NTUA-Appathon 2019-2020

Διαδίκτυο και Εφαρμογές

Ξανθόπουλος Βασίλειος-Χρήστος 03115186

Εισαγωγή

Στο πλαίσιο του μαθήματος υλοποιήθηκε μια διαδικτυακή εφαρμογή η οποία παίρνει σαν είσοδο από τον χρήστη ένα φάρμακο και επιστρέφει όλες τις κλινικές μελέτες που έχουν διεξαχθεί και χρησιμοποιούν αυτό το φάρμακο.

Επιπλέον εξετάζει αν το φάρμακο εμπεριέχεται στην περίληψη και στα κριτήρια καταλληλότητας της κλινικής μελέτης.

Τα δεδομένα που χρησιμοποιήθηκαν βρίσκονται εδω: https://clinicaltrials.gov/

Ο κώδικας της εφαρμογής καθώς και οι οδηγίες εγκατάστασης βρίσκονται εδω: https://github.com/vilaras/Appathon-NTUA

Αρχιτεκτονική Συστήματος

Το σύστημα ακολουθεί το σχεδιαστικό μοντέλο server/client.

- Ο cilent δέχεται την είδοσο από τον χρήστη, φτιάχνει ένα κατάλληλο αίτημα με βάση το API που προσφέρει ο server, στέλνει το αίτημα, λαμβάνει τα αποτέλεσματα και τα παρουσιάζει στον χρήστη.

- Ο server δέχεται τα αιτήματα από τον client, συλλέγει τα απαραίτητα δεδομένα κάνοτας ερωτήματα στη βάση δεδομένων και τα επιστρέφει.

Υπάρχει ένα ακόμα κομμάτι στο σύστημα που χρησιμοποιείται μόνο κατά την αρχικοποίηση και είναι υπεύθυνο να κάνει parse όλα τα δεδομένα του https://clinicaltrials.gov/ και να κατασκευάσει την βάση δεδομένων.

Εγκατάσταση

Τα βασικά βήματα που πρέπει να ακολουθήσει κάποιος για να τρέξει την εφαρμογή locally στον υπολογιστή του είναι:

1. Εγκατάσταση των πακέτων συστήματος: git, nodejs, npm, mongodb
2. Αντιγραφή του github repository: https://github.com/vilaras/Appathon-NTUA
3. Κατασκευή της βάσης δεδομένων
4. Εγκατάσταση των πακέτων node: Express.js, React.js, mongoose, ...

Αναλυτικές οδηγίες για όλα τα βήματα υπάρχουν στο github repository.

Τεχνολογίες (1)

Οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής είναι:

Για το client side:

- React.js: Ενα JavaScript Framework (τυπικά βιβλιοθήκη) για την κατασκευή User Interfaces
- MaterializeCSS: Ενα Responsive CSS Framework βασισμένο στο Material Design

Τεχνολογίες (2)

Για το server side:

- Express.js: Ενα JavaScript Framework για την κατασκευή Web εφαρμογών.
- Mongoose: Μια ODM βιβλιοθήκη για τη MongoDB
- xml2js: Ενας parser για μετατροπή αρχείων απο XML σε JS Objects

User Interface

Το User Interface είναι δομημένο με τη λογική των components. Ολα τα component βρίσκονται στον φάκελο */clients/src/components* και είναι τα εξής:

- Navbar
- Modal
- Pagination
- PreLoader
- Study
- StudyList
- StudySearch
- Tabs

Server API (1)

Ο server εξυπηρετεί αιτήματα στα εξής endpoints:

- *GET /api/studies?limit&drug_prefix/*
- *GET /api/studies/:drug?limit*
- *GET /api/drugs?limit*
- *GET /api/drugs/:prefix?limit*

Το πρώτο endpoint επιστρέφει κλινικές μελέτες. Προαιρετικά παίρνει παραμέτρους *limit* και *drug_prefix.* Η παράμετρος *limit* περιορίζει το πλήθος των αποτελεσμάτων ενω η δεύτερη κάνει filter τις κλινικές μελέτες με κριτήριο το αν υπάρχει φαρμακο σε αυτές που ξεκινάει με την παράμετρο *drug_prefix.* Το δεύτερο επιστρέφει όλες τις κλινικές μελέτες που περιέχουν ένα συγκεκριμένο φάρμακο και δέχεται επίσης προαιρετικό όρισμα *limit*.

Όλες οι συγκρίσεις είναι case insensitive.

Server API(2)

Το τρίτο και το τέταρτο enpdoint επιστρέφουν μόνο ονομασίες φαρμάκων και χρησιμοποιούνται για τη λειτουργία Autocomplete κατά την αναζήτηση του χρήστη για φάρμακα. Περιορίζουν τα αποτελέσματα που επιστρέφουν σε 10 εκτός αν περαστεί επιπλέον παράμετρος *limit.*

Ο server επίσης εξυπηρετεί ένα endpoint που χρησιμοποιείται για την αρχικοποίηση της βάσης δεδομένων:

- *GET /populate_db/*

Επικοινωνία Client/Server (1)

Η επικοινωνία μεταξύ client και server γίνεται ασύγχρονα μέσω του fetch API. Ο client κάνει ένα αίτημα σε κάποιο από τα διαθέσιμα endpoint του server, αυτός επιστρέφει τα αποτελέσματα σε μορφή JSON, στη συνέχεις ο client τα επεξεργάζεται και τα εμφανίζει στον χρήστη.

Επικοινωνία Client/Server (2)

Π.χ. αυτός είναι ο κώδικας στην πλευρά του client όταν ο χρήστης δίνει ένα φάρμακο προς αναζήτηση.

Το input του χρήστη γίνεται encode για να αποφευχθούν σφάλματα απο ειδικούς χαρακτήρες στο URI όπως %, $, /, ...

```
const updateStudies = value => {
    setLoading(true);
    const encoded_uri = `/api/studies/` + encodeURIComponent(value);
    fetch(encoded_uri)
        .then(res => res.json())
        .then(res => res.map(study => removeDuplicates(study)))
        .then(res => createFiltered(res, value))
        .then(res => setStudies(res))
        .then(() => setLoading(false))
};
```

Επικοινωνία Client/Server (3)

Από την πλευρά του ο server δέχεται το αίτημα, συλλέγει τα δεδομένα απο τη βάση, τα σορτάρει με βάση το id και τα επιστρέφει σε μορφή JSON

```
// @route   GET /api/studies/:drug
// @desc    Fetch studies containing given drug
// @access  Public
router.get('/:drug', (req, res) => {
    const lim = parseInt(req.query.limit, 10);
    Study.find(
        { "drugs": { $regex: escapeRegExp(req.params.drug) , $options : "i"} },
    )
    .limit(lim)
    .sort({ "study_id": 1 })
    .then(studies => res.json(studies))
});
```