HTTP αιτήματα που στέλνει το frontend με αποδέκτη το backend κατά την διάρκεια των τεστ ανακατευθύνονται στην βιβλιοθήκη MSW, η οποία ελέγχει τα αιτήματα και μιμείται τις απαντήσεις το backend. Με αυτόν τον τρόπο, η λειτουργία του frontend απομονώνεται πλήρως και διασφαλίζεται ότι το περιβάλλον εκτέλεση των τεστ δεν επηρεάζεται από το τυχόν αστοχίες του backend. Η διαδικασία ολοκληρώνεται με την εκτέλεση assert statements, δηλαδή ισχυρισμών που επιβεβαιώνουν ότι το περιεχόμενο της σελίδας μετά την αλληλεπίδραση που προσομοιώνεται είναι το αναμενόμενο.

Στη συνέχεια παρουσιάζεται ένα παράδειγμα των βημάτων ενός τυπικού τεστ. Το συγκεκριμένο τεστ αφορά την σελίδα που εμφανίζει τις αποθηκευμένες χαρτογραφήσεις της εφαρμογής (βλ. Κεφ. 5.3).

1. Αρχικά διαμορφώνεται ο MSW για την προσομοίωση των απαντήσεων του backend.

Όταν/αν το frontend στείλει HTTP GET αιτήματα στην υπηρεσία `/api/floodmaps/` του backend θα λάβει ένα JSON στο οποίο:

- 1.1. Σε κάθε περίπτωση αναγράφεται ότι υπάρχουν συνολικά 16 αποθηκευμένες χαρτογραφήσεις.
- 1.2. Στην περίπτωση που το αίτημα δεν έχει παράμετρο ή έχει παράμετρο `page=1`, περιλαμβάνονται τα δεδομένα των πρώτων 12 χαρτογραφήσεων.
- 1.3. Στην περίπτωση που το αίτημα έχει παράπετρο `page=2`, περιλαμβάνονται τα δεδομένα των επόμενων 4 χαρτογραφήσεων.
2. Φορτώνεται η σελίδα `/floodmaps` που θα ελεγχθεί.
3. Επιβεβαιώνεται μέσω assert statements ότι το περιεχόμενο της σελίδας περιλαμβάνει τις πρώτες 12 χαρτογραφήσεις.
4. Επιλέγεται το κουμπί της δεύτερης σελίδας αποτελεσμάτων.
5. Επιβεβαιώνεται ότι το περιεχόμενο της νέας σελίδας περιλαμβάνει τις επόμενες 4 χαρτογραφήσεις.

Επιτυχής ολοκλήρωση του παραπάνω βημάτων συνεπάγεται ότι το frontend, για την σελίδα `/floodmaps` που παρουσιάζει τις αποθηκευμένες χαρτογραφήσεις, στέλνει τα αναμενόμενα HTTP αιτήματα στο backend και εμφανίζει τα δεδομένα των χαρτογραφήσεων σύμφωνα με τις απαιτήσεις.

## 7. Λειτουργίες και Αρχιτεκτονική του Backend

Το backend υλοποιεί τη λογική που εκτελείται στον διακομιστή της εφαρμογής και είναι υπεύθυνο για τη διαχείριση όλων των αιτημάτων που αποστέλλονται από το frontend. Οι υπηρεσίες που παρέχει ανταποκρίνονται πλήρως στις απαίτησεις των λειτουργιών του frontend που περιγράφηκαν στο Κεφάλαιο 5. Ειδικότερα, το backend υλοποιεί τις ακόλουθες λειτουργίες:

- Αποθήκευση και ανάκτηση δεδομένων χαρτογραφήσεων.
- Εκτέλεση εργασιών του FLOODPY και αποστολή ενημερώσεων σχετικά με την πρόοδό τους.
- Διαχείριση λογαριασμών χρηστών και υλοποίηση μηχανισμών αυθεντικοποίησης.

Οι λειτουργίες αυτές καθορίζουν τις τεχνικές προδιαγραφές του backend, οι οποίες με τη σειρά τους διαμορφώνουν τη συνολική αρχιτεκτονική του συστήματος.

### 7.1 Τεχνικές Προδιαγραφές

Οι πιο θεμελιώδεις τεχνικές προδιαγραφές αφορούν τις δυνατότητες εξυπηρέτησης HTTP αιτημάτων και διαχείρισης δεδομένων. Το frontend στέλνει HTTP αιτήματα στο backend, αρκετά από τα οποία περιέχουν δεδομένα σε μορφή JSON, και αναμένει απαντήσεις με περιεχόμενο στην ίδια μορφή. Για τον σκοπό αυτό, το backend υλοποιεί ένα σύνολο HTTP υπηρεσιών ικανών να αποκωδικοποίησουν δεδομένα JSON, να εκτελέσουν προκαθορισμένη επεξεργασία και να κωδικοποιήσουν απαντήσεις σε μορφή JSON. Κατά τη διάρκεια της επεξεργασίας, οι υπηρεσίες έχουν πρόσβαση σε βάση δεδομένων μέσω της οποίας μπορούν να ανακτούν και να αποθηκεύουν δεδομένα. Οι περισσότερες υπηρεσίες πραγματοποιούν σύγχρονη επεξεργασία που συμπεριλαμβάνει έλεγχο των παραμέτρων του εισερχόμενου αιτήματος, πρόσβαση στη βάση δεδομένων και κωδικοποίηση της τελικής απάντησης.

Ωστόσο, οι υπηρεσίες δημιουργίας νέων χαρτογραφήσεων έχουν επιπλέον τη δυνατότητα ασύγχρονης εκτέλεσης μακροχρόνιων εργασιών. Όπως αναφέρθηκε στο Κεφάλαιο 5.6, η ολοκλήρωση μιας εργασίας του FLOODPY απαιτεί αρκετό χρόνο και υπολογιστικούς πόρους. Επομένως, οι υπηρεσίες που εκτελούν το FLOODPY αποκρίνονται ασύγχρονα, δηλαδή απαντούν αμέσως με την κατάσταση αποδοχής του αιτήματος δημιουργίας νέας χαρτογράφησης (επιβεβαίωση/απόρριψη) και ταυτόχρονα υποβάλλουν την αντίστοιχη εργασία του FLOODPY για εκτέλεση σε μεταγενέστερο χρόνο. Οι εργασίες που υποβάλλονται καταχωρούνται και

εκτελούνται στο παρασκήνιο όταν οι απαραίτητοι πόροι είναι διαθέσιμοι. Επιπρόσθετα, για να διασφαλιστεί ότι οι υπολογιστικά ακριβές εργασίες του FLOODPY δεν επηρεάζουν άλλες υπηρεσίες του backend, η εκτέλεσή τους πραγματοποιείται από ξεχωριστές διεργασίες. Με αυτόν τον τρόπο, επιτυγχάνεται επιπλέον απομόνωση πιθανων σφαλμάτων και διευκολύνεται η ανεξάρτητη κλιμάκωση των υπολογιστικών πόρων που διατίθενται για τις εργασίες του FLOODPY.

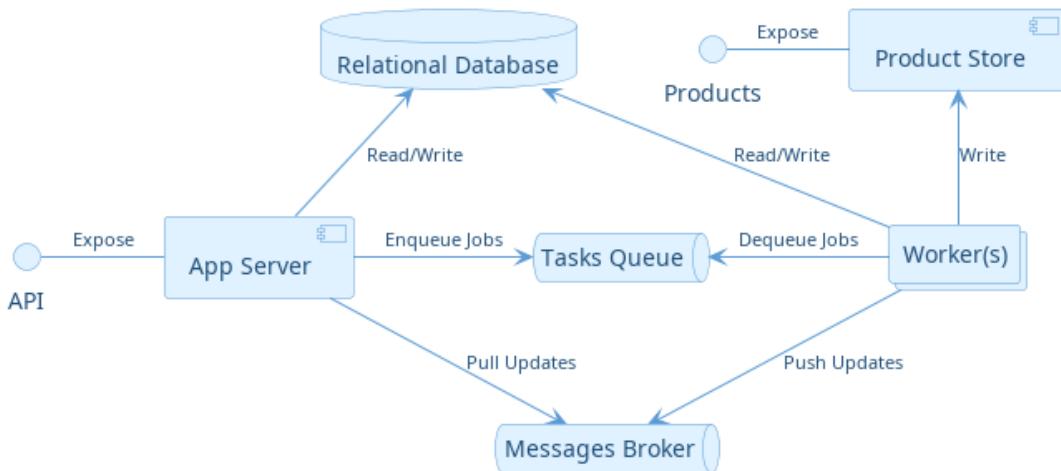
Κατά τη διάρκεια της εκτέλεσης εργασιών, το backend υποστηρίζει την αποστολή ενημερώσεων σχετικά με την πρόοδο των εκτελούμενων βημάτων μέσω του πρωτοκόλλου WebSocket. Η αναγκαιότητά του εν λόγω πρωτοκόλλου και το τρόπος λειτουργίας του στο frontend αναλύθηκαν στο Κεφάλαιο 5.2.1. Στο backend οι υπηρεσίες WebSocket επιτρέπουν τη δημιουργία αμφίδρομων διαύλων επικοινωνίας για την αποστολή ενημερώσεων σε συνδεδεμένους πελάτες σε πραγματικό χρόνο. Το περιεχόμενο των ενημερώσεων καθορίζεται εντός των ξεχωριστών διεργασιών που εκτελούν τις εργασίες του FLOODPY και προωθείται από τις υπηρεσίες του backend μέσω μηνυμάτων WebSocket. Οι υπηρεσίες του backend είναι ανεξάρτητες από τις διεργασίες που εκτελούν τις εργασίες του FLOODPY, και συνεπώς το backend στέλνει ενημερώσεις ακόμα και σε περιπτώσεις που κάποια εργασία αποτύχει ή τερματιστεί απόσμενα.

Ιδιαίτερα σημασία έχουν και οι τεχνικές προδιαγραφές που αφορούν την υποστήριξη ασφαλούς διαχείριση λογαριασμών χρηστών και μεθόδων αυθεντικοποίησης. Οι λειτουργίες αυτές εντάσσονται στις τεχνικές προδιαγραφές, καθώς η υλοποίησή τους απαιτεί την εκτέλεση κρίσιμων διαδικασιών, όπως η αποστολή email, η ασφαλής διαχείριση κωδικών πρόσβασης (hashing/salting) και η επαλήθευση JWTs, οι οποίες βασίζονται σε καθιερωμένα πρότυπα ασφαλείας. Αυτές οι διαδικασίες είναι εξαιρετικά σημαντικές για την συνολική ασφάλεια της εφαρμογής και δεν υλοποιούνται σαν τις συνήθεις λειτουργίες. Αντίθετα, αντιμετωπίζονται ως τεχνικές απαιτήσεις που πρέπει να ικανοποιηθούν μέσω της χρήσης αξιόπιστων framework ή βιβλιοθηκών, η ενσωμάτωση των οποίων αποτελεί σημαντικό παράγοντα κατά το σχεδιασμό της εφαρμογής.

## 7.2 Αρχιτεκτονική

Η υλοποίηση του backend οργανώθηκε σε ανεξάρτητα components (τμήματα), τα οποία αλληλεπιδρούν μεταξύ τους για την σύνθεση της συνολικής λειτουργικότητα του συστήματος

(Εικόνα 7.1). Σε κάθε component αντιστοιχεί μία ή περισσότερες διεργασίες που μπορούν να εκτελούνται είτε στο ίδιο μηχάνημα με τις διεργασίες άλλων component είτε σε ξεχωριστά μηχάνημα. Τα components διαθέτουν προκαθορισμένες λειτουργίες και σαφώς ορισμένες διεπαφές, οι οποίες εξυπηρετούν τόσο την επικοινωνία μεταξύ τους όσο και τις συναλλαγές με το frontend. Στη συνέχεια, παρουσιάζονται η αρχιτεκτονική του backend και οι κύριες λειτουργίες των component που την απαρτίζουν.



**Εικόνα 7.1: Αρχιτεκτονική του backend.**

Το βασικό component του backend, που είναι υπεύθυνο για την υλοποίηση των προδιαγραφών εξυπηρέτησης HTTP αιτημάτων, ονομάζεται App Server (Εξυπηρετητής Εφαρμογής). Όταν ο διακομιστής (web server) λαμβάνει ένα αίτημα για κάποια από τις υπηρεσίες του backend, το κατευθύνει στον App Server, εκτός αν αφορά τη λήψη στατικών προϊόντων του FLOODPY. Ο App Server, ανάλογα με το αίτημα και την υπηρεσία που κλήθηκε, εκτελεί την προκαθορισμένη λογική. Για τις ανάγκες της λογικής αυτής, έχει πρόσβαση στο component Relational Database, το οποίο φιλοξενεί μια σχεσιακή βάση δεδομένων και επιτρέπει την ανάγνωση και την αποθήκευση δεδομένων.

Ο App Server περιλαμβάνει επίσης τη λογική αυθεντικοποίησης και διαχείρισης λογαριασμών χρηστών. Οι λειτουργίες αυτές απαιτούν πρόσβαση στη βάση δεδομένων, επομένως η ενσωμάτωσή τους στον App Server είναι πρακτική επιλογή. Με τον τρόπο αυτό, διευκολύνεται και ο έλεγχος της ταυτότητας των χρηστών κατά την πρόσβασή τους σε προστατευμένες υπηρεσίες. Γενικά, η δημιουργία ξεχωριστού component στο backend για την

διαχείριση λογαριασμών και την αυθεντικοποίηση κρίθηκε μη απαραίτητη λόγω της σημαντικής αύξησης στην πολυπλοκότητα της υλοποίησης που θα επέφερε και της σχετικά μικρής κλίμακας της εφαρμογής.

Ωστόσο, ο App Server δεν μπορεί να επιβαρυνθεί περαιτέρω με την εκτέλεση των εργασιών του FLOODPY. Όπως αναφέρθηκε στις τεχνικές προδιαγραφές, οι εργασίες αυτές είναι χρονοβόρες, υπολογιστικά ακριβές και πρέπει να εκτελούνται ασύγχρονα εκτός του περιβάλλοντος εξυπηρέτησης υπηρεσιών. Συνεπώς, ο App Server αναθέτει την εκτέλεσή τους σε ένα σύνολο components που ονομάζονται Workers και σχεδιάστηκαν για αυτό το σκοπό. Πιο συγκεκριμένα, όταν ο App Server λαμβάνει ένα αίτημα για την εκτέλεση μιας εργασίας, την εναποθέτει στην ουρά Tasks Queue και απαντά στο αίτημα ασύγχρονα. Ο πρώτος διαθέσιμος Worker αφαιρεί την εργασία από την Tasks Queue, την εκτελεί και αποστέλλει τα προϊόντα που παράγονται στο Product Store.

Το Product Store είναι το component όπου αποθηκεύονται προϊόντα εργασιών του FLOODPY και εξυπηρετούνται αιτήματα για την λήψη τους. Όταν ο διακομιστής λαμβάνει αιτήματα για τη λήψη στατικών προϊόντων, όπως απεικονίσεις πλημμυρισμένων περιοχών, τα προωθεί στο Product Store, το οποίο διαθέτει τα ζητούμενα δεδομένα. Επιπλέον, όταν ένας Worker ολοκληρώνει επιτυχώς μια εργασία του FLOODPY, αποστέλλει τα παραγόμενα προϊόντα στο Product Store για αποθήκευση και μελλοντικό διαμοιρασμό. Αξίζει να σημειωθεί ότι τα προϊόντα των εργασιών του FLOODPY είναι geo-registered εικόνες, δηλαδή περιλαμβάνουν τις γεωγραφικές συντεταγμένες των περιοχών που απεικονίζουν. Επομένως, το Product Store υποστηρίζει τα κατάλληλα πρωτόκολλα για την αποδοτική αποθήκευση και την διανομή geo-registered εικόνων.

Κατά τη διάρκεια της εκτέλεσης εργασιών του FLOODPY, οι Workers στέλνουν ενημερώσεις προόδου στον App Server, ο οποίος τις διανέμει μέσω υπηρεσιών WebSocket. Ειδικότερα, όταν μια εργασία αλλάζει κατάσταση, ο Worker που την εκτελεί εναποθέτει σχετική ενημέρωση στο component Messages Broker (Εικόνα 7.1). Εφόσον υπάρχει τουλάχιστον ένας πελάτης συνδεδεμένος σε υπηρεσία του App Server που να παρακολουθεί την εκτελούμενη εργασία, ο App Server λαμβάνει την ενημέρωση από τον Messages Broker και την αποστέλλει στον συνδεδεμένο πελάτη μέσω μηνύματος WebSocket. Σε περίπτωση που κανένας πελάτης δεν έχει συνδεθεί για να λάβει την ενημέρωση, ο Messages Broker την διαγράφει έπειτα από προκαθορισμένο χρονικό διάστημα.

## 8. Τεχνολογίες του Backend

Για την ανάπτυξη του backend αξιοποιήθηκε ένα σύνολο τεχνολογιών που περιλαμβάνει frameworks, βιβλιοθήκες, εξυπηρετητές και βάσεις δεδομένων. Οι τεχνολογίες αυτές επιλέχθηκαν με γνώμονα την κάλυψη των προδιαγραφών του backend και την υποστήριξη των απαιτήσεων της αρχιτεκτονικής του. Ιδιαίτερη έμφαση δόθηκε στην ελαχιστοποίηση της εισαγωγής περιττής πολυπλοκότητας, ώστε να διασφαλίζονται η επεκτασιμότητα και η αποδοτικότητα της υλοποίησης.

Στη συνέχεια, παρουσιάζονται οι πιο θεμελιώδεις τεχνολογίες που αξιοποιήθηκαν. Ειδικότερα, αναλύονται οι βασικές δυνατότητες που προσφέρουν, καθώς και ο τρόπος με τον οποίο οι δυνατότητες αυτές αντιστοιχούν στις ανάγκες και στις λειτουργίες των component της αρχιτεκτονικής. Για λόγους σαφήνειας, περιγράφονται μόνο οι τεχνολογίες που αποτελούν τον πυρήνα της υλοποίησης του backend και αποφεύγεται η παράθεση τεχνικών λεπτομερειών. Οι επιμέρους τεχνολογίες που αφορούν δευτερεύουσες λειτουργικότητες περιγράφονται σε επόμενο κεφάλαιο (βλ. Κεφ. 9).

### 8.1 Django

Η ανάπτυξη του App Server προϋποθέτει τη δυνατότητα εκτέλεσης κώδικα κατά την εξυπηρέτηση εισερχόμενων αιτημάτων. Αυτή η απαίτησης μπορεί να υλοποιηθεί σε πολλές γλώσσες προγραμματισμού με τη χρήση βιβλιοθηκών και frameworks που διευκολύνουν τον προγραμματισμό στο περιβάλλον του backend. Ωστόσο, δεδομένου ότι το FLOODPY είναι γραμμένο σε Python και η ενσωμάτωση πολλαπλών γλωσσών προγραμματισμού μπορεί να αυξήσει την πολυπλοκότητα της υλοποίησης, η Python αποτελεί την πιο πρακτική και αποτελεσματική επιλογή. Επιπρόσθετα, η Python διαθέτει πολλά δημοφιλή και εύχρηστα frameworks για την δημιουργία web εφαρμογών, όπως τα Flask, Django και FastAPI.

Ανάμεσα στα διαθέσιμα web frameworks της Python, επιλέχθηκε το Django. Το βασικό κριτήριο που οδήγησε σε αυτή την επιλογή είναι η ενσωματωμένη σελίδα διαχείρισης δεδομένων που προσφέρει το Django στους διαχειριστές της εφαρμογής [44]. Μέσω αυτής της σελίδας, οι διαχειριστές μπορούν να δώσουν προνόμια σε άλλους λογαριασμού χρηστών και να διορθώσουν τυχόν σφάλματα στη βάση δεδομένων, χωρίς να απαιτούνται εξειδικευμένες τεχνικές γνώσεις. Επιπλέον, το Django παρέχει πολλές ενσωματωμένες δυνατότητες για τη

διαχείριση λογαριασμών και την αυθεντικοποίηση [45], ενώ υποστηρίζει τις τεχνικές προδιαγραφές ασφαλείας που αναφέρθηκαν στο Κεφάλαιο 7.1, όπως το hashing/salting των κωδικών πρόσβασης [46]. Παράλληλα, προσφέρει προστασία από κοινά κενά ασφαλείας, όπως το SQL Injection και το XSS (Cross-Site Scripting) [47], καθιστώντας το μια εξαιρετική επιλογή για την ασφαλή ανάπτυξη του App Server.

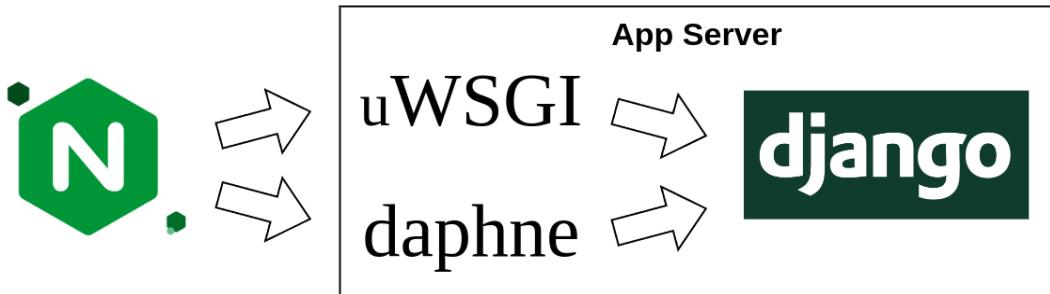
Για να ενισχυθούν περαιτέρω οι δυνατότητες του Django, χρησιμοποιήθηκε το Django Rest Framework (DRF). Το DRF διευκολύνει την δημιουργία υπηρεσιών παρέχοντας υποστήριξη για λειτουργίες, όπως η σελιδοποίηση αποτελεσμάτων, η εφαρμογή φίλτρων αναζήτησης και η διαχείριση δεδομένων μέσω serializers [48]-[50]. Ταυτόχρονα, επιτρέπει την υλοποίηση λειτουργιών ασφαλείας, όπως ο έλεγχος εισερχόμενων δεδομένων, η αυθεντικοποίηση μέσω token, η επιβολή πολιτικών throttling και η διαχείριση εξουσιοδοτήσεων [51]-[53]. Η ενσωμάτωση αυτών των δυνατοτήτων, η σημασία των οπίων αναλύεται σε επόμενα κεφάλαιο, συνέβαλε στη μείωση του όγκου και της πολυπλοκότητας του κώδικα, ενισχύοντας παράλληλα την ασφάλεια και την αποδοτικότητα του backend.

### 8.1.1 Ενσωμάτωση

Η χρήση του Django στον App Server προϋποθέτει την εγκατάσταση εξειδικευμένων εξυπηρετητών που υποστηρίζουν την εκτέλεσή του. Όλα τα αιτήματα που αποστέλλονται στο backend λαμβάνονται αρχικά από τον διακομιστή nginx και όσα από αυτά απαιτούν δυναμική επεξεργασία δρομολογούνται στον App Server. Για τη διαχείριση αιτημάτων στον App Server μέσω λογικής που έχει υλοποιηθεί με κώδικα Python, απαιτείται ένας εξυπηρετητής ικανός να εκτελεί αυτόν τον κώδικα και να του παρέχει τα στοιχεία των αιτημάτων ως είσοδο. Η διαδικασία αυτή έχει προτυποποιηθεί με τις προδιαγραφές WSGI (Web Server Gateway Interface) και ASGI (Asynchronous Server Gateway Interface) που καθορίζουν τον τρόπο επικοινωνίας μεταξύ εξυπηρετητών και web εφαρμογών Python για σύγχρονες και ασύγχρονες υπηρεσίες αντίστοιχα [54], [55].

Το Django υποστηρίζει τα πρωτόκολλα WSGI και ASGI και μπορεί να εκτελεστεί από συμβατούς εξυπηρετητές, όπως ο uWSGI για το WSGI και ο daphne για το ASGI. Συνεπώς, για την ενσωμάτωση του Django στο backend, έγινε εγκατάσταση των εξυπηρετητών uWSGI και daphne στον App Server (Εικόνα 8.1). Με τον τρόπο αυτό, οταν ο nginx λαμβάνει ένα αίτημα για τον App Server, το δορομιλογεί τους εξυπηρετητές uWSGI ή daphne ανάλογα με τον αν-

αφορά σύγχρονη ή ασύγχρονη υπηρεσία. Οι εξυπηρετητές αυτοί εκτελούν την λογική που έχει υλοποιηθεί στο περιβάλλον του Django για το εν λόγω αίτημα και επιστρέφουν την απάντηση που προκύπτει στον nginx.



**Εικόνα 8.1:** Εσωμάτωση του Django στον App Server [56], [57].

### 8.1.2 Βασικές Δυνατότητες

Όπως επισημάνθηκε, το Django και το DRF προσφέρουν πλήθος λειτουργιών για την αποτελεσματική ανάπτυξη υπηρεσιών. Ωστόσο, ορισμένες δυνατότητες είναι τόσο θεμελιώδεις που καθορίζουν τη δομή της υλοποίησης και χρησιμοποιούνται σε όλες τις υπηρεσίες. Αυτές οι βασικές δυνατότητες είναι οι εξής:

- Request Routing
- Django ORM
- Serializers

Στη συνέχεια, παρουσιάζονται συνοπτικά αυτές οι λειτουργίες, ώστε να αναδειχθούν τα κύρια χαρακτηριστικά της υλοποίησης των υπηρεσιών.

Η δυνατότητα Request Routing αναφέρεται στην διαδικασία δρομολόγησης κάθε εισερχόμενου αιτήματος προς την κατάλληλη συνάρτηση ή μέθοδο, προκειμένου να εκτελεστεί η προκαθορισμένη λογική και να επιστραφεί η απάντηση που προκύπτει. Στο Django, αυτή η διαδικασία βασίζεται σε ένα σύνολο κανόνων δρομολόγησης που αντιστοιχίζουν URLs σε views. Τα views είναι είτε συναρτήσεις είτε μέθοδοι κλάσεων που ορίζουν την επεξεργασία που εκτελείται σε ένα αίτημα και δημιουργούν την αντίστοιχη απάντηση. Η ουσία της ανάπτυξης υπηρεσιών στο πλαίσιο του Django έγκειται στην διαμόρφωση των κανόνων δρομολόγησης και στην υλοποίηση των views, ώστε σε κάθε αίτημα να πραγματοποιούνται οι απαιτούμενες λειτουργίες και να επιστρέφεται η επιθυμητή απάντηση.

To ORM (Object-Relational Mapping) είναι μια τεχνική που επιτρέπει τη διαχείριση του περιεχομένου σχεσιακών βάσεων δεδομένων με αντικειμενοστραφή τρόπο, διευκολύνοντας την αποθήκευση και ανάκτηση πληροφοριών μέσω κώδικα Python. To Django επιτρέπει τη σύνδεση με σχεσιακές βάσεις δεδομένων και την δημιουργία κλάσεων, γνωστών ως models, που αντιστοιχούν σε πίνακες της συνδεδεμένης βάσης. Κάθε model περιγράφει τη δομή ενός πίνακα, καθορίζει τα πεδία του, καθώς και τις σχέσεις του με άλλους πίνακες [58]. Επιπρόσθετα, το Django προσφέρει ένα σύνολο μεθόδων για τη διαχείριση δεδομένων, όπως την προσθήκη, την επεξεργασία, την διαγραφή και την ανάκτηση περιεχομένου από τους πίνακες, με τη χρήση αντικειμένων των models. Για παράδειγμα, στον κώδικα ορίζεται ένα model με πεδία τα δεδομένα των χαρτογραφήσεων και το Django δημιουργεί στη βάση τον αντίστοιχο πίνακα με τα πεδία και το όνομα του model (Εικόνα 8.2). Επακολούθως, η διαχείριση των χαρτογραφήσεων στη βάση πραγματοποιείται μέσω κώδικα Python, καλώντας μεθόδους επί των αντικειμένων του model. Με τον τρόπο αυτό, διευκολύνεται η αλληλεπίδραση με τη βάση δεδομένων χωρίς να απαιτείται άμεση συγγραφή και εκτέλεση προτάσεων SQL.

	Floodmap
o	id : int8 «PK»
	name : varchar(40)
	min_lat : float8
	min_lng : float8
	max_lat : float8
	max_lng : float8
	flood_date : timestampz
	days_before_flood : int4
	days_after_flood : int4
	owner_id : int8 «FK» -- User(id)

```
class Floodmap(models.Model):
    name = models.CharField(max_length=40)
    min_lat = models.FloatField()
    min_lng = models.FloatField()
    max_lat = models.FloatField()
    max_lng = models.FloatField()
    flood_date = models.DateTimeField()
    days_before_flood = models.IntegerField()
    days_after_flood = models.IntegerField()
    owner = models.ForeignKey(settings.AUTH_USER_MODEL,
        db_index=True, null=True, on_delete=models.SET_NULL)
```

**Εικόνα 8.2:** Αντιστοιχία μεταξύ του model (αριστερά) και του πίνακα της σχεσιακής βάσης δεδομένων (δεξιά) για τα δεδομένα των χαρτογραφήσεων.

Οι serializers είναι κλάσεις που ενσωματώνουν το ORM του Django και επιτρέπουν την μετατροπή δεδομένων μεταξύ αντικειμένων των models και ενσωματωμένων τύπων δεδομένων της Python [50]. Η ανάκτηση πληροφοριών από πίνακες της βάσης δεδομένων μέσω του ORM επιστρέφει αντικείμενα των αντίστοιχων models. Οι serializers μετατρέπουν αυτά τα αντικείμενα σε τύπους δεδομένων της Python, όπως dictionaries, επιτρέποντας την επεξεργασία τους και την ενσωμάτωσή τους σε απαντήσεις υπηρεσιών. Επιπλέον, οι serializers εκτελούν και την αντίστροφη διαδικασία, δηλαδή λαμβάνουν δεδομένα από dictionaries, τα επιβεβαιώνουν και τα

μετατρέπουν σε αντικείμενα των models, τα οποία μπορούν να αποθηκευτούν στη βάση δεδομένων. Αυτές οι διαδικασίες πραγματοποιούνται πολύ συχνά, ιδιαίτερα σε υπηρεσίες που απαιτούν είτε τη λήψη αποθηκευμένων πληροφοριών είτε την αποθήκευση νέων πληροφοριών από δεδομένα που υποβάλλουν οι χρήστες.

## 8.2 Celery

Προκειμένου να εκτελούνται οι εργασίες του FLOODPY ασύγχρονα από Worker components εκτός του περιβάλλοντος του App Server, χρησιμοποιήθηκε το κατανεμημένο σύστημα εκτέλεσης εργασιών Celery. Σύμφωνα με τις προδιαγραφές και την αρχιτεκτονική του backend, οι χρονοβόρες και υπολογιστικά απαιτητικές εργασίες του FLOODPY εκτελούνται ασύγχρονα σε ανεξάρτητα Worker components, ώστε να μην επιβαρύνεται η εξυπηρέτηση άλλων υπηρεσιών στον App Server. Για το σκοπό αυτό, ενσωματώθηκε στο backend το σύστημα Celery που επιτρέπει την διανομή φορτίου σε πολλαπλά μηχανήματα ή threads [59]. Το Celery υλοποιεί τις προδιαγραφές των Worker components μέσω της εκτέλεσης διεργασιών-εργατών, οι οποίες έχουν τη δυνατότητα να λαμβάνουν και να εκτελούν ασύγχρονα μακροχρόνιες εργασίες. Επιπλέον, λόγω της ενσωμάτωσης του Celery με το Django [60], οι διεργασίες-εργάτες έχουν πρόσβαση στις λειτουργίες αποθήκευσης και ανάκτησης δεδομένων μέσω του Django ORM. Συνεπώς, η υλοποίηση των Worker components μέσω των διεργασιών-εργατών του Celery επιτρέπει την ασύγχρονη εκτέλεση των εργασιών του FLOODPY και την άμεση αποθήκευση αποτελεσμάτων που προκύπτουν στη βάση δεδομένων.

Επιπλέον, το Celery παρέχει τη δυνατότητα εγκατάστασης ουράς μηνυμάτων, τα χαρακτηριστικά της οποίας αντιστοιχούν στις απαιτησεις του Tasks Queue component (Εικόνα 7.1). Η ουρά μηνυμάτων του Celery επιτρέπει την αποθήκευση αιτημάτων εργασιών έως ότου μια διεργασία-εργάτης καταστεί διαθέσιμη. Η πρώτη διαθέσιμη διεργασία-εργάτης λαμβάνει το αίτημα εργασία που εισήχθει πρώτο στην ουρά (FIFO) και προχωρά στην εκτέλεσή του. Η υποκείμενη τεχνολογία που χρησιμοποιείται για την ουρά μηνυμάτων είναι το Redis: ένα in-memory, key-value store που παρέχει εξαιρετικές επιδόσεις και δυνατότητες κλιμακωσμότητας [61]. Δεδομένων των υψηλών επιδόσεων και των δυνατοτήτων της, η ουρά μηνυμάτων του Celery χρησιμοποιήθηκε για την υλοποίηση του Tasks Queue component. Το Tasks Queue αξιοποιεί τις λειτουργίες της ουράς για την αποθήκευση των αιτημάτων εκτέλεσης

εργασιών του FLOODPY που υποβάλλονται από τον App Server και τη διάθεσή τους στους Workers, σύμφωνα με τις απαιτήσεις της αρχιτεκτονικής.

## 8.3 Channels

Για την αποστολή ενημερώσεων σχετικά με την πρόοδο των εργασιών, ενσωματώθηκε στο Django framework η βιβλιοθήκη Channels. Το Django δεν παρέχει ενσωματωμένη υποστήριξη για το πρωτόκολλο WebSocket. Ωστόσο, η ανάγκη για αποδοτική και έγκαιρη αποστολή μηνυμάτων στο frontend σχετικά με την εξέλιξη των εργασιών του FLOODPY, καθιστά απαραίτητη την υποστήριξη του εν λόγω πρωτοκόλλου. Επομένως, οι δυνατότητες του Django επεκτάθηκαν μέσω της βιβλιοθήκης Channels επιτρέποντας την υλοποίηση αμφίδρομης επικοινωνίας σε πραγματικό χρόνο με WebSocket.

Η βιβλιοθήκη Channels υποστηρίζει τη δημιουργία υπηρεσιών WebSocket μέσω της υλοποίησης κλάσεων γνωστών ως consumers. Οι consumers λειτουργούν παρόμοια με τα views του Django. Όταν μια υπηρεσία λαμβάνει ένα αίτημα, καλείται η μέθοδος που αντιστοιχεί στον τύπο του αιτήματος, η οποία ορίζεται στον consumer που είναι υπεύθυνος για την εξυπηρέτηση της συγκεκριμένης υπηρεσίας. Για παράδειγμα, έστω ότι ένα αίτημα εγκατάστασης αμφίδρομου διαύλου επικοινωνίας λαμβάνεται στην υπηρεσία `/ws/api/floodmaps/123/updates/`. Η υπηρεσία αυτή είναι υπεύθυνη για την αποστολή ενημερώσεων σχετικά με την πρόοδο της εργασίας του FLOODPY για την χαρτογράφηση με αναγνωριστικό αριθμό 123 και εξυπηρετείται από τον consumer `UpdatesConsumer`. Συνεπώς, κατά την λήψη του αιτήματος δημιουργείται ένα νέο αντικείμενο του consumer `UpdatesConsumer` και καλείται η μέθοδος `connect` για την εγκατάσταση του αμφίδρομου διαύλου επικοινωνίας. Ο consumer διαθέτει επιπλέον τις μεθόδους `send_json` για την αποστολή μηνυμάτων με περιεχόμενο JSON, `receive` για τη διαχείριση εισερχόμενων μηνυμάτων, και `disconnect` για την αποσύνδεση από τον δίαυλο [62].

Η ενσωμάτωση της βιβλιοθήκης Channels στο Django καθιστά επίσης δυνατή την επικοινωνία μεταξύ εξωτερικών διεργασιών και consumers μέσω ενός μηχανισμού που ονομάζεται channel layers. Ο μηχανισμός αυτός επιτρέπει σε διεργασίες να στέλνουν μηνύματα σε groups, από όπου οι consumers μπορούν να τα λαμβάνουν [63]. Για την προσωρινή αποθήκευση των μηνυμάτων στα groups χρησιμοποιείται το Redis, που λειτουργεί ως διαμεσολαβητής υψηλών επιδόσεων για την επικοινωνία μεταξύ διεργασιών και consumers.

Η δυνατότητα επικοινωνίας μεταξύ διεργασιών και consumers αξιοποιείται στο backend για την διαβίβαση ενημερώσεων σχετικά με την πρόοδο των εργασιών του FLOODPY από τους Workers προς τον App Server. Πιο συγκεκριμένα, οι Workers αποστέλλουν τις ενημερώσεις προόδου σε groups που αφορούν στις εκτελούμενες εργασίες. Αυτές οι ενημερώσεις αποθηκεύονται προσωρινά στο component Messages Broker της αρχιτεκτονικής (Εικόνα 7.1) που υλοποιείται μέσω Redis. Στη συνέχεια, οι ενημερώσεις λαμβάνονται από τους consumers που εκτελούνται στον App Server και διανέμονται σε πελάτες που παρακολουθούν τις εργασίες ως μηνύματα WebSocket. Η διαδικασία αυτή περιγράφεται αναλυτικά στο Κεφάλαιο 9.3.

## 8.4 GeoServer

Προκειμένου να ικανοποιηθούν οι προδιαγραφές του Product Store σχετικά με την αποθήκευση και τη διανομή των προϊόντων των εργασιών του FLOODPY, χρησιμοποιήθηκε ο εξυπηρετητής GeoServer. Τα προϊόντα που παράγονται από τις εργασίες είναι geo-registered εικόνες, δηλαδή εικόνες που περιλαμβάνουν τις γεωγραφικές συντεταγμένες των περιοχών που εμφανίζουν. Αυτές οι εικόνες αποθηκεύονται ως αρχεία GeoTIFF και παρουσιάζονται ως επιστρώσεις (layers) πάνω στο χάρτη του frontend. Ο GeoServer είναι ένας εξυπηρετητής γεωχωρικών δεδομένων [64], ο οποίος ενσωματώθηκε στο backend, καθώς επιτρέπει την αποθήκευση εικόνων GeoTIFF και τη διάθεση του περιεχομένου τους μέσω κατάλληλων υπηρεσιών.

Για την αποθήκευση και τον διαμοιρασμό των γεωχωρικών δεδομένων μιας χαρτογράφησης, ο GeoServer ακολουθεί μια διαδικασία που περιλαμβάνει τη δημιουργία workspaces, stores και layers [65]. Μετά την παραγωγή των προϊόντων από το FLOODPY, δημιουργείται στον GeoServer ένα workspace, το οποίο λειτουργεί ως φάκελος για την ομαδοποίηση των προϊόντων της χαρτογράφησης. Κάθε παραγόμενη εικόνα GeoTIFF εισάγεται στον GeoServer και αντιπροσωπεύονται από ένα store. Για την διάθεση των προϊόντων, από κάθε store δημιουργείται ένα layer που καθιστά τα γεωχωρικά δεδομένα του υποκείμενου GeoTIFF διαθέσιμα μέσω υπηρεσιών Web Map Service (WMS). Όλες αυτές οι διαδικασίες εκτελούνται αυτόματα από τους Workers, μέσω αποστολής αιτημάτων στο API διαχείρισης του GeoServer.

Οι υπηρεσίες WMS που προσφέρει κάθε layer αξιοποιούνται από το frontend για την προβολή των προϊόντων του FLOODPY σε έναν διαδραστικό χάρτη. Ειδικότερα, το WMS είναι

ένα πρότυπο που επιτρέπει τη λήψη geo-registered εικόνων μέσω αιτημάτων HTTP [66]. Κάθε αίτημα WMS περιλαμβάνει παραμέτρους όπως το ζητούμενο layer, τις συντεταγμένες της περιοχής ενδιαφέροντος και τον επιθυμητό τύπο εικόνας (π.χ. PNG, JPEG). Το frontend αποστέλλει αιτήματα WMS στον GeoServer μέσω της βιβλιοθήκης Leaflet για την λήψη των προϊόντων. Οι εικόνες που λαμβάνονται ενσωματώνονται ως επιστρώσεις στον διαδραστικό χάρτη του frontend.

## 9. Υπηρεσίες του Backend

Κατά την επεξεργασία εισερχόμενων αιτημάτων, κάθε υπηρεσία του backend ακολουθεί μια διαδικασία που έχει σχεδιαστεί ώστε να ανταποκρίνεται στις απαιτήσεις της αντίστοιχης λειτουργίας του frontend. Η υλοποίηση των διαδικασιών αυτών βασίζεται στην αλληλεπίδρασης των component που απαρτίζουν το backend, καθώς και στην αξιοποίηση των δυνατοτήτων που προσφέρουν οι ενσωματωμένες τεχνολογίες. Στην παρούσα ενότητα αναλύονται τα επιμέρους βήματα επεξεργασίας κάθε υπηρεσίας, ενώ παράλληλα περιγράφονται η αλληλεπίδραση μεταξύ των component και ο τρόπος με τον οποίο γίνεται χρήση των επιλεγμένων τεχνολογιών.

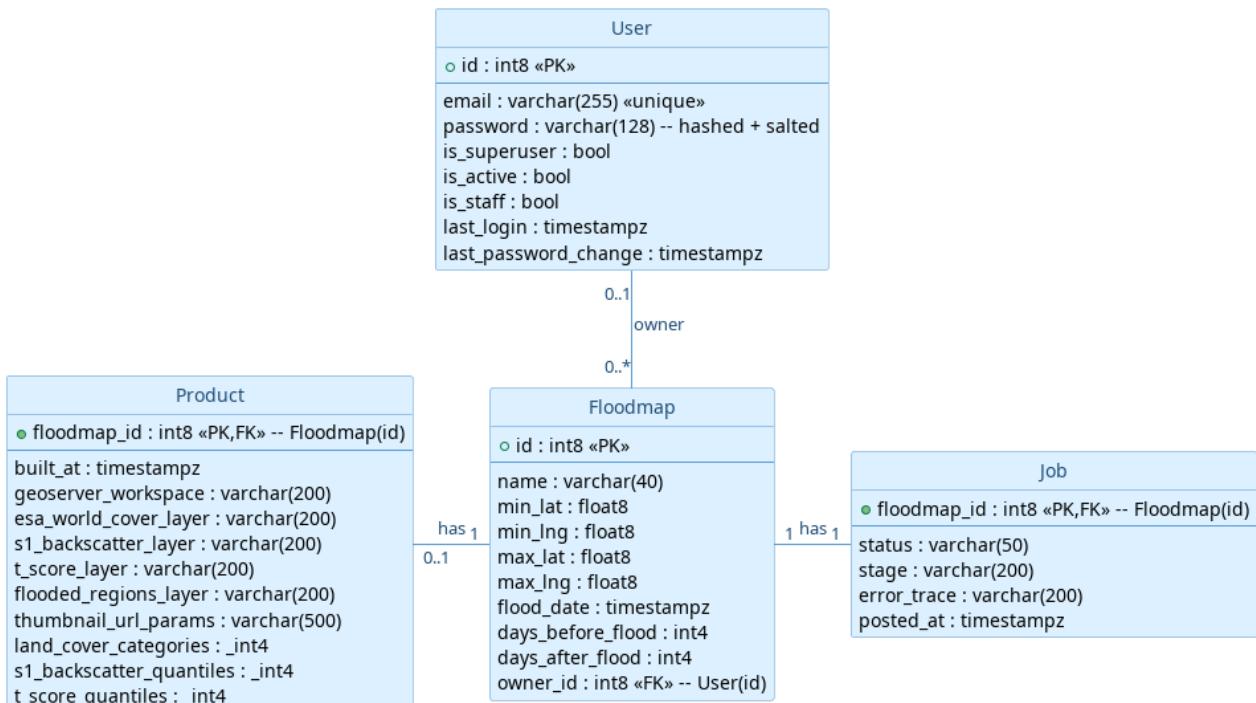
### 9.1 Διαχείριση Χαρτογραφήσεων

Μια από τις βασικές λειτουργίες το backend είναι η διαχείριση των δεδομένων των χαρτογραφήσεων. Τα δεδομένα κάθε χαρτογράφησης περιλαμβάνουν τα στοιχεία της πλημμύρας που χαρτογραφείται, καθώς και τις πληροφορίες της εργασίας του FLOODPY που παράγει το περιεχόμενο της χαρτογράφησης. Επιπλέον, αν η εργασία του FLOODPY έχει ολοκληρωθεί επιτυχώς, τα δεδομένα της χαρτογράφησης περιλαμβάνουν πληροφορίες για τα προϊόντα που δημιουργήθηκαν από την εργασία. Τα δεδομένα αυτά αποθηκεύονται στο Relational Database component, που βασίζεται στη σχεσιακή βάσης δεδομένων PostgreSQL, και το backend τα διαχειρίζεται μέσω του Django ORM.

Στη βάση PostgreSQL, τα δεδομένα οργανώνονται σε τρεις πίνακες, η δομή και το περιεχόμενο των οποίων φαίνονται στο διάγραμμα της Εικόνας 9.1. Για κάθε χαρτογράφηση, ο πίνακας `Floodmap` αποθηκεύει πληροφορίες της αντίστοιχης πλημμύρας, όπως το όνομά της, την ημερομηνία της και τις συντεταγμένες του bounding box της περιοχής που έπληξε. Στην περίπτωση που το αίτημα χαρτογράφησης υποβλήθηκε από αυθεντικοποιημένο χρήστη, τότε αποθηκεύεται στον πίνακα `Floodmap` και ο αναγνωριστικός αριθμός του χρήστη αυτού. Ταυτόχρονα, για κάθε πλημμύρα δημιουργείται ακριβώς μία εργασία του FLOODPY, οι πληροφορίες της οποίας αποθηκεύονται στον πίνακα `Job`. Αυτός ο πίνακας περιέχει δεδομένα όπως η κατάσταση και το στάδιο και η ημερομηνία δημιουργίας κάθε εργασίας του FLOODPY. Εφόσον μία εργασία ολοκληρωθεί, ο πίνακας `Products` αποθηκεύει μεταδεδομένα σχετικά με τα προϊόντα που παράχθηκαν. Αυτά τα μεταδεδομένα περιέχουν τις απαραίτητες πληροφορίες ώστε

το frontend, αφού τα λάβει, να μπορεί να αιτηθεί τη λήψη των αντίστοιχων προϊόντων από τον GeoServer και να τα απεικονίσει εντός διαδραστικού χάρτη (βλ. Κεφ. 9.2).

Για την διαχείριση των δεδομένων των χαρτογραφήσεων, το backend διαθέτει τρεις βασικές υπηρεσίες. Οι υπηρεσίες αυτές επιτρέπουν τη δημιουργία νέων χαρτογραφήσεων, καθώς και τη λήψη δεδομένων είτε για μία είτε για πολλαπλές χαρτογραφήσεις. Όπως όλες οι υπηρεσίες του backend, οι υπηρεσίες διαχείρισης χαρτογραφήσεων δέχονται και αποστέλλουν πληροφορίες κωδικοποιημένες σε μορφή JSON εντός HTTP μηνυμάτων. Επιπλέον, στο πλαίσιο εξυπηρέτησης αιτημάτων πραγματοποιούν αναγνώσεις και εγγραφές στους πίνακες **Floodmap** και **Job** της βάσης δεδομένων για την ανάκτηση και την αποθήκευση δεδομένων χαρτογραφήσεων. Οι λεπτομέρειες για τις συναλλαγές με τη βάση δεδομένων, καθώς και για τα επιμέρους βήματα που ακολουθούνται κατά την εξυπηρέτηση αιτημάτων από τις εν λόγω υπηρεσίες, περιγράφονται στις επόμενες ενότητες.



**Εικόνα 9.1:** Πίνακες της σχεσιακής βάσης δεδομένων του backend.

### 9.1.1 Δημιουργία Χαρτογράφησης

Η δημιουργία μιας νέας χαρτογράφησης προϋποθέτει την υποβολή ενός έγκυρου JSON με τις απαραίτητες παραμέτρους στην υπηρεσία `/api/floodmaps/` του backend. Όπως

περιγράφηκε στο Κεφάλαιο 5.1, το frontend διαθέτει μια σελίδα όπου οι χρήστες, μέσω ενός wizard, μπορούν να εισάγουν τα απαιτούμενα δεδομένα για τη δημιουργία μιας νέας χαρτογράφησης. Στο τελικό βήμα του wizard, τα δεδομένα που έχουν συλλεχθεί υποβάλλονται στην υπηρεσία `/api/floodmaps/` μέσω HTTP POST αιτήματος, με τη μορφή που φαίνεται στο Παράδειγμα 9.1. Το αίτημα δρομολογείται στο component App Server (Εικόνα 9.2), που υλοποιεί την λογική εξυπηρέτησης HTTP αιτημάτων σε Python και Django. Αν σώμα του μηνύματος δεν είναι έγκυρο JSON, η υπηρεσία απαντά άμεσα στο αίτημα με HTTP status κώδικα 400 (Bad Request). Σε διαφορετική περίπτωση, το Django, μέσω της λειτουργίας Request Routing (βλ. Κεφ. 8.1.2), καλεί την προκαθορισμένη μέθοδο του view που πραγματοποιεί την επεξεργασία του αιτήματος.

```
{
    "name": "Test Flood 1",
    "bbox": {
        "min_lat": 37.923,
        "min_lng": 23.622,
        "max_lat": 38.019,
        "max_lng": 23.842
    },
    "flood_date": "2024-10-08T00:00:00Z",
    "days_before_flood": 30,
    "days_after_flood": 2,
}
```

**Παράδειγμα 9.1:** JSON με τα δεδομένα που υποβάλλονται για την δημιουργία μιας νέας χαρτογράφησης.

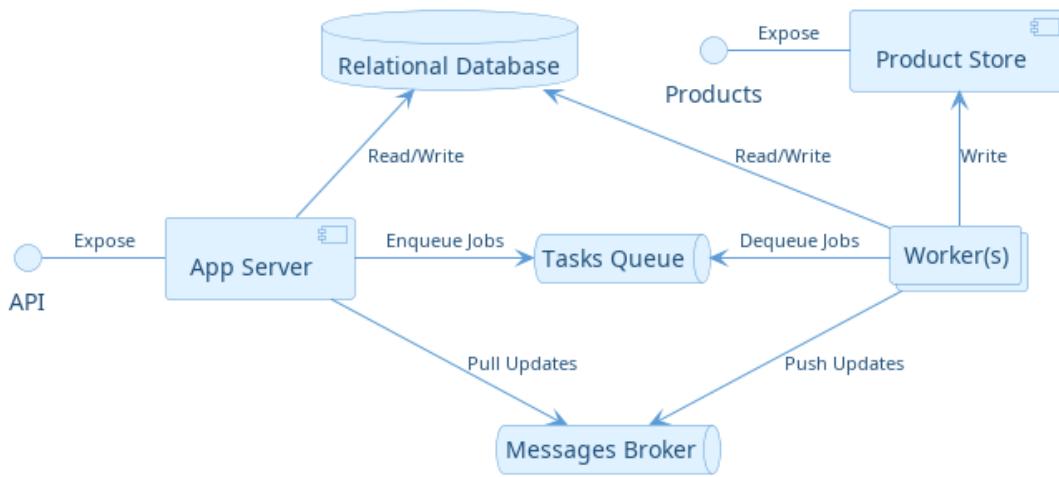
Η μέθοδος που διαχειρίζεται τα αιτήματα δημιουργίας νέων χαρτογραφήσεων αρχικά εισάγει τα δεδομένα του εισερχόμενου αιτήματος σε έναν serializer. Πράγματι, τα δεδομένα του αιτήματος, αφού μετατραπούν από το Django σε Python dictionary, χρησιμοποιούνται ως είσοδος στον `FloodmapSerializer`. Ο serializer αυτός ελέγχει την ορθότητα των υποβληθέντων δεδομένων, διασφαλίζοντας ότι όλα τα απαιτούμενα πεδία είναι διαθέσιμα και ότι οι τιμές τους είναι έγκυρες. Για παράδειγμα, οι έλεγχοι επιβεβαιώνουν τις τιμές των γεωγραφικών συντεταγμένων ώστε να ισχύουν οι συνθήκες:

```
-90 ≤ min_lat < max_lat ≤ 90
-180 ≤ min_lng < max_lng ≤ 180
```

καθώς και την τιμή της ημερομηνίας της πλημμύρας, η οποία πρέπει να βρίσκεται μεταξύ της ημερομηνίας εκτόξευσης του δορυφόρου Sentinel-1 και της τρέχουσας ημερομηνίας. Παρόλο που τα δεδομένα που υποβάλλονται υπόκεινται σε ελέγχους στο frontend, είναι απαραίτητο να επιβεβαιώνονται με ιδιαίτερη προσοχή και στο backend, διότι οι έλεγχοι του frontend γίνεται να παρακαμφούν.

Τα επόμενα βήματα της υπηρεσία καθορίζονται από το αποτέλεσμα του ελέγχου εγκυρότητας των υποβληθέντων δεδομενών. Αν τα δεδομένα δεν είναι έγκυρα, η υπηρεσία απαντά στο χρήστη με ένα μήνυμα που περιέχει λεπτομέρειες για τους ελέγχους που απέτυχαν, το οποίο έχει HTTP status κώδικα 400 (Bad Request). Αντίθετα, εάν τα δεδομένα είναι έγκυρα, αποθηκεύονται στη βάση δεδομένων. Ειδικότερα, ο `FloodmapSerializer` ανοίγει μια ατομική συναλλαγή με την βάση δεδομένων PostgreSQL που βρίσκεται στο Relational Database component και δημιουργεί από μια εγγραφή στους πίνακες `Floodmap` και `Job` μέσω του ORM. Για την εγγραφή στον πίνακα `Floodmap` χρησιμοποιούνται τα υποβληθέντα δεδομένα, ενώ προστίθεται και ο αναγνωριστικός αριθμός του χρήστη που υπέβαλε το αίτημα, εφόσον είναι αυθεντικοποιημένος. Η εγγραφή στον πίνακα `Job` περιέχει πληροφορίες για την εργασία του FLOODPY, από την εκτέλεση της οποίας μπορεί να δημιουργηθεί το περιεχόμενο της νέας χαρτογράφησης. Η χρήση της ατομική συναλλαγή εξασφαλίζει ότι αν οποιαδήποτε από τις δύο εγγραφές αποτύχει, η συναλλαγή ακυρώνεται συνολικά, αποτρέποντας την αποθήκευση ημιτελών δεδομένων χαρτογραφήσεων στη βάση.

Αφότου η νέα χαρτογράφηση καταχωρηθεί στη βάση δεδομένων, η υπηρεσία ελέγχει αν απαιτείται η άμεση εκτέλεση της εργασίας του FLOODPY για την παραγωγή του περιεχομένου της χαρτογράφησης. Σύμφωνα με όσα αναφέρθηκαν στο Κεφάλαιο 5.6, η εκτέλεση μιας εργασίας επιφέρει υψηλό υπολογιστικό κόστος και ως εκ τούτου απαιτεί την έγκριση διαχειριστή. Επομένως, η υπηρεσία δημιουργίας χαρτογραφήσεων εξετάζει τα δικαιώματα του λογαριασμού του χρήστη που υπέβαλε το αίτημα. Αν ο χρήστης έχει δικαιώματα διαχειριστή, η υπηρεσία ενημερώνει την εγγραφή που δημιούργησε στον πίνακα `Job`, ώστε να καταγραφεί ότι η εργασία έχει εγκριθεί από διαχειριστή. Πιο συγκεκριμένα, η τιμή του σταδίου (πεδίο *stage*) αλλάζει από *Pending approval* σε *Waiting in queue*. Στη συνέχεια, η υπηρεσία εκκινεί την εργασία, εισάγοντας το σχετικό αίτημα στο Tasks Queue (Εικόνα 9.2). Στην περίπτωση που ο χρήστης δεν είναι διαχειριστής, η εργασία του FLOODPY δεν εκτελείται και το στάδιο της εγγραφή στον πίνακα `Job` διατηρεί την τιμή *Pending approval*.



**Εικόνα 9.2:** Αρχιτεκτονική του backend.

Ανεξάρτητα από το αν η εργασία του FLOODPY εκτελείται άμεσα ή αναμένει έγκριση διαχειριστή, η υπηρεσία ενημερώνει τον χρήστη ότι η επεξεργασία του αιτήματος ολοκληρώθηκε επιτυχώς. Η απάντηση περιλαμβάνει ένα JSON που περιέχει όλες τις πληροφορίες της νέας χαρτογράφησης που έχουν αποθηκευτεί στους πίνακες Floodmap και Job. Στο Παράδειγμα 9.2 φαίνεται το περιεχόμενο της επιτυχημένης απάντηση που αντιστοιχεί στο αίτημα που παρουσιάζεται στο Παράδειγμα 9.1. Το μήνυμα της απάντησης έχει HTTP status κώδικα 201 (Created), υποδεικνύοντας ότι το αίτημα ολοκληρώθηκε επιτυχώς και ότι η νέα χαρτογράφηση καταχωρήθηκε.

```
{
  "id": 34,
  "name": "Test Flood 1",
  "bbox": {
    "min_lat": 37.923,
    "min_lng": 23.622,
    "max_lat": 38.019,
    "max_lng": 23.842
  },
  "flood_date": "2024-10-08T00:00:00Z",
  "days_before_flood": 30,
  "days_after_flood": 2,
  "job": {
    "status": "Progressing",
    "stage": "Pending approval",
    "posted_at": "2024-12-03T19:31:48.946338Z"
  }
}
```

}

**Παράδειγμα 9.2:** *JSON με τα δεδομένα μιας χαρτογράφησης, για την οποία η εργασία του FLOODPY βρίσκεται σε εξέλιξη.*

### 9.1.2 Λήψη Χαρτογράφησης

Το backend παρέχει επιπλέον μια υπηρεσία που επιτρέπει την λήψη των δεδομένων μιας επιλεγμένης χαρτογράφησης. Όταν ένας χρήστης επισκέπτεται τη σελίδα μιας χαρτογράφησης, το frontend απαιτείται να λάβει τα δεδομένα της από το backend (βλ. Κεφ. 5.2). Για το σκοπό αυτό, το backend επιτρέπει τη ανάκτηση των εν λόγω δεδομένων μέσω HTTP GET αιτήματος στο `/api/floodmaps/<id>/`, όπου `<id>` ο αναγνωριστικός αριθμός της επιθυμιτής χαρτογράφησης. Η υπηρεσία επιστρέφει τα δεδομένα υπό την μορφή JSON, όπως φαίνεται στο Παράδειγμα 9.2. Αν η εργασία του FLOODPY για την ζητούμενη χαρτογράφηση έχει ολοκληρωθεί επιτυχώς, τότε το JSON που επιστρέφει η υπηρεσία περιλαμβάνει επίσης τα μεταδεδομένα για τα προϊόντα που παρήγαγε η εργασία του FLOODPY. Η μορφή των μεταδεδομένων αυτών φαίνεται στο Παράδειγμα 9.3 και περιγράφεται στο Κεφάλαιο 9.2.

Για την αποστολή μιας χαρτογράφησης, η υπηρεσία διαβάζει τα διαθέσιμά δεδομένα από τη βάση και τα μετατρέπει στην κατάλληλη μορφή. Πιο συγκεκριμένα, η υπηρεσία εξυπηρετείται στο App Server component και τα δεδομένα ανακτώνται από το Relational Database component. Ο αναγνωριστικός αριθμός της χαρτογράφησης `<id>` χρησιμοποιείται για την αναζήτηση σχετικών εγγραφών στους πίνακες `Floodmap`, `Job` και `Product`. Η αναζήτηση είναι ιδιαίτερα αποδοτική, καθώς ο αναγνωριστικός αριθμός αποτελεί το primary key των πινάκων. Στην περίπτωση που οι πίνακες δεν διαθέτουν δεδομένα για την ζητούμενη χαρτογράφηση, η υπηρεσία απαντά με HTTP status κώδικα 404 (Not Found). Διαφορετικά, η αναζήτηση επιστρέφει ένα αντικείμενο του Django ORM που περιέχει τα αποθηκευμένα δεδομένα. Μέσω του `FloodmapSerializer`, η υπηρεσία εξάγει τα δεδομένα από το αντικείμενο σε ένα Python dictionary. Κατόπιν, το dictionary μετατρέπεται σε JSON και συμπεριλαμβάνεται στην απάντηση της υπηρεσίας, η οποία έχει HTTP status κώδικα 200 (OK).

```
"product": {  
    "built_at": "2024-11-11T11:57:50.075102Z",  
    "geoserver_workspace": "floodmap_8",  
    "esa_world_cover_layer": "esa_world_cover_layer_8",  
    "s1_backscatter_layer": "s1_backscatter_layer_8",  
    "t_score_layer": "t_score_layer_8",  
    "flooded_regions_layer": "flooded_regions_layer_8",  
}
```

```

    "thumbnail_url_params":  

"service=WMS&version=1.1.0&request=GetMap&layers=thumbnail_layer_group_185&bbox=23.622,37.923,  

23.842,38.019&width=568&height=247&srs=EPSG:4326&format=image/png",  

    "land_cover_categories": [10, 20, 30, 40, 50, 60, 80, 90],  

    "s1_backscatter_quantiles": [-17.937391147613525, 0.33864215552795707],  

    "t_score_quantiles": [-30.528865633010863, 67.16878479003913],  

}

```

**Παράδειγμα 9.3:** JSON με τα μεταδεδομένα των προϊόντων που παρήγαγε η εργασία του FLOODPY για μια χαρτογράφηση.

### 9.1.3 Λήψη Συνόλου Χαρτογραφήσεων

Εκτός από τη δυνατότητα ανάκτησης μιας επιλεγμένης χαρτογράφησης ανά αίτημα, το backend διαθέτει μια υπηρεσία που επιτρέπει την λήψη πολλαπλών χαρτογραφήσεων βάσει καθορισμένων φίλτρων αναζήτησης. Αυτή η υπηρεσία υποστηρίζει δύο σελίδες του frontend που προβάλλουν πολλαπλές χαρτογραφήσεις. Η πρώτη σελίδα παρουσιάζει όλες τις αποθηκευμένες χαρτογραφήσεις της εφαρμογής, για τις οποίες οι εργασίες του FLOODPY έχουν ολοκληρωθεί (βλ. Κεφ. 5.2), ενώ η δεύτερη σελίδα παρουσιάζει μόνο τις χαρτογραφήσεις του αυθεντικοποιημένου χρήστη (βλ. Κεφ. 5.5). Για την υποστήριξη αυτών των λειτουργιών, η υπηρεσία ανταποκρίνεται σε HTTP GET αιτήματα στο /api/floodmaps/, επιστρέφοντας δεδομένα σε σελιδοποιημένη μορφή, αποτελούμενα από χαρτογραφήσεις που ικανοποιούν τις παραμέτρους του εκάστοτε αιτήματος.

Κατά την εξυπηρέτηση ενός αιτήματος, η επεξεργασία ξεκινά με την εξαγωγή των φίλτρων αναζήτησης. Αφού το αίτημα δρομολογηθεί στον App Server, το Django καλεί την προκαθορισμένη μέθοδο του view της υπηρεσίας και της παρέχει τις παραμέτρους του αιτήματος σε ένα dictionary. Τότε, οι τιμές των παραμέτρων στο dictionary υποβάλλονται σε ελέγχους εγκυρότητας μέσω ενός serializer. Οι επιτρεπτές παράμετροι, η σημασία τους και οι έγκυρες τιμές που μπορούν να λάβουν, έχουν ως εξής:

- **flood\_name:** Συμβολοσειρά που πρέπει να περιέχουν τα ονόματα των ζητούμενων χαρτογραφήσεων. Αποτελείται από 2 έως 40 χαρακτήρες.
- **min\_lat, max\_lat:** Ελάχιστο και μέγιστο γεωγραφικό πλάτος που πρέπει να έχει η περιοχή ενδιαφέροντος των ζητούμενων χαρτογραφήσεων. Αριθμός μεταξύ -90 και 90.
- **min\_lng, max\_lng:** Ελάχιστο και μέγιστο γεωγραφικό μήκος που πρέπει να έχει η περιοχή ενδιαφέροντος των ζητούμενων χαρτογραφήσεων. Αριθμός μεταξύ -180 και 180.

- **from\_date, to\_date**: Ημερομηνίες μεταξύ των οποίων πρέπει να είναι οι ημερομηνίες των πλημμυρών των ζητούμενων χαρτογραφήσεων. Από 3 Απριλίου 2014 (εκτόξευση δορυφόρου Sentinel-1) έως τρέχουσα ημερομηνία.
- **owned**: Σημαία που καθορίζει αν οι ζητούμενες χαρτογραφήσεις απαιτείται να ανήκουν στον αυθεντικοποιημένο χρήστη ή όχι. Τιμές *true* ή *false*, με την τιμή *true* να προϋποθέτει ότι ο χρήστης είναι αυθεντικοποιημένος. Αν δεν οριστεί θεωρείται *false*.
- **succeeded, progressing, failed**: Σημαίες που καθορίζουν τις επιτρεπτές καταστάσεις που μπορούν να έχουν οι εργασίες του FLOODPY για τις ζητούμενες χαρτογραφήσεις. Τιμές *true* ή *false*. Αν δεν οριστούν, τα *progressing* και *failed* είναι *false*, ενώ το *succeeded* είναι *true*. Επομένως, από προεπιλογή η υπηρεσία επιστρέφει μόνο τις χαρτογραφήσεις με επιτυχημένες εργασίες.

Αν η τιμή κάποιας παραμέτρου δεν είναι έγκυρη, η υπηρεσία απαντά με ένα μήνυμα που επεξηγεί τον έλεγχο εγκυρότητας που απέτυχε και έχει HTTP status κώδικα 400 (Bad Request).

Στην περίπτωση που όλες οι παράμετροι του εισερχόμενου αιτήματος είναι έγκυρες, η υπηρεσία λαμβάνει τα δεδομένα των χαρτογραφήσεων από τη βάση δεδομένων και εφαρμόζει τα φίλτρα που ορίζονται από τις παραμέτρους του αιτήματος. Τα δεδομένα των χαρτογραφήσεων ανακτώνται μέσω του Django ORM από τους πίνακες `Floodmap`, `Job` και `Product` της βάσης δεδομένων. Στη συνέχεια, από τα αποτελέσματα αφαιρούνται οι χαρτογραφήσεις που δεν ικανοποιούν τις παραμέτρους του αιτήματος. Για την δημιουργία φίλτρων που αντιστοιχούν στις παραμέτρους και την εφαρμογή τους στα αποτελέσματα της αναζήτησης στη βάση δεδομένων, χρησιμοποιήθηκε η βιβλιοθήκη `django-filter` [67]. Η βιβλιοθήκη αυτή απλοποίησε σημαντικά την λογική της υλοποίησης, επιτρέποντας την δημιουργία φίλτρων μέσω του ορισμού μια εξειδικευμένης κλάσης, χωρίς να απαιτείται να οριστούν πολύπλοκες συναλλαγών με τη βάση δεδομένων.

Στο τελευταίο βήμα της υπηρεσίας, τα φίλτραρισμένα δεδομένα σελιδοποιούνται και μετατρέπονται στην κατάλληλη μορφή για αποστολή. Προκειμένου να περιοριστεί ο όγκος των δεδομένων που επιστρέφει η υπηρεσία με κάθε αίτημα, το σύνολο των χαρτογραφήσεων που προκύπτει μετά από την εφαρμογή των φίλτρων χωρίζεται σε σελίδες, καθεμία από τις οποίες περιλαμβάνει έως 12 χαρτογραφήσεις. Η υπηρεσία υποστηρίζει την παράμετρο `page` μέσω της οποίας μπορεί να επιλεχθεί η ζητούμενη σελίδα. Για την υλοποίηση αυτής της λειτουργίας χρησιμοποιήθηκε η κλάση σελιδοποίησης (pagination class) από το Django Rest Framework

[48]. Μέσω αυτής της κλάσης, από τα φιλτραρισμένα δεδομένα επιλέγεται η σελίδα – δηλαδή η δωδεκάδα εγγραφών – που ορίζεται στην παράμετρο `page`. Ταυτόχρονα, συμπεριλαμβάνεται στην απάντηση ένας αριθμός που εκφράζει το συνολικό πλήθος των αποτελεσμάτων (Παράδειγμα 5.4, πεδίο `count`), ο οποίος επιτρέπει την σελιδοποίηση στο frontend (βλ. Κεφ. 5.3). Το τελικό αποτέλεσμα μετατρέπεται από αντικείμενο του Django ORM σε Python dictionary μέσω του `FloodmapSerializer` και αποστέλλεται εντός της απάντησης της υπηρεσίας με HTTP status κώδικα 200 (OK).

```
{
    "count": 16,
    "next": "http://127.0.0.1:8000/api/floodmaps/?page=2",
    "previous": null,
    "results": [
        {
            "id": 1,
            "name": "Test Flood 1",
            ...
        },
        {
            "id": 2,
            "name": "Test Flood 2",
            ...
        },
        ...
        {
            "id": 12,
            "name": "Test Flood 12",
            ...
        },
    ]
}
```

**Παράδειγμα 9.4:** JSON με σελιδοποιημένα δεδομένα αποθηκευμένων χαρτογραφήσεων.

## 9.2 Εκτέλεση Εργασίας FLOODPY

Η εκτέλεση της εργασίας του FLOODPY για μια χαρτογράφηση μπορεί να προκληθεί με δύο τρόπους. Ο πρώτος τρόπος είναι η αυτόματη εκκίνησή της κατά την δημιουργία της χαρτογράφησης και συμβαίνει μόνο για χαρτογραφήσεις που δημιουργούνται από διαχειριστές. Ο δεύτερος τρόπος αφορά κάθε άλλη χαρτογράφηση και προϋποθέτει την έγκριση της εργασίας από κάποιο διαχειριστή.

Για την υλοποίηση του δεύτερου τρόπου, το backend διαθέτει μια υπηρεσία στην οποία έχουν πρόσβαση μόνο οι διαχειριστές. Η υπηρεσία αυτή επιτρέπει την έγκριση και την απόρριψη εργασιών του FLOODPY με την αποστολή HTTP PATCH αιτημάτων στο `/api/jobs/<id>/`, όπου `<id>` ο αναγνωριστικός αριθμός της σχετικής χαρτογράφησης. Κάθε αίτημα που αποστέλλεται στην υπηρεσία δρομολογείται στον App Server, όπου καλείται η προκαθορισμένη μέθοδος `εξυπηρέτησης`. Τότε, η υπηρεσία διαπιστώνει αν το αίτημα προέρχεται από χρήστη με δικαιώματα διαχειριστή. Στην περίπτωση που ο χρήστης δεν είναι διαχειριστής, η υπηρεσία επιστρέφει ένα μήνυμα με HTTP status κώδικα 401 (Unauthorized), χωρίς να τροποποιήσει την κατάσταση της εργασίας. Διαφορετικά, η υπηρεσία εξετάζει το σώμα του αιτήματος, το οποίο πρέπει να είναι ένα JSON που να περιέχει την τιμή `approved` ως `true` για έγκριση της εργασίας και ως `false` για απόρριψή της. Αν το σώμα του αιτήματος έχει οποιαδήποτε άλλη μορφή, η υπηρεσία ανταποκρίνεται με ένα μήνυμα που έχει HTTP status κώδικα 400 (Bad Request) και περιγράφει το αναμενόμενο περιεχόμενο του αιτήματος.

Για έγκυρα αιτήματα έγκριση ή απόρριψη, η υπηρεσία ενημερώνει ανάλογα την κατάσταση της εργασίας του FLOODPY. Ειδικότερα, μέσω του Django ORM και του αναγνωριστικού αριθμού `<id>`, η υπηρεσία διαβάζει την εγγραφή του πίνακα `Job` για την χαρτογράφηση από τη βάση δεδομένων. Για αιτήματα απόρριψης, το πεδίο `status` της εγγραφής λαμβάνει την τιμή `Failed`, ενώ για αιτήματα έγκρισης το πεδίο `stage` μεταβαίνει από την τιμή `Pending approval` στην τιμή `Waiting in queue` και το πεδίο `status` διατηρεί την τιμή `Progressing`. Επιπρόσθετα, για αιτήματα έγκρισης, μετά την ολοκλήρωση της συναλλαγής με τη βάση δεδομένων η υπηρεσία εκκινεί ασύγχρονα την εκτέλεση της εργασία του FLOODPY. Σε κάθε περίπτωση, η υπηρεσία επιστρέφει ένα κενό μήνυμα με HTTP status κώδικα 204 (No Content).

Η εκτέλεση των εργασιών του FLOODPY, είτε προκαλείται αυτόματα είτε έπειτα από έγκριση, πραγματοποιείται ασύγχρονα εκτός του App Server. Πράγματι, κατά την επεξεργασία ενός αιτήματος που προκαλεί την εκτέλεση μιας εργασίας, η υπηρεσία στέλνει ένα μήνυμα με τον αναγνωριστικό αριθμό της σχετικής χαρτογράφηση από τον App Server προς την ουρά Tasks Queue (Εικόνα 9.2). Η αποστολή του μηνύματος υλοποιήθηκε μέσω της βιβλιοθήκης Celery, ενώ για την αποθήκευσή του στο Tasks Queue χρησιμοποιήθηκε το Redis (βλ. Κεφ. 8.2). Στη συνέχεια, η υπηρεσία ανταποκρίνεται στο αίτημα χωρίς να αναμένει την ολοκλήρωση της εργασίας του FLOODPY. Κατά συνέπεια, το φορτίο στον App Server μειώνεται, καθώς το

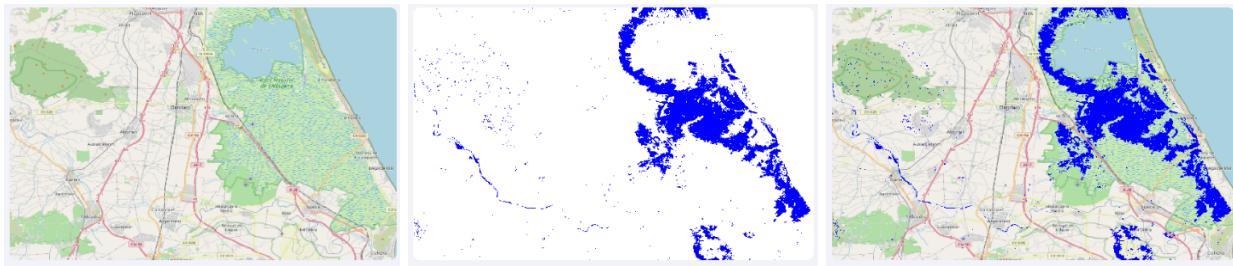
thread που επεξεργάζεται το αίτημα δεν επιβαρύνεται με την εκτέλεση της εργασίας ούτε δεσμεύεται μέχρι την ολοκλήρωσή της.

Η εκτέλεση των εργασιών του FLOODPY πραγματοποιείται από τα Worker components. Οι Workers είναι ανεξάρτητης διεργασίες του Celery που έχουν πρόσβαση στη βάση δεδομένων και στο ORM του Django. Όταν ένας Worker είναι διαθέσιμος αφαιρεί το πρώτο μήνυμα από την ουρά Tasks Queue και διαβάζει τον αναγνωριστικό αριθμό της χαρτογράφησης που περιέχει. Χρησιμοποιώντας τον αναγνωριστικό αριθμό και το ORM, ο Worker λαμβάνει τις παραμέτρους της εν λόγω χαρτογράφησης από τη βάση δεδομένων και τις παρέχει σε μία κλάση οδηγό που επιτρέπει την εκτέλεση των βημάτων του FLOODPY για τις δοσμένες παραμέτρους. Η κλάση οδηγός διαθέτει μια μέθοδο για κάθε βήμα επεξεργασίας του FLOODPY και ο Worker καλεί τις μεθόδους αυτές είτε μέχρι να ολοκληρωθεί η εκτέλεση του FLOODPY είτε μέχρι να συμβεί κάποιο σφάλμα. Επιπλέον, ανάλογα με το αποτέλεσμα κάθε μεθόδου – δηλαδή κάθε βήματος της εργασίας – ο Worker ενημερώνει τα πεδία *status* και *stage* του πίνακα *Job*. Επομένως, η εγγραφή της χαρτογράφησης στον πίνακα *Job* περιέχει κάθε στιγμή την ακριβή κατάσταση της εργασίας του FLOODPY.

Η κλάση οδηγός εκτελεί εργασίες του FLOODPY υλοποιώντας μια τροποποιημένη και επαυξημένη έκδοση της λογικής που περιέχει το αποθετήριο του FLOODPY [68]. Όπως αναφέρθηκε στο Κεφάλαιο 2.3, η εκτέλεση του FLOODPY γίνεται μέσω κώδικα Python στο περιβάλλον ενός Jupyter Notebook. Η κλάση οδηγός ενσωματώνει αυτόν τον κώδικα, ώστε να εκτελείται στο περιβάλλον του backend από τους Workers. Ωστόσο, αντί να απεικονίζει τα προϊόντα που παράγονται εντός του Jupyter Notebook, τα μετατρέπει σε εικόνες GeoTIFF, τις οποίες αποθηκεύει ώστε να είναι διαθέσιμες στον GeoServer του Product Store component (Εικόνα 9.2). Επίσης, η κλάση οδηγός, για κάθε χαρτογράφηση, δημιουργεί στον GeoServer ένα workspace μέσω κλήσεων στο REST API του. Το workspace μιας χαρτογράφησης περιέχει ένα store και ένα layer για κάθε εικόνα GeoTIFF που δημιουργείται από το FLOODPY για την χαρτογράφηση αυτή.

Επιπλέον της εκτέλεσης του FLOODPY και της αποθήκευσης των προϊόντων του, η κλάση οδηγός δημιουργεί και μία εικόνα προεπισκόπησης (thumbnail) για κάθε χαρτογράφηση. Για το σκοπό αυτό χρησιμοποιήθηκε η βιβλιοθήκη contextily [69]. Μέσω αυτής της βιβλιοθήκης γίνεται λήψη ψηφίδων (tiles) του χάρτη του Open Street Map για την περιοχή ενδιαφέροντος. Η εικόνα της περιοχής ενδιαφέροντος που σχηματίζουν οι ψηφίδες μετατρέπεται σε GeoTIFF και

αποθηκεύεται στον GeoServer μαζί με τα προϊόντα του FLOODPY. Για την παραγωγή της εικόνας προεπισκόπησης της χαρτογράφησης δημιουργείται στον GeoServer ένα layer group που συνδυάζει το layer της εικόνας της περιοχής ενδιαφέροντος και το layer των πλημμυρισμένων περιοχών του παράγει το FLOODPY. Παρακάτω φαίνεται ένα παράδειγμα των δύο layers και του layer group που τα συνδυάζει (Εικόνα 9.3). Το layer group χρησιμοποιείται για τη λήψη της εικόνας προεπισκόπησης που παρουσιάζεται στην κάρτα κάθε χαρτογράφησης (βλ. Κεφ. 5.3).



**Εικόνα 9.3:** Παράδειγμα συνδυασμού layers για την δημιουργία προεπισκόπησης πλημμύρας.

Αποτελείται από τα layers της περιοχής ενδιαφέροντος (αριστερά) και των πλημμυρισμένων περιοχών (κέντρο) που συνδυάζονται στο layer group της προεπισκόπησης (δεξιά)

Προκειμένου η εικόνα προεπισκόπησης και τα προϊόντα που παράγονται για κάθε χαρτογράφηση να μπορούν να ληφθούν από το frontend, η κλάση οδηγός αποθηκεύει μεταδεδομένα στον πίνακα Product. Τα μεταδεδομένα που αποθηκεύονται για κάθε χαρτογράφηση είναι τα ακόλουθα:

- Το όνομα του workspace στον GeoServer.
- Τα ονόματα των layers στον GeoServer για τις απεικονίσεις των πλημμυρισμένων περιοχών, των αλλαγών του t-score, των συντελεστών οπισθοσκέδασης, και της κάλυψης του εδάφους ESA WorldCover.
- Τις παραμέτρους για τη λήψη της εικόνας προεπισκόπησης.
- Τις κατηγορίες της απεικόνισης της κάλυψης του εδάφους WorldCover για επεξήγηση των χρωμάτων.
- Τις ακραίες τιμές των απεικονίσεων με τις αλλαγές του t-score και με τους συντελεστές οπισθοσκέδασης, για την εμφάνιση υπομνημάτων με το gradient των χρωμάτων των εν λόγω απεικονίσεων.

Τα παραπάνω μεταδεδομένα είναι απαραίτητα για την παρουσίαση των χαρτογραφήσεων στο frontend και συμπεριλαμβάνονται στις απαντήσεις της υπηρεσίας λήψης χαρτογράφησης (Παράδειγμα 9.3).

Αξίζει να σημειωθεί ότι η σχεδίαση των Worker components υποστηρίζει τη χρήση πολλαπλών κλάσεων οδηγών. Συγκεκριμένα, για τον έλεγχο της λειτουργίας της εφαρμογής υλοποιήθηκε μια δεύτερη κλάση οδηγός που προσομοιώνει την εκτέλεση του FLOODPY. Επιπλέον, μέσω της ανάπτυξης νέων κλάσεων οδηγών διευκολύνεται η μελλοντική ενσωμάτωση πρόσθετων προγραμμάτων χαρτογράφησης φυσικών καταστροφών, τα οποία μπορούν να εκτελούνται παράλληλα με το FLOODPY. Αρκεί να δημιουργηθεί μια κλάση οδηγός που να επιτρέπει την εκτέλεση του ζητούμενου προγράμματος χαρτογράφησης στο περιβάλλον των Worker components. Η σχεδίαση αυτή επιτρέπει την εύκολη επέκταση της web εφαρμογής, ώστε να παραμένει επίκαιρη και συμβαδίζει με τις εξελίξεις στον τομέα της χαρτογράφησης φυσικών καταστροφών.

### 9.3 Αποστολή Ενημερώσεων

Το backend διαθέτει μια υπηρεσία WebSocket που επιτρέπει την αποστολή ενημερώσεων αναφορικά με την κατάσταση της εργασίας του FLOODPY κάθε χαρτογράφησης. Όταν ένας χρήστης επισκέπτεται την σελίδα μιας χαρτογράφησης, της οποίας η εργασία του FLOODPY βρίσκεται σε εξέλιξη, το frontend ανοίγει μία σύνδεση WebSocket με το backend ώστε λάβει ενημερώσεις για την κατάσταση της εργασίας (βλ. Κεφ. 5.2.1). Για το σκοπό αυτό, το backend διαθέτει μια υπηρεσία WebSocket στο /ws/api/floodmaps/<id>/updates/, όπου <id> ο αναγνωριστικός αριθμός της χαρτογράφησης για την οποία ζητούνται ενημερώσεις.

Όταν ένα αίτημα σύνδεσης λαμβάνεται στην υπηρεσία WebSocket δρομολογείται στην προκαθορισμένη μέθοδο εξυπηρέτησης, η οποία υλοποιήθηκε μέσω της βιβλιοθήκης Channels (βλ. Κεφ. 8.3) και εκτελείται στον App Server. Κατά την επεξεργασία του αιτήματος σύνδεσης, το πρώτο βήμα της υπηρεσίας είναι να αποδεχτεί την σύνδεση και να διαβάσει την κατάσταση της σχετικής εργασίας του FLOODPY από τη βάση δεδομένων. Για την ανάκτηση της κατάσταση, η υπηρεσία χρησιμοποιεί το Django ORM και τον αναγνωριστικό αριθμό <id> που περιέχει το αίτημα, και λαμβάνει από τον πίνακα Job την σχετική εγγραφή. Ακολούθως, η υπηρεσία στέλνει ένα μήνυμα WebSocket στο frontend με την κατάσταση της εργασίας, η οποία προκύπτει κυρίως στα περιεχόμενα των πεδίων status και stage της εγγραφής του πίνακα Job.

Αν η εργασία δεν βρίσκεται σε εξέλιξη, η υπηρεσία μετά την πρώτη ενημέρωση κλείνει την σύνδεση WebSocket, καθώς η κατάσταση της εργασίας δεν αναμένεται να αλλάξει και συνεπώς δεν χρειάζεται να αποσταλούν περεταίρω ενημερώσεις στο frontend. Διαφορετικά, η υπηρεσία διατηρεί την σύνεση WebSocket και εγγράφεται σε ένα κανάλι του Messages Broker component, από το οποίο λαμβάνει μηνύματα για όταν μεταβάλλεται η κατάσταση της ζητούμενης εργασίας του FLOODPY. Για κάθε μήνυμα στο εν λόγω κανάλι, η υπηρεσία στέλνει την αντίστοιχη ενημέρωση στο frontend μέσω της σύνδεσης WebSocket. Στην περίπτωση που ληφθεί από το κανάλι του Messages Broker μήνυμα που δηλώνει ότι η εργασία απέτυχε ή ολοκληρώθηκε, η υπηρεσία αφού στείλει τη σχετική ενημέρωση το frontend κλείνει στην σύνδεση WebSocket.

Τα μηνύματα που λαμβάνει η υπηρεσία WebSocket από τον Messages Broker υποβάλλονται από τους Workers που εκτελούν τις εργασίες του FLOODPY. Πιο συγκεκριμένα, ο Messages Broker διαθέτει ένα κανάλι για κάθε εκτελούμενη εργασία του FLOODPY. Κάθε φορά που η κατάσταση μιας εργασίας μεταβάλλεται, ο Worker που την εκτελεί υποβάλλει ένα μήνυμα στον Messages Broker με την νέα κατάσταση της εργασίας. Τα μηνύματα που υποβάλλονται στον Messages Broker λαμβάνονται από τον App Server και προωθούνται στο frontend από την υπηρεσία WebSocket. Η υλοποίηση των λειτουργιών υποβολής και λήψης μηνυμάτων από και προς τον Messages Broker βασίστηκε στις δυνατότητες της βιβλιοθήκης Channels, ενώ για την αποθήκευση μηνυμάτων στον Messages Broker χρησιμοποιήθηκε το Redis.

## 9.4 Δημιουργία και Διαχείριση Λογαριασμού

Καθοριστική σημασία για την ασφάλεια της εφαρμογής έχουν οι λειτουργίες του backend που επιτρέπουν τη δημιουργία και τη διαχείριση λογαριασμών χρηστών. Το frontend διαθέτει σελίδες μέσω των οποίων κάθε χρήστης μπορεί να εγγραφεί στην εφαρμογή και να διαχειριστεί τον λογαριασμό του (βλ. Κεφ. 5.4). Για την υποστήριξη των εν λόγω σελίδων, το backend διαθέτει το ακόλουθο σύνολο υπηρεσιών:

- /auth/users/
  - Δημιουργία νέου λογαριασμού μέσω HTTP POST αιτήματος που περιέχει τη διεύθυνση email και τον κωδικό πρόσβασης του χρήστη που εγγράφεται. Αποστέλλει email επιβεβαίωσης που περιέχει ένα ζεύγος αναγνωριστικών token και UID. Δημιουργεί μια εγγραφή στον πίνακα User (Εικόνα 9.1).

- Διαγραφή λογαριασμού μέσω HTTP DELETE αιτήματος που περιέχει τον κωδικό πρόσβασης του χρήστη. Προϋποθέτει ότι ο χρήστης είναι αυθεντικοποιημένος. Αφαιρεί την σχετική εγγραφή από τον πίνακα User.
- **/auth/users/activation/**  
 Ενεργοποίηση νέου λογαριασμού μέσω HTTP POST αιτήματος που περιέχει το token και το UID που έχουν αποσταλεί στο χρήστη μέσω email μετά το αίτημα δημιουργίας λογαριασμού.
- **/auth/users/set\_password/**  
 Άλλαγή κωδικού πρόσβασης μέσω HTTP POST αιτήματος που περιέχει τον τρέχον κωδικό πρόσβασης του χρήστη. Προϋποθέτει ότι ο χρήστης είναι αυθεντικοποιημένος.
- **/auth/users/reset\_password/**  
 Επαναφορά κωδικού πρόσβασης μέσω HTTP POST αιτήματος που περιέχει τη διεύθυνση email του χρήστη. Αποστέλλει email επιβεβαίωσης που περιέχει ένα ζεύγος αναγνωριστικών token και UID.
- **/auth/users/reset\_password\_confirm/**  
 Επιβεβαίωση επαναφοράς κωδικού πρόσβασης μέσω HTTP POST αιτήματος που περιέχει το νέο κωδικό πρόσβασης, καθώς και τα token και UID που έχουν αποσταλεί στο χρήστη μέσω email μετά το αίτημα επαναφοράς του κωδικού πρόσβασης.

Για την ασφαλή υλοποίηση των παραπάνω υπηρεσιών χρησιμοποιήθηκε η βιβλιοθήκη Djoser [70]. Αυτή η βιβλιοθήκη εγκαταστάθηκε στον App Server στο περιβάλλον του Django και παρέχει όλη την απαραίτητη λειτουργικότητα των υπηρεσιών που παρουσιάστηκαν. Μέσω της ενσωμάτωσης μιας ολοκληρωμένη λύσης για τις υπηρεσίες δημιουργίας και διαχείρισης λογαριασμών επιτυγχάνεται ένα υψηλό επίπεδο ασφάλειας. Πράγματι, με την χρήση του Djoser αποφεύγεται η εκ νέου υλοποίηση κρίσιμων λειτουργιών όπως ο élégeγχος και η αποθήκευση κωδικών πρόσβασης. Αντιθέτως, οι λειτουργίες αυτές παρέχονται έτοιμες από το Djoser, το οποίο υποστηρίζεται από μια ευρεία κοινότητα μηχανικών και έχει ελεγχθεί στο περιβάλλον παραγωγής πολλών web εφαρμογών, όπως προκύπτει από τον υψηλό αριθμό forks και stars του αποθετηρίου της βιβλιοθήκης στο GitHub [71].

## 9.5 Αυθεντικοποίηση

Προκειμένου κάθε εγγεγραμμένος χρήστης να έχει τη δυνατότητα σύνδεσης στην εφαρμογή με τον λογαριασμό του, το backend διαθέτει μια υπηρεσία αυθεντικοποίησης. Η υπηρεσία αυτή εξυπηρετεί HTTP POST αιτήματα στο `/auth/users/login/`, τα οποία αποστέλλονται από την σελίδα σύνδεσης του frontend και περιέχουν τα διαπιστευτήρια του χρήστη. Κάθε αίτημα δρομολογείται στον App Server, όπου η υπηρεσία ελέγχει τα απεσταλμένα διαπιστευτήρια μέσω σύγκρισης με τα δεδομένα του πίνακα `User`. Στην περίπτωση που τα διαπιστευτήρια δεν βρεθούν στην βάση δεδομένων, η υπηρεσία επιστρέφει ένα μήνυμα αποτυχίας αυθεντικοποίησης με HTTP status κώδικα 401 (Unauthorized). Διαφορετικά, η υπηρεσία εκδίδει για τον χρήστη ένα ζεύγος access και refresh tokens, τα υπογράφει ώστε να μην μπορούν να αλλοιωθούν, και τα συμπεριλαμβάνει στην απάντηση. Ειδικότερα, η απάντηση έχει HTTP status κώδικα 200 (OK) και περιέχει το access token στο σώμα σε μορφή JSON και το refresh token ως HTTP Only cookie. Για να εξασφαλιστεί ότι η υπηρεσία ακολουθεί αξιόπιστες πρακτικές ασφαλείας, ο έλεγχος των διαπιστευτηρίων πραγματοποιείται από το Django και η έκδοσης των tokens γίνεται μέσω της βιβλιοθήκης Simple JWT [51].

Τα access και refresh tokens χρησιμοποιούνται για την διατήρηση της ταυτότητας του χρήστη μέχρι την αποσύνδεσή του. Όπως αναλύθηκε στο Κεφάλαιο 5.4.2, κάθε αίτημα του frontend μετά την αυθεντικοποίηση του χρήστη συμπεριλαμβάνει το access token στο Authorization HTTP header. Επομένως, το backend μπορεί να διαπιστώσει αν ένα εισερχόμενο αίτημα προέρχεται από τον αυθεντικοποιημένο χρήστη, επιβεβαιώνοντας την υπογραφή του access token και ελέγχοντας το χρήστη για τον οποίο έχει εκδοθεί. Επιπλέον, στην περίπτωση που το access token έχει λήξει, το frontend το ανανεώνει αποστέλλοντας HTTP POST αίτημα στην υπηρεσία `/auth/users/refresh/`. Στη συγκεκριμένη υπηρεσία, το backend επαληθεύει την εγκυρότητα του refresh token που περιλαμβάνεται ως HTTP Only cookie στο αίτημα και, εφόσον είναι έγκυρο, αποστέλλει ένα νέο access token στο χρήστη. Τέλος, κατά την αποσύνδεση του χρήστη, το frontend στέλνει ένα HTTP POST αίτημα στην υπηρεσία `/auth/users/logout/`, όπου το backend αφαιρεί το refresh token από τα cookies. Μετά την αποσύνδεση, το frontend δεν συμπεριλαμβάνει το access token σε επόμενα αιτήματα.

Η διαδικασία αυθεντικοποίησης που υλοποιήθηκε είναι εξαιρετικά αξιόπιστη και έχει σχεδιαστεί με τρόπο που προστατεύει την εφαρμογή από συνήθη κενά ασφαλείας. Πιο συγκεκριμένα, συμπεριλαμβάνει μέτρα που επιτρέπουν την ακύρωση tokens που έχουν εκδοθεί

και αποτρέπουν επιτυχώς τις επιθέσεις Cross-Site Request Forgery (CSRF) και Cross-Site Scripting (XSS).

Για την προστασία της εφαρμογής από επιθέσεις CSRF, τα access tokens δεν αποθηκεύονται στα cookies. Οι επιθέσεις CSRF συμβαίνουν όταν μια κακόβουλη ιστοσελίδα εκμεταλλεύεται διαπιστευτήρια αποθηκευμένα σε cookies για να πραγματοποιήσει ανεπιθύμητες ενέργειες σε έναν ιστότοπο στον οποίο ο χρήστης έχει αυθεντικοποιηθεί [72]. Για παράδειγμα, μια κακόβουλη ιστοσελίδα μπορεί να περιέχει μια κρυφή φόρμα που υποβάλλεται αυτόματα και αποστέλλει ένα αίτημα στην υπηρεσία δημιουργίας νέας χαρτογράφησης του backend. Επειδή τα cookies συμπεριλαμβάνονται σε κάθε αίτημα, κατά την επίσκεψη ενός αυθεντικοποιημένου χρήστη στην κακόβουλη ιστοσελίδα το αίτημα από την κρυφή φόρμα περιέχει τα cookies του χρήστη. Συνεπώς, αν το access token αποθηκευταν στα cookies, το αίτημα θα θεωρούταν ότι προέρχεται από τον χρήστη και στην περίπτωση που ο χρήστης έχει δικαιώματα διαχειριστή η αντίστοιχη εργασία του FLOODPY θα εκτελούνταν αυτόματα, οδηγώντας σε σπατάλη πόρων. Ωστόσο, το access token αποστέλλεται μέσω του Authorization HTTP header και δεν αποθηκεύεται στα cookies, ενώ το refresh token που αποθηκεύονται στα cookies δεν επαρκεί για την εκτέλεση επιζήμιων ενεργειών.

Για να αποτραπεί η υποκλοπή refresh tokens μέσω επιθέσεων XSS, η αποστολή τους πραγματοποιείται αποκλειστικά μέσω HTTP Only cookies. Αν ένας κακόβουλος πράκτορας αποκτήσει πρόσβαση στο refresh token ενός χρήστη, μπορεί να εκδίδει νέα access tokens και να εκτελεί ενέργειες εκ μέρους του χρήστη, έως ότου το refresh token λήξει ή ακυρωθεί. Για την υποκλοπή του refresh token, ο κακόβουλος πράκτορας μπορεί να πραγματοποιήσει επιθέσεις XSS με σκοπό να εισάγει στον φυλλομετρητή του χρήστη κακόβουλο κώδικα που θα δημιουργήσει το token [73]. Για παράδειγμα, ο κακόβουλος πράκτορας μπορεί να προσπαθήσει να δημιουργήσει μια χαρτογράφηση με όνομα:

```
<script> ... malicious code ... </script>
```

Ο στόχος είναι το όνομα να ερμηνευθεί ως κώδικας και να εκτελεστεί στον φυλλομετρητή του χρήστη όταν αυτός επισκεφθεί την σελίδα της χαρτογράφησης. Σε αυτό το υποθετικό σενάριο – το οποίο προφανώς δεν λειτουργεί – ο κακόβουλος κώδικας θα επιχειρούσε να διαβάσει το refresh token από τα cookies και θα το έστελνε σε μία υπηρεσία συλλογής κλεμμένων token. Ωστόσο, επειδή το refresh token αποθηκεύεται ως HTTP Only cookie, η JavaScript του φυλλομετρητή δεν έχει πρόσβαση σε αυτό. Επομένως, ακόμα κι αν ο κακόβουλος πράκτορας

καταφέρει να εισάγει και να εκτελέσει κώδικα στο φυλλομετρητή του χρήστη, δεν μπορεί να αποκτήσει πρόσβαση στο refresh token. Επιπλέον, όπως αναφέρθηκε στο Κεφάλαιο 5.4.2, το access token αποθηκεύεται στη μνήμη του φυλλομετρητή αντί για το localStorage, δυσκολεύοντας την υποκλοπή του μέσω επιθέσεων XSS [74].

Παρά τις υψηλές προδιαγραφές ασφαλείας που ακολουθεί η εφαρμογή, η πιθανότητα διαρροής tokens δεν μπορεί να αποκλειστεί πλήρως. Επομένως, είναι απαραίτητη η ενσωμάτωση ενός μηχανισμού ακύρωσης εκδοθέντων tokens. Πράγματι, στην περίπτωση απώλειας ενός token ο χρήστης στον οποίο ανήκει πρέπει να έχει την δυνατότητα να το ακυρώσει άμεσα. Για τον σκοπό αυτό, αποθηκεύεται στη βάση δεδομένων η χρονική στιγμή της τελευταίας αλλαγής του κωδικού πρόσβασης κάθε χρήστη. Κατά την αποκωδικοποίηση οποιουδήποτε access ή refresh token, ελέγχεται η χρονική στιγμή έκδοσής του. Ένα token θεωρείται έγκυρο μόνο αν η ημερομηνία έκδοσής του είναι μεταγενέστερη από τη χρονική στιγμή της τελευταίας αλλαγής του κωδικού πρόσβασης του χρήστη για τον οποίο εκδόθηκε. Συνεπώς, κάθε φορά που ένας χρήστης αλλάζει τον κωδικό πρόσβασης του λογαριασμού του, όλα τα tokens που είχαν εκδοθεί πριν από την αλλαγή καθίστανται αυτομάτως μη έγκυρα.

## 10. Επιβεβαίωση του Backend

Για την διασφάλιση της ορθής λειτουργίας του backend υλοποιήθηκε ένα σύνολο αυτοματοποιημένων τεστ. Η στρατηγική που ακολουθήθηκε είναι παρόμοια με αυτή που εφαρμόστηκε για την επιβεβαίωση του frontend. Πιο συγκεκριμένα, ελέγχονται οι βασικές λειτουργίες του backend στο επίπεδο του API των υπηρεσιών, χωρίς να επιδιώκεται πλήρης κάλυψη του κώδικα. Με τη χρήση του testing framework του Django [75], αποστέλλονται HTTP αιτήματα στις υπηρεσίες του backend, και οι απαντήσεις που επιστρέφουν ελέγχονται μέσω assert statements. Με τον τρόπο αυτό, προσομοιώνεται η αλληλεπίδραση του backend με το frontend. Αξίζει να σημειωθεί ότι σε ορισμένα τεστ, πριν από την αποστολή αιτημάτων πραγματοποιούνται προσωρινές εγγραφές στη βάση δεδομένων, ώστε να επαληθευτεί η ορθή ανάκτηση τους από τις αντίστοιχες υπηρεσίες. Επιπλέον, λειτουργίες που απαιτούν πρόσβαση σε εξωτερικές υπηρεσίες, όπως η εκτέλεση του FLOODPY, δεν ελέγχονται στα αυτοματοποιημένα τεστ, ωστόσο επιβεβαιώνεται ότι καλούνται οι μέθοδοι που τις υλοποιούν.

Στη συνέχεια, παρουσιάζεται ένα παράδειγμα των βημάτων ενός τεστ του backend. Το συγκεκριμένο τεστ ελέγχει την υπηρεσία δημιουργίας χαρτογράφησης (βλ. Κεφ. 9.1.1), προσομοιώνοντας την αποστολή ενός αιτήματος από χρήστη χωρίς δικαιώματα διαχειριστή.

1. Δημιουργείται ένας προσωρινός λογαριασμός χρήστη χωρίς δικαιώματα διαχειριστή.
2. Προσομοιώνεται η αυθεντικοποίηση του χρήστη μέσω μιας βοηθητικής συνάρτησης του testing framework του Django. Η αυθεντικοποίηση προσομοιώνεται, διότι ο στόχος του παρόντος τεστ δεν είναι η επιβεβαίωση της λειτουργίας των υπηρεσιών αυθεντικοποίησης, και κάθε τεστ απομονώνει μια υπηρεσία.
3. Αποστέλλεται στην υπηρεσία δημιουργίας χαρτογράφησης στο `/api/floodmaps/` ένα έγκυρο HTTP POST αίτημα με προκαθορισμένο περιεχόμενο. Το αίτημα αποστέλλεται από τον λογαριασμό του αυθεντικοποιημένου χρήστη.
4. Επιβεβαιώνεται μέσω assert statements ότι ο HTTP status κώδικας της απάντησης είναι 201 (Created).
5. Επιβεβαιώνεται μέσω assert statements ότι το σώμα της απάντησης είναι ένα έγκυρο JSON που περιέχει τις παραμέτρους της χαρτογράφησης που δημιουργήθηκε.
6. Γίνεται ανάκτηση των εγγραφών της βάσης δεδομένων για τη νέα χαρτογράφηση.
7. Επιβεβαιώνεται ότι οι πληροφορίες των εγγραφών που ανακτήθηκαν από τη βάση δεδομένων αντιστοιχούν στο περιεχόμενο του αιτήματος εστάλη στο βήμα 3.
8. Επιβεβαιώνεται ότι η συνάρτηση που προκαλεί την εκτέλεση του FLOODPY για την χαρτογράφηση δεν κλήθηκε και ότι η εργασία του FLOODPY βρίσκεται σε κατάσταση αναμονής έγκρισης από διαχειριστή (*Pending Approval*).

Επιτυχής ολοκλήρωση των παραπάνω βημάτων επαληθεύει ότι η υπηρεσία δημιουργίας χαρτογράφησης, για έγκυρα αιτήματα από μη διαχειριστές, αποστέλλει την αναμενόμενη απάντηση, αποθηκεύει τις επιθυμητές πληροφορίες στη βάση δεδομένων και δεν εκτελεί αυτόματα την εργασία του FLOODPY. Με ανάλογο τρόπο ελέγχονται όλες οι υπηρεσίες του backend για έγκυρα και μη έγκυρα αιτήματα.

## 11. Εγκατάσταση της Web Εφαρμογής

Η εγκατάσταση της εφαρμογής προϋποθέτει την λήψη και την ενσωμάτωση ενός συνόλου βιβλιοθηκών και προγραμμάτων, τα οποία διαφέρουν ανάλογα με το αν η εγκατάσταση προορίζεται για περιβάλλον ανάπτυξης ή παραγωγής. Σε κάθε περίπτωση, απαιτούνται

εξυπηρετητές για την εκτέλεση της λογικής των υπηρεσιών του backend και το διαμοιρασμό των αρχείων του frontend. Επιπλέον, όπως προκύπτει από την περιγραφή του backend, είναι απαραίτητη εκτέλεση πολλαπλών προγραμμάτων που περιλαμβάνουν:

- Τη βάση δεδομένων PostgreSQL.
- To Redis.
- Τον GeoServer.
- Τουλάχιστον μια διεργασία–εργάτη Celery.

Στο περιβάλλον ανάπτυξης, η ενσωμάτωση των εν λόγω προγραμμάτων πρέπει να γίνεται αυτόμata, με απλό και αποδοτικό τρόπο, ώστε να αποφεύγονται προβλήμata και καθυστερήσεις κατά την υλοποίηση νέων λειτουργιών. Παράλληλα στο περιβάλλον παραγωγής, η εγκατάσταση απαιτείται να διασφαλίζει την ομαλή λειτουργία, την αξιοπιστία και την ασφάλεια της εφαρμογής.

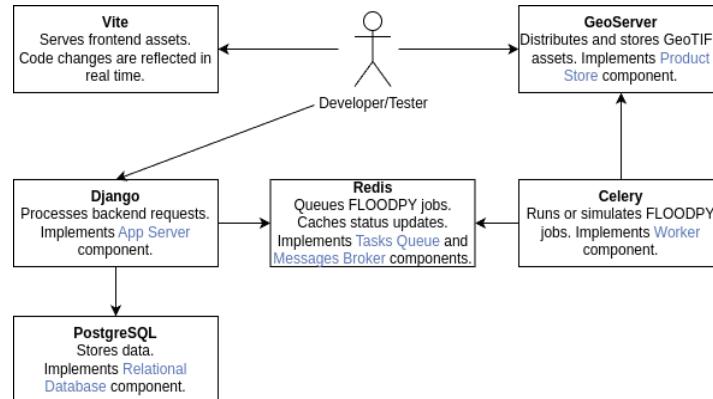
Για την ικανοποίηση των απαιτήσεων της εγκατάστασης στα περιβάλλοντα ανάπτυξης και παραγωγής χρησιμοποιήθηκε το Docker Compose [76]. Ειδικότερα, η εγκατάσταση χωρίστηκε σε Docker Containers, καθένα από τα οποία δημιουργείται από ένα Docker Image. Κάθε Container λειτουργεί ως απομονωμένο περιβάλλον που περιέχει όλα τα αρχεία, τις βιβλιοθήκες και τα προγράμμata τα οποία είναι απαραίτηta για την εκτέλεση ενός τμήμαtος tης εφαρμογής. Για tα προγράμμata Redis, PostgreSQL και GeoServer χρησιμοποιήθηκan έτoιma Docker Images από tο απoθetήriο Docker Hub. Ωstόso, απaiτήthηke η υλοποίηση και tρiών eξeiδiκeuμénw Images πou εpiτrέpoυn:

1. Tηn δηmioυrgia κai tοn δiamoiraσmό tοu fronteнд stο pеriβálloν aνáptuξης.
2. Tηn δrōmoloγηs aiτemátow κai tοn δiamoiraσmό tοu fronteнд stο pеriβálloν paraγωgήs.
3. Tηn eξupherétiηs tωn uptereσiώn tοu backeнд κai tηn ektélese tοu FLOODPY aneξaprtήtw pеriβálloνtοs.

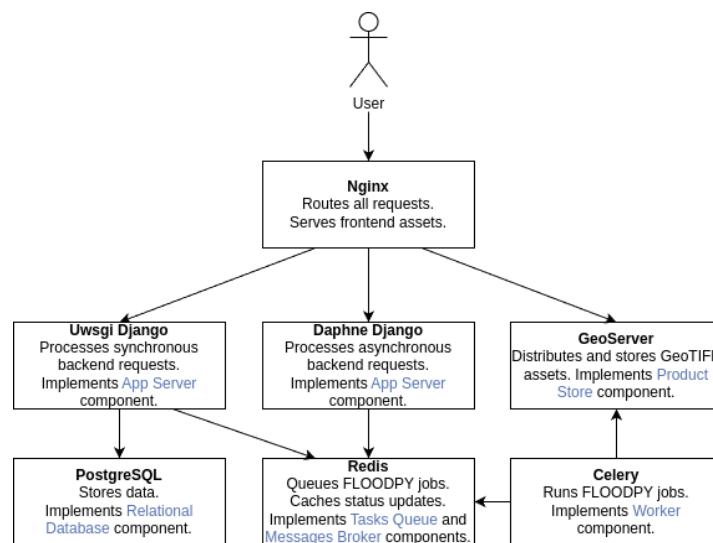
H δiaδikasía dηmioυrgiaς, rύthmisηs, κai ektéleseηs tωn Containers automatoπoieίtai mέsω tου Docker Compose. Me mίa evtolή, tο Docker Compose dηmioυrgieί oλa ta aparaítηta Containers apό ta antistoiχa Images κai ta paraμetropoieί chriηsimopoiώntaς prokaθorisiMénves metabhlētēs pеriβálloνtοs (environment variables).

H eγkatastasη tηs eφarμogήs πou prokúptei apό tηn chriήsη tοu Docker Compose diaphrérei análoga μe tο pеriβálloν γia tο opoio prooρizetai. Stο pеriβálloν aνáptuξηs, tο fronteнд diamoirázetai mέsω tοu Vite, tο opoio epitrépeι tηn autómatη ananéwsoη tοu

περιεχομένου των σελίδων σε κάθε αλλαγή κώδικα (βλ. Κεφ. 4.4). Όμοια, για την εκτέλεση των υπηρεσιών του backend, στο περιβάλλον ανάπτυξης χρησιμοποιείται ένας εξυπηρετητής που είναι ενσωματωμένος στο Django και εφαρμόζει τις αλλαγές στον κώδικα χωρίς να απαιτείται επανεκκίνηση του. Ωστόσο, στο περιβάλλον παραγωγής, λόγω των απαιτήσεων αποδοτικότητας και ασφάλειας, ο nginx χρησιμοποιείται για τον διαμοιρασμό του frontend, καθώς και για την δρομολόγηση όλων των εισερχόμενων αιτημάτων στο backend. Επιπλέον, οι υπηρεσίες του backend εκτελούνται από τον εξυπηρετητή uWSGI με εξαίρεση την υπηρεσία WebSocket που εκτελείται από τον εξυπηρετητή daphne. Στις παρακάτω εικόνες (Εικόνα 11.1, Εικόνα 11.2) παρουσιάζονται λεπτομέρειες της δομή των Docker Containers για τα περιβάλλοντα ανάπτυξης και παραγωγής.



**Εικόνα 11.1:** Δομή των Docker Containers της εγκατάστασης στο περιβάλλον ανάπτυξης.



**Εικόνα 11.2:** Δομή των Docker Containers της εγκατάστασης στο περιβάλλον παραγωγής.

## 12. Σύνοψη και Μελλοντικές Επεκτάσεις

Η ανάπτυξη της web εφαρμογής που περιγράφηκε στην παρούσα εργασία αποτελεί μια σημαντική προσπάθεια υλοποίησης ενός εύχρηστου, αποδοτικού και αξιόπιστου συστήματος για τη χαρτογράφηση πλημμυρών. Με βασικούς στόχους την απλότητα και την προσβασιμότητα, η εφαρμογή επιτρέπει σε χρήστες χωρίς τεχνικές ή επιστημονικές γνώσεις να χαρτογραφούν πλημμύρες μέσω του FLOODPY από το περιβάλλον του φυλλομετρητή. Επιπλέον, η αποθήκευση των απεικονίσεων που δημιουργούνται καθιστά δυνατή τη δημιουργία ενός αρχείου χαρτογραφήσεων, το οποίο διευκολύνει τη συγκριτική ανάλυση πλημμυρικών φαινομένων και την αναγνώριση ευάλωτων περιοχών. Συνολικά, η εφαρμογή που υλοποιήθηκε μπορεί να συμβάλλει ουσιαστικά τόσο στην βελτίωση της διαχείρισης των πλημμυρών όσο και στον αποτελεσματικό σχεδιασμό προληπτικών δράσεων.

Περαιτέρω βελτίωση της εφαρμογής μπορεί να περιλαμβάνει την ενσωμάτωση νέων προγραμμάτων χαρτογράφησης, καθώς και την προσθήκη τεχνικών δυνατοτήτων όπως log aggregation, monitoring και container orchestration. Με την ενσωμάτωση προγραμμάτων χαρτογράφησης επιπλέον του FLOODPY, θα ήταν δυνατή η αξιολόγηση των αποτελεσμάτων που παράγονται διαφορετικές μέθοδοι, επιτρέποντας την επιλογή του προγράμματος που παράγει τα καλύτερα αποτελέσματα σε κάθε περίπτωση. Παράλληλα, η προσθήκη συστημάτων για log aggregation και monitoring θα επέτρεπε την κεντρική διαχείριση των logs της εφαρμογής και την παρακολούθηση των αιτημάτων που εξυπηρετούνται. Με αυτόν τον τρόπο, θα διασφαλιζόταν η αποτελεσματική ανάλυση της απόδοσης του συστήματος, ενώ παράλληλα θα διευκολυνόταν η έγκαιρη ανίχνευση και επίλυση σφαλμάτων. Τέλος, θα μπορούσε να γίνει δυναμική κατανομή των Docker Containers της εφαρμογής σε πολλαπλά υπολογιστικά συστήματα με τη χρήση εργαλείων όπως τα Docker Swarm ή Kubernetes για την διασφάλιση της εύρυθμης λειτουργίας της εφαρμογής σε μεγάλη κλίμακα. Οποιαδήποτε πρόταση για επέκταση μπορεί να υποβληθεί στο αποθετήριο της εφαρμογής στο GitHub [77].

## Βιβλιογραφία

- [1] D. Guha-Sapir, P. Hoyois, P. Wallemacq, and R. Below, “Annual Disaster Statistical Review 2016: The numbers and trends - World,” ReliefWeb. Accessed: Dec. 05, 2024. [Online]. Available: <https://reliefweb.int/report/world/annual-disaster-statistical-review-2016-numbers-and-trends>
- [2] L. Alfieri, L. Feyen, and G. Di Baldassarre, “Increasing flood risk under climate change: a pan-European assessment of the benefits of four adaptation strategies,” *Climatic Change*, vol. 136, no. 3–4, pp. 507–521, Mar. 2016, doi: 10.1007/s10584-016-1641-1.
- [3] K. Karamvasis and V. Karathanassi, “FLOMPY: An Open-Source Toolbox for Floodwater Mapping Using Sentinel-1 Intensity Time Series,” *Water*, vol. 13, no. 21, p. 2943, Oct. 2021, doi: 10.3390/w13212943.
- [4] “Synthetic Aperture Radar (SAR),” NASA Earthdata. Accessed: Dec. 05, 2024. [Online]. Available: <https://www.earthdata.nasa.gov/learn/backgrounder/what-is-sar>
- [5] T. L. Toan, “Introduction to Sar Remote Sensing,” esa. Accessed: Dec. 05, 2024. [Online]. Available: [https://eo4society.esa.int/wp-content/uploads/2021/05/2021\\_SARBasics\\_TLeToan\\_theory.pdf](https://eo4society.esa.int/wp-content/uploads/2021/05/2021_SARBasics_TLeToan_theory.pdf)
- [6] “S1 Mission,” Sentiwiki. Accessed: Dec. 05, 2024. [Online]. Available: <https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-1>
- [7] “S1 Applications,” Sentiwiki. Accessed: Dec. 05, 2024. [Online]. Available: <https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/product-overview/polarimetry>
- [8] J. Kittler and J. Illingworth, “Minimum error thresholding,” *Pattern Recognition*, vol. 19, no. 1, pp. 41–47, Jan. 1986, doi: 10.1016/0031-3203(86)90030-0.
- [9] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979, doi: 10.1109/tsmc.1979.4310076.
- [10] “Home,” Copernicus EMS. Accessed: Dec. 05, 2024. [Online]. Available: <https://emergency.copernicus.eu/>
- [11] kleok, “GitHub - kleok/FLOODPY: Flood Python Toolbox,” GitHub. Accessed: Dec. 06, 2024. [Online]. Available: <https://github.com/kleok/FLOODPY>
- [12] “Technology,” 2024 Stack Overflow Developer Survey. Accessed: Dec. 08, 2024. [Online]. Available: <https://survey.stackoverflow.co/2024/technology#1-web-frameworks-and-technologies>
- [13] “Your First Component – React.” Accessed: Dec. 08, 2024. [Online]. Available: <https://react.dev/learn/your-first-component>
- [14] “Writing Markup with JSX – React.” Accessed: Dec. 08, 2024. [Online]. Available: <https://react.dev/learn/writing-markup-with-jsx>
- [15] “Feature Overview v6.28.0,” React Router. Accessed: Dec. 08, 2024. [Online]. Available: <https://reactrouter.com/en/v6.28.0/feature-overview>

<https://reactrouter.com/en/main/start/overview#data-fetchers>

[16] “ActionFunction,” React Router API Reference. Accessed: Dec. 08, 2024. [Online]. Available: <https://reactrouter.com/en/main/route/action>

[17] “LoaderFunction,” React Router API Reference. Accessed: Dec. 08, 2024. [Online]. Available: <https://reactrouter.com/en/main/route/loader>

[18] “Material UI: React components that implement Material Design.” Accessed: Dec. 08, 2024. [Online]. Available: <https://mui.com/material-ui/>

[19] “Accessibility,” Base UI. Accessed: Dec. 08, 2024. [Online]. Available: <https://mui.com/base-ui/getting-started/accessibility/>

[20] “About npm,” npm Docs. Accessed: Dec. 08, 2024. [Online]. Available: <https://docs.npmjs.com/about-npm>

[21] “Introduction,” Rollup. Accessed: Dec. 08, 2024. [Online]. Available: <https://rollupjs.org/introduction/#the-why>

[22] “Why Vite,” vitejs. Accessed: Dec. 08, 2024. [Online]. Available: <https://vite.dev/guide/why>

[23] “Frequently Asked Questions,” Rollup. Accessed: Dec. 08, 2024. [Online]. Available: <https://rollupjs.org/faqs/#what-is-tree-shaking>

[24] “Getting Started,” vitejs. Accessed: Dec. 08, 2024. [Online]. Available: <https://vite.dev/guide/>

[25] “Features,” vitejs. Accessed: Dec. 08, 2024. [Online]. Available: <https://vite.dev/guide/features.html#hot-module-replacement>

[26] Atlassian, “User Stories,” Atlassian. Accessed: Dec. 08, 2024. [Online]. Available: <https://www.atlassian.com/agile/project-management/user-stories>

[27] “React Leaflet,” React Leaflet. Accessed: Dec. 08, 2024. [Online]. Available: <https://react-leaflet.js.org/>

[28] “Copyright and License,” OpenStreetMap. Accessed: Dec. 08, 2024. [Online]. Available: <https://www.openstreetmap.org/copyright>

[29] “Slippy map,” OpenStreetMap Wiki. Accessed: Dec. 08, 2024. [Online]. Available: [https://wiki.openstreetmap.org/wiki/Slippy\\_map](https://wiki.openstreetmap.org/wiki/Slippy_map)

[30] “Nominatim.” Accessed: Dec. 11, 2024. [Online]. Available: <https://nominatim.org/>

[31] “Bounding box,” OpenStreetMap Wiki. Accessed: Dec. 08, 2024. [Online]. Available: [https://wiki.openstreetmap.org/wiki/Bounding\\_box](https://wiki.openstreetmap.org/wiki/Bounding_box)

[32] heyman, “GitHub - heyman/leaflet-areaselect: Leaflet plugin for letting users select an area of the map using a rectangle, and get the bounding box,” GitHub. Accessed: Dec. 08, 2024. [Online]. Available:

<https://github.com/heyman/leaflet-areaselect>

[33] jquense, “GitHub - jquense/yup: Dead simple Object schema validation,” GitHub. Accessed: Dec. 08, 2024. [Online]. Available: <https://github.com/jquense/yup>

[34] “Get Started.” Accessed: Dec. 08, 2024. [Online]. Available: <https://react-hook-form.com/get-started>

[35] “Server-sent events - Web APIs,” MDN Web Docs. Accessed: Dec. 17, 2024. [Online]. Available: [https://developer.mozilla.org/en-US/docs/Web/API/Server-sent\\_events](https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events)

[36] “The WebSocket API (WebSockets) - Web APIs,” MDN Web Docs. Accessed: Dec. 17, 2024. [Online]. Available: [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)

[37] “WorldCover,” WORLDCOVER. Accessed: Dec. 18, 2024. [Online]. Available: <https://esa-worldcover.org/en>

[38] auth0.com, “JWT.IO - JSON Web Tokens Introduction,” Auth0. Accessed: Dec. 18, 2024. [Online]. Available: <https://jwt.io/introduction>

[39] D. Hardt, “RFC 6749: The OAuth 2.0 Authorization Framework.” Accessed: Dec. 18, 2024. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6749#section-1.4>

[40] “Set-Cookie - HTTP,” MDN Web Docs. Accessed: Dec. 19, 2024. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie#httponly>

[41] I. Sommerville, *Software Engineering, Global Edition*. Pearson Higher Ed, 2016.

[42] jsdom, “GitHub - jsdom/jsdom: A JavaScript implementation of various web standards, for use with Node.js,” GitHub. Accessed: Dec. 20, 2024. [Online]. Available: <https://github.com/jsdom/jsdom>

[43] “React Testing Library,” Testing Library. Accessed: Dec. 20, 2024. [Online]. Available: <https://testing-library.com/docs/react-testing-library/intro/>

[44] “The Django admin site,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.djangoproject.com/en/5.1/ref/contrib/admin/>

[45] “User authentication in Django,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.djangoproject.com/en/5.1/topics/auth/>

[46] “Password management in Django,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.djangoproject.com/en/5.1/topics/auth/passwords/>

[47] “Security in Django,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.djangoproject.com/en/5.1/topics/security/>

[48] T. Christie, “Pagination,” Django REST framework. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.django-rest-framework.org/api-guide/pagination/>

- [49] T. Christie, “Filtering,” Django REST framework. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.django-rest-framework.org/api-guide/filtering/>
- [50] T. Christie, “Serializers,” Django REST framework. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.django-rest-framework.org/api-guide/serializers/>
- [51] “jazzband/djangorestframework-simplejwt: A JSON Web Token authentication plugin for the Django REST Framework.,” GitHub. Accessed: Dec. 20, 2024. [Online]. Available: <https://github.com/jazzband/djangorestframework-simplejwt>
- [52] T. Christie, “Throttling,” Django REST framework. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.django-rest-framework.org/api-guide/throttling/>
- [53] T. Christie, “Permissions,” Django REST framework. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.django-rest-framework.org/api-guide/permissions/>
- [54] “What is WSGI? — WSGI.org.” Accessed: Dec. 20, 2024. [Online]. Available: <https://wsgi.readthedocs.io/en/latest/what.html>
- [55] “ASGI Documentation — ASGI 3.0 documentation.” Accessed: Dec. 20, 2024. [Online]. Available: <https://asgi.readthedocs.io/en/latest/>
- [56] “Django,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.djangoproject.com/>
- [57] “nginx.” Accessed: Dec. 20, 2024. [Online]. Available: <https://nginx.org/en/index.html>
- [58] “Models,” Django Project. Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.djangoproject.com/en/5.1/topics/db/models/>
- [59] “Introduction to Celery — Celery 5.5.0rc4 documentation.” Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.celeryq.dev/en/latest/getting-started/introduction.html>
- [60] “First steps with Django — Celery 5.5.0rc4 documentation.” Accessed: Dec. 20, 2024. [Online]. Available: <https://docs.celeryq.dev/en/latest/django/first-steps-with-django.html>
- [61] “What is Redis?: An Overview.” Accessed: Dec. 20, 2024. [Online]. Available: <https://redis.io/learn/develop/node/nodecrashcourse/whatisredis>
- [62] “Consumers — Channels 4.2.0 documentation.” Accessed: Dec. 20, 2024. [Online]. Available: <https://channels.readthedocs.io/en/latest/topics/consumers.html#websocketconsumer>
- [63] “Tutorial Part 2: Implement a Chat Server — Channels 4.2.0 documentation.” Accessed: Dec. 20, 2024. [Online]. Available: [https://channels.readthedocs.io/en/latest/tutorial/part\\_2.html#enable-a-channel-layer](https://channels.readthedocs.io/en/latest/tutorial/part_2.html#enable-a-channel-layer)
- [64] “About,” GeoServer. Accessed: Dec. 20, 2024. [Online]. Available: <https://geoserver.org/about/>
- [65] “Publishing a Image — GeoServer 2.27.x User Manual.” Accessed: Dec. 20, 2024. [Online].

Available: <https://docs.geoserver.org/latest/en/user/gettingstarted/image-quickstart/index.html>

[66] “Web Map Service,” Open Geospatial Consortium. Accessed: Dec. 20, 2024. [Online]. Available: <https://www.ogc.org/publications/standard/wms/>

[67] “django-filter 24.3 documentation.” Accessed: Dec. 27, 2024. [Online]. Available: <https://django-filter.readthedocs.io/en/stable/index.html>

[68] kleok, “FLOODPY/Floodpyapp\_stat.ipynb at main · kleok/FLOODPY,” GitHub. Accessed: Dec. 31, 2024. [Online]. Available: [https://github.com/kleok/FLOODPY/blob/main/Floodpyapp\\_stat.ipynb](https://github.com/kleok/FLOODPY/blob/main/Floodpyapp_stat.ipynb)

[69] “contextily: context geo tiles in Python — contextily 1.6.3.dev2+geafeb3c.d20241118 documentation.” Accessed: Dec. 31, 2024. [Online]. Available: <https://contextily.readthedocs.io/en/latest/index.html>

[70] “Introduction — djoser 2.3.1 documentation.” Accessed: Jan. 03, 2025. [Online]. Available: <https://djoser.readthedocs.io/en/latest/introduction.html>

[71] “sunscrapers/djoser: REST implementation of Django authentication system.,” GitHub. Accessed: Jan. 03, 2025. [Online]. Available: <https://github.com/sunscrapers/djoser>

[72] “Cross-Site Request Forgery Prevention,” OWASP Cheat Sheet Series. Accessed: Jan. 03, 2025. [Online]. Available: [https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)

[73] “Cross Site Scripting (XSS),” OWASP Foundation. Accessed: Jan. 03, 2025. [Online]. Available: <https://owasp.org/www-community/attacks/xss/>

[74] “HTML5 Security,” OWASP Cheat Sheet Series. Accessed: Jan. 03, 2025. [Online]. Available: [https://cheatsheetseries.owasp.org/cheatsheets/HTML5\\_Security\\_Cheat\\_Sheet.html#local-storage](https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage)

[75] T. Christie, “Testing,” Django REST framework. Accessed: Jan. 06, 2025. [Online]. Available: <https://www.djangoproject.com/api-guide/testing/>

[76] ““Docker Compose,”” Docker Documentation. Accessed: Jan. 09, 2025. [Online]. Available: <https://docs.docker.com/compose/>

[77] adamsigi, “GitHub - adamsigi/FloodViewer,” GitHub. Accessed: Jan. 11, 2025. [Online]. Available: <https://github.com/adamsigi/FloodViewer>