HY - 359 23/01/22

Αναφορά ομαδικού project

Σπυρίδων Τζαγκαράκης csd4279
Παπασηφάκης Στράτος csd4278

Περιεχόμενα:

1. **Η Αρχιτεκτονική του συστήματος.**

Η αρχιτεκτονική του συστήματός μας μπορεί να περιγραφεί με ενα αρκτικόλεξο, το LEAN stack. Κάθε ένα από τα γράμματα που παρουσιάζονται αναφέρεται σε μια τεχνολογία. Έχουμε:

L - SQLite3 (database) + knex (query maker)

E - Express (backend web application framework)

A - Angular (frontend development platform)

N - nginx && node (web server, development environment)

SQLite3 (npm link)

- Σχεσιακή βάση δεδομένων που στηρίζεται πάνω στην απλότητα και την ευκολία χρήσης.
- Είναι super light, αρκεί ένα `npm` πακέτο στην `node` ώστε να στηθεί. Δεν χρειάζεται περίπλοκα προγράμματα διαχείρισης που τρέχουν στο background. Το καταφέρνει αυτό γιατί...
- Οργανώνεται σε ένα αρχείο του συστήματος. Επομένως, αρκεί απλά ένας reader του πρωτοκόλλου της ώστε να την επεξεργαστεί.

Knex (npm link)

- Μια βιβλιοθήκη, ένα `npm` πακέτο, που απλοποιεί την δημιουργία query προς σχεσιακές βάσεις δεδομένων.
- Δίνει αρκετά shortcuts για να χειριζόμαστε κλήσεις ως προς την βάση, μιας και είναι ασύγχρονες.

Express (npm link)

- διάσημη βιβλιοθήκη, `npm` πακέτο, που απλοποιεί την δημιουργία web based application στην node.
- Την χρησιμοποιούμε για την δημιουργία του REST API.

Angular (npm link , site link)

- Εργαλείο που οργανώνει την παραγωγή frontend κώδικα.
- Κάνει πολλά πράγματα για εμάς, παρέχει πολλά abstractions που υλοποιούν το κωδικα μας.
- Δημιουργει **single-page** web applications.
- Είναι **component based.**

nginx (open source)

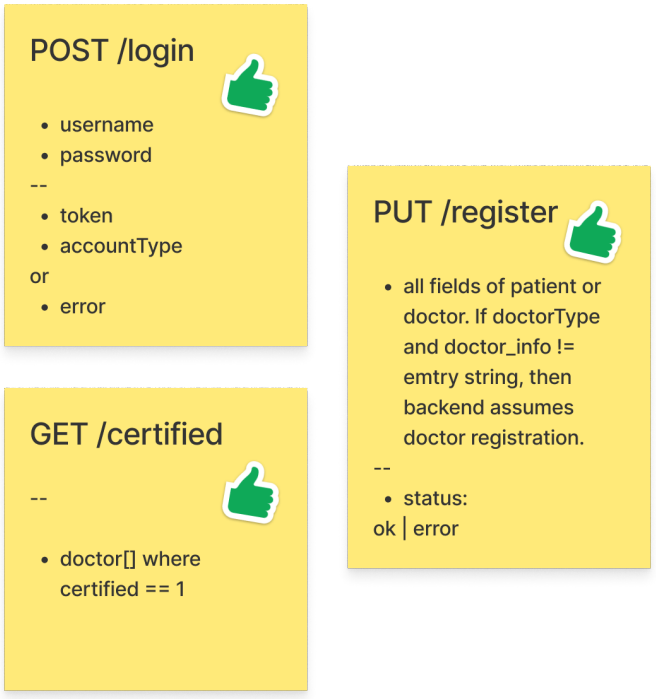
- Είναι ένας web server που θα μπορούσε να χρησιμοποιηθεί για να σερβίρει αρχεία.
- **Δεν θα το χρησιμοποιήσουμε για το πρότζεκτ** μιας και θα το τρέχουμε στα δικά μας συστήματα. Η Angular μας παρέχει έναν development web server για να σερβίρουμε το frontend μας.

2. Το Backend, λογική, τεχνολογίες και λεπτομέρειες σχεδίασης.

Το Backend είναι ουσιαστικά ένα REST API στο port 3000. Χειριζεται κλησεις ως προς την βαση δεδομενων του συστηματος. Οργανωνεται με router modules που γινονται linked στον γενικο router πριν σηκωθεί το backend. Υπαρχουν 4 api router modules, τα οποια με την σειρα τους χρησιμοποιούν 3-4 middleware modules καθως και εναν adaptor για την βαση δεδομενων. Δεν παρουσιάζουμε τα middleware παρα μονο τα supported routers και τα endpoint τους. Για περισσοτερες πληροφοριες μπορειτε να κοιταξετε τον κωδικα, ειναι αρκετα intuitive. Έχουμε:

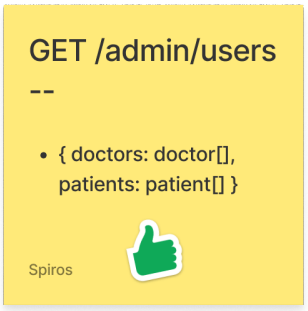
1. Open router // δεχεται κλησεις στο /api/open/

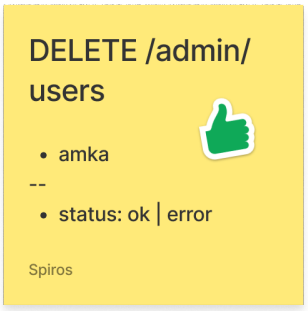
Περιέχει όλες τις κλήσεις του συστήματος οι οποίες δεν απαιτούν authentication. (Οπως /login, /register ...). Αναλυτικά, περιέχει:

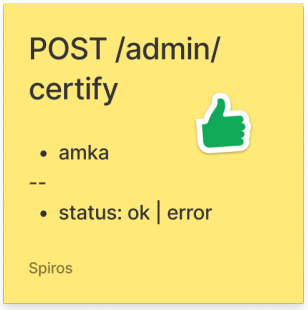


2. Admin router // δεχεται κλησεις στο /api/admin/

Περιέχει όλες τις κλήσεις του συστήματος οι οποίες απαιτούν authentication με admin generated token. Αναλυτικά, περιέχει (το token δεν αναφέρεται στα παρακάτω notes αλλά συμπεριλαμβάνεται στα headers του αιτήματος):

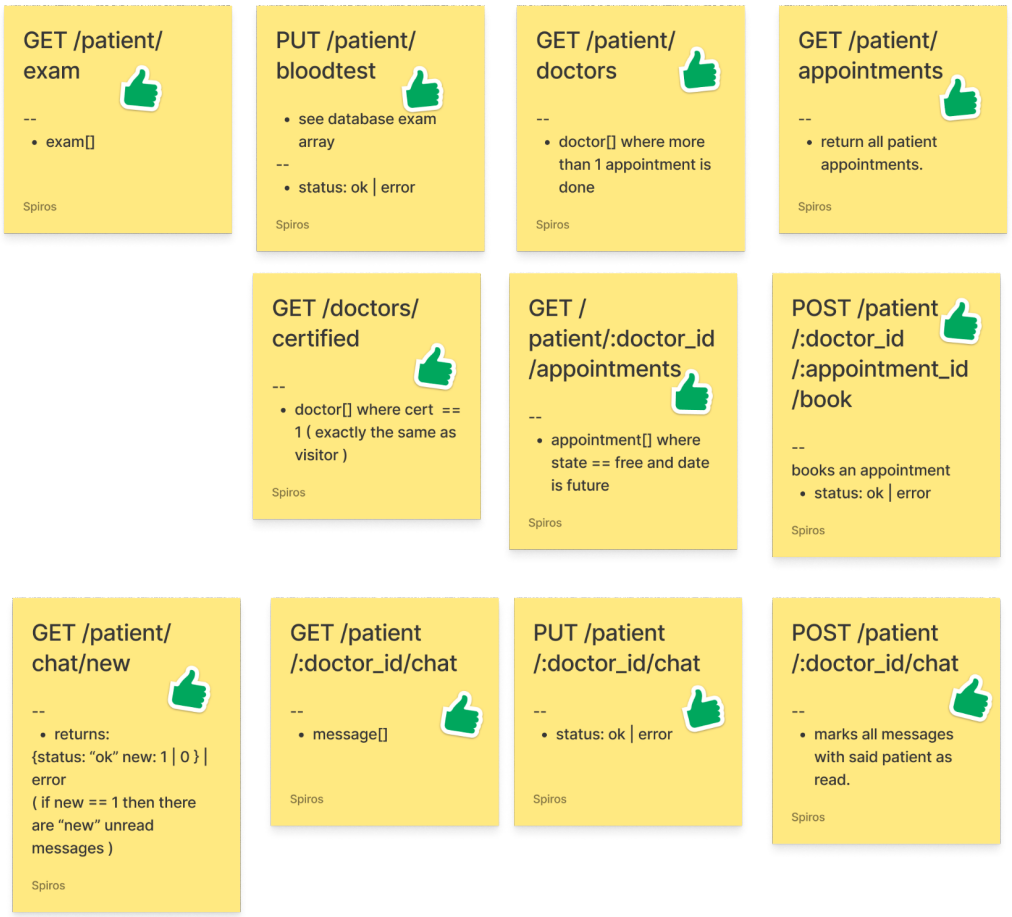






3. Patient router // δεχεται κλησεις στο /api/patient/

Περιέχει όλες τις κλήσεις του συστήματος οι οποίες απαιτούν authentication με patient generated token. Αναλυτικά, περιέχει (το token δεν αναφέρεται στα παρακάτω notes αλλά συμπεριλαμβάνεται στα headers του αιτήματος):



4. Doctor router // δεχεται κλησεις στο /api/doctor/

Περιέχει όλες τις κλήσεις του συστήματος οι οποίες απαιτούν authentication με doctor generated token. Αναλυτικά, περιέχει (το token δεν αναφέρεται στα παρακάτω notes αλλά συμπεριλαμβάνεται στα headers του αιτήματος):

GET /doctor/appointments

--

- appointment[]

Spiros

PUT /doctor/appointment

- (doctorID)
- date
- price

--

- status: ok | error

Spiros

GET /doctor/patients

--

- patient[] where one or more appointments is done with said doctor

Spiros

GET /doctor/:patient_id/bloodtests

- patientID

(must check if there is a done appointment between them)

--

- exams[]

Spiros

POST /doctor/appointment

- newState: String
- appointmentID

If newStage is done, add a pair of patient - doctor relationship

--

- status: ok | error

Spiros

GET /doctor/:patient_id/chat

--

- message[]

Spiros

PUT /doctor/:patient_id/:exam_id/treatment

- examID
- string: drugs
- string: examinations
- duration

--

- status: ok | error

Spiros

GET /doctor/chat/new

--

- returns:

all patients with unread messages.

Spiros

PUT /doctor/:patient_id/chat

--

- status: ok | error

Spiros

POST /doctor/:patient_id/chat

--

- marks all messages with said patient as read.

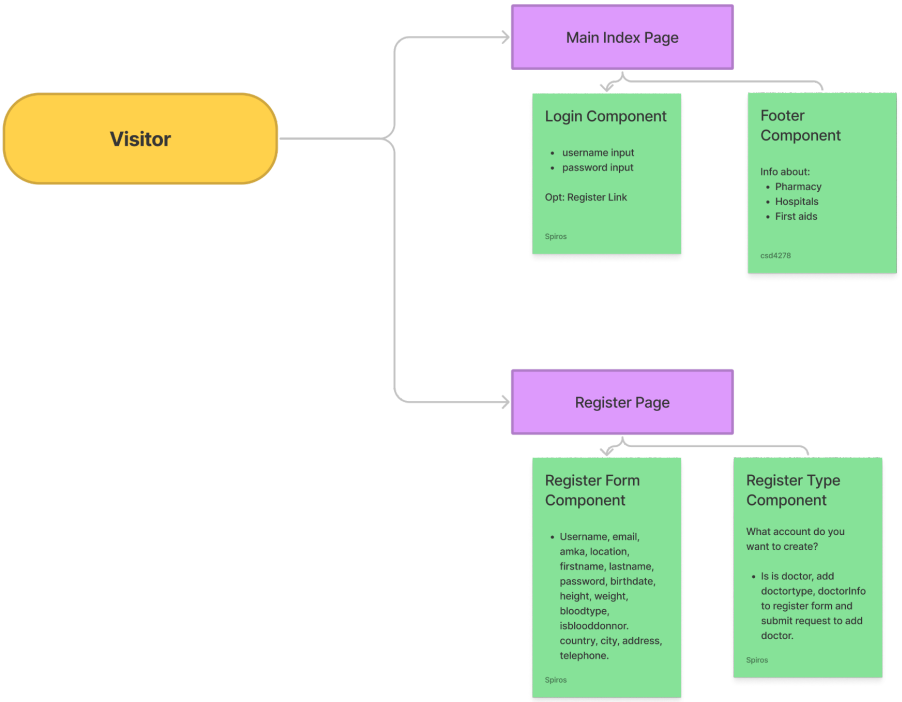
Spiros

3. Το Frontend, λογική, τεχνολογίες και λεπτομέρειες σχεδίασης.

Το frontend είναι μια διαδικτυακη εφαρμογη που σηκώνεται στο port 4200. Όπως είπαμε, η Angular είναι component-based framework. Τα components ειναι πακεταρισμενα και απομονωμενα κομματια κωδικα που περιεχουν την δικη τους δομη (html), στιλ (css) και λογικη (js). Πρακτικα, δημιουργουμε επαναχρησιμοποιουμενα components και τα προσθετουμε στις “σελιδες” του συστηματος (οπου και αυτες ειναι component αλλα θα αναφερονται ως σελιδες για να μην υπαρχει μπερδεμα). Οι κεντρικες σελιδες του συστηματος βρισκονται στο “frontend/src/app/pages/”. Τα components τους στο “frontend/src/app/general/”. Οι ελευθερες σελιδες (σελιδες που δεν απαιτουν token για πλοήγηση) ειναι:

1. (/index) Το index page του συστηματος. Περιεχει login, register επιλογες καθως και παρουσιαζει τους certified γιατρους του συστηματος.
2. (/register) Το register page του συστηματος. Περιεχει μια απλη φορμα για την εγγραφη user στο συστημα. Ο χρηστης, με την επιλογή doctor account εμφανιζει κρυμμενες επιλογες που απαιτουνται για την εγγραφη γιατρου.

Η δομη των component που χρησιμοποιηθηκαν:

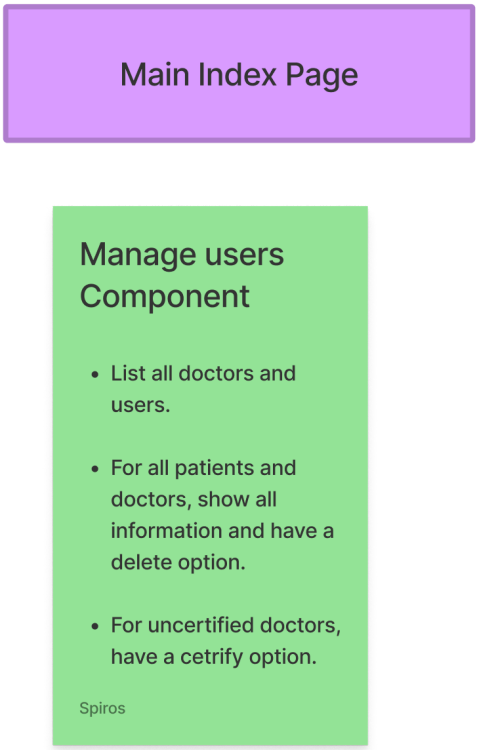


(απο δω και περα το footer, header, navigation component δεν θα αναφέρεται στις φωτο γιατι υπαρχει παντου)

Οι σελιδες του admin:

1. (/admin/index) Το admin page. Περιεχει λιστες γιατρων και patient με επιλογες certify και delete. Certify επιλογη εμφανιζεται μονο σε doctors που δεν ειναι certified.

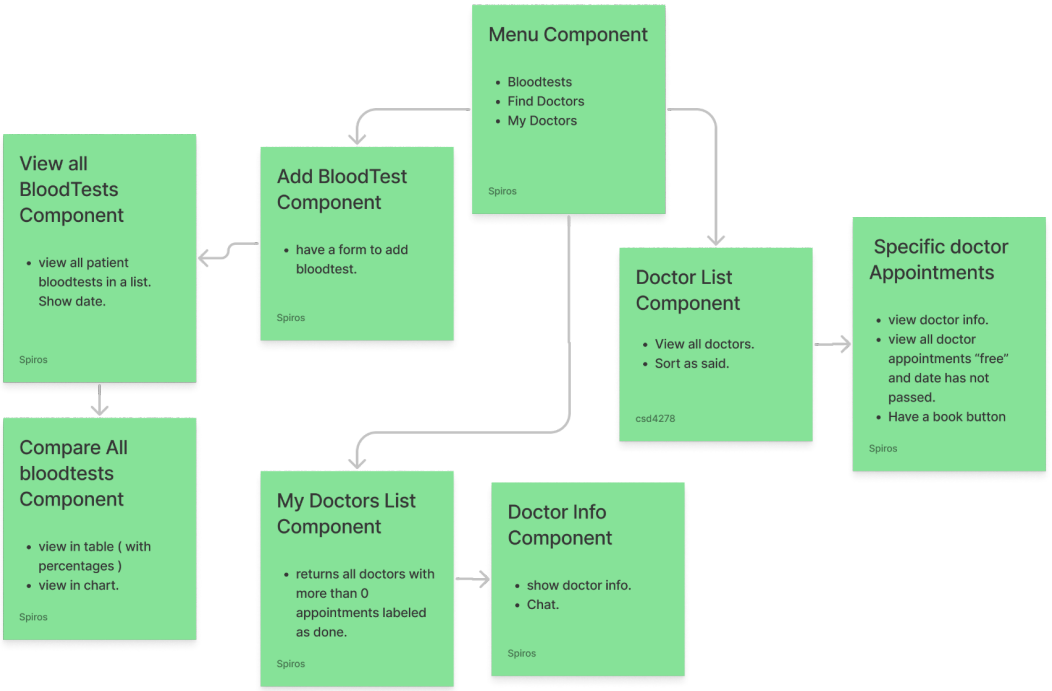
Η δομη των component που χρησιμοποιηθηκαν:



Οι σελιδες του patient:

1. (/patient/index) Το index page του patient. Περιεχει τα διαθεσιμα ραντεβου σορταρισμενα κατα ημερομηνια. Περιεχει το τελευταιο treatment που του εχει δοθει. Περιεχει ευρεση νεων γιατρων.
2. (/patient/mydoctors) Το page περιεχει τους γιατους του χρηστη. Πατωντας στο more κουμπι τους, ο χρηστης μπορει να δει περισσοτερες πληροφοριες για αυτους, τα ραντεβου μαζι του καθως και να συνομιλησει μαζι τους. Αμα υπαρχουν μη αναγνωρισμενα μυνηματα, ειδικο notification φαινεται στο entry του γιατρου.
3. (/patient/mybloodtests) Ενα page που περιεχει επιλογες για τις εξετασεις του χρηστη. Μπορει να προσθεσει νεες εξετασεις καθως και να δει τις εξετασεις του σε λιστα και γραφημα.

Η δομη των component που χρησιμοποιηθηκαν:



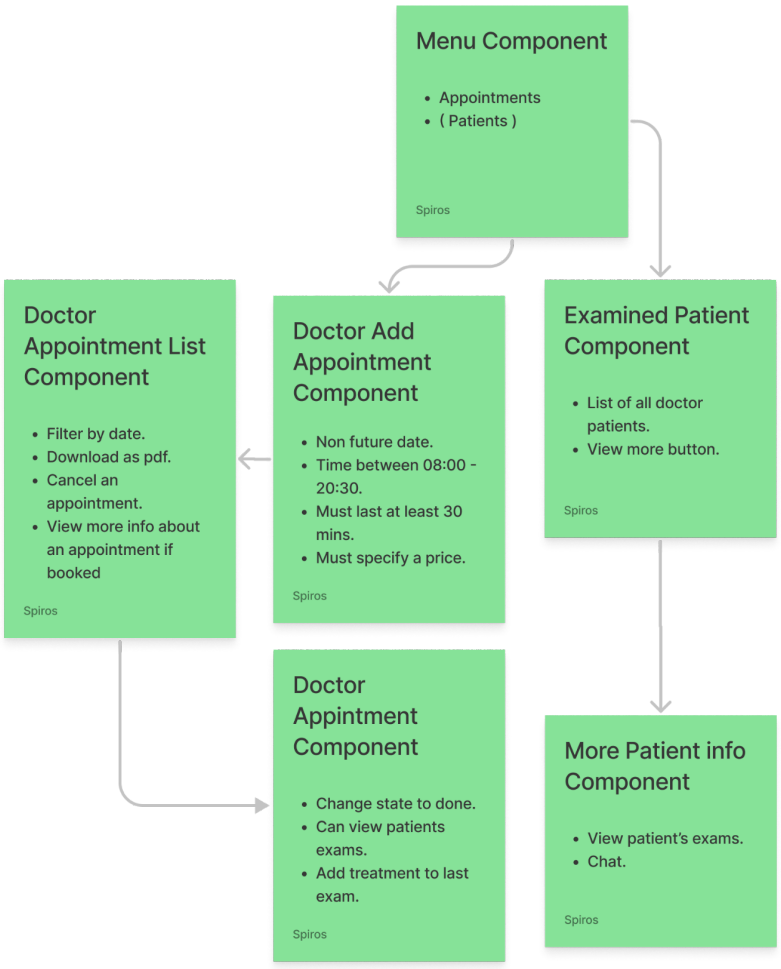
Οι σελιδες του doctor:

1. (/doctor/index) Περιεχει ολους τους ασθενεις του γιατρου σε μια λιστα. Πατωντας σε εναν ασθενη μπορει να δει τα ραντεβου τους, τις εξετασεις τους και το chat τους. Πανοντας more σε ενα booked

ραντεβου τους, μπορει να προσθεσει treatment, να δει τις εξετασεις τους και να ολοκληρωσει το ραντεβου.

2. (/doctor/appointments) Περιεχει τα ραντεβου του γιατρου σε μια λιστα με ικανοτητα επεξεργασιας της καταστασης τους καθως και δυνατοτητα αναρτησης νεων ραντεβου.

Η δομη των component που χρησιμοποιηθηκαν:



(τα components που παρουσιαζονται εδω, υπαρχουν για ευκολια κατανοησης. Στο κανονικο συστημα υπαρχουν περισσοτερα components τα οποια δεν υπαρχει λογος να αναφερθούν.)

Γενικα στο frontend μπορουν να ειπωθούν πολλα για τα components και τα services αλλα δεν νομιζουμε πως εχει καποια σημασια να τα αναφερουμε, οπως και να αναφερουμε τι κανει η καθε συναρτηση σε αυτα. Το ονομα του καθε component ειναι αρκετο για να καταλαβει κανεις τι πανω κατω παιζει.

Για παραδειγμα ενα component στο '/general' με ονομα 'doctor-list-entry' περιεχει λογικη, δομη και στυλ για να αναπαρασταθει ενας γιατρος σε μορφη λιστας. Τα περισσοτερα απο αυτα τα component παιρνουν input args απο τα υψηλοτερα στην ιεραρχια components, τα οποια κανουν query το backend μεσω του REST API στο port 3000.

Μια υπηρεσια που ειναι αξια αναφορας ειναι η StatefulNavigationService. Γενικα οι υπηρεσιες στην Angular ειναι κωδικας που μπορει να γινεται inject σε components για να μην χρειαζεται να ξαναγραφεται. Μια υπηρεσια μπορει να οριστει ως singleton, δηλαδη να υπαρχει μια φορα ο κωδικας της και να μοιραζεται μεταξυ των component που την χρησιμοποιούν. Αυτο, μας δινει την δυνατοτητα να κουβαλαμε - σωζουμε κατασταση στην εφαρμογη μας. Αυτην την δυνατοτητα εκμεταλλεύεται η StatefulNavigationService ωστε να κραταει τα selected users, appointments πραγμα που κανει την εφαρμογη μας να μην χανει κατασταση κατα το back navigation καθως και να επιτρεπει το forward navigation. Η υπηρεσια χρησιμοποιείται οπου χρειαζεται διατηρηση καταστασης στην εφαρμογη.

4. Ενδεικτικά screenshot της τελικής εφαρμογής.

Τα screenshot ειναι πολλα και μεγαλα σε μεγεθος. Μπορειτε να τα βρειτε στον φακελο /misc. Ειναι φωτογραφιες με ονομα ‘screenshot-x.png’. Τα screenshot εχουν παρθει απο το τελικο συστημα.

5. Η βάση του συστήματος.

Εχει δημιουργηθει σε sqlite3 συμφωνα με το pdf / tutorial της ασκησης 3. Εχει απλοποιηθει λιγο γιατι δεν γινοταν 1-1 mapping αλλα γενικα κανει την ιδια δουλεια. Ειναι οργανωμενη στο αρχειο “backend/database/database.sqlite3”. Για να δειτε τα περιεχομενα αρκει να ανοιξετε το αρχειο με vs code, θα σας προτεινει να κατεβασετε ενα extension για να δειτε το περιεχομενο του. Διαφορετικα μια εφαρμογη οπως η “db browser for SQLite” ειναι αρκετη για ανοιξετε το αρχειο. Στο “backend/database” υπαρχει και ο adaptoras της βασης που χρησιμοποιει το knex για να δημιουργησει τις συνδεσεις. Πιθανων εχει αρκετες διαφορες με το original σας.

6. Σύνδεση βάσης με frontend. Token based authentication.

Το frontend επικοινωνει με την βαση μεσω του REST API που εχει υλοποιηθει. Βεβαιως, δεν ειναι ολα τα routes publicly accessible. Τα routes του doctor, patient, admin ειναι προστατευμενα απο τα public καθως και μεταξυ τους. Επισης, υπαρχει προστασια μεταξυ διαφορετικων doctor, patient (με αλλα λογια, δεν μπορει ενας doctor να λαβει πληροφοριες ενος αλλου). Αυτην, την τριπλη προστασια διαχειριζεται το Authentication middleware χρησιμοποιωντας jsonwebtokens. Βρισκεται στον φακελο “backend/authentication”. Περιεχει μια κλάση με μια μεθοδο καθως και μια public μεθοδο generate_token. Παρακατω περιγραφεται αναλυτικα το authentication της εφαρμογης.

Η συναρτηση generate_token δημιουργει και κρυπτογραφει ενα νεο unique token βαση του τυπου και του id του account. Η generate token καλειται επειτα απο ενα επιτυχες login και το output της γυριζει ως json παραμετρος του /login. Το frontend, επειτα απο ενα επιτυχες login, σωζει το token στο local storage του browser και το προσθετει σε οποιοδηποτε request στειλει στο backend απο εκει και περα. Το καταφερνει αυτο προσθετοντας στα http requests ενα authentication header με

value το token αυτο. Οταν το το backend λαβει ενα request που βρισκεται υπο προστασια, το authentication middleware function “authenticate” κοιταει αμα το request εχει token. Αμα δεν εχει στελνει 406. Διαφορετικα αποκρυπτογραφει το token. Αμα δεν το καταφερει, στελνει 406. Επειτα, κοιταει το περιεχομενο του ωστε να διαπιστωσει για ποιο τυπο λογαριασμου αναφερεται. Αμα ο τυπος του λογαριασμου ειναι διαφορετικος απο τον τυπο που αναμενεται σε αυτο το route τοτε στελνει 406. Διαφορετικα, σωζει το account id που βρισκοταν στο token και χρησιμοποιώντας αυτο εκτελει τις κλησεις στην βαση. Ετσι, δεν μπορει με κανεναν τροπο, εκτος απο tampering, να δει ενας λογαριασμος τα δεδομενα ενος αλλου. Το παραπανω βασιζεται στο οτι καθε user του συστηματος εχει ενα unique id το οποιο σωζεται στον αναλογο πινακα της βασης.

7. Επιπλέον βιβλιοθήκες που χρησιμοποιήθηκαν.

Για να παρουσιαστουν οι χαρτες, τα γραφήματα, η client side δημιουργια pdf χρειαστηκαν επιπλεον εξωτερικες βιβλιοθηκες. Επισης, για την δημιουργια και κωδικοποιηση / αποκωδικοποιηση τον web token χρειαστηκε εξωτερικη βιβλιοθηκη.

Για τα γραφιματα χρησιμοποιηθηκε ενας angular wrapper του google Google Visualization API. Το npm πακετο ειναι το εξης. Το πακετο περιεχει ενα module και components για την αναπαρασταση γραφων.

Για τα maps χρησιμοποιηθηκε ενα official angular Google Map API extension. Το npm πακετο ειναι το εξης. Το πακετο περιεχει module και components για να αναπαρασταση χαρτων. Το api key που χρησιμοποιηθηκε ειναι σε free trial για 90 ακομα μερες, επειτα, οι χαρτες μπορει να μην λειτουργουν.

Για το client size pdf creation χρησιμοποιηθηκε το jsPDF το οποίο, αν και κακως (γιατι δεν ειναι optimized και ανεβασε πολυ το size της εφαρμογης), προστεθηκε ως bundle στην angular εφαρμογη. Το npm πακετο ειναι το εξης.

Για τα jsonwebtoken χρησιμοποιηθηκε αυτο το πακετο.

Με το καλο...