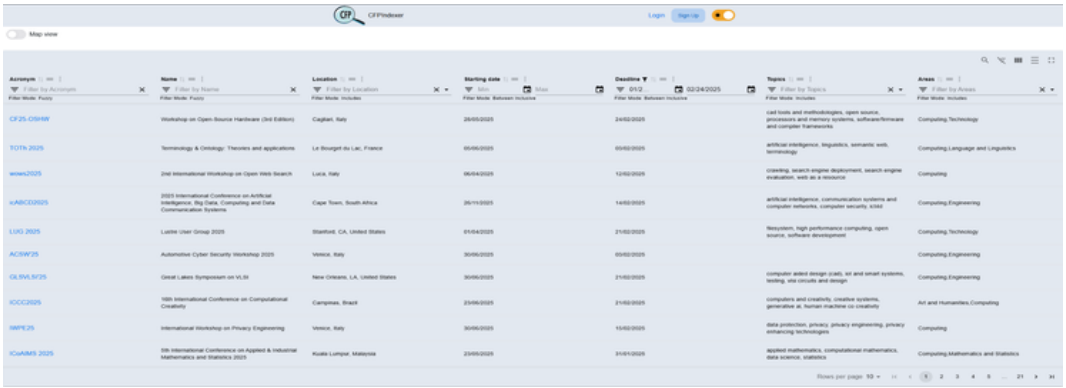
Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διαδικτυακό σύστημα διαχείρισης ανακοινώσεων ’Call for Papers’ με υπηρεσία αυτόματης συγκέντρωσης δεδομένων



Φοιτητής:
Αθανάσιος Κοντούδης
Αριθμός Μητρώου: 154474

Επιβλέπων:
Στέφανος Ουγιάρογλου

29 Ιανουαρίου 2025

Title of Dissertation A web-based 'Call for Papers' management system with a web service for automatic data aggregation
Code of Dissertation 23101
Student's full name Athanasios Kontoudis
Supervisor's full name Stefanos Ougiaroglou
Date of undertaking 29-10-2024
Date of completion 29-01-2025

We hereby affirm the authorship of this paper as well as the acknowledgement and credit of whichever assistance We received in its composition. We have, furthermore, noted the various sources from which We extracted data, ideas, visual or written material, in paraphrase or exact quotation. Moreover, we affirm the exclusive composition of this paper by myself only, for the purpose of it being a dissertation, in the Department of Information and Electronic Engineering of the I.H.U.

This paper constitutes the intellectual property of Athanasios Kontoudis the student that composed it. According to the open-access policy, the author/composer offers the International Hellenic University authorisation to use the right to reproduce, borrow, publicly present and digitally distribute the paper globally, in electronic form and media of all kinds, for teaching or research purposes, voluntarily. Open access to the full text, by no means grants the right to trespass the intellectual property of the author/composer, nor does it authorise the reproduction, republication, duplication, selling, commercial use, distribution, publication, downloading, uploading, translation, modification of any kind, in part or summary of the paper, without the explicit written consent of the authors.

The approval of this dissertation by the Department of Information and Electronic Engineering of the International Hellenic University, does not necessarily entail the adoption of the author's views, on behalf of the Department.

Abstract

This application is an integrated tool for managing and searching data related to calls for papers (CFPs) and draws its data from the EasyChair platform. By performing automatic data extraction and partial data rectification, the application organizes and stores the information in a structured database, facilitating efficient information retrieval. Users are able to perform advanced searches for CFPs using various filters, ensuring targeted results tailored to their needs. In addition, users can subscribe to receive email updates, ensuring timely information on new or relevant calls. Finally, the application allows for the creation and publication of new CFPs directly within the system, simplifying the process for contributors and organisers. This solution aims to improve accessibility and simplify workflows in academia and research.

Περίληψη

Η παρούσα εφαρμογή αποτελεί ένα ολοκληρωμένο εργαλείο για τη διαχείριση και αναζήτηση δεδομένων που αφορούν προσκλήσεις για εργασίες (call for papers - CFPs) και αντλεί τα δεδομένα της από την πλατφόρμα EasyChair. Πραγματοποιώντας αυτόματη άντληση και μερική διορθώσει των δεδομένων, η εφαρμογή οργανώνει και αποθηκεύει τις πληροφορίες σε μια δομημένη βάση δεδομένων, διευκολύνοντας την αποδοτική ανάκτηση πληροφοριών. Οι χρήστες έχουν τη δυνατότητα να πραγματοποιούν προηγμένες αναζητήσεις για CFPs με χρήση διαφόρων φίλτρων, εξασφαλίζοντας στοχευμένα αποτελέσματα προσαρμοσμένα στις ανάγκες τους. Επιπλέον, παρέχεται η δυνατότητα εγγραφής χρηστών για λήψη ενημερώσεων μέσο email, διασφαλίζοντας έγκαιρη πληροφόρηση σχετικά με νέες ή σχετικές προσκλήσεις. Τέλος, η εφαρμογή επιτρέπει τη δημιουργία και δημοσίευση νέων CFPs απευθείας μέσα από το σύστημα, απλοποιώντας τη διαδικασία για συντελεστές και διοργανωτές. Η λύση αυτή στοχεύει στη βελτίωση της προσβασιμότητας και την απλοποίηση των ροών εργασίας στον ακαδημαϊκό και ερευνητικό χώρο.

Contents

Chapter 1

Εισαγωγή

1.1 Διοργάνωση επιστημονικών συνεδρίων και cfps

Ο όρος Call for papers (cfps) είναι η πρόσκληση επιστημόνων όλων των κλάδων σε ένα συνέδριο, στο οποίο θα καταθέσουν ερευνητικά έγγραφα τα οποία θα εξετασθούν και θα συζητηθούν από τους συμμετέχοντες. Έπειτα θα αποφασιστεί ποια από αυτά τα έγγραφα θα δημοσιευτούν σε κάποιο άρθρο η επιστημονικό βιβλίο. Τα συνέδρια αυτά λαμβάνουν μέρος είτε σε φυσικές τοποθεσίες είτε μετά την πανδημία διαδικτυακά. Η οργάνωση όπως και προώθηση αυτών των συνεδριών είναι δύσκολη, όμως εδώ και κάποια χρόνια υπάρχει η πλατφόρμα `EasyChair`[1] που υποστηρίζει πως κάνει την διοργάνωση πιο εύκολη.

1.2 Κίνητρο

Τα βασικά κίνητρα για την δημιουργία της εφαρμογής μας είναι δυο και έχουν να κάνουν με την πλατφόρμα του `EasyChair`[1], που είναι και η διαδεδομένη λύση για την διοργάνωση επιστημονικών συνεδρίων. Η πλατφόρμα δεν μας δίνει όμως την δυνατότητα να προωθήσουμε τα συνέδρια μας αν δεν διαχειρίζεσαι αυτή το συνέδριο , δηλαδή δεν έχουμε δημιουργήσει ένα call for paper σε αυτήν. Η αποκλειστικότητα αυτή προκαλεί μια ιδιαίτερη δυσκολία στην προώθηση των συνεδρίων. Το δεύτερο μεγαλύτερο πρόβλημα είναι πως η διαδικασία εύρεσης συνεδρίων που ενδιαφέρουν τον χρήστη είναι αρκετα δύσχρηστη. Η πλατφόρμα προσφέρει περιορισμένα φίλτρα αναζήτησης συνεδρίων (cfps) τα οποία μάλιστα δεν είναι εύκολα προσβάσιμα από τους χρήστες. Αυτά μπορεί να επηρεάσει βεβαίως και την προώθηση των συνεδρίων μιας και είναι εξαιρετικά πιθανό να μην βρει ο χρήστης το συνέδριο μας λόγο των φίλτρων αυτών.

1.3 Συνεισφορά

Η εφαρμογή μας έχει ως κύριο σκοπό την δημιουργία μιας πλατφόρμας που διευκολύνει τους χρήστες της αλλά και μη, να έχουν άμεση πρόσβαση σε αναγνώσεις επιστημονικών συνεδρίων. Μετά από την καταχώρηση των συνεδρίων η οποία γίνεται είτε στην πλατφόρμα μας είτε στην πλατφό-ρμα του `EasyChair`[1], η αναζήτηση των συνεδρίων γίνεται με ιδιαίτερη ευκολία

από ένα σημείο της εφαρμογής. Επίσης δεν θα υπάρχει κανένας περιορισμός στα φίλτρα τα οποία μπορεί να εφαρμόσει ο χρήστης. Ο χρήστης επίσης μπορεί να λαμβάνει ειδοποιήσεις για τα συνέδρια τα οποία τον ενδιαφέρουν άμεσα, μάλιστα μπορεί να λαμβάνει ειδοποιήσεις για συνέδρια τα οποία ανήκουν σε κατηγορίες αλλά και σε όλα τα φίλτρα που μπορούν να εφαρμοσθούν στην αναζήτηση. Η προώθηση των συνεδρίων στην εφαρμογή μας δεν έχει κάποιο κριτήριο ως προς την προέλευση τους (που διοργανώνεται), καθιστώντας την πλατφόρμα μας πιο ελκυστική προς όλους τους χρήστες.

1.4 Οργάνωση εργασίας

1.4.1 Η πλατρφόρμα EasyChair

Η πλατφόρμα του `EasyChair`[1] είναι η πλατφόρμα η οποία ήταν και το κίνητρο της δημιουργίας της δικιάς μας πλατφόρμας. Το `EasyChair`[1] παρέχει στους χρήστες την δυνατότητα δημιου-ργίας συνεδρίων για την κατάθεση ερευνών, η οποίες εκ των υστέρων μπορεί να δημοσιευθούν σε κάποιο επιστημονικό περιοδικό ή βιβλίο. Ο σκοπός της είναι να είναι εύχρηστη και ίσως μια πύλη για όλους τους επιστήμονες να διαφημίζουν τα συνέδρια τους πολύ εύκολα. Όμως όπως θα δούμε και αργότερα έχει μερικά βασικά προβλήματα τα οποία επιτυγχάνουν το αντίθετο αποτέλεσμα.

1.4.2 Τεχνολογίες

Οι τεχνολογίες οποίες χρησιμοποιήθηκαν για να υλοποιηθεί η πλατφόρμα μας είναι πολλές. Για την υλοποίηση της άντλησης δεδομένων αυτοματοποιημένα χρησιμοποιήθηκε η γλώσσα `python`[2] και η βιβλιοθήκη `BeautifulSoup`[3] που είναι δανικά για αυτήν την δουλεία. Για την υλοποίηση της διεπαφής (UI) αλλά και του και του διακομιστή (server - backend) χρησιμοποιήθηκε το framework `Next.js`[4] το οποίο προσφέρει την δυνατότητα ανάπτυξης διαδικτυακών εφαρμογών εύκολα με την χρήση `JavaScript`[5] ή `TypeScript`[6]. Επίσης παρέχει εργαλεία για να είναι πιο αποδοτικές οι εφαρμογές σε ότι αφορά το την κατάταξη τους στις μηχανές αναζήτησης. Επιπροσθέτως υπάρχει δυνατότητα εγκατάστασης επιπροσθέτως πακέτων (npm packages) τα οποία βοηθούν στην ευκολία ανάπτυξης της εφαρμογής. Ως βάση δεδομένων έχουμε μια `mariadb`[7] βάση η οποία είναι μια sql βάση. Τέλος για μερικές προγραμματισμένες διεργασίες γίνεται η χρήση `Node.js`[8] που είναι είναι μια πλατφόρμα ανάπτυξης λογισμικού βασισμένη σε περιβάλλον `JavaScript`[5] και η χρήση του bash.

1.4.3 Υλοποίηση του CFPIndexer

Η υλοποίηση του CFPIndexer έγινε χρησιμοποιώντας τις τεχνολογίες που προαναφέρθηκαν. Αρχικά τοπικά και με την βοήθεια του GitHub για την οργάνωση του κώδικα. Το πλεονέκτημα της χρήσης του `Nextjs`[4] είναι ότι μπορούμε να αναπτυσουμε ταυτόχρονα το front και το back end της εφαρμογής, και έτσι η υλοποίηση του project έγινε με βάση τις απαιτήσεις του.

1.4.4 Παρουσίαση του CFPIndexer

Το CFPIndexer έχει μερικές πολύ βασικές λειτουργίες οι οποίες διευκολύνουν τους χρήστες της, οι οποίες παρουσιάζονται αναλυτικότερα στο παρακάτω κεφάλαιο.

1.4.5 Συμπεράσματα και Μελλοντικές επεκτάσεις

Τα συμπεράσματα τα οποία θα αναλύσουμε και αργότερα είναι ότι ο χρόνος αναζήτησης έχει μειωθεί δραστικά, καθώς και η επιτυχία εύρεσης των συνεδρίων είναι πολύ πιο μεγάλη και χρηστική. Ενώ οι μελλοντικές επεκτάσεις είναι η προσθήκη περισσοτέρων πηγών καθώς και η χρήση διάφορων τεχνολογιών τεχνητής νοημοσύνης για να βοηθήσουν την πλατφόρμα αλλά και τους χρήστες σε διάφορες λειτουργίες.

Chapter 2

Η πλατρφόρμα EasyChair

2.1 Παρουσίαση

Η πλατφόρμα του EasyChair έχει σχεδιαστή για να προσφέρει στους χρήστες την δυνατότητα να διοργανώνουν επιστημονικά συνέδρια τα οποία καλούν ενδιαφερόμενους επιστήμονες να καταθέσουν ιδέες, οι οποίες αργότερα μπορεί να δημοσιευθούν σε κάποιο επιστημονικό βιβλίο η περιοδικό. Η πλατφόρμα υπόσχεται να κάνει την διαδικασία αυτή πολύ εύκολη και φιλική προς τους χρήστες. Επίσης κάνει προώθηση των συνεδρίων στους χρήστες της αλλά και στους απλούς επισκέπτες της ιστοσελίδας της. Παρέχει επίσης την δυνατότητα αναζήτησης στους επισκέπτες της η οποία όπως θα αναλύσουμε αργότερα είναι αρκετά προβληματική και δύσχρηστη.
Μερικες ακομα παροχες του EasyChair ειναι οι εξης:

- **Υπηρεσίες δημοσίευση**ς: Η υπηρεσία αυτή βοηθά τους χρήστες να δημοσιεύσουν τις έρευνες οι οποίες θα παρουσιαστούν και ότι είναι διασυνδεμένες άρτια με την δημιουργία των συνεδρίων.

- **Smart slides**: Η υπηρεσία αυτή βοηθά τους συμμετάσχοντες αλλά και τους διοργανωτές των συνεδρίων στο να καταθέσουν τις διαφάνειες για την παρουσίαση τους.

- **Smart CFP**: Είναι η λειτουργία - υπηρεσία του, που δίνει την δυνατότητα στους επισκέπτες να βρούνε cfps που τους ενδιαφέρουν και υπόσχεται στους δημιουργούς τους να τα προωθήσουν σε ένα πολύ μεγάλο κοινό το οποίο ανέρχεται στα περίπου 1,8 εκατομμύρια.

2.2 Μειονεκτήματα

Τα μειονεκτήματα του EasyChair αν και δεν είναι πολλά είναι σημαντικά. Το πρώτο από αυτά το οποίο δεν είναι τόσο σημαντικό οσο το άλλο, είναι η ουσιαστικά η μη προώθηση συνεδρίων τα οποία δεν έχουν δημιουργηθεί σε αυτό. Το EasyChair προωθεί τα συνέδρια και τα cfps τα οποία έχουν δημιουργηθεί από εγγεγραμμένους χρήστες σε αυτό. Αν κάποιος θέλει να προωθήσει στο κοινό το CFP του είναι αναγκασμένος να το δημιουργήσει στο EasyChair εκ νέου. Ένα πρόβλημα είναι η δομή του και η οργάνωση του το οποίο αναγκάζει τους χρήστες να πρέπει να κάνουν πολλές ενέργειες για απλά πράγματα, παραδείγματος χάριν να

βρούνε την ιστοσελίδα που παρουσιάζονται όλα τα CFP. Η διαδικασία της δημιουργίας ενός cfp ή ενός συνεδρίου είναι εν μέρει προβληματική διότι δεν ελέγχονται τα δεδομένα τα οποία εισάγουν οι χρήστες και αυτό μπορεί να τα κάνει δυσεύρετα διότι δεν υπάρχον τα ανάλογα φίλτρα και η λογική τους. Όσον άφορα την διαδικασία εύρεσης των CFP τα φίλτρα τα οποία είναι διαθέσιμα καθώς και ο τρόπος φιλτραρίσματος είναι ιδιαίτερα προβληματικός και θα αναλυθεί περαιτέρω στο επόμενο υποκεφάλαιο.

2.3 Αναζήτηση cfp

Η αναζήτηση των CFPs στο EasyChair είναι το σημαντικότερο μειονέκτημα και ο κυρίως λόγος που δημιουργήθηκε το CFPIndexer. Είναι εξαιρετικά δύσχρηστη, γιατί ο χρήστης πρέπει να πλοηγηθεί σε πολλές σελίδες για να βρει τα βασικά φίλτρα και να τα εφαρμόσει. Επιπλέον, αν θέλει να εφαρμόσει σύνθετα φίλτρα και κριτήρια ανα-ζήτησης, αυτό δεν είναι δυνατό. Για παράδειγμα, η πλατφόρμα έχει 21 επιστημονικά πεδία που τα ονομάζει ”areas”. Η πλατφόρμα παρέχει την επιλογή να αναζητήσουμε βάσει επιστημονικού πεδίου, αλλά η δυνατότητα αυτή αρχικά είναι δύσκολα προσβάσιμη και απαιτεί από τους χρήστες να προβούν σε πολλές ενέργειες για να πραγματοποιήσουν την αναζήτηση με φίλτρα. Αν ο χρήστης θέλει όμως να εφαρμόσει ένα φίλτρο με περισσότερα από ένα επιστημονικά πεδία, η δυνατότητα αυτή δεν παρέχεται από την πλατφόρμα. Επίσης, το EasyChair προσφέρει φίλτρα ανα-ζήτησης στα topics, που είναι ουσιαστικά οι λέξεις-κλειδιά που συσχετίζονται με το CFP, και στις χώρες που λαμβάνουν μέρος τα CFPs. Το EasyChair δεν δίνει στον χρήστη τη δυνατότητα να εφαρμόσει όλα αυτά τα φίλτρα μαζί, με αποτέλεσμα ο χρήστης να αναγκάζεται να εφαρμόσει ένα φίλτρο και μετά να κάνει αναζήτηση με την τιμή του πεδίου του οποίου θέλει να φιλτράρει. Το αποτέλεσμα είναι πολλές φορές το EasyChair να μην επιστρέφει τα σωστά αποτελέσματα αναζήτησης και ο χρήστης να μην βρίσκει το επιθυμητό αποτέλεσμα. Αυτό βλάπτει έμμεσα και την προώθηση των CFPs μας.

Chapter 3

Τεχνολογίες

3.1 Τεχνολογίες για την Κατασκευή Πλατφόρμας

Για την κατασκευή μιας πλατφόρμας ή εφαρμογής, οι τεχνολογίες που χρησιμοποιούνται συνήθως διαχωρίζονται σε δύο βασικές κατηγορίες: το *frontend*, που αποτελεί τη διεπαφή χρήστη, και το *backend*, το οποίο περιλαμβάνει τη λογική και τη διαχείριση των δεδομένων.

3.1.1 Frontend

Το *frontend* είναι η διεπαφή που επιτρέπει στους χρήστες να αλληλεπιδρούν με την εφαρμογή. Κύριος στόχος του είναι η παρουσίαση δεδομένων και η δημιουργία μιας ευχάριστης και λειτουργικής εμπειρίας χρήστη (*User Interface/UX*). Βασικές τεχνολογίες για την ανάπτυξη του *frontend* είναι:

- Η **HTML (Hypertext Markup Language)**[9], που παρέχει τη δομή των ιστοσελίδων.
- Η **CSS (Cascading Style Sheets)**[10], που χρησιμοποιείται για τη μορφοποίηση και το στυλ.
- Η **JavaScript**[5], η οποία επιτρέπει τη δυναμικότητα και τη διαδραστικότητα.
- Σύγχρονα *frameworks* όπως το **React**[11] ή το **Angular**, που διευκολύνουν την ανάπτυξη αρθρωτών και προσαρμοστικών εφαρμογών.

3.1.2 Backend

Το *backend* είναι υπεύθυνο για τη διαχείριση αιτημάτων από το *frontend* ή από εξωτερικά APIs και περιλαμβάνει τη λογική της εφαρμογής και τη διαχείριση δεδομένων. Επιπλέον, το *backend* διασυνδέεται με βάσεις δεδομένων για την αποθήκευση και επεξεργασία των δεδομένων. Δημοφιλείς τεχνολογίες για την ανάπτυξη του *backend* περιλαμβάνουν:

- **Node.js**[8], που επιτρέπει την ανάπτυξη server-side εφαρμογών με `JavaScript`[5].
- **Python**[2], που χρησιμοποιείται ευρέως με *frameworks* όπως το Django ή το Flask.
- **Java**, **PHP** και **Ruby**, οι οποίες προσφέρουν λύσεις για εφαρμογές μεγαλύτερης κλίμακας.

3.1.3 Τεχνολογίες που χρησιμοποιήθηκαν για την Κατασκευή του CFPIndexer

Για την ανάπτυξη της πλατφόρμας **CFPIndexer**, επιλέχθηκαν σύγχρονες τεχνολογίες που εξασφαλίζουν αποδοτικότητα, ταχύτητα και ευελιξία. Στο *frontend* χρησιμοποιήθηκε το **Next.js**[4], ένα *JavaScript framework*[5] που υποστηρίζει τόσο *server-side rendering (SSR)* όσο και *static site generation (SSG)*, προσφέροντας ευκολία ανάπτυξης και παραμετροποίησης. Η **TypeScript**[6], που χρησιμοποιείται σε συνδυασμό με το `Next.js`[4], εξασφάλισε αυστηρότητα και αξιοπιστία στον κώδικα.
Στο *backend*, η γλώσσα **Python**[2] επιλέχθηκε για την αυτοματοποίηση της διαδικασίας άντλησης δεδομένων από την πλα-τφόρμα **EasyChair**[1]. Χρησιμοποιήθηκαν ισχυρές βιβλιοθήκες της `Python`[2], που διευκόλυναν τη διασύνδεση με APIs και τη διαχείριση δεδομένων.
Όλες οι παραπάνω τεχνολογίες συνδυάστηκαν για να δημιουργηθεί μια ολοκληρωμένη πλατφόρμα που καλύπτει τις ανάγκες ταχύτητας, ευελιξίας και αξιοπιστίας.

3.2 Τεχνολογίες Frontend

3.2.1 Hypertext Markup Language (HTML)

Η `HTML`[9] είναι μια γλώσσα περιγραφής περιεχομένου που χρησιμοποιείται για τη διαμόρφωση της δομής μιας ιστοσελίδας. Παρόλο που δεν είναι γλώσσα προγραμματισμού, παρέχει τις βασικές οδηγίες για τη διάταξη του περιεχομένου μέσω ειδικών ετικετών (*tags*). Οι περιηγητές (*browsers*) ερμηνεύουν αυτές τις ετικέτες και αποδίδουν την τελική μορφή στον χρήστη. Στην πλατφόρμα **CFPIndexer**, η `HTML`[9] χρησιμεύει ως βασικό στοιχείο για την κατασκευή της διεπαφής.

3.2.2 Cascading Style Sheets (CSS)

Η `CSS`[10] είναι ένα σύστημα κανόνων που εφαρμόζεται για τη μορφοποίηση της εμφάνισης των στοιχείων μιας ιστοσελίδας. Στην πλατφόρμα **CFPIndexer** χρησιμοποιήθηκε το *framework* **Tailwind CSS**[12], το οποίο είναι ένα *utility-first CSS framework* που διευκολύνει την ταχύτατη ανάπτυξη. Η `Tailwind`[12] επιτρέπει τη γρήγορη παραμετροποίηση και είναι ιδανική για τη δημιουργία διεπαφών που προσαρμόζονται σε διάφορα μεγέθη οθόνης.

3.2.3 React

Το **React**[11] είναι μια βιβλιοθήκη που αναπτύχθηκε από τη Meta και χρησιμοποιείται για τη δημιουργία διαδραστικών και δυναμικών διεπαφών. Ένα από τα βασικά πλεονεκτήματά του είναι η διαχείριση του *DOM* μέσω `JavaScript`[5], επιτρέποντας την αλλαγή της δομής της σελίδας χωρίς πλήρη επαναφόρτωση.
Στην πλατφόρμα **CFPIndexer**, το `React`[11] χρησιμοποιήθηκε σε συνδυασμό με **TypeScript**[6], γεγονός που ενίσχυσε την αξιοπιστία του κώδικα. Επιπλέον, το `React`[11] παρέχει τη δυνατότητα ανάπτυξης επαναχρησιμοποιήσιμων στοιχείων (*components*), καθιστώντας τη διαδικασία ανάπτυξης πιο αποδοτική και εύκολη στη συντήρηση.

Μερικες απο τις βιβλιοθηκες που χρησιμοποιηθηκαν κατα την αναπτυξη της πλατφορμας ειναι οι εξης:

- **NextUI**, είναι μια βιβλιοθήκη από components τα οποία χρησιμοποιούνται για να προσφέρουν διαδραστικότητα μεταξύ εφαρμογής και χρήστη. Τα components τα οποία έχουν χρησιμο-ποιηθεί είναι κυρίως για εισαγωγή δεδομένων. Επίσης, το NavBar έχει σχεδιαστεί και έχει υλοποιηθεί με τη χρήση αυτής της βιβλιοθήκης. Copy

- **Material React Table**[13], που χρησιμοποιείται στην πλατφόρμα μας για τους πίνακες που υπάρχουν. Η βιβλιοθήκη αυτή εισάγει πίνακες οι οποίοι είναι άκρως παραμετροποιήσιμοι. Επίσης, παρέχει δυνατότητες όπως φιλτράρισμα σε κάθε στήλη αλλά και αναδιοργάνωση των στηλών από τον χρήστη.

- **Pigeon Maps**, είναι μια βιβλιοθήκη η οποία εισάγει χάρτη στον οποίο μπορούμε να προσθέσουμε διάφορες πινέζες, διανύσματα και σύμβολα. Στην περίπτωση του CFPIndexer έχουμε βάλει πινέζες στις πόλεις που έχουμε συνέδρια για τα ανάλογα φίλτρα που έχουμε εφαρμό-σει.

3.2.4 Typescript

Η `TypeScript`[6] είναι μια επέκταση της `JavaScript`[5], η οποία έχει αναπτυχθεί από τη Microsoft και εισάγει έννοιες αντικειμενοστραφούς προγραμματισμού στη `JavaScript`[5]. Η χρήση της `TypeScript`[6] προσφέρει στον προγραμματιστή προειδοποιήσεις για σφάλματα, καθώς και αυτόματη συμπλήρωση κώδικα κατά την ανάπτυξη, όταν χρησιμοποιείται οποιοδήποτε σύγχρονο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Ο κώδικας της TypeScript μεταφράζεται σε `JavaScript`[5] κατά τη διαδικασία της μεταγλώττισης και με αυτόν τον τρόπο υποστη-ρίζεται και μπορεί να εκτελεστεί σε όλες τις πλατφόρμες και περιβάλλοντα που μπορούν να εκτελέσουν κώδικα `JavaScript`[5]. Για αυτούς τους λόγους επιλέχθηκε η ανάπτυξη του κώδικα να γίνει με τη χρήση της `TypeScript`[6], καθώς παρέχει μεγαλύτερη ασφάλεια, ευκολία και ποιότητα στο τελικό αποτέλεσμα. Με τη χρήση διαφόρων package managers, οι οποίοι μπορούν εξαιρετικά εύκολα να προσθέσουν διάφορες βιβλιοθήκες που επιλύουν διάφορα προβλήματα, μειώνεται δραστικά ο χρόνος ανάπτυξης του κώδικα, προσφέροντας παράλληλα μεγάλη ασφά-λεια, καθώς χρησιμοποιούνται βιβλιοθήκες που έχουν ελεγχθεί από χιλιάδες χρήστες. Μερικές από τις βιβλιοθήκες που χρησιμοποιήθηκαν κατά την ανάπτυξη της πλατφόρμας είναι:

- **NextAuth.js**[14], που παρέχει ένα σύστημα αυθεντικοποίησης χρηστών το οποίο είναι άκρως ασφαλές. Η αυθεντικοποίηση μπορεί να γίνει με τη χρήση διαφόρων παρόχων όπως το Google, το `GitHub`[15] και άλλοι. Επίσης, δίνει τη δυνατότητα υλοποίησης ενός απλού συστήματος με χρήση email και κωδικού πρόσβασης. Στην περίπτωση του CFPIndexer, χρησιμοποιούμε τον τρόπο αυθεντικοποίησης με email/κωδικό πρόσβασης και τον πάροχο Google. Μετά την είσοδο, χρησιμοποιούμε συνεδρίες (sessions) για να επιβεβαιώνουμε την ταυτότητα κάθε χρήστη σε όλες τις σελίδες της πλατφόρμας. Copy

- **jsonwebtoken**[16], που χρησιμοποιείται ευρέως σε διάφορα frameworks για τη δημιουργία διακριτικών (tokens) τα οποία περιέχουν κρυπτογραφημένες πληροφορίες συνήθως για

τον χρήστη και επισυνάπτονται σε διάφορα αιτήματα ή απαντήσεις προς ή από διακομιστές.

- **Nodemailer**[17], μια βιβλιοθήκη που χρησιμοποιήθηκε για την αποστολή διαφόρων emails προς τους εγγεγραμμένους χρήστες της εφαρμογής.

3.2.5 Next.js

Το **Next.js**[4] είναι ένα προηγμένο framework για εφαρμογές `React`[11] που προσφέρει μια συνδυασμένη προσέγγιση της απεικόνισης στην πλευρά του διακομιστή (Server-Side Rendering, SSR), της δημιουργίας στατικών ιστοσελίδων (Static Site Generation, SSG) και τεχνικών απεικόνισης στην πλευρά του πελάτη. Αναπτύχθηκε από την `Vercel`[18], με στόχο την απλοποίηση της ανάπτυξης πολύπλοκων εφαρμογών `React`[11], παρέχοντας έναν τυποποιημένο τρόπο δημιουργίας πλήρως λειτουργικών και βελτιστοποιημένων εφαρμογών ιστού. Λόγω της χρήσης αυτών των δυνατοτήτων, οι ιστοσελίδες έχουν πολύ υψηλή βαθμολογία στις μηχανές αναζήτησης, καθιστάμενες πιο αποδοτικές. Το βασικό χαρακτηριστικό που κάνει την ανάπτυξη κώδικα ευκολότερη είναι ότι δίνει τη δυνατότητα ανάπτυξης frontend και backend ταυτόχρονα μέσα από το ίδιο αποθετήριο (repository) με τη χρήση της ίδιας γλώσσας. Για αυτό τον λόγο, η ανάπτυξη εφαρμογών με τέτοια frameworks καλείται και monorepo. Τα χαρακτηριστικά που χρησιμοποιήθηκαν κατά την ανάπτυξη περιλαμβάνουν:

- Τον δρομολογητή (router) του `Next.js`[4] που βοηθά στην πλοήγηση εντός της πλατφόρμας
- Την υλοποίηση API
- Τα components
- Τα server actions για την υλοποίηση frontend και backend αντίστοιχα

3.2.6 Python

Η **Python**[2] είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου, με ενσωματωμένες δομές δεδομένων και δυναμικές ιδιότητες. Είναι αρκετά εύκολη στη χρήση για την ανάπτυξη ολοκληρωμένων ή εξειδικευμένων εφαρμογών. Επίσης, είναι η γλώσσα που συνήθως διδάσκεται σε άτομα χωρίς προγραμματιστική εμπειρία, ιδιαίτερα για εφαρμογές στατιστικής και εξόρυξης δεδομένων. Μπορεί να εκτελεστεί σε όλες τις πλατφόρμες που υποστηρίζουν το περιβάλλον της. Κατά τη διάρκεια σχεδιασμού της πλατφόρμας, αποφασίστηκε να χρησιμοποιηθεί η `Python`[2] για την εξόρυξη δεδομένων (CFPs) από την πλατφόρμα του `EasyChair`[1]. Η χρήση της `Python`[2] για αυτή τη λειτουργία είναι ιδιαιτέρως διαδεδομένη. Η βιβλιοθήκη που χρησιμοποιήθηκε είναι η **Beautiful Soup**[3], ένας parser για αρχεία `HTML`[9] και XML που δημιουργεί μια δομή προσβάσιμη μέσω προγραμματισμού. Ουσιαστικά, επιτρέπει την πλοήγηση μέσα σε ιστοσελίδες και την εξαγωγή των επιθυμητών δεδομένων.

3.2.7 MariaDB

Η **MariaDB**[7] είναι ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων. Το πρόγραμμα εκτελεί έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών χρηστών σε ένα σύνολο βάσεων δεδομένων. Η `MariaDB`[7] έχει υλοποιηθεί ως κλάδος της MySQL και είναι μια σχεσιακή βάση δεδομένων που κάνει χρήση πρωτευόντων κλειδιών. Ο σχεδιασμός του σχήματος της βάσης για την πλατφόρμα του CFPIndexer έγινε βάσει των αναγκών και των λειτουργιών που έπρεπε να υλοποιηθούν.

3.3 Συμπέρασμα

Η επιλογή των παραπάνω τεχνολογιών για την ανάπτυξη της πλατφόρμας **CFPIndexer** ήταν καθοριστική για την επίτευξη των στόχων της. Ο συνδυασμός τους εξασφάλισε μια ευέλικτη, αποδοτική και σύγχρονη εφαρμογή, προσαρμοσμένη στις απαιτήσεις των χρηστών.

Chapter 4

Υλοποίηση του CFPIndexer

Σε αυτό το κεφάλαιο θα αναλύσουμε τον τρόπο με τον οποίο υλοποιήθηκε το CFPIndexer καθώς και τις τεχνολογίες που χρησιμοποιήθηκαν. Τα επόμενα υποκεφάλαια αποδομούν τις βασικές λειτουργίες και απαιτήσεις του CFPIndexer καθώς και πιθανά σενάρια χρήσης της εφαρμογής - πλατφόρμας.

4.1 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις που ήταν ο γνώμονας για να υλοποιηθεί το CFPIndexer ήταν πολλές, σε αυτό το υποκεφάλαιο θα αναλύσουμε τις βασικές με την δομή "User Stories". Η δομή αυτή είναι μια γενική επεξήγηση με περιγραφικό τρόπο των απαιτήσεων και των χαρακτηριστικών μιας εφαρμογής, από την οπτική ενός χρήστη της με σκοπό την ευκολότερη κατανόηση των υλοποιήσεών τους από τους χρήστες. Τα "User Stories" χρησιμοποιούνται ευρέως κατά το στάδιο της παρουσίασης νέων λειτουργιών που έχουν υλοποιηθεί κατά την διάρκεια του σταδίου της υλοποίησης.

4.1.1 Αναζήτηση CFP

Ο χρήστης θέλει να βρει τα CFP για τα οποία ισχύουν τα εξής φίλτρα: η τοποθεσία του να είναι Ελλάδα, η κατηγορία να είναι "Computing" και η ημερομηνία έναρξης από 01/04/2025 έως τις 30/04/2025. Τα βήματα που πρέπει να ακολουθήσει είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί θα δει την λίστα των CFP που έχουν deadline μέσα στο διάστημα ενός μήνα από την σημερινή ημερομηνία.
- Ο χρήστης θέτει την ημερομηνία 01/04/2025 στο αριστερό πεδίο της στήλης "Starting date".
- Ο χρήστης θέτει την ημερομηνία 30/04/2025 στο δεξί πεδίο της στήλης "Starting date".
- Ο χρήστης θέτει την τιμή "Computing" από την λίστα που εμφανίζεται μόλις πατήσει επάνω στο πεδίο "Areas".

- Ο χρήστης θέτει την τιμή ”Greece” από την λίστα που εμφανίζεται μόλις πατήσει επάνω στο πεδίο ”Location”.
- Η εφαρμογή κατά την διάρκεια των αλλαγών επαναφορτώνει την λίστα με τα αποτελέσματα των φίλτρων και ο χρήστης τελικά βλέπει τα αποτελέσματα τα οποία αναζητεί.

4.1.2 Πλοήγηση στα CFP με φίλτρα και προβολή πληροφοριών CFP

Ο χρήστης θέλει να βρει τα CFP για τα οποία ισχύουν τα εξής φίλτρα: η κατηγορία να είναι ”Computing” και η ημερομηνία έναρξης από 01/04/2025 έως τις 30/04/2025. Τα βήματα που πρέπει να ακολουθήσει είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί θα δει την λίστα των CFP που έχουν deadline μέσα στο διάστημα ενός μήνα από την σημερινή ημερομηνία.
- Ο χρήστης θέτει την ημερομηνία 01/04/2025 στο αριστερό πεδίο της στήλης ”Starting date”.
- Ο χρήστης θέτει την ημερομηνία 30/04/2025 στο δεξί πεδίο της στήλης ”Starting date”.
- Ο χρήστης θέτει την τιμή ”Computing” από την λίστα που εμφανίζεται μόλις πατήσει επάνω στο πεδίο ”Areas”.
- Η εφαρμογή κατά την διάρκεια των αλλαγών επαναφορτώνει την λίστα με τα αποτελέσματα των φίλτρων και ο χρήστης τελικά βλέπει τα αποτελέσματα τα οποία αναζητεί.
- Ο χρήστης μπορεί να αλλάξει τον τρόπο με τον οποίο εμφανίζονται τα CFP σε μορφή χάρτη ή πίνακα και να πλοηγηθεί ανάλογα στα αποτελέσματα.
- Όταν βρει το CFP που τον ενδιαφέρει πατάει επάνω στην τιμή της στήλης acronym.
- Αν η πλοήγηση γίνεται με την χρήση του χάρτη τότε ο χρήστης πατάει επάνω στην πινέζα και ανακατευθύνεται σε έναν πίνακα με τα ανάλογα αποτελέσματα στα οποία κάνει ακριβώς την ίδια ενέργεια με το προηγούμενο βήμα.
- Τέλος ο χρήστης ανακατευθύνεται στην σελίδα του CFP το οποίο διάλεξε.

4.1.3 Εγγραφή στην πλατφόρμα CFPIndexer

Ο χρήστης θέλει να εγγραφεί στην πλατφόρμα με την χρήση email/κωδικού, ώστε να έχει τις δυνατότητες εγγεγραμμένων χρηστών. Τα βήματα που πρέπει να ακολουθήσει είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί θα δει το κουμπί ”sign up” επάνω δεξιά στην μπάρα πλοήγησης.
- Πατώντας το κουμπί ανακατευ-θύνεται στην σελίδα εγγραφής στην εφαρμογή.
- Ο χρήστης συμπληρώνει το email του στο ανάλογο πεδίο.

- Ο χρήστης συμπληρώνει τον κωδικό του στο ανάλογο πεδίο.
- Ο χρήστης συμπληρώνει ξανά τον κωδικό του στο πεδίο ”Confirm password”.
- Ο χρήστης πατάει το κουμπί ”Signup” αν δεν υπάρξει κάποιο πρόβλημα ο χρήστης ανακατευθύνεται στην σελίδα σύνδεσης στην πλατφόρμα.

4.1.4 Σύνδεση στην πλατφόρμα CFPIndexer

Ο χρήστης θέλει να εγγραφεί στην πλατφόρμα είτε με την χρήση email/κωδικού είτε με την χρήση λογαριασμού google. Τα βήματα που πρέπει να ακολουθήσει είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί θα δει το κουμπί ”Login” επάνω δεξιά στην μπάρα πλοήγησης.
- Πατώντας το κουμπί ανακατευθύνεται στην σελίδα σύνδεσης στην εφαρμογή.
- Ο χρήστης συμπληρώνει το email του στο ανάλογο πεδίο.
- Ο χρήστης συμπληρώνει τον κωδικό του στο ανάλογο πεδίο.
- Ο χρήστης πατάει το κουμπί ”Login” αν δεν υπάρξει κάποιο πρόβλημα ο χρήστης ανακατευθύνεται στην αρχική σελίδα της πλατφόρμας.
- Αν ο χρήστης θέλει να κάνει είσοδο με τον λογαριασμό απλά πατάει το κουμπί ”Continue with Google”.
- Διαλέγει τον λογαριασμό αποδέχεται τους όρους.
- Αν δεν υπάρχει κάποιο πρόβλημα ανακατευθύνεται στην αρχική σελίδα της εφαρμογής.

4.1.5 Εγγραφή στις ειδοποιήσεις σε διάφορα φίλτρα των CFP

Ο χρήστης θέλει να εγγραφεί στις ειδοποιήσεις της πλατφόρμας για κάποια φίλτρα των CFP που τον ενδιαφέρουν. Τα βήματα που πρέπει να ακολουθήσει για να λαμβάνει email για όλα τα CFP με χώρα την Ελλάδα είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί εφόσον έχει κάνει σύνδεση, θα δει το κουμπί ”Profile” επάνω δεξιά στην μπάρα πλοήγησης.
- Πατώντας το κουμπί ανακατευθύνεται στην σελίδα σύνδεσης στο προφίλ του.
- Ο χρήστης πατάει στο κουμπί ”Subscriptions”.
- Εμφανίζονται οι εγγραφές του χρήστη.
- Ο χρήστης πατάει στο κουμπί ”countries” και εμφανίζονται όλες οι χώρες.
- Διαλέγει την Ελλάδα.
- Πατάει το κουμπί update.
- Ο χρήστης έχει εγγραφεί στις ειδοποιήσεις για τα CFP που λαμβάνουν χώρα στην Ελλάδα.

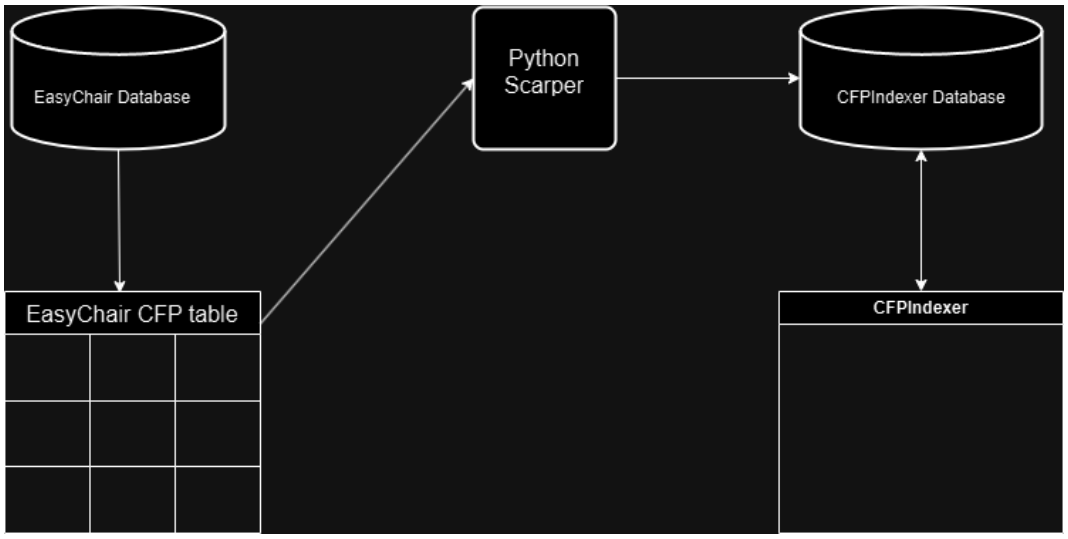
4.1.6 Δημιουργία κλειδιού API για την χρήση του

Ο χρήστης θέλει να δημιουργήσει κλειδί API για την χρήση του API της πλατφόρμας. Τα βήματα που πρέπει να ακολουθήσει είναι:

- Ο χρήστης πρέπει να πλοηγηθεί στην αρχική σελίδα της πλατφόρμας CFPIndexer εκεί εφόσον έχει κάνει σύνδεση, θα δει το κουμπί "Profile" επάνω δεξιά στην μπάρα πλοήγησης.
- Πατώντας το κουμπί ανακατευθύνεται στην σελίδα σύνδεσης στο προφίλ του.
- Ο χρήστης πατάει στο κουμπί "Request API Key".
- Εφόσον δημιουργηθεί το κλειδί θα αλλάξει η τιμή του ανάλογου πεδίου.
- Ο χρήστης θα πατήσει το κουμπί εντός του πεδίου το οποίο αντιγράφει το κλειδί και μπορεί να το χρησιμοποιήσει.

4.2 Αρχιτεκτονική εφαρμογής

Η αρχιτεκτονική της εφαρμογής είναι αυτή που παρουσιάζεται στο επόμενο διάγραμμα ροής.



4.3 Βάση Δεδομένων

Η βάση δεδομένων που χρησιμοποιήθηκε είναι μια σχεσιακή βάση δεδομένων που υλοποιείται με τη χρήση της `MariaDB`[7]. Στη βάση συνδέονται τα δύο κύρια μέρη της πλατφόρμας. Το Python script που είναι υπεύθυνο για την άντληση των δεδομένων, μόλις γίνει η επεξεργασία τους, τα εισάγει στη βάση ως εγγραφές στους ανάλογους πίνακες, οι οποίοι θα αναλυθούν παρακάτω. Το δεύτερο κύριο μέρος είναι η ιστοσελίδα της πλατφόρμας, η οποία συνδέεται στη βάση για να κάνει προβολή, τροποποίηση και εισαγωγή δεδομένων σε αυτήν.
Παρακάτω θα αναλυθούν οι πίνακες της βάσης με τη σειρά που πρέπει να δημιουργηθούν, ώστε να είναι δυνατόν να υλοποιηθούν και οι απαιτούμενες σχέσεις μεταξύ τους.

4.3.1 Πίνακας `locations` - Τοποθεσίες CFP

Ο πίνακας `locations` είναι ο πίνακας που περιγράφει μια τοποθεσία στην οποία πραγματοποιείται ένα CFP. Τα πεδία του είναι τα εξής:
Το πεδίο `Id` είναι ένας αυτόματα αυξανόμενος αριθμός, που αποτελεί το κύριο κλειδί του πίνακα. Το πεδίο `city` περιγράφει το όνομα της πόλης, και είναι τύπου αλφαριθμητικού (`VARCHAR`) έως 255 χαρακτήρες. Το πεδίο `state` αναφέρεται στην πολιτεία ή περιοχή και είναι επίσης τύπου αλφαριθμητικού (`VARCHAR`) έως 255 χαρακτήρες, ενώ το πεδίο `country` περιγράφει τη χώρα, με τον ίδιο τύπο δεδομένων. Τα πεδία `latitude` και `longitude` περιγράφουν τις γεωγραφικές συντεταγμένες της τοποθεσίας και είναι τύπου δεκαδικού (`DECIMAL`). Τέλος, το πεδίο `created_at` είναι τύπου χρονοσήμανσης (`TIMESTAMP`), και χρησιμοποιείται για λόγους παρακολούθησης.
Επιπροσθέτως, υπάρχει ένας περιορισμός (*constraint*) στον πίνακα, ο οποίος απαιτεί ο συνδυασμός των πεδίων `city`, `state` και `country` να είναι μοναδικός, ώστε να αποτρέπονται διπλότυπες εγγραφές.

4.3.2 Πίνακας `topics` - Θέματα CFP

Ο πίνακας `topics` περιγράφει τα θέματα που σχετίζονται με τα CFP. Τα πεδία του πίνακα είναι τα εξής:
Το πεδίο `id` είναι ένας αυτόματα αυξανόμενος αριθμός (`AUTO_INCREMENT`), που αποτελεί το κύριο κλειδί (`PRIMARY KEY`) του πίνακα. Το πεδίο `topic_id` είναι τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 20 χαρακτήρες και χρησιμοποιείται για να αποθηκεύσει τον μοναδικό αναγνωριστικό κωδικό του θέματος και ο οποίος ειναι είτε ο αριθμός που έχει στην πλατφόρμα του `EasyChair`[1] είτε ενας μοναδικός αριθμός 18 χαρακτήρων τύπου `UUID_SHORT`. Το πεδίο `name` είναι επίσης τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 255 χαρακτήρες και περιγράφει το όνομα του θέματος.
Το πεδίο `created_at` είναι τύπου χρονοσήμανσης (`TIMESTAMP`), με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`) και χρησιμοποιείται για λόγους παρακολούθησης της ημερομηνίας δημιουργίας της εγγραφής.
Επιπλέον, υπάρχει ένας περιορισμός (*constraint*) στον πίνακα, ο οποίος απαιτεί ο συνδυασμός των πεδίων `topic_id` και `name` να είναι μοναδικός, αποτρέποντας έτσι τη δημιουργία διπλότυπων εγγραφών.

4.3.3 Πίνακας `areas` - Γνωστικές Περιοχές CFP

Ο πίνακας `areas` περιγράφει τις γνωστικές περιοχές που σχετίζονται με τα CFP. Τα πεδία του πίνακα είναι τα εξής:
Το πεδίο `id` είναι ένας αυτόματα αυξανόμενος αριθμός (`AUTO_INCREMENT`), που αποτελεί το κύριο κλειδί (`PRIMARY KEY`) του πίνακα. Το πεδίο `area_id` είναι τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 16 χαρακτήρες και χρησιμοποιείται για να αποθηκεύσει τον μοναδικό αναγνωριστικό κωδικό της γνωστικής περιοχής. Το πεδίο `name` είναι επίσης τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 255 χαρακτήρες και περιγράφει το όνομα της γνωστικής περιοχής.
Το πεδίο `link` είναι τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 255 χαρακτήρες

και χρησιμοποιείται για να αποθηκεύσει έναν σύνδεσμο (URL) που σχετίζεται με τη γνωστική περιοχή.
Το πεδίο `created_at` είναι τύπου χρονοσήμανσης (`TIMESTAMP`), με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`) και χρησιμοποιείται για λόγους παρακολούθησης της ημερομηνίας δημιουργίας της εγγραφής.
Επιπλέον, υπάρχει ένας περιορισμός (*constraint*) στον πίνακα, ο οποίος απαιτεί ο συνδυασμός των πεδίων `area_id`, `name` και `link` να είναι μοναδικός. Αυτό αποτρέπει τη δημιουργία διπλότυπων εγγραφών που περιέχουν τις ίδιες τιμές και στα τρία αυτά πεδία.

4.3.4 Πίνακας `userSubscriptionTypes` - Τύποι Συνδρομών Χρηστών

Ο πίνακας `userSubscriptionTypes` περιγράφει τους διαφορετικούς τύπους συνδρομών χρηστών στην πλατφόρμα. Τα πεδία του πίνακα είναι τα εξής:
Το πεδίο `id` είναι ένας αυτόματα αυξανόμενος αριθμός (`AUTO_INCREMENT`), που αποτελεί το κύριο κλειδί (`PRIMARY KEY`) του πίνακα. Το πεδίο `description` είναι τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 255 χαρακτήρες και περιέχει μια περιγραφή του τύπου συνδρομής.
Το πεδίο `created_at` είναι τύπου χρονοσήμανσης (`TIMESTAMP`), με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`) και χρησιμοποιείται για λόγους παρακολούθησης της ημερομηνίας δημιουργίας της εγγραφής.
Ο πίνακας αυτός επιτρέπει την καταγραφή και την οργάνωση των διαφορετικών τύπων συνδρομών χρηστών, διευκολύνοντας τη διαχείριση των σχετικών δεδομένων.
Οι τρεις εγγραφές που έχει ο πίνακας είναι ανάλογες των τριών προηγούμενων πινάκων, δηλαδή ”Area”, ”Topic” και ”Location”.

4.3.5 Πίνακας `users` - Χρήστες Πλατφόρμας

Ο πίνακας `users` διαχειρίζεται τις πληροφορίες που σχετίζονται με τους χρήστες της πλατφόρμας. Τα πεδία του πίνακα είναι τα εξής:
Το πεδίο `uuid` είναι ένας μοναδικός αναγνωριστικός αριθμός τύπου `UUID`[19], το οποίο λειτουργεί ως κύριο κλειδί (`PRIMARY KEY`) του πίνακα.Η τιμή, δημιουργείται αυτόματα ως μοναδική τιμή (`DEFAULT UUID()`).
Το πεδίο `username` είναι τύπου αλφαριθμητικού (`VARCHAR`) με μέγιστο μήκος 255 χαρακτήρες και μπορεί να είναι κενό (`NULL`). Το πεδίο `email` είναι επίσης τύπου `VARCHAR` με μέγιστο μήκος 255 χαρακτήρες και καταγράφει τη διεύθυνση email του χρήστη. Το πεδίο `password` είναι τύπου `VARCHAR` και περιέχει τον κωδικό πρόσβασης του χρήστη σε κρυπτογραφημένη μορφή.
Το πεδίο `token` είναι τύπου `TEXT` και αποθηκεύει ένα μοναδικό διακριτικό (`token`) για την αυθεντικοποίηση στο API της πλατφόρμας. Το πεδίο `token_expiration_date` είναι τύπου `DATE` και αποθηκεύει την ημερομηνία λήξης του `token`, αν υπάρχει.
Το πεδίο `email_notifications` είναι τύπου `BOOLEAN` και προσδιορίζει αν ο χρήστης επιθυμεί να λαμβάνει ειδοποιήσεις μέσω email, με προεπιλεγμένη τιμή `FALSE`. Το πεδίο `isActivated` είναι επίσης τύπου `BOOLEAN` και υποδεικνύει αν ο λογαριασμός του χρήστη έχει ενεργοποιηθεί, με προεπιλεγμένη τιμή `FALSE`. Το πεδίο `push_notifications` είναι τύπου `BOOLEAN` και δηλώνει αν ο χρήστης έχει ενεργοποιήσει ειδοποιήσεις μέσω συσκευών,

επίσης με προεπιλεγμένη τιμή FALSE και το οποίο θα χρησιμοποιηθεί σε μελλοντικές υλοποιήσεις και επεκτάσεις του κωδικά.
Το πεδίο created_at είναι τύπου TIMESTAMP και καταγράφει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (DEFAULT CURRENT_TIMESTAMP).
Ο πίνακας users παρέχει τη βασική υποδομή για την αποθήκευση και διαχείριση δεδομένων χρήστη στην πλατφόρμα.

4.3.6 Πίνακας cfps - Ανακοινώσεις Πρόσκλησης για Συνεδρίες (Call for Papers - CFPs)

Ο πίνακας cfps διαχειρίζεται τις πληροφορίες σχετικά με ανακοινώσεις πρόσκλησης για συνεδρίες (Call for Papers - CFPs). Τα πεδία του πίνακα είναι τα εξής:
Το πεδίο id είναι ένας ακέραιος αριθμός (INT) που αυξάνεται αυτόματα (AUTO_INCREMENT) και χρησιμεύει ως το κύριο κλειδί (PRIMARY KEY) του πίνακα.
Το πεδίο cfp_link είναι τύπου VARCHAR με μέγιστο μήκος 255 χαρακτήρες. Αποθηκεύει τον μοναδικό σύνδεσμο της ανακοίνωσης (NOT NULL UNIQUE).
Το πεδίο acronym είναι τύπου VARCHAR με μέγιστο μήκος 255 χαρακτήρες και αποθηκεύει το ακρώνυμο της ανακοίνωσης, ενώ το πεδίο name είναι τύπου VARCHAR και περιέχει το πλήρες όνομα της ανακοίνωσης.
Το πεδίο location_id είναι τύπου INT και αποτελεί ξένο κλειδί (FOREIGN KEY) που αναφέρεται στο πεδίο id του πίνακα locations. Σε περίπτωση διαγραφής της σχετικής τοποθεσίας, η τιμή του πεδίου γίνεται NULL (ON DELETE SET NULL).
Τα πεδία deadline και starting_date είναι τύπου DATE και περιέχουν αντίστοιχα την προθεσμία υποβολής και την ημερομηνία έναρξης της εκδήλωσης. Το πεδίο deadline μπορεί να είναι NULL.
Το πεδίο is_easychair είναι τύπου BOOLEAN και υποδεικνύει αν η ανακοίνωση σχετίζεται με την πλατφόρμα EasyChair[1], με προεπιλεγμένη τιμή FALSE.
Το πεδίο description είναι τύπου TEXT και αποθηκεύει την περιγραφή της ανακοίνωσης.
Το πεδίο author είναι τύπου UUID[19] και αποτελεί ξένο κλειδί (FOREIGN KEY) που αναφέρεται στο πεδίο uuid[19] του πίνακα users. Εάν ο χρήστης διαγραφεί, η τιμή του πεδίου γίνεται NULL (ON DELETE SET NULL).
Το πεδίο created_at είναι τύπου TIMESTAMP και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (DEFAULT CURRENT_TIMESTAMP). Το πεδίο last_modified_date είναι τύπου DATE και περιέχει την ημερομηνία τελευταίας τροποποίησης της εγγραφής.
Ο πίνακας cfps παρέχει τη βασική υποδομή για την αποθήκευση και διαχείριση ανακοινώσεων Call for Papers στην πλατφόρμα.

4.3.7 Πίνακας cfpsToAreas - Συσχέτιση Ανακοινώσεων CFP με Γνωστικές Περιοχές

Ο πίνακας cfpsToAreas διαχειρίζεται τη συσχέτιση μεταξύ ανακοινώσεων Call for Papers (CFPs) και γνωστικών περιοχών, επιτρέποντας την ταξινόμηση κάθε ανακοίνωσης με βάση

τις θεματικές περιοχές της. Τα πεδία του πίνακα περιγράφονται παρακάτω:
Το πεδίο `id` είναι ένας ακέραιος αριθμός (`INT`) που αυξάνεται αυτόματα (`AUTO_INCREMENT`) και χρησιμεύει ως το κύριο κλειδί (`PRIMARY KEY`) του πίνακα.
Το πεδίο `cfp_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που ανα-φέρεται στο πεδίο `id` του πίνακα `cfps`. Η σχέση αυτή υποδεικνύει την ανακοίνωση CFP που σχετίζεται με τη γνωστική περιοχή. Σε περίπτωση διαγραφής της σχετικής ανακοίνωσης, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `area_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `areas`. Η σχέση αυτή υποδεικνύει τη γνωστική περιοχή που σχετίζεται με την ανακοίνωση CFP. Σε περίπτωση διαγραφής της σχετικής γνωστικής περιοχής, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `created_at` είναι τύπου `TIMESTAMP` και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`).
Ο πίνακας `cfpsToAreas` προσφέρει τη δυνατότητα για αποτελεσματική διαχείριση των θεματικών περιοχών που σχετίζονται με κάθε ανακοίνωση CFP, διευκολύνοντας την κατηγοριοποίηση και την αναζήτηση με βάση τις γνωστικές περιοχές.

4.3.8 Πίνακας `cfpsToTopics` - Συσχέτιση Ανακοινώσεων CFP με Θεματικές Ενότητες

Ο πίνακας `cfpsToTopics` διαχειρίζεται τη συσχέτιση μεταξύ ανακοινώσεων Call for Papers (CFPs) και θεματικών ενοτήτων, επιτρέποντας την κατηγοριοποίηση κάθε ανακοίνωσης με βάση τις θεματικές ενότητες που καλύπτει. Τα πεδία του πίνακα περιγράφονται παρακάτω:
Το πεδίο `id` είναι ένας ακέραιος αριθμός (`INT`) που αυξάνεται αυτόματα (`AUTO_INCREMENT`) και χρησιμεύει ως το κύριο κλειδί (`PRIMARY KEY`) του πίνακα.
Το πεδίο `cfp_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `cfps`. Η σχέση αυτή υποδεικνύει την ανακοίνωση CFP που σχετίζεται με τη θεματική ενότητα. Σε περίπτωση διαγραφής της σχετικής ανακοίνωσης, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `topic_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `topics`. Η σχέση αυτή υποδεικνύει τη θεματική ενότητα που σχετίζεται με την ανακοίνωση CFP. Σε περίπτωση διαγραφής της σχετικής θεματικής ενότητας, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `created_at` είναι τύπου `TIMESTAMP` και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`).
Ο πίνακας `cfpsToTopics` διευκολύνει τη διαχείριση των θεματικών ενοτήτων που σχετίζονται με κάθε ανακοίνωση CFP, καθιστώντας δυνατή την ταξινόμηση και την αναζήτηση με βάση τις θεματικές ενότητες.

4.3.9 Πίνακας `userSubscriptions` - Συνδρομές Χρηστών

Ο πίνακας `userSubscriptions` χρησιμοποιείται για τη διαχείριση των συνδρο-μών χρηστών σε συγκεκριμένους τύπους συνδρομών που παρέχονται από την πλατφόρμα. Τα πεδία του

πίνακα περιγράφονται ως εξής:
Το πεδίο `id` είναι τύπου ακέραιου (`INT`), αυξάνεται αυτόματα (`AUTO_INCREMENT`) και αποτελεί το κύριο κλειδί (`PRIMARY KEY`) του πίνακα.
Το πεδίο `subscription_type_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `userSubscriptionTypes`. Υποδεικνύει τον τύπο συνδρομής στον οποίο ανήκει η εγγραφή. Σε περίπτωση διαγραφής του σχετικού τύπου συνδρομής, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `user_id` είναι τύπου `UUID`[19] και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `uuid`[19] του πίνακα `users`. Υποδεικνύει τον χρήστη που σχετίζεται με τη συγκεκρι-μένη εγγραφή συνδρομής. Σε περίπτωση διαγραφής του σχετικού χρήστη, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `created_at` είναι τύπου `TIMESTAMP` και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`).
Ο πίνακας περιέχει έναν μοναδικό περιορισμό (`UNIQUE CONSTRAINT`) με την ονομασία `uc_subType_user`, ο οποίος διασφαλίζει ότι κάθε συνδυασμός `subscription_type_id` και `user_id` είναι μοναδικός. Αυτό σημαίνει ότι κάθε χρήστης μπορεί να έχει μόνο μία συνδρομή για κάθε τύπο συνδρομής.
Ο πίνακας `userSubscriptions` επιτρέπει τη συσχέτιση χρηστών με τύπους συνδρομών, παρέχοντας ευελιξία και ακρίβεια στη διαχείριση των συνδρομών.

4.3.10 Πίνακας `subscriptions` - Συνδρομές

Ο πίνακας `subscriptions` χρησιμοποιείται για τη διαχείριση των συνδρομών που σχετίζονται με τοποθεσίες, γνωστικές περιοχές και θέματα στην πλατφόρμα. Τα πεδία του πίνακα περιγράφονται ως εξής:
Το πεδίο `id` είναι τύπου ακέραιου (`INT`), αυξάνεται αυτόματα (`AUTO_INCREMENT`) και αποτελεί το κύριο κλειδί (`PRIMARY KEY`) του πίνακα.
Το πεδίο `location_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `locations`. Υποδεικνύει την τοποθεσία που σχετίζεται με τη συνδρομή. Σε περίπτωση διαγραφής της σχετικής τοποθεσίας, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `area_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `areas`. Υποδεικνύει τη γνωστική περιοχή που σχετίζεται με τη συνδρομή. Σε περίπτωση διαγραφής της σχετικής γνωστικής περιοχής, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `topic_id` είναι τύπου `INT` και αποτελεί ξένο κλειδί (`FOREIGN KEY`) που αναφέρεται στο πεδίο `id` του πίνακα `topics`. Υποδεικνύει το θέμα που σχετίζεται με τη συνδρομή. Σε περίπτωση διαγραφής του σχετικού θέματος, η εγγραφή διαγράφεται αυτόματα (`ON DELETE CASCADE`).
Το πεδίο `created_at` είναι τύπου `TIMESTAMP` και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (`DEFAULT CURRENT_TIMESTAMP`).
Το πεδίο `unique_key` είναι τύπου `VARCHAR(255)` και δημιουργείται αυτόματα (`GENERATED ALWAYS AS`) ως μια συνδυαστική αναπαράσταση των πεδίων `topic_id`, `area_id`, και

location_id. Η τιμή του πεδίου παράγεται μέσω της συνάρτησης CONCAT, η οποία ενώνει τις τιμές των σχετικών πεδίων με διαχωριστικό τον χαρακτήρα '-'. Εάν κάποιο από τα πεδία είναι NULL, αντικαθίσταται με την τιμή '0'. Το πεδίο αποθηκεύεται μόνιμα στη βάση δεδομένων (STORED).
Ο πίνακας περιέχει μοναδικό περιορισμό (UNIQUE) στο πεδίο unique_key, διασφαλίζοντας ότι κάθε συνδυασμός τοποθεσίας, γνωστικής περιοχής και θέματος είναι μοναδικός.
Ο πίνακας subscriptions παρέχει μια ευέλικτη και επεκτάσιμη δομή για τη διαχείριση συνδρομών που σχετίζονται με πολλαπλές οντότητες της πλατφόρμας.

4.3.11 Πίνακας logs - Καταγραφές Συμβάντων

Ο πίνακας logs χρησιμοποιείται για την αποθήκευση καταγραφών (logs) που αφορούν την λειτουργία της πλατφόρμας. Τα πεδία του πίνακα περιγράφονται ως εξής:
Το πεδίο id είναι τύπου ακέραιου (INT), αυξάνεται αυτόματα (AUTO_INCREMENT) και αποτελεί το κύριο κλειδί (PRIMARY KEY) του πίνακα.
Το πεδίο type είναι τύπου ENUM και δέχεται μία από τις ακόλουθες προκαθορισμένες τιμές: 'Debug', 'Info', 'Warning', 'Error'. Το πεδίο αυτό περιγράφει το επίπεδο της καταγραφής, δηλαδή αν πρόκειται για μήνυμα αποσφαλμάτωσης (Debug), πληροφορίας (Info), προειδοποίησης (Warning) ή σφάλματος (Error).
Το πεδίο source είναι τύπου VARCHAR(255) και αποθηκεύει την πηγή του συμβάντος, δηλαδή την οντότητα ή το υποσύστημα της πλατφόρμας που προκάλεσε την καταγραφή.
Το πεδίο source_method είναι τύπου VARCHAR(255) και αποθηκεύει το όνομα της μεθόδου ή της λειτουργίας που προκάλεσε την καταγραφή.
Το πεδίο message είναι τύπου TEXT και αποθηκεύει το μήνυμα της καταγραφής. Το μήνυμα αυτό μπορεί να περιέχει λεπτομέρειες σχετικά με το συμβάν.
Το πεδίο created_at είναι τύπου TIMESTAMP και αποθηκεύει την ημερομηνία και ώρα δημιουργίας της εγγραφής, με προεπιλεγμένη τιμή την τρέχουσα χρονική στιγμή (DEFAULT CURRENT_TIMESTAMP).
Ο πίνακας logs παρέχει μια ολοκληρωμένη δομή για την καταγραφή, την παρακολούθηση και την ανάλυση των συμβάντων που λαμβάνουν χώρα στην πλατφόρμα.

4.3.12 Πίνακας userSubscriptions_users - Συνδρομές Χρηστών

Ο πίνακας userSubscriptions_users χρησιμοποιείται για την αποθήκευση της σχέσης μεταξύ των χρηστών και των συνδρομών τους στην πλατφόρμα. Αποτελεί έναν πίνακα συσχέτισης (junction table) μεταξύ των πινάκων userSubscriptions και users, επιτρέποντας την αποθήκευση πολλαπλών σχέσεων συνδρομών για κάθε χρήστη. Η δομή του πίνακα περιλαμβάνει τα εξής πεδία:
Το πεδίο userSubscriptions_user_id είναι τύπου INT και αναφέρεται στο id του πίνακα userSubscriptions, το οποίο αντιπροσωπεύει μια συγκεκριμένη συνδρομή στην πλατφόρμα. Το πεδίο αυτό αποτελεί το πρώτο μέρος του σύνθετου πρωτεύοντος κλειδιού (PRIMARY KEY) του πίνακα.
Το πεδίο users_id είναι τύπου UUID[19] και αναφέρεται στο μοναδικό αναγνωριστικό (uuid[19]) του χρήστη από τον πίνακα users. Αυτό το πεδίο αποτελεί το δεύτερο μέρος του σύνθετου πρωτεύοντος κλειδιού (PRIMARY KEY).

Το πρωτεύον κλειδί του πίνακα είναι το σύνθετο κλειδί, το οποίο αποτελείται από τα πεδία `userSubscriptions_user_id` και `users_id`, εξασφαλίζοντας έτσι ότι κάθε ζεύγος συνδρομής-χρήστη είναι μοναδικό στον πίνακα.
Το πεδίο `userSubscriptions_user_id` είναι επίσης ξένο κλειδί που αναφέρεται στο `id` του πίνακα `userSubscriptions`. Αν διαγραφεί μια εγγραφή από τον πίνακα `userSubscriptions`, τότε όλες οι σχετικές εγγραφές στον πίνακα `userSubscriptions_users` διαγράφονται αυτόματα μέσω της εντολής `ON DELETE CASCADE`.
Το πεδίο `users_id` είναι ξένο κλειδί που αναφέρεται στο `uuid`[19] του πίνακα `users`. Αν διαγραφεί ένας χρήστης από τον πίνακα `users`, τότε όλες οι σχετικές εγγραφές στον πίνακα `userSubscriptions_users` διαγράφονται αυτόματα μέσω της εντολής `ON DELETE CASCADE`.
Ο πίνακας `userSubscriptions_users` παρέχει τη δυνατότητα αποθήκευσης πολλαπλών συνδρομών για κάθε χρήστη, επιτρέποντας τη διαχείριση των συνδρομητικών σχέσεων της πλατφόρμας με αποδοτικό και κανονικοποιημένο τρόπο.

4.3.13 Πίνακας `subscriptions_userSubscriptions` - Σχέσεις Συνδρομών και Συνδρομητικών Υπηρεσιών

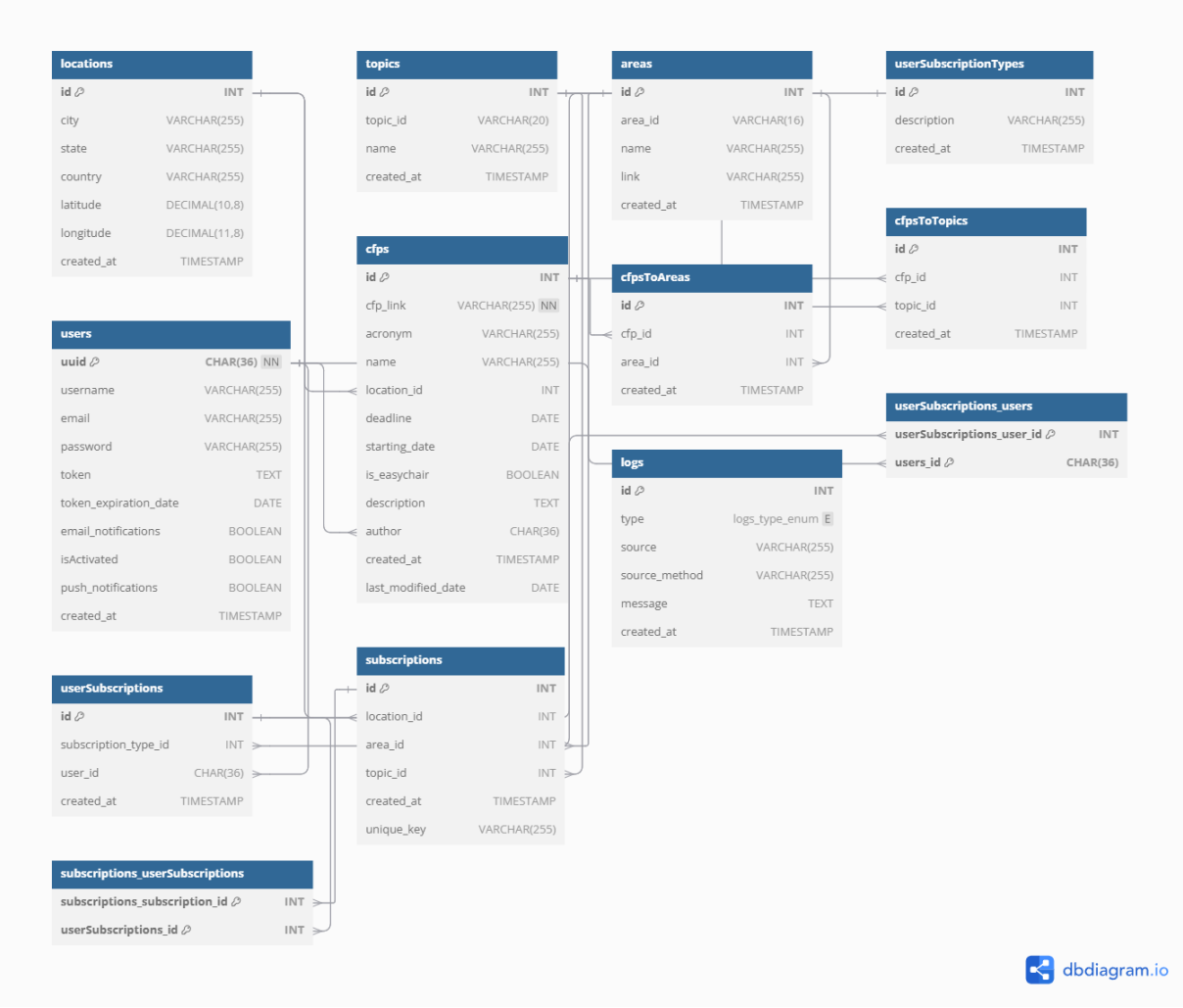
Ο πίνακας `subscriptions_userSubscriptions` χρησιμοποιείται για την αποθήκευση της σχέσης μεταξύ των συνδρομών και των συνδρομητικών υπηρεσιών που παρέχονται στους χρήστες στην πλατφόρμα. Αποτελεί έναν πίνακα συσχέτισης (junction table) μεταξύ των πινάκων `subscriptions` και `userSubscriptions`, επιτρέποντας την αποθήκευση πολλαπλών συνδρομών για κάθε συνδρομητική υπηρεσία και αντίστροφα. Η δομή του πίνακα περιλαμβάνει τα εξής πεδία:
Το πεδίο `subscriptions_subscription_id` είναι τύπου `INT` και αναφέρεται στο `id` του πίνακα `subscriptions`, το οποίο αντιπροσωπεύει μια συγκεκριμένη συνδρομή στην πλατφόρμα. Το πεδίο αυτό αποτελεί το πρώτο μέρος του σύνθετου πρωτεύοντος κλειδιού (`PRIMARY KEY`) του πίνακα.
Το πεδίο `userSubscriptions_id` είναι τύπου `INT` και αναφέρεται στο `id` του πίνακα `userSubscriptions`, το οποίο αντιπροσωπεύει μια συγκεκριμένη εγγραφή συνδρομής χρήστη. Αυτό το πεδίο αποτελεί το δεύτερο μέρος του σύνθετου πρωτεύοντος κλειδιού (`PRIMARY KEY`).
Το πρωτεύον κλειδί του πίνακα είναι το σύνθετο κλειδί, το οποίο αποτελείται από τα πεδία `subscriptions_subscription_id` και `userSubscriptions_id`, εξασφαλίζοντας ότι κάθε ζεύγος συνδρομής-χρήστη είναι μοναδικό στον πίνακα.
Το πεδίο `subscriptions_subscription_id` είναι ξένο κλειδί που αναφέρεται στο `id` του πίνακα `subscriptions`. Αν διαγραφεί μια εγγραφή από τον πίνακα `subscriptions`, τότε όλες οι σχετικές εγγραφές στον πίνακα `subscriptions_userSubscriptions` διαγράφονται αυτόματα μέσω της εντολής `ON DELETE CASCADE`.
Το πεδίο `userSubscriptions_id` είναι ξένο κλειδί που αναφέρεται στο `id` του πίνακα `userSubscriptions`. Αν διαγραφεί μια εγγραφή από τον πίνακα `userSubscriptions`, τότε όλες οι σχετικές εγγραφές στον πίνακα `subscriptions_userSubscriptions` διαγράφονται αυτόματα μέσω της εντολής `ON DELETE CASCADE`.
Ο πίνακας `subscriptions_userSubscriptions` παρέχει τη δυνατότητα διαχείρισης των σχέσεων μεταξύ συνδρομών και χρηστών, διασφαλίζοντας την αποτελεσματική και κανονικοποιημένη αποθήκευση των δεδομένων που αφορούν τις συνδρομές στην πλατφόρμα.

4.3.14 Trigger `before_insert_topics`

Τα triggers είναι μηχανισμοί στη MySQL που επιτρέπουν την εκτέλεση συγκεκριμένων ενεργειών αυτόματα όταν συμβαίνουν αλλαγές στους πίνακες, όπως η εισαγωγή, η ενημέρωση ή η διαγραφή εγγραφών. Στη συγκεκριμένη περίπτωση, δημιουργείται ένα trigger που αφορά τον πίνακα `topics`. Το trigger περιγράφεται ως εξής:
Το trigger `before_insert_topics` ενεργοποιείται πριν από την εισαγωγή (`BEFORE INSERT`) μιας νέας εγγραφής στον πίνακα `topics`. Αν το πεδίο `topic_id` της νέας εγγραφής είναι κενό ή `NULL`, το trigger ορίζει το `topic_id` σε μια νέα τιμή, η οποία παράγεται μέσω της συνάρτησης `UUID_SHORT()`, η οποία δημιουργεί έναν μοναδικό αριθμό αναγνωριστικού. Έτσι, εξασφαλίζεται ότι κάθε νέα εγγραφή στον πίνακα `topics` θα έχει πάντα ένα μοναδικό `topic_id`, ακόμα κι αν δεν παρέχεται εκ των προτέρων από τον χρήστη.
Το trigger αυτό εξασφαλίζει τη διατήρηση της συνέπειας και της μοναδικότητας των δεδομένων στον πίνακα `topics`, αυτοματοποιώντας διαδικασίες που διαφορετικά θα απαιτούσαν επιπλέον έλεγχο από τον χρήστη ή την εφαρμογή.

4.3.15 Διάγραμμα βάσης

Το διάγραμμα της βάσης ειναι το εξής:



4.4 Ανάκτηση Δεδομένων από το EasyChair μέσω Python

Η ανάκτηση δεδομένων από το `EasyChair`[1] πραγματοποιείται με τη χρήση της Python και των πακέτων `requests`[20] και `BeautifulSoup`[3]. Η συγκεκριμένη τεχνική εξαγωγής δεδομένων ονομάζεται *scraping* και χρησιμοποιείται ευρέως για την εξαγωγή και τη συλλογή δεδομένων από οποιαδήποτε ιστοσελίδα. Το script που έχουμε υλοποιήσει θα εκτελείται από ένα `bash` script μία φορά την εβδομάδα.

Η υλοποίηση του *scraper* περιλαμβάνει διάφορες τεχνικές που διασφαλίζουν ότι δεν υπερφορτώνεται η ιστοσελίδα από την οποία αντλούμε τα δεδομένα. Αφού αντληθούν τα δεδομένα, γίνεται η επεξεργασία τους, ώστε να διαμορφωθούν κατάλληλα, να διορθωθούν τυχόν λάθη και να εμπλουτιστούν με διάφορες πληροφορίες, οι οποίες είναι απαραίτητες για ορισμένες λειτουργίες της πλατφόρμας `CFPIndexer`.

Εφόσον έχουν ολοκληρωθεί όλες οι απαιτούμενες ενέργειες στα δεδομένα, αυτά εισάγονται στη βάση δεδομένων με τη χρήση του πακέτου (*package*) της `MariaDB`[7]. Στις επόμενες ενότητες θα αναλύσουμε λεπτομερώς τον κώδικα που υλοποιεί τον *scraper*.

4.4.1 Παρουσίαση του Κώδικα

Ο ακόλουθος κώδικας αναλαμβάνει την αυτοματοποιημένη λήψη δεδομένων (scraping) από την πλατφόρμα `EasyChair`[1] καθώς και την εισαγωγή τους σε βάση δεδομένων `MariaDB`[7] Εφαρμόζονται τεχνικές αποφυγής υπερφόρτωσης της απομακρυσμένης ιστοσελίδας ενώ παράλληλα καταγράφονται όλα τα βήματα και πιθανά σφάλματα μέσω ενός μηχανισμού καταγραφής *logging* Στις επόμενες ενότητες περιγράφονται οι σημαντικότερες λειτουργίες όπως υλοποιούνται μέσα στον κώδικα

Διαχείριση Καταγραφών (Logs)

Κλάση `Log`

Η κλάση `Log` καθορίζει την εσωτερική δομή των μηνυμάτων καταγραφής Κάθε μήνυμα έχει τύπο καταγραφής πηγή προέλευσης μέθοδο και περιγραφή

`logsToInsert` και Ρυθμίσεις `logs`

Τα μηνύματα καταγραφής συγκεντρώνονται στη λίστα `logsToInsert` Έπειτα εισάγονται στη βάση δεδομένων μέσω της `insert_logs` Τα επίπεδα `logs (log levels)` ρυθμίζουν αν εμφανίζονται σφάλματα ειδοποιήσεις ή πρόσθετες πληροφορίες

4.4.2 Βοηθητικές Κλάσεις και Δομές Δεδομένων

Κλάση `Location`

Χρησιμοποιείται για τη διαχείριση γεωγραφικών πληροφοριών Περιέχει πόλη πολιτεία χώρα γεωγραφικό μήκος και πλάτος Ορίζει `hash` και μέθοδο σύγκρισης ώστε να τοποθετείται σε σύνολα

Κλάση `Topic`

Αναπαριστά ένα θεματικό πεδίο με όνομα και ταυτότητα `topic_id` Παρέχει μεθόδους εισαγωγής και ενημέρωσης θεμάτων στη βάση δεδομένων

Κλάση `Area`

Αναφέρεται σε επιστημονικό πεδίο ή τομέα ενδιαφέροντος Περιλαμβάνει όνομα `area_id` και `link`

Κλάση `Cfp`

Περιλαμβάνει όλες τις πληροφορίες που αφορούν μια πρόσκληση υποβολής εργασιών Διαθέτει `cfp_link` όνομα και συντομογραφία παράλληλα με `location` ημερομηνίες και σχετικούς `topics` και `areas`

Data Transfer Objects

Οι `cfpsToAreasDTO` και `cfpsToTopicsDTO` αντιστοιχούν τα `Cfp` σε `Area` ή `Topic` αντίστοιχα

4.4.3 Διαδικασία Συλλογής Δεδομένων (Scraping)

`get_page_contents`

Επικοινωνεί με τη σελίδα μέσω `GET HTTP requests` Στα αιτήματα αυτά τροποποιούμε τις κεφαλίδες Ορίζουμε την παράμετρο `User-Agent` σε `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36` που χρησιμοποιείται ώστε να μην απορρίπτονται τα αιτήματα από τον παραλήπτη ως spam bot traffic Μόλις ληφθούν τα δεδομένα τα οποία αποτελούν μια ιστοσελίδα τα εισάγουμε σε αντικείμενο τύπου `BeautifulSoup`[3] και το επιστρέφουμε

`get_areas`

Ανακτά τα `areas` από πίνακα του `EasyChair`[1] και είναι το πρώτο `request`[20] που κάνει ο `scraper` για να πάρει όλα τα γνωστικά πεδία Τα δεδομένα που αντλούνται από αυτήν τη μέθοδο χρησιμοποιούνται στη συνέχεια για την άντληση των CFP Κάθε `area` προστίθεται στο σύνολο γνωστών `areas` εάν δεν υπάρχει ήδη μέσω `add_area`

`get_cfps_by_area`

Δέχεται σελίδα και αντικείμενο `Area`. Εντοπίζει και επιστρέφει όλα τα `Cfp` που συσχετίζονται με το συγκεκριμένο `area`. Συμπληρώνει πληροφορίες για `acronym`, `location`, `deadline`, `startDate` και `topics`. Έπειτα από τη λήψη των CFP από κάθε κατηγορία, η `main` μέθοδος τα καταχωρεί σε μια λίστα. Μόλις ολοκληρωθεί η διαδικασία άντλησης όλων των CFP από όλες τις κατηγορίες, τα δεδομένα αποθηκεύονται προσωρινά σε ένα αρχείο `cfp.json`[21].

`cleanup_and_merge_cfps`

Η συνάρτηση `cleanup_and_merge_cfps` ελέγχει αν υπάρχει το αρχείο `cleaned.json`[21] και αν είναι παλιότερο από μία ημέρα ή αν δεν υπάρχει καθόλου. Αν ισχύει κάποιο από τα παραπάνω, διαβάζει τα δεδομένα από το `cfp.json`[21] και τα αποθηκεύει σε μια μεταβλητή `data`. Διαφορετικά, τα δεδομένα λαμβάνονται από το `cleaned.json`[21]. Στη συνέχεια, δημιουργείται ένα κενό λεξικό `cfpHashMap` με κλειδί το `cfp_link` του κάθε συνεδρίου `conference`. Αν εντοπιστεί `cfp_link` που δεν έχει ήδη εισαχθεί στο `cfpHashMap`, προστίθεται άμεσα. Αν βρεθεί διπλότυπο `cfp_link` και το αντικείμενο περιέχει `areas`, συνδυάζονται οι `areas` του νεότερου αντικειμένου με αυτές του ήδη αποθηκευμένου. Με τον τρόπο αυτό δημιουργείται ένα ενιαίο και ενημερωμένο `cfpHashMap` που συγκεντρώνει πληροφορίες από διάφορες πηγές και αποφεύγονται διπλότυπα ή ελλιπή δεδομένα. Τέλος, η συνάρτηση επιστρέφει το τελικό `cfpHashMap` για περαιτέρω επεξεργασία.

4.4.4 Διαδικασίες Εισαγωγής στη Βάση

`connect_to_database`

Δημιουργεί σύνδεση με τη βάση `MariaDB`[7] και επιστρέφει `conn` και `cur`, τα οποία χρησιμοποιούνται σε κάθε σημείο του κώδικα που απαιτεί αλληλεπίδραση με τη βάση δεδομένων.

`insert_unique_areas`

Φορτώνει τις `areas` που υπάρχουν ήδη στη βάση. Εντοπίζει και εισάγει μόνο νέες `areas` που απουσιάζουν

`insert_unique_topics`

Συλλέγει όλα τα `topics` από τα `Cfp`. Εισάγει μόνο όσα δεν υπάρχουν ήδη στη βάση.

`insert_unique_locations`

Φορτώνει τις υπάρχουσες `locations`. Εντοπίζει νέες τοποθεσίες και τις εισάγει. Προσθέτει την ειδική `Online`, εφόσον δεν υπάρχει ήδη στη βάση, η οποία αργότερα θα χρησιμοποιείται για τα CFP που οργανώνονται διαδικτυακά.

`update_locations_cords`

Για όσες `locations` δεν έχουν συντεταγμένες, εκτελεί αναζήτηση στη βοηθητική βάση που περιλαμβάνει όλες τις πόλεις του πλανήτη και ενημερώνει `latitude` και `longitude`. Η βάση αυτή χρησιμοποιείται και για τον καθαρισμό δεδομένων που αφορούν τις τοποθεσίες.

`insert_cfps_and_junction_tables`

Δημιουργεί ή ενημερώνει εγγραφές `Cfp` στον πίνακα `cfps` Ενημερώνει τους πίνακες γέφυρας `cfpsToTopics` και `cfpsToAreas` για τη συσχέτιση πολλών προς πολλά

4.4.5 Πρόσθετες Λειτουργίες

insertLocationToSubscriptions,insertAreaToSubscriptions,insertTopicToSubscriptions

Εισάγουν στον πίνακα `subscriptions` τις νέες εγγραφές από τους πίνακες `locations`, `areas` και `topics`, με στόχο να είναι διαθέσιμες οι εγγραφές στον πίνακα `subscriptions`. Οι εγγραφές αυτές χρησιμοποιούνται στη συνέχεια για να μπορούν οι χρήστες να εγγράφονται σε αυτές, μέσω των `junction` πινάκων.

4.4.6 Κύρια Συνάρτηση `main`

Η `main` καθορίζει τη ροή εκτέλεσης του κώδικα και συγκεντρώνει όλα τα επιμέρους βήματα σε ένα ολοκληρωμένο σενάριο.

Αρχικά, καλείται η `connect_to_database` για να δημιουργηθεί σύνδεση (`conn`) και αντικείμενο `cursor` (`cur`) προς τη `MariaDB`[7]. Στη συνέχεια, γίνεται ανάκτηση της σελίδας `area.cgi` με την `get_page_contents` και εξαγωγή των `areas` μέσω `get_areas`. Τα `areas` εισάγονται στη βάση με τη `insert_unique_areas`.
Ακολουθεί λήψη `CFPs` ανά `area` με τη `get_cfps_by_area` και ενσωμάτωσή τους σε μια κεντρική λίστα `cfps`. Για να αποφευχθεί υπερφόρτωση της σελίδας, εφαρμόζεται `time.sleep` με τυχαίο διάστημα. Στο τέλος, δημιουργείται αρχείο `cfp.json`[21], όπου αποθηκεύονται οι πληροφορίες κάθε `Cfp`.
Η `cleanup_and_merge_cfps` συνδυάζει τα δεδομένα του `cfp.json`[21] με όσα περιέχει το `cleaned.json`[21], δημιουργεί ένα ενιαίο `cfpHashMap` και αποφεύγει διπλότυπα `CFPs`. Το `cleaned.json`[21] ενημερώνεται με το τελικό αποτέλεσμα.
Πραγματοποιείται έλεγχος για νέες `locations` με τη `insert_unique_locations`, και όπου χρειάζεται γίνεται ενημέρωση συντεταγμένων μέσω `update_locations_cords`. Αμέσως μετά, εντοπίζονται και εισάγονται τυχόν νέα `topics` με τη `insert_unique_topics`.
Με τη `get_cfps_to_objects_maps` δημιουργούνται λεξικά που συνδέουν `CFPs` με `locations`, `topics` και `areas`. Η `insert_cfps_and_junction_tables` ενημερώνει ή εισάγει εγγραφές στον πίνακα `cfps` και στους πίνακες `cfpsToTopics` και `cfpsToAreas`.
Δημιουργείται ή ανανεώνεται το `cleanedLocations.json`[21], για να αποθηκευτούν τα δεδομένα τοποθεσιών. Στη συνέχεια, καλούνται οι `insertTopicToSubscriptions`, `insertLocationToSubscriptions` και `insertAreaToSubscriptions` ώστε να εγγραφούν στον πίνακα `subscriptions` οι νέες καταχωρήσεις σε `topics`, `locations` και `areas`.
Τέλος, καλείται η `insert_logs` για την αποθήκευση μηνυμάτων καταγραφής, κλείνει η σύνδεση (`conn.close()`) με τη βάση δεδομένων και ολοκληρώνεται η εκτέλεση της `main`.

4.5 Υλοποίηση Backend

Η πλατφόρμα `Next.js`[4] επιτρέπει την ανάπτυξη τόσο του *backend* όσο και του *frontend* ενιαία, μέσα στο ίδιο περιβάλλον και στον ίδιο φάκελο εργασίας. Αυτή η προσέγγιση, γνωστή ως *monorepo*, συμβάλλει στην ευκολότερη διαχείριση της αρχιτεκτονικής του έργου, καθώς όλα τα μέρη του κώδικα (διαδρομές, λογική διακομιστή, στοιχεία παρουσίασης) βρίσκονται σε ένα ενιαίο σημείο.
Στο πλαίσιο της ανάπτυξης της πλατφόρμας `CFPIndexer`, χρησιμοποιήθηκε η `TypeScript`[6], μια επέκταση της `JavaScript`, η οποία προσθέτει μεταξύ άλλων ισχυρή τυποποίηση (στατικές δηλώσεις τύπων) και αντικειμενοστρεφή χαρακτηριστικά. Με αυτόν τον τρόπο, ενισχύεται η αξιοπιστία της εφαρμογής και περιορίζονται πιθανά σφάλματα, καθώς ο μεταγλωττιστής της `TypeScript`[6] εντοπίζει παραβάσεις τύπων και ασυμβατότητες πριν την εκτέλεση.
Στο *backend* μέρος της `Next.js`[4] εφαρμογής, μπορούν να υλοποιηθούν *API Routes* που εκτελούνται σε περιβάλλον διακομιστή, επιτρέποντας την επεξεργασία αιτημάτων `HTTP`, την επικοινωνία με βάσεις δεδομένων και την αλληλεπίδραση με εξωτερικές υπηρεσίες (π.χ. απομακρυσμένα *APIs*). Επιπρόσθετα, ο κώδικας του *backend* μπορεί να αξιοποιεί όλες τις βιβλιοθήκες και τα εργαλεία του οικοσυστήματος `Node.js`[8], ενώ ταυτόχρονα διατηρεί τη συνέπεια της τυποποίησης (*type safety*) που προσφέρει η `TypeScript`[6].
Με αυτό το ενιαίο (*monorepo*) μοντέλο, η μετάβαση από την ανάπτυξη του *frontend* στη δημιουργία των τελικών `API routes` γίνεται ομαλά, χωρίς την ανάγκη επιπρόσθετων μεταβάσεων μεταξύ

διαφορετικών αποθετηρίων ή υποδομών. Παράλληλα, η κεντρικοποιημένη διαχείριση εξαρτήσεων διευκολύνει τη συντήρηση του κώδικα και τη συνεχή ενσωμάτωση νέων λειτουργιών, ελαχιστοποιώντας τη διπλή εργασία και την πολυπλοκότητα.
Στο κεφάλαιο αυτό, εστιάζουμε στις λειτουργίες του *backend* για την πλατφόρμα `CFPIndexer`. Συγκεκριμένα, παρουσιάζουμε τη δομή των *API routes*, τις κεντρικές `middleware` τεχνικές που διασφαλίζουν την ομαλή επεξεργασία των αιτημάτων, καθώς και τις συνδέσεις με τη βάση δεδομένων. Επιπλέον, θα δούμε πώς ο τύπος δεδομένων (`TypeScript`[6]) μειώνει δραστικά την πιθανότητα σφαλμάτων και καθιστά την τελική λύση πιο ποιοτική και αξιόπιστη.

4.5.1 Αυθεντικοποίηση - Authentication

Η πλατφόρμα CFPIndexer παρέχει δυνατότητες στους χρήστες που εγγράφονται σε αυτήν. Η αυθεντικοποίηση είναι μέρος του backend της πλατφόρμας αλλά και του frontend. Σε αυτό το υποκεφάλαιο θα αναλυθεί ο τρόπος υλοποίησης της αυθεντικο-ποίησης στο backend. Για την υλοποίηση χρησιμοποιείται το πακέτο `NextAuth.js`[14], το οποίο είναι ένα πακέτο που παρέχει έτοιμες κλάσεις και μεθόδους που υλοποιούν τα βασικά χαρακτηριστικά αυθεντικοποίησης. Γίνεται η χρήση του για να παρέχεται εγγραφή, σύνδεση στην πλατφόρμα, καθώς και αυθεντικοποίηση μέσω session και token σε κάθε αίτημα προς τα άλλα services της πλατφόρμας. Οι τρόποι εγγραφής/σύνδεσης είναι δύο: με email/κωδικό και με λογαριασμό Google.

Κώδικας εγγραφής

Παρακάτω ακολουθεί ο κώδικας που υλοποιεί την εγγραφή στην πλατφόρμα CFPIndexer. Αφού ο χρήστης δώσει την ηλεκτρονική διεύθυνση ταχυδρομείου του (email) και τον κωδικό που επιθυμεί, ελέγχουμε αν υπάρχει ήδη η διεύθυνση. Αν όχι, καταχωρούμε τον χρήστη και στέλνουμε το email επιβεβαίωσης, το οποίο περιλαμβάνει έναν μοναδικό σύνδεσμο για την ενεργοποίηση του λογαριασμού.

```
1  export async function signup(email: string, password: string):
   ↪  Promise<boolean> {
2      const pool = mariadb.createPool({ host: process.env.DATABSE_URL, user:
       ↪  'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber: true });
3      let conn;
4      if (email && password) {
5          try {
6              conn = await pool.getConnection();
7              const rows = await conn.query("SELECT count(uuid) from
               ↪  cfps_scraper.users WHERE email = ? ", [email]);
8              if (rows[0]['count(uuid)'] == 0) {
9                  const signupResult = await conn.query('INSERT INTO
                   ↪  cfps_scraper.users(email,password) value (?,?)', [email,
                   ↪  password]);
10                 if (signupResult) {
11                     if (signupResult['affectedRows'] == 1) {
12                         if (process.env.JWT_SECRET) {
```

```
                            const rows = await conn.query<User[]>("SELECT
                            ↪  uuid,email,username from cfps_scraper.users
                            ↪  WHERE email = ? ", [email]);
                            const emailActivationSend = await
                            ↪  sendAccountVerifyEmail(rows[0]);
                            console.log(`Email sent:
                            ↪  ${emailActivationSend}`);
                        }
                        return true;
                    }
                }
                return false;
            }
            return false;
        } catch (error) {
            console.log(error);
            return false;
        } finally {
            pool.end();
        }
    }
    pool.end();
    return false;
}
```

Κώδικας σύνδεσης στην πλατφόρμα

Παρακάτω ακολουθεί ο κώδικας που υλοποιεί τη σύνδεση στην πλατφόρμα CFPIndexer με τους δύο τρόπους που παρέχει. Η λειτουργία αυτή υλοποιείται με τη χρήση του πακέτου `NextAuth.js`[14]. Για κάθε τρόπο δημιουργούμε έναν `Provider`, ο οποίος καθορίζει τη λογική που ακολουθούμε.

```
providers: [
        CredentialsProvider({
            credentials: {
                email: {},
                password: {},
            },
            async authorize(credentials) {
                if (credentials === null) throw new
                ↪  InvalidLoginError("Credentials werent filled!");
                const pool = mariadb.createPool({host:
                ↪  process.env.DATABSE_URL, user: 'it154474',
                ↪  connectionLimit: 5, port:3333, bigIntAsNumber: true});
```

```
10                  let conn;
11                  if(credentials?.email && credentials.password){
12                      try {
13                          conn = await pool.getConnection();
14                          const rows = await conn.query("SELECT uuid, username,
                             ↪  email, password, isActivated from
                             ↪  cfps_scraper.users WHERE email =
                             ↪  ?",[credentials?.email]);
15
16                          const match = await
                             ↪  bcrypt.compare(credentials.password as string,
                             ↪  rows[0].password);
17
18                          if(match) {
19                              const user: User = {
20                                  uuid: rows[0].uuid,
21                                  username: rows[0].username,
22                                  email: rows[0].email,
23                                  isActivated: rows[0].isActivated == 1 ? true
                                     ↪  : false
24                              };
25                              return user;
26                          }else {
27                              throw new InvalidLoginError("Email/password dont
                                 ↪  match or exist!");
28                          }
29                      } catch (error: unknown) {
30                          if (error instanceof InvalidLoginError) {
31                            throw new InvalidLoginError(error.code || "Unknown
                               ↪  error code");
32                          }
33
34                          if (typeof error === "object" && error !== null &&
                             ↪  "code" in error) {
35                            throw new InvalidLoginError((error as { code:
                               ↪  string }).code);
36                          }
37                          throw new InvalidLoginError("There was an error
                             ↪  during login!");
38                      } finally{
39                          pool.end();
40                      }
41                  }
42                  throw new InvalidLoginError("There was an error during
                   ↪  login!");
```

```
            },
        }),
        CredentialsProvider({
          id: 'update-user',
          credentials: {},
          authorize(credentials) {
            return JSON.parse((credentials as { user: string }).user);
          },
        }),
        Google({
            clientId: process.env.GOOGLE_CLIENT_ID,
            clientSecret: process.env.GOOGLE_CLIENT_SECRET,
            authorization: {
                params: {
                    prompt: "consent",
                    access_type: "offline",
                    response_type: "code",
                },
            },

        })
        ]
```

Δημιουργία API κλειδιού

Παρακάτω ακολουθεί ο κώδικας που υλοποιεί τη δημιουργία του `API` κλειδιού για τη χρήση των `API endpoints` της πλατφόρμας CFPIndexer. Αρχικά, ελέγχεται αν υπάρχει το πεδίο `JWT_SECRET` μέσα στο αρχείο `.env`. Έπειτα, ελέγχεται αν υπάρχει χρήστης με το ίδιο `uuid`[19] στη βάση. Αν ναι, το κρυπτογραφεί και το καταχωρεί μαζί με την ημερομηνία λήξης του στην εγγραφή του εκάστοτε χρήστη στη βάση.

```
export async function generateBearerToken(user: User): Promise<string | null>
↪  {
    if (process.env.JWT_SECRET) {
        const pool = mariadb.createPool({ host: process.env.DATABSE_URL,
        ↪  user: 'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber:
        ↪  true });
        try {
            const conn = await pool.getConnection();
            const isActivated = await conn.query(`SELECT isActivated FROM
            ↪  cfps_scraper.users WHERE uuid = ?;`, [user.uuid]);
            if (isActivated[0].isActivated == 0) {
                return 'NotActiveError'
```

```
            } else if (isActivated[0].isActivated == 1) {
                const token = jwt.sign(
                    { uuid: user.uuid },            // Payload
                    process.env.JWT_SECRET,       // Secret key (use an
                    ↪  environment variable)
                    { expiresIn: '365d' }        // Set token expiration
                );
                const today = new Date(); // Get today's date
                const nextYear = new Date(today); // Create a copy of the
                ↪  current date
                nextYear.setFullYear(today.getFullYear() + 1); // Add one
                ↪  year to the current year
                const query = `
                                UPDATE cfps_scraper.users
                                SET token = ?,
                                    token_expiration_date = ?
                                WHERE uuid = ?
                                `;

                const params = [token, nextYear.toISOString().split('T')[0],
                ↪  user.uuid];
                const rows = await conn.query(query, params);

                if (rows['affectedRows'] == 1) {
                    return token;
                }
            } else {
                return null;
            }
        } catch (e) {
            console.log(e);
            return null;
        } finally {
            pool.end();
        }
    }
    return null;
}
```

4.5.2 Ενεργειες CFP

Στην πλατφορμα CFPIndexer γινονται διαφορες ενεργεις στο πινακα cfps. Αυτες θα αναλυσουμε στο υποκεφαλαιο αυτο. Οι ενεργιες ειναι η προβολη ολων των εγγραφων φιλτραρισμενες η μη,

η εισαγωγη εγγραφων στο πινακα οπως και η προβολη μεμονομενων εγγραφων.

Προβολή εγγραφών πίνακα με φίλτρα ή χωρίς

Στην αρχική σελίδα της πλατφόρμας εμφανίζεται ένας πίνακας με τις εγγραφές του πίνακα `cfps`. Ο πίνακας αυτός δεν έχει κάποιο όριο ως προς τον αριθμό των `CFP`, αλλά εξ αρχής εφαρμόζει κάποια φίλτρα ώστε να επιστρέψει τις εγγραφές των οποίων η προθεσμία (`deadline`) είναι μέσα στον επόμενο μήνα. Ο χρήστης μπορεί να αλλάξει ή να αφαιρέσει τα φίλτρα, με αποτέλεσμα η πλατφόρμα να κάνει ένα νέο αίτημα προς τον διακομιστή (`backend`) για να επιστραφούν οι νέες εγγραφές.

```
export async function getCfps(input: string | null): Promise<CFP[] | null> {
    try {
        let data: { id: string, value: string[] }[] | null = null;
        if (input) {
            data = JSON.parse(input);
        }
        const conditions: string[] = [];
        const params: string[] = [];
        if (data) {
            data.forEach(filter => {
                if (filter.id == 'startingDate') {
                    if (filter.value[0]) {
                        conditions.push(`c.starting_date >= ?`);
                        params.push(`${filter.value[0]}`);
                    }
                    if (filter.value[1]) {
                        conditions.push(`c.starting_date <= ?`);
                        params.push(`${filter.value[1]}`);
                    }
                }
                if (filter.id == 'deadline') {
                    if (filter.value[0]) {
                        conditions.push(`c.deadline >= ?`);
                        params.push(`${filter.value[0]}`);
                    }
                    if (filter.value[1]) {
                        conditions.push(`c.deadline <= ?`);
                        params.push(`${filter.value[1]}`);
                    }
                }
                % MORE CODE HERE
                if (filter.id == 'acronym') {
                    if (filter.value) {
                        conditions.push(`c.acronym LIKE ?`);
```

```
                    params.push(`%${filter.value}%`);
                }
            }
        });

    }
    const pool = mariadb.createPool({ host: process.env.DATABSE_URL,
    ↪ user: 'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber:
    ↪ true });
    const conn = await pool.getConnection();

    let query = `
        SELECT
            c.cfp_link AS cfpLink,
            c.acronym AS acronym,
            c.name AS cfpName,
            c.deadline AS deadline,
            c.starting_date AS startingDate,
            l.city AS city,
            l.state AS state,
            l.country AS country,
            l.latitude as latitude,
            l.longitude as longitude,
            GROUP_CONCAT(DISTINCT t.name ORDER BY t.name ASC SEPARATOR
            ↪ '%') AS topics,
            GROUP_CONCAT(DISTINCT a.name ORDER BY a.name ASC) AS areas
        FROM cfps_scraper.cfps c
        LEFT JOIN cfps_scraper.locations l ON c.location_id = l.id
        LEFT JOIN cfps_scraper.cfpsToTopics cft ON c.id = cft.cfp_id
        LEFT JOIN cfps_scraper.topics t ON cft.topic_id = t.id
        LEFT JOIN cfps_scraper.cfpsToAreas cfa ON c.id = cfa.cfp_id
        LEFT JOIN cfps_scraper.areas a ON cfa.area_id = a.id
    `;

    if (conditions.length > 0) {
        query += ` WHERE ${conditions.join(' AND ')}`;
    } else {
        query += `WHERE c.deadline >= CURDATE() AND c.deadline <=
        ↪ DATE(NOW() + INTERVAL 30 DAY)`;
    }

    // Add GROUP BY
    query += `
        GROUP BY c.id, c.cfp_link, c.acronym, c.name, c.deadline,
        ↪ c.starting_date, l.city, l.state, l.country
```

```
        `;
        query += ';';

        // console.log(query)
        // console.log(params)
        const rows = await conn.query<CFP[]>(query, params);

        pool.end();
        return rows;
    } catch (e) {
        console.log(e);
        return null;
    }
}
```

Εισαγωγή CFP στον πίνακα

Οι εγγεγραμμένοι χρήστες της πλατφόρμας έχουν τη δυνατότητα να δημιουργήσουν το δικό τους `CFP`. Εφόσον συνδεθούν, υπάρχει ο σχετικός σύνδεσμος ο οποίος τους μεταφέρει στη σελίδα δημιουργίας του `CFP`. Εφόσον συμπληρώσουν όλα τα απαραίτητα πεδία και πατήσουν το κουμπί δημιουργίας, το `frontend` κάνει ένα αίτημα προς το `backend` με αυτά.
Τα στοιχεία αυτά ελέγχονται και, αν υπάρχουν νέες εγγραφές για τα πεδία τοποθεσίας ή θεμάτων, αποθηκεύονται πρώτα όλες οι νέες υποεγγραφές. Στη συνέχεια, γίνεται η αποθήκευση του νέου `CFP`.
Κατά αυτόν τον τρόπο, μπορεί να δημιουργηθεί πιο εύκολα από τον χρήστη ένα `CFP` χωρίς να χρειαστεί να κάνει επιπλέον ενέργειες, όπως η εισαγωγή ενός θέματος από κάποιο άλλο σημείο της πλατφόρμας.

```
export async function insertCfp(data: ICfpData): Promise<boolean> {
    try {
        const pool = mariadb.createPool({ host: process.env.DATABSE_URL,
        ↪  user: 'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber:
        ↪  true, insertIdAsNumber: true });
        const conn = await pool.getConnection();
        const toInsert: string[] = [];
        let location_id;
        const topicsInsertedId = [];
        const cfpObject = new CreateCFPDTO(data);
        console.log(cfpObject);
        if (cfpObject.topics && cfpObject.topics.length > 0) {
            cfpObject.topics.forEach(item => {
                const isNum = item.match(/^\d+$/) !== null;
                if (!isNum) {
```

```
                    toInsert.push(...item.split(';'));
                    cfpObject.topics.pop();
                }

            });
            console.log(toInsert)
            if (toInsert.length > 0) {
                for (const topic of toInsert) {
                    const topics = await conn.query('INSERT INTO
                    ↪  cfps_scraper.topics (name) VALUES (?);', [topic]);
                    if (topics['insertId']) {
                        topicsInsertedId.push(topics['insertId']);
                        try {
                            await conn.query('INSERT INTO
                            ↪  cfps_scraper.subscriptions(location_id,
                            ↪  area_id, topic_id) VALUES (?, ?, ?)', [null,
                            ↪  null, topics['insertId']]);
                        } catch (e) {
                            console.log("Failed to insert");
                            console.log(e);
                        }
                    }
                }
            }
        }
        const location = await getLocationId(
            cfpObject.country,
            cfpObject.state,
            cfpObject.city,
            conn
        );
        if (location.length == 1) {
            location_id = location[0].id;
        } else {
            const locationInsert = await conn.query('INSERT INTO
            ↪  cfps_scraper.locations (city,state,country) VALUES (?,?,?);',
            ↪  [cfpObject.city, cfpObject.state, cfpObject.country]);
            if (locationInsert['insertId']) {
                location_id = locationInsert['insertId'];
                try {
                    await conn.query('INSERT INTO
                    ↪  cfps_scraper.subscriptions(location_id, area_id,
                    ↪  topic_id) VALUES (?, ?, ?)', [location_id, null,
                    ↪  null]);
```

```
                } catch (e) {
                    console.log("Failed to insert");
                    console.log(e);
                }
            }
        }
        const cfpInsert = await conn.query('INSERT INTO
        ↪  cfps_scraper.cfps(cfp_link,
        acronym,
        name,
        location_id,
        deadline,
        starting_date,
        description,
        author)
            VALUES (?,?,?,?,?,?,?,?);',
        [ cfpObject.cfpLink,
            cfpObject.acronym,
            cfpObject.cfpName,
            location_id,
            new Date(cfpObject.deadline),
            new Date(cfpObject.startingDate),
            cfpObject.description,
            cfpObject.author]);
        % MORE CODE HERE
        pool.end();
    } catch (e) {
        console.log(e);
        return false;
    }
    return true;
}
```

Προβολή συγκεκριμένης εγγραφής CFP

Όταν ο χρήστης επισκεφθεί τον σύνδεσμο του `CFP` ή εφόσον δημιουργήσει ένα και η πλατφόρμα τον ανακατευθύνει σε αυτή τη σελίδα, ο κώδικας ο οποίος επιστρέφει το σωστό `CFP` είναι ο παρακάτω. Γίνεται αναζήτηση στον πίνακα με το πεδίο `cfp_link` και, εάν βρεθεί, επιστρέφεται το ανάλογο αντικείμενο στο `frontend`.

```
export async function getCfp(link: string): Promise<CFP> {
    const pool = mariadb.createPool({ host: process.env.DATABSE_URL, user:
    ↪  'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber: true });
```

```
    const conn = await pool.getConnection();
    try {
        const rows = await conn.query<CFP[]>(`
            SELECT
                c.cfp_link as cfpLink,
                c.acronym as acronym,
                c.name AS cfpName,
                c.deadline as deadline,
                c.starting_date as startingDate,
                l.city as city,
                l.state as state,
                l.country as country,
                GROUP_CONCAT(DISTINCT t.name ORDER BY t.name ASC SEPARATOR
                ↪  '%') AS topics,
                GROUP_CONCAT(DISTINCT a.name ORDER BY a.name ASC) AS areas,
                c.is_easychair,
                c.description,
                u.username as author
            FROM cfps_scraper.cfps c
            LEFT JOIN cfps_scraper.locations l ON c.location_id = l.id
            LEFT JOIN cfps_scraper.cfpsToTopics cft ON c.id = cft.cfp_id
            LEFT JOIN cfps_scraper.topics t ON cft.topic_id = t.id
            LEFT JOIN cfps_scraper.cfpsToAreas cfa ON c.id = cfa.cfp_id
            LEFT JOIN cfps_scraper.areas a ON cfa.area_id = a.id
            LEFT JOIN cfps_scraper.users u ON c.author = u.uuid
            WHERE
            c.cfp_link = ?
            GROUP BY c.id, c.cfp_link, c.acronym, c.name, c.deadline,
            ↪  c.starting_date, l.city, l.state, l.country`, [link]);

        return rows[0];
    } catch (err) {
        throw err;
    } finally {
        conn.release();
        pool.end();
    }
}
```

API

Η πλατφόρμα παρέχει στους εγγεγραμμένους χρήστες τη δυνατότητα χρήσης του `API` της. Το `API` παρέχει τα `CFP` υπό τη μορφή `JSON`[21] και μπορεί να εφαρμόσει τα ίδια φίλτρα όπως και παραπάνω.

Η μόνη διαφορά είναι πως προαπαιτεί την παροχή του `API key` υπό τη μορφή `Bearer Token`, το οποίο, εφόσον αυθεντικοποιηθεί, επιτρέπεται η συνέχιση εξυπηρέτησης του αιτήματος.

```
export async function isTokenValid(token: string): Promise<boolean> {
    if (process.env.JWT_SECRET) {
        try {
            const decoded = jwt.verify(token, process.env.JWT_SECRET);
            if (typeof decoded === 'object' && 'uuid' in decoded) {
                const pool = mariadb.createPool({ host:
                ↪  process.env.DATABSE_URL, user: 'it154474',
                ↪  connectionLimit: 5, port: 3333, bigIntAsNumber: true });
                const conn = await pool.getConnection();
                const rows = await conn.query("SELECT token,
                ↪  token_expiration_date FROM cfps_scraper.users WHERE uuid
                ↪  = ? AND token = ? AND token_expiration_date >
                ↪  CURDATE();", [decoded.uuid, token]);
                if (rows.length) {
                    pool.end();
                    return true;
                }
            }
            return false;
        } catch (e) {
            console.log(e);
            return false;
        }
    }
    return false;
}
```

4.5.3 Βοηθητικές μέθοδοι

Το `CFPIndexer` απαιτεί την ύπαρξη διαφόρων άλλων μεθόδων, οι οποίες υλοποιούν διαφορετικές απαιτήσεις είτε για κάποιο αίτημα από το `frontend`, είτε για κάποιον άλλο σκοπό, όπως η μαζική αποστολή email.

Αποστολή email

Η πλατφόρμα στέλνει διάφορα email για να ενημερώνει τους χρήστες είτε για τα νέα `CFP`, είτε για λόγους διαχείρισης λογαριασμών. Η αποστολή των ειδοποιήσεων γίνεται μέσω της κλήσης μιας μεθόδου καθημερινά, η οποία βρίσκει τα νέα `CFP` ή τα πρόσφατα τροποποιημένα. Η μέθοδος παρέχει τη δυνατότητα να τεθεί δυναμικά το πόσες μέρες πριν έγιναν οι αλλαγές ώστε να συμπεριληφθούν στα αποτελέσματα.

```
export async function sendSubscriptionEmails(days: number): Promise<boolean> {
    try {
        const pool = mariadb.createPool({ host: process.env.DATABSE_URL, user: 'it154474', connectionLimit: 5, port: 3333, bigIntAsNumber: true, insertIdAsNumber: true });
        const conn = await pool.getConnection();
        const emailHashMap: Record<string, ShortCfp[]> = {};
        const initialEmailList = await conn.query<UserSubscriptionWithCfp[]>(
            `SELECT
                u.uuid AS user_id,
                u.username,
                u.email,
                us.id AS subscription_id,
                s.topic_id,
                s.area_id,
                s.location_id,
                c.id AS cfp_id,
                c.cfp_link,
                c.name AS cfp_name,
                c.acronym AS cfp_acronym,
                c.created_at AS cfp_created_at
            FROM
                cfps_scraper.users u
            JOIN
                cfps_scraper.userSubscriptions us ON u.uuid = us.user_id
            JOIN
                cfps_scraper.subscriptions_userSubscriptions sus ON sus.userSubscriptions_id = us.id
            JOIN
                cfps_scraper.subscriptions s ON s.id = sus.subscriptions_subscription_id
            LEFT JOIN
                cfps_scraper.cfpsToTopics ct ON s.topic_id = ct.topic_id
            LEFT JOIN
                cfps_scraper.cfpsToAreas ca ON s.area_id = ca.area_id
            LEFT JOIN
                cfps_scraper.cfps c ON (ct.cfp_id = c.id OR ca.cfp_id = c.id OR s.location_id = c.location_id)
            WHERE
               DATE(c.created_at) = DATE(NOW() - INTERVAL ${days} DAY) OR
               DATE(c.last_modified_date) = DATE(NOW() - INTERVAL ${days} DAY);`);

```

```
38          initialEmailList.forEach(record => {
39              if (emailHashMap[record.email]) {
40                  const array: ShortCfp[] = emailHashMap[record.email];
41                  const exists = array.find(x => x.id == record.cfp_id);
42                  if (exists == undefined) {
43                      array.push({
44                          id: record.cfp_id,
45                          name: record.cfp_acronym,
46                          cpf_link: record.cfp_link,
47                          acronym: record.cfp_acronym
48                      })
49                      emailHashMap[record.email] = array;
50                  }
51              } else {
52                  emailHashMap[record.email] = [{
53                      id: record.cfp_id,
54                      name: record.cfp_acronym,
55                      cpf_link: record.cfp_link,
56                      acronym: record.cfp_acronym
57                  }];
58              }
59          })
60          pool.end();
61          const sendEmailResult = await
            ↪  sendSubscriptionEmailsUtil(emailHashMap);
62          if (sendEmailResult) {
63              return true;
64          } else {
65              return false;
66          }
67      } catch (e) {
68          console.log(e)
69          return false;
70      }
71  }
```

Αποστολή email για επαναφορά κωδικού

Η μέθοδος αυτή βοηθάει τον χρήστη της πλατφόρμας να επαναφέρει τους κωδικούς του, ώστε να μπορεί ξανά να συνδεθεί στην πλατφόρμα. Η ενέργεια αυτή καλείται από το `frontend`, όπου ο χρήστης συμπληρώνει τη διεύθυνση email του. Αν υπάρχει ο χρήστης, του αποστέλλεται ένα email για να εισαγάγει νέους κωδικούς.
Το email αυτό περιέχει έναν σύνδεσμο που ισχύει για 5 λεπτά.

```
export async function forgotPassword(email: string): Promise<boolean> {
    if (email) {
    if (process.env.JWT_SECRET) {
        try {
            const pool = mariadb.createPool({ host: process.env.DATABSE_URL,
            ↪  user: 'it154474', connectionLimit: 5, port: 3333,
            ↪  bigIntAsNumber: true });
            const conn = await pool.getConnection();
            const queriedUser = await conn.query<User[]>("SELECT
            ↪  uuid,email,username,isActivated FROM cfps_scraper.users WHERE
            ↪  email = ?;", [email]);
            if (queriedUser[0]) {
                const token = jwt.sign(
                    {
                        uuid: queriedUser[0].uuid,
                        use: 'forgotPassword'
                    },// Payload
                    process.env.JWT_SECRET,       // Secret key (use an
                    ↪  environment variable)
                    { expiresIn: '5m' }        // Set token expiration
                );
                const mail: Email = {
                    from: {
                        name: "CFPIndexer",
                        address: "tsatsotourkoskappa@gmail.com"
                    },
                    subject: 'Password reset',
                    text: `
                            Hello,\n
                            There was a request to change your password!\n
                            If you did not make this request then please
                            ↪  ignore this email.\n
                            Otherwise, please click this link to change your
                            ↪  password:
                            ↪  ${'http://localhost:3000/resetPassword/' +
                            ↪  token}`,
                    html: `
                    <h2>Hello,</h2>
                    <p>There was a request to change your password!</p>

                    <p>If you did not make this request then please ignore
                    ↪  this email.</p>

                    <p>Otherwise, please click this
```

```
                    <a href="http://localhost:3000/resetPassword/${token}">
                        link
                    </a> to change your password</p>`
                };

                const user: User = {
                    uuid: '',
                    email,
                    username: '',
                    isActivated: false
                }
                const result = await senEmail(user, mail);
                pool.end();
                return result;
            } else {
                pool.end();
                return false;
            }
        } catch (e) {
            console.log(e);
            return false;
        }
    } else {
        return false
    }
    } else {
        return false;
    }
}
```

4.5.4 Προγραμματισμένες ενέργειες

Η πλατφόρμα χρησιμοποιεί προγραμματισμένες ενέργειες (`cron jobs`[22]), οι οποίες αντλούν δεδομένα από την πλατφόρμα του `EasyChair`[1] μία φορά την εβδομάδα, καθώς και μια διεργασία που αποστέλλει email στους χρήστες.
Η λειτουργία αυτή υλοποιείται με τη χρήση `bash`, καθώς ο διακομιστής χρησιμοποιεί λειτουργικό σύστημα βασισμένο σε `Linux`.
Ο κώδικας για την ενέργεια που αντλεί δεδομένα είναι ο εξής:

```
0 3 * * 0 /usr/bin/python3 /path/to/your/script/main.py >> /path/
   to/log/output.log 2>&1
```

Ο παραπάνω κώδικας εκτελεί το `Python` script που αντλεί τα δεδομένα και αποθηκεύει την έξοδό του στο αρχείο `output.log`.

Ανάλογο `cron job`[22] είναι και η διεργασία που καλεί τη μέθοδο για την αποστολή των `subscription emails`:

```
0 4 * * * /usr/bin/node /path/to/your/script/sendEmail.js >> /
   path/to/log/sendEmail.log 2>&1
```

Ο παραπάνω κώδικας εκτελεί το `Node.js`[8] script που αποστέλλει τα email και καταγράφει την έξοδο στο αρχείο `sendEmail.log`.

4.6 Υλοποίηση Frontend

Σε αυτό το υποκεφάλαιο θα παρουσιαστούν τμήματα κώδικα, τα οποία έχουν αναπτυχθεί με τη χρήση της `TypeScript`[6] και της `React.js`[11]. Η `React.js` είναι ένα framework που διευκολύνει την ανάπτυξη εφαρμογών ιστού, οι οποίες δεν χρειάζεται να επαναφορτώνουν ολόκληρη τη σελίδα όταν υπάρχει κάποια αλλαγή, καθώς τροποποιεί τα στοιχεία του DOM. Με αυτόν τον τρόπο, οι εφαρμογές γίνονται πιο ευχάριστες στη χρήση και δίνουν την εντύπωση ότι είναι ταχύτερες.
Επιπλέον, η `React.js`[11] υποστηρίζει την επεκτασιμότητά της μέσω διαφόρων πακέτων και βιβλιοθηκών, τα οποία επιλύουν σύνθετα προβλήματα και προσφέρουν απλές λύσεις στους προγραμματιστές που τις αξιοποιούν για την ανάπτυξη των εφαρμογών τους.

Κώδικας Εγγραφής

Σε αυτό το τμήμα κώδικα, που παρουσιάζεται παρακάτω, υλοποιείται η εγγραφή του χρήστη στο σύστημα με τη χρήση email και κωδικού πρόσβασης. Πιο συγκεκριμένα, το τμήμα αυτό αποτελεί τη μέθοδο που εκτελεί την κλήση της μεθόδου εγγραφής, δηλαδή ένα αίτημα προς το `backend` της πλατφόρμας.
Όπως φαίνεται και στην εικόνα, ο διαχωρισμός των δύο μερών της εφαρμογής (`frontend` και `backend`) δεν είναι απόλυτα διακριτός λόγω της χρήσης των `server actions` της `Next.js`[4]. Επιπρόσθετα, το τμήμα κώδικα που παρουσιάζεται περιλαμβάνει τη χρήση του πακέτου `bcryptjs`[23], το οποίο, με την κλήση της μεθόδου `hashSync()`, κρυπτογραφεί τον κωδικό πρόσβασης του χρήστη πριν την καταχώρισή του στη βάση δεδομένων.

```
1  import bcrypt from "bcryptjs-react";
2  import { signup } from '@app/actions';
3  import { useRouter } from 'next/navigation';
4  import 'react-toastify/dist/ReactToastify.css';
5  import { Bounce, ToastContainer, toast } from 'react-toastify';
6
7  /* MORE CODE HERE */
8
9  const handleSignup = async () => {
10   const hash = bcrypt.hashSync(password, 10);
11   signup(email, hash).then(res => {
12     console.log(res);
```

```
13      if (res) {
14        router.push('/login')
15      } else {
16        toast.error('Failed to signup.', {
17          position: "top-right",
18          autoClose: 5000,
19          hideProgressBar: false,
20          closeOnClick: true,
21          pauseOnHover: true,
22          draggable: true,
23          progress: undefined,
24          theme: selectedTheme,
25          transition: Bounce,
26        });
27      }
28    }).catch(error => {
29      console.log(error);
30      toast.error('Error signing up', {
31        position: "top-right",
32        autoClose: 5000,
33        hideProgressBar: false,
34        closeOnClick: true,
35        pauseOnHover: true,
36        draggable: true,
37        progress: undefined,
38        theme: selectedTheme,
39        transition: Bounce,
40      });
41    });
42  };
43
```

Κώδικας Σύνδεσης

Σε αυτό το τμήμα κώδικα, που παρουσιάζεται παρακάτω, υλοποιείται η σύνδεση του χρήστη στο σύστημα με τη χρήση `Google account`. Πιο συγκεκριμένα, το τμήμα αυτό αποτελεί τη μέθοδο που εκτελεί την κλήση της μεθόδου `signIn`, η οποία είναι μια μέθοδος από το πακέτο της `NextAuth.js`[14] και υλοποιεί τη σύνδεση/εγγραφή στην πλατφόρμα με τη χρήση του `Google` λογαριασμού.
Μόλις ο χρήστης πατήσει το κουμπί για σύνδεση, γίνεται η κλήση της `handleGoogleLogin`, η οποία καλεί την προαναφερθείσα μέθοδο της `NextAuth.js`[14]. Εφόσον η σύνδεση πραγματοποιηθεί με επιτυχία, ο χρήστης ανακατευθύνεται στην αρχική σελίδα.

```
"use client"

import { signIn } from "next-auth/react";

export default function Login() {
  const handleGoogleLogin = async () => {
    await signIn("google", { redirectTo: '/' });
  };

  return (
    <button onClick={handleGoogleLogin} className="flex items-center bg-white
    ↪  dark:bg-gray-800 border border-gray-300 rounded-lg shadow-md px-6
    ↪  py-2 text-sm font-medium text-gray-800 dark:text-white
    ↪  hover:bg-gray-200 focus:outline-none focus:ring-2 focus:ring-offset-2
    ↪  focus:ring-gray-500">
      <svg className="h-6 w-6 mr-2" xmlns="http://www.w3.org/2000/svg"
      ↪  width="800px" height="800px" viewBox="0 0 48 48" version="1.1">
        <title>Google-color</title>
        <desc>Created with Sketch.</desc>
        <g id="Icons" stroke="none" strokeWidth="1" fill="none"
        ↪  fillRule="evenodd">
          <g id="Color-" transform="translate(-401.000000, -860.000000)">
            <g id="Google" transform="translate(401.000000, 860.000000)">
              /* SVG PATHS */
            </g>
          </g>
        </g>
      </svg>
      <span>Continue with Google</span>
    </button>
  );
}

```

Κώδικας Προβολής CFP υπό Μορφή Πίνακα

Σε αυτό το τμήμα κώδικα, που παρουσιάζεται παρακάτω, υλοποιείται η προβολή των `CFP` υπό τη μορφή πίνακα. Ο πίνακας υλοποιείται με τη χρήση ενός πακέτου που ονομάζεται `material-react-table`[13], το οποίο παρέχει πλήρως παραμετροποιήσιμους πίνακες με χαρακτηριστικά όπως η σελιδοποίηση, η αναζήτηση ανά στήλη ή γενική αναζήτηση.
Στο `component`, στο οποίο γίνεται χρήση αυτού του πακέτου, όπως φαίνεται στο παρακάτω τμήμα κώδικα, η αναζήτηση πραγματοποιείται παρακολουθώντας αλλαγές στη μεταβλητή `columnFilters`, που περιέχει τα φίλτρα των στηλών. Μόλις υπάρξει αλλαγή, καλείται η

αντίστοιχη μέθοδος του `backend`. Όταν επιστραφούν τα δεδομένα, αυτά αποθηκεύονται στη μεταβλητή `data`.
Επιπροσθέτως, στο τμήμα κώδικα φαίνονται τα φίλτρα στη στήλη της τοποθεσίας, τα οποία αποτελούν ουσιαστικά μια λίστα από την οποία ο χρήστης μπορεί να επιλέξει τιμές.

```
import { useMemo, useState, useEffect } from 'react';
import { MaterialReactTable, MRT_ColumnDef, MRT_ColumnFiltersState,
 ↪  useMaterialReactTable } from 'material-react-table';
import { getCfps, getCountries } from '../../../actions';
import { CFP } from '../../../types/dbTypes';
import { LocalizationProvider } from
 ↪  '@mui/x-date-pickers/LocalizationProvider';

export default function Cfps() {

  const [data, setData] = useState<CFP[]>([]);
  const [columnFilters, setColumnFilters] =
   ↪  useState<MRT_ColumnFiltersState>([]);

  useEffect(() => {
    async function fetchFilteredData() {
      const result = await getCfps(JSON.stringify(columnFilters));
      if (result) {
        setData(result);
      }
    }
    fetchFilteredData();
  }, [columnFilters]);

  const columns = useMemo<MRT_ColumnDef<CFP>[]>(() => [
    {
      accessorFn: (row) => `${row.city ? row.city + ", " : ""} ${row.state ?
       ↪  row.state + ", " : ""} ${row.country}`,
      id: 'location', //simple recommended way to define a column
      header: 'Location',
      filterVariant: 'multi-select',
      filterSelectOptions: countries,
      Cell: ({ cell }) => <span>{cell.getValue() as string}</span>,
       ↪  //optional custom cell render
    },
  ], []);

  const table = useMaterialReactTable({
    columns,
    data,
```

```
    enableColumnOrdering: true, //enable some features
    enableGlobalFilter: true,
    manualFiltering: true, //turn off client-side filtering
    onColumnFiltersChange: setColumnFilters, //hoist internal columnFilters
     ↪  state to your state
    state: {
      isLoading,
      columnFilters
    }
  });

  return (
    <div className="max-h-full pt-6">
      <LocalizationProvider dateAdapter={AdapterDayjs}>
        <ThemeProvider theme={theme}>
          <MaterialReactTable table={table} />
        </ThemeProvider>
      </LocalizationProvider>
    </div>
  );
}
```

Κώδικας Εγγραφής Χρηστών σε Φίλτρα CFP

Σε αυτό το τμήμα κώδικα, που παρουσιάζεται παρακάτω, υλοποιείται η εγγραφή των χρηστών σε φίλτρα των `CFP`. Όταν ο χρήστης πατήσει το κουμπί `update`, δημιουργείται ένα αντικείμενο το οποίο περιέχει τα γνωστικά αντικείμενα, τις τοποθεσίες και τα θέματα που έχει επιλέξει. Το αντικείμενο αυτό προστίθεται ως παράμετρος στη μέθοδο `addSubscription`, η οποία ανήκει στο backend.
Μόλις ληφθεί η απάντηση, εμφανίζεται το αντίστοιχο μήνυμα με τη χρήση του πακέτου `react-toastify`[24]. Αξίζει να σημειωθεί ότι η κλήση είναι ασύγχρονη, πράγμα που σημαίνει ότι η μέθοδος αναμένει την απάντηση του backend πριν εκτελέσει τις επόμενες γραμμές κώδικα.

```
<Button
  color="primary"
  variant="shadow"
  className="max-w-xs"
  onPress={async () => {
    const data = {
      uuid: session?.user.uuid ?? '',
      selectedAreas,
      selectedLocations: selectedCountries,
```

```
      selectedTopics: Object.entries(rowTopicsSelection)
        .filter((field) => field[1] == true)
        .map((field) => { return field[0].toString() })
    }
    const res = await addSubscriptions(data);
    if (res) {
      toast.success("Subscriptions updated.", {
        position: "top-right",
        autoClose: 5000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
        theme: selectedTheme,
        transition: Bounce,
      });
    } else {
      toast.error("Couldn't update subscriptions!", {
        position: "top-right",
        autoClose: 5000,
        hideProgressBar: false,
        closeOnClick: true,
        pauseOnHover: true,
        draggable: true,
        progress: undefined,
        theme: selectedTheme,
        transition: Bounce,
      });
    }
  }}
>
  Update
</Button>
```

4.7 GitHub Repository

Το GitHub[15] είναι μια διαδικτυακή πλατφόρμα που χρησιμοποιείται ευρέως για την αποθήκευση, διαχείριση και συνεργασία σε έργα λογισμικού. Πρόκειται για έναν εξαιρετικά δημοφιλή χώρο όπου προγραμματιστές, ομάδες ανάπτυξης λογισμικού και οργανισμοί φιλοξενούν τα αποθετήρια (*repositories*) τους, στα οποία αποθηκεύεται ο πηγαίος κώδικας και η ιστορία αλλαγών του. Ακολουθούν βασικές πληροφορίες για το GitHub:

4.7.1 Κύρια Χαρακτηριστικά

Έκδοση και Έλεγχος Κώδικα με Git

Το `GitHub`[15] βασίζεται στο *Git*, ένα σύστημα ελέγχου εκδόσεων (*version control system*). Το Git επιτρέπει την παρακολούθηση αλλαγών στον κώδικα, τη διαχείριση εκδόσεων και την επιστροφή σε προηγούμενες εκδόσεις εάν χρειαστεί.

Αποθετήρια (Repositories)

Τα αποθετήρια είναι οι "φάκελοι" όπου αποθηκεύεται ο πηγαίος κώδικας. Μπορούν να είναι:

- **Δημόσια:** Ανοιχτά προς όλους.
- **Ιδιωτικά:** Προσβάσιμα μόνο από συγκεκριμένους χρήστες.

Συνεργασία

Οι χρήστες μπορούν να συνεργάζονται σε έργα λογισμικού μέσω των λειτουργιών που προσφέρει η πλατφόρμα, όπως:

- Η διαχείριση παραρτημάτων (*branches*).
- Οι αιτήσεις έλξης (*pull requests*).
- Η επίλυση θεμάτων (*issues*).

4.7.2 Χρήση του GitHub στην Next.js

Η *Next.js*[4] δίνει τη δυνατότητα στον προγραμματιστή να δημιουργεί *cloud instances*, τα οποία εκτελούν τον κώδικα της εφαρμογής. Αυτά τα *instances* αποτελούν ουσιαστικά την πλατφόρμα του *CFPIndexer* στην περίπτωσή μας. Η δημιουργία αυτών γίνεται αυτόματα από την εταιρεία που έχει αναπτύξει τη *Next.js*[4], τη *Vercel*[18].
Ο τρόπος με τον οποίο δημιουργούνται είναι μέσω της παρακολούθησης αλλαγών στο δηλωμένο *GitHub repository*[15] της εφαρμογής. Μόλις εντοπιστούν αλλαγές, η *Vercel*[18] εκκινεί τη διαδικασία του *build* της εφαρμογής και, εφόσον δεν προκύψει κάποιο σφάλμα, την εκτελεί στο *live* περιβάλλον.

Chapter 5

Παρουσίαση του CFPIndexer

Σε αυτό το κεφάλαιο παρουσιάζονται οι βασικές οθόνες της διεπαφής χρήστη (UI) της πλατφόρμας CFPIndexer, οι οποίες έχουν σχεδιαστεί για την εξυπηρέτηση των βασικών λειτουργιών της. Η περιγραφή εστιάζει τόσο στη λειτουργικότητα όσο και στην εμπειρία χρήστη (UX), παρέχοντας αναλυτική τεκμηρίωση των χαρακτηριστικών και των δυνατοτήτων κάθε επιμέρους οθόνης. Επιπλέον, δίνεται έμφαση στη λογική σχεδίασης της πλατφόρμας, καθώς και στον τρόπο που η αρχιτεκτονική της UI υποστηρίζει την αποτελεσματική αναζήτηση και κατηγοριοποίηση περιεχομένου.

5.1 Αρχική Σελίδα

Η αρχική σελίδα αποτελεί την πρώτη οθόνη με την οποία αλληλεπιδρά ο χρήστης.

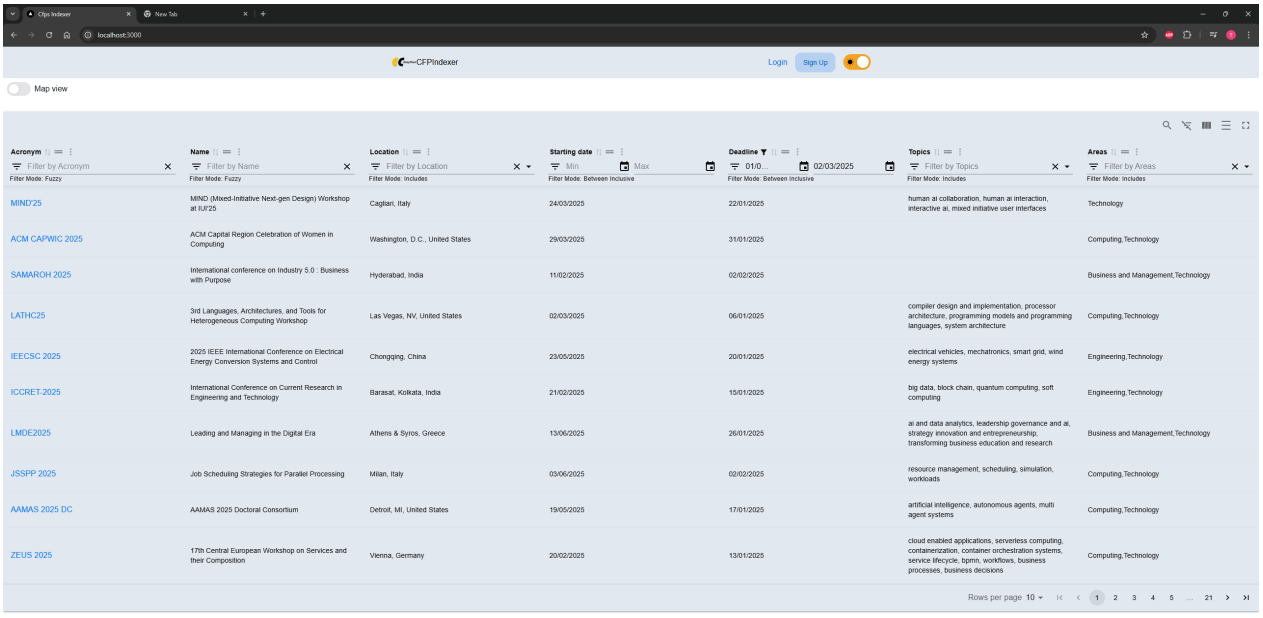


Figure 5.1: Αρχική σελίδα, πίνακας CFPs

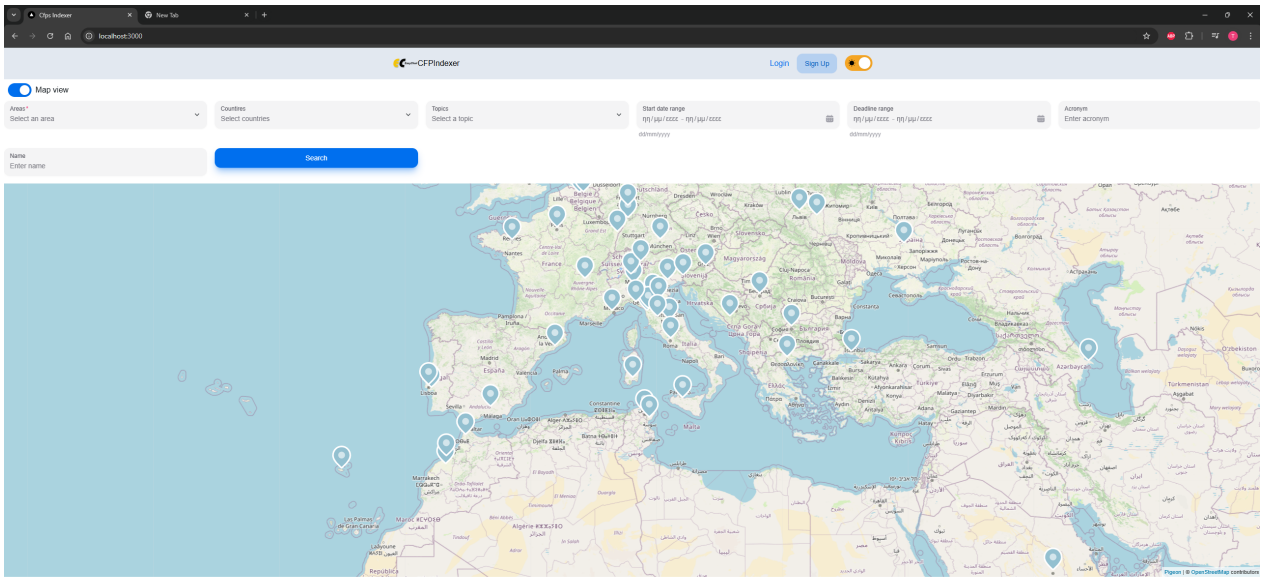


Figure 5.2: Αρχική σελίδα, χάρτης CFPs

5.2 Οθόνη Αναζήτησης - Αρχική Σελίδα με Αναζήτηση

Η οθόνη αναζήτησης αποτελεί το κεντρικό εργαλείο για την εύρεση CFPs.

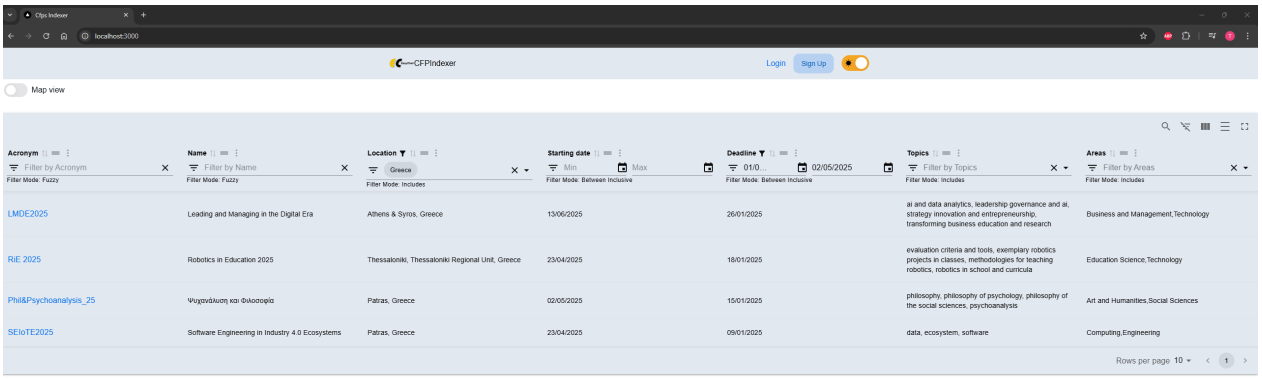


Figure 5.3: Αρχική σελίδα, πίνακας CFPs φιλτραρισμένος

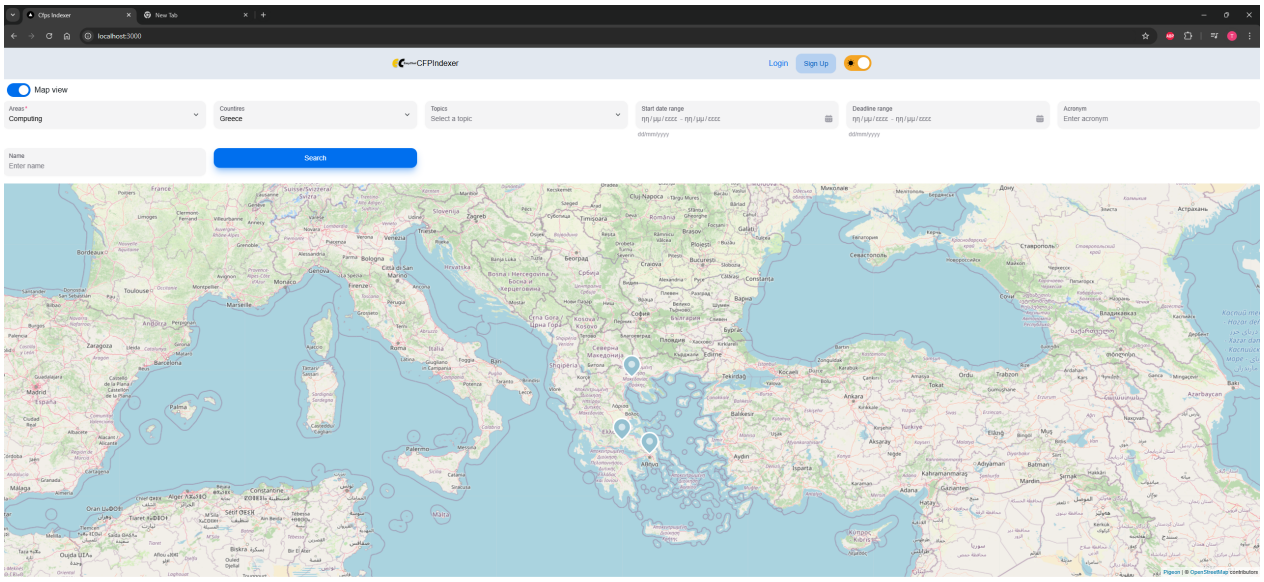


Figure 5.4: Αρχική σελίδα, χάρτης CFPs φιλτραρισμένος

5.3 Οθόνη Προβολής Αναλυτικών Πληροφοριών CFP

Σε αυτήν την ενότητα περιγράφεται η οθόνη όπου προβάλλονται οι λεπτομέρειες κάθε ξεχωριστού CFP.

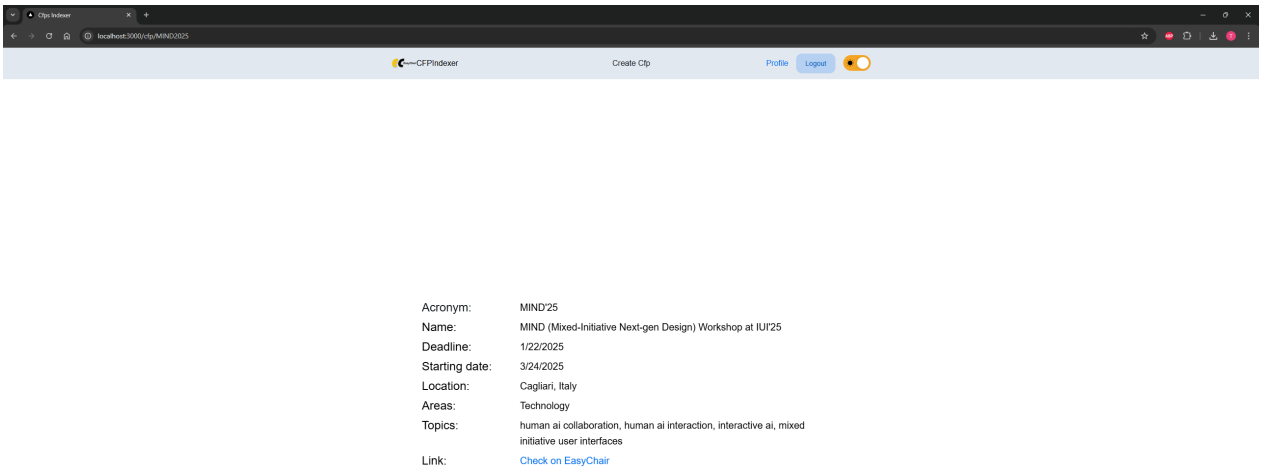


Figure 5.5: Σελίδα CFP

5.4 Σελίδες Χρήστη

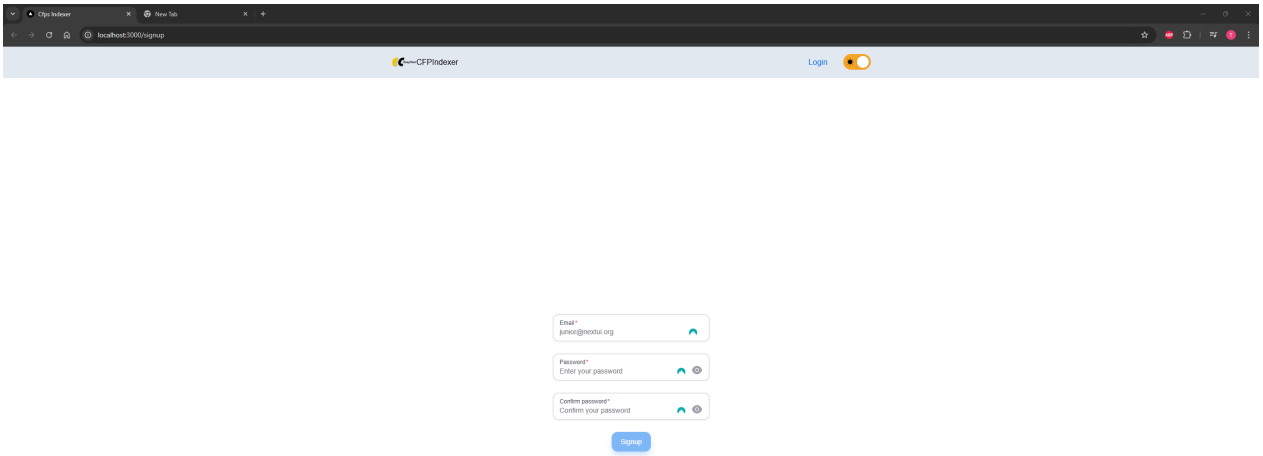


Figure 5.6: Σελίδα εγγραφής

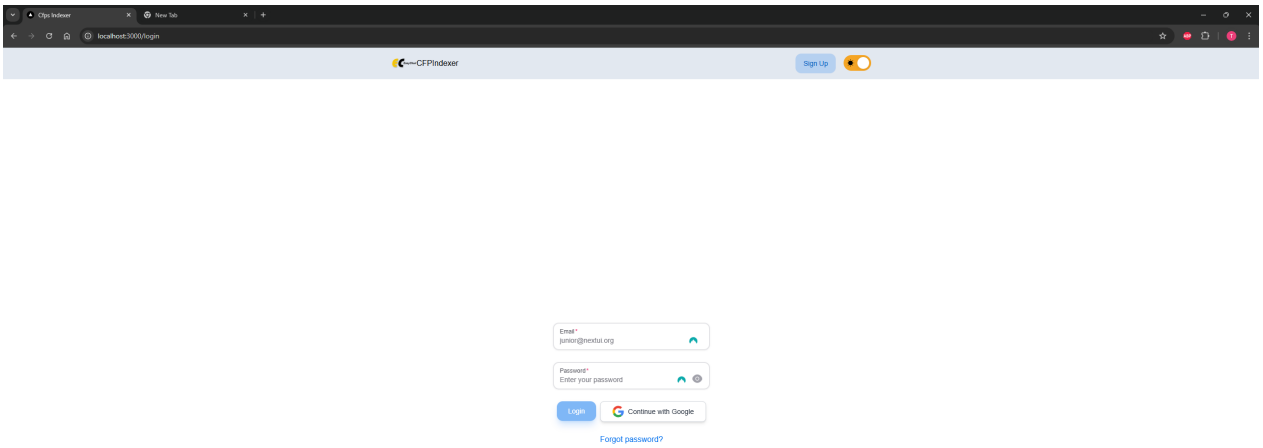


Figure 5.7: Σελίδα σύνδεσης

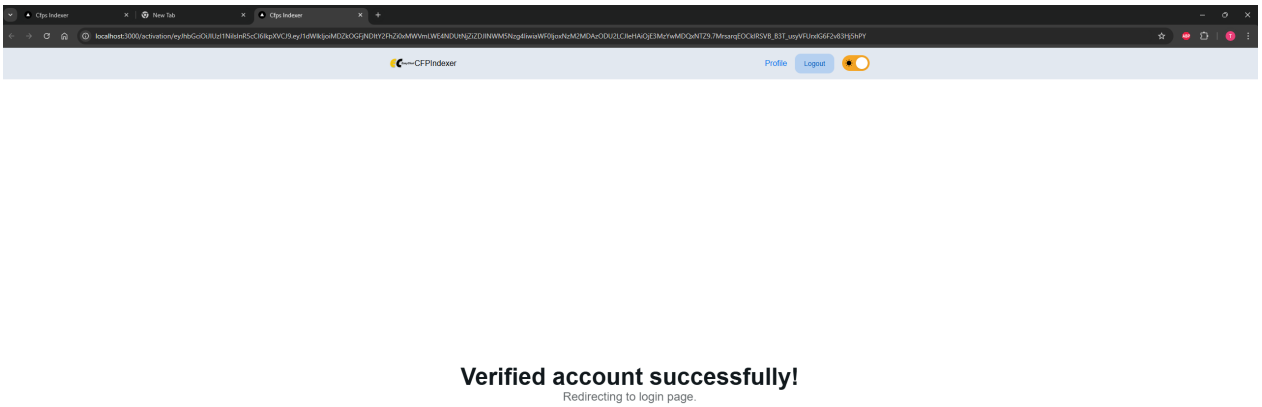


Figure 5.8: Σελίδα επιβεβαίωσης λογαριασμού

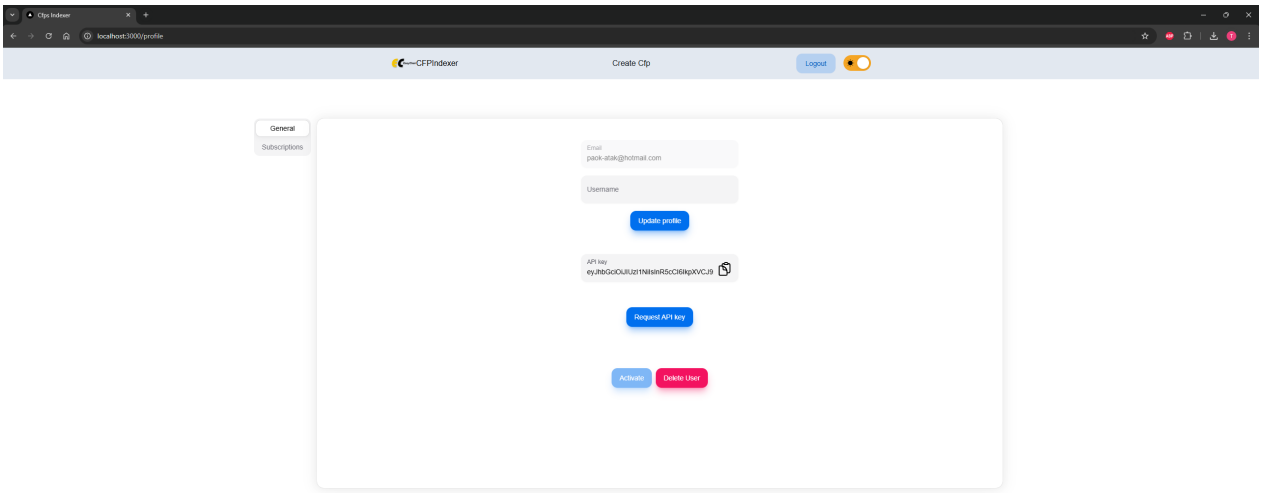


Figure 5.9: Γενική σελίδα προφίλ

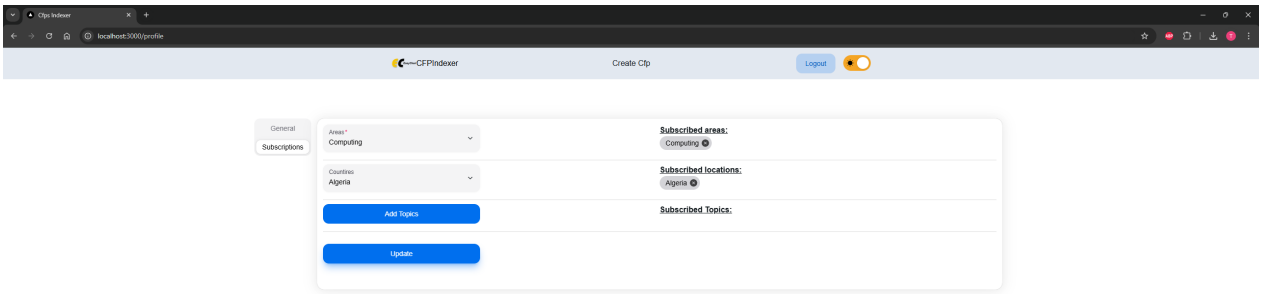


Figure 5.10: Σελίδα εγγραφών σε φίλτρα

5.4.1 Σελίδες Συνδεδεμένων Μελών

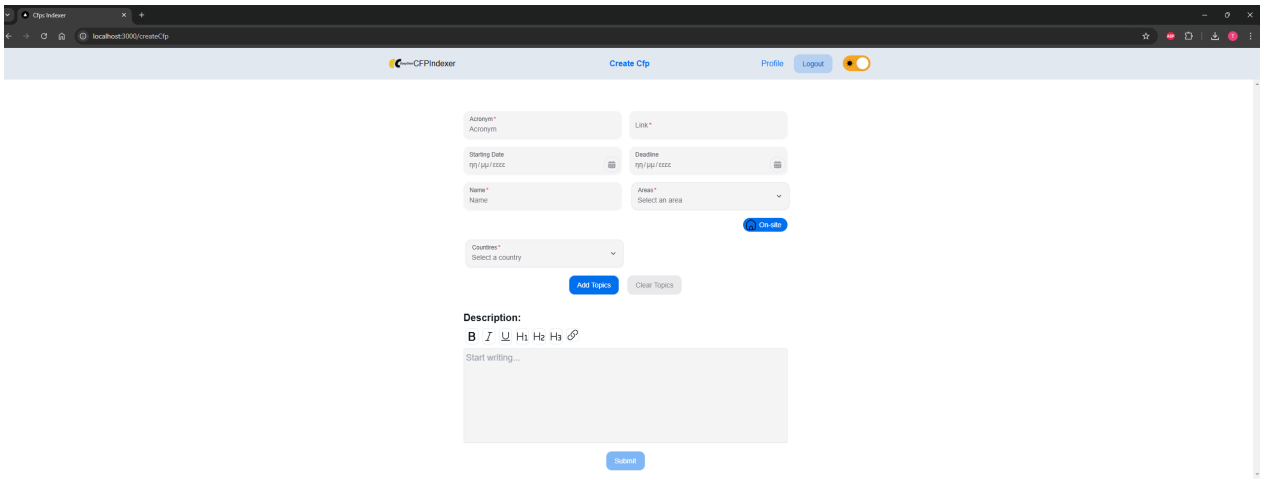


Figure 5.11: Σελίδα δημιουργίας CFP

Chapter 6

Συμπεράσματα και Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα

Η πλατφόρμα EasyChair αποτελεί ένα πολύτιμο εργαλείο για την επιστημονική κοινότητα, καθώς αντιμετωπίζει σημαντικές ανάγκες, όπως η οργάνωση και διαχείριση των προσκλήσεων για υποβολή εργασιών (CFPs). Παρόλα αυτά, ο τρόπος υλοποίησής της παρουσιάζει ορισμένα σημαντικά προβλήματα σε βασικές λειτουργίες. Ένα από τα κύρια ζητήματα είναι η δυσκολία στην αναζήτηση και φιλτράρισμα των επιθυμητών CFPs, καθώς και η αδυναμία συνδυαστικής χρήσης πολλαπλών φίλτρων.
Η πλατφόρμα CFPIndexer σχεδιάστηκε με σκοπό να αντιμετωπίσει αυτά τα προβλήματα, προσφέροντας μια πιο αποδοτική και φιλική προς τον χρήστη εμπειρία. Συγκεκριμένα, η αναζήτηση στα CFPs έχει γίνει πιο εύχρηστη και κεντροποιημένη, εξαλείφοντας την πολυπλοκότητα που παρατηρείται στην EasyChair. Επιπλέον, η πλατφόρμα υλοποιεί ένα API που είναι εύκολα προσβάσιμο από τους χρήστες, γεγονός που επιτρέπει τη χρήση της ως πηγή δεδομένων για άλλες πλατφόρμες. Με τον τρόπο αυτό, ενισχύεται η δυνατότητα συνεργασίας και η ανάπτυξη εργαλείων που μπορούν να ωφελήσουν περαιτέρω την επιστημονική κοινότητα.
Συνολικά, η πλατφόρμα CFPIndexer κατάφερε να βελτιώσει σημαντικά την εμπειρία των χρηστών, μειώνοντας τον αριθμό των απαιτούμενων ενεργειών και καθιστώντας τις διαδικασίες απλές και αποτελεσματικές. Με τον συνδυασμό αυτών των πλεονεκτημάτων, η πλατφόρμα όχι μόνο διευκολύνει τους επισκέπτες και τους χρήστες της, αλλά προσφέρει και μια καινοτόμο προσέγγιση που ανταποκρίνεται στις σύγχρονες ανάγκες της επιστημονικής κοινότητας.

6.2 Μελλοντικές Επεκτάσεις

Οι μελλοντικές επεκτάσεις του CFPIndexer επικεντρώνονται κυρίως στη βελτίωση της ακρίβειας, του ελέγχου και του εμπλουτισμού των δεδομένων κατά την εισαγωγή τους στη βάση. Ενδεικτικά, ένας τομέας προς ανάπτυξη είναι η επαλήθευση των τοποθεσιών που εισάγουν οι χρήστες. Στόχος είναι η χρήση πιο προηγμένων μεθόδων για να διασφαλιστεί ότι οι δηλωθείσες τοποθεσίες είναι υπαρκτές και ακριβείς.
Επιπλέον, η ενσωμάτωση τεχνητής νοημοσύνης αποτελεί σημαντική μελλοντική προοπτική.

Για παράδειγμα, μπορούν να αναπτυχθούν αλγόριθμοι που προτείνουν γνωστικές περιοχές ή θέματα ενδιαφέροντος με βάση τα δεδομένα που εισάγουν οι χρήστες. Αυτό θα επιτρέψει την περαιτέρω εξατομίκευση της εμπειρίας χρήστη και την καλύτερη εξυπηρέτηση των αναγκών του.
Μία ακόμη προτεινόμενη επέκταση αφορά την προσθήκη περισσότερων πλατφορμών από τις οποίες θα αντλούνται δεδομένα. Παράλληλα, η δημιουργία ενός πίνακα διαχείρισης (admin panel) θα επιτρέψει τη δυναμική προσθήκη νέων πηγών δεδομένων και τη γενικότερη διαχείριση της πλατφόρμας. Αυτή η δυνατότητα συνεπάγεται αλλαγές στον τρόπο άντλησης δεδομένων, ώστε η διαδικασία να γίνεται πιο παραμετροποιήσιμη μέσω της διεπαφής χρήστη, χωρίς να απαιτείται η άμεση επεξεργασία του κώδικα για κάθε νέα πλατφόρμα.
Τέλος, λαμβάνοντας υπόψη τη ραγδαία εξέλιξη των τεχνολογιών, βασική μελλοντική κατεύθυνση αποτελεί η αναβάθμιση των τεχνολογιών που χρησιμοποιούνται, με στόχο τη βελτίωση της απόδοσης και της ευχρηστίας της πλατφόρμας. Με τον τρόπο αυτό, το CFPIndexer θα μπορεί να ανταποκριθεί στις αυξημένες απαιτήσεις των χρηστών και να παραμείνει ανταγωνιστικό σε ένα ταχέως μεταβαλλόμενο τεχνολογικό περιβάλλον.

Bibliography

[1] E. Team, *Easychair platform*, Accessed: 2024-11-01, 2024. [Online]. Available: https://easychair.org.

[2] J. M. Zelle, *Python Programming: An Introduction to Computer Science*, 2nd. Franklin, Beedle & Associates, 2010.

[3] V. G. Nair, *Getting started with beautiful soup*. Packt Publishing Ltd, 2014.

[4] *Next.js documentation*, Vercel. [Online]. Available: https://nextjs.org/docs.

[5] D. Crockford, *JavaScript: The Good Parts: The Good Parts*. " O'Reilly Media, Inc.", 2008.

[6] S. Fenton, *Pro TypeScript: Application-Scale JavaScript Development*, 2nd. Apress, 2017.

[7] R. J. Dyer, *Learning MySQL and MariaDB: Heading in the right direction with MySQL and MariaDB*. " O'Reilly Media, Inc.", 2015.

[8] P. Teixeira, *Professional Node. js: Building Javascript based scalable software*. John Wiley & Sons, 2012.

[9] B. Lawson and R. Sharp, *Introducing html5*. New Riders, 2011.

[10] E. A. Meyer, *CSS: The Definitive Guide: The Definitive Guide*. " O'Reilly Media, Inc.", 2006.

[11] A. Fedosejev, *React. js essentials*. Packt Publishing Ltd, 2015.

[12] T. Labs, *Tailwind css documentation*, Accessed: 2024-12-15, 2024. [Online]. Available: https://tailwindcss.com/docs.

[13] M. R. T. Team, *Material react table documentation*, Accessed: 2024-10-17, 2024. [Online]. Available: https://www.material-react-table.com.

[14] N. Maintainers, *Nextauth.js documentation*, Accessed: 2024-12-03, 2024. [Online]. Available: https://next-auth.js.org.

[15] *Github documentation*, GitHub Docs. [Online]. Available: https://docs.github.com.

[16] Auth0, *Jsonwebtoken package documentation*, Accessed: 2024-12-02, 2024. [Online]. Available: https://www.npmjs.com/package/jsonwebtoken.

[17] A. Reinman, *Nodemailer documentation*, Accessed: 2024-12-02, 2024. [Online]. Available: https://nodemailer.com/about/.

[18] Vercel, *Vercel platform documentation*, Accessed: 2024-12-10, 2024. [Online]. Available: https://vercel.com/docs.

[19] *Universally unique identifier (uuid) specification*, RFC 4122. [Online]. Available: https://datatracker.ietf.org/doc/rfc4122/.

[20] K. Reitz and the Requests Contributors, *Python requests documentation*, Accessed: 2024-11-10, 2024. [Online]. Available: https://docs.python-requests.org/en/latest/.

[21] L. Bassett, *Introduction to JavaScript object notation: a to-the-point guide to JSON*. " O'Reilly Media, Inc.", 2015.

[22] *Understanding cron jobs in linux*, Linux Foundation. [Online]. Available: https://linuxfoundation.org.

[23] D. Walton, *Bcryptjs library documentation*, Accessed: 2024-12-03, 2024. [Online]. Available: https://www.npmjs.com/package/bcryptjs.

[24] F. Khdra, *React toastify documentation*, Accessed: 2024-12-16, 2024. [Online]. Available: https://fkhadra.github.io/react-toastify/.