ΦΑΡΜΑΚΑΠΟΘΗΚΗ

ΤΕΚΤΟΝΙΔΟΥ ΧΡΥΣΑΝΘΗ 4101
ΔΕΛΗΓΙΩΡΓΗ ΕΥΑΓΓΕΛΙΑ 4213
ΜΙΧΕΛΑΚΗΣ ΣΤΕΦΑΝΟΣ 4149
ΣΑΜΑΗΛΙΔΗΣ ΑΝΑΣΤΑΣΙΟΣ 4238
ΣΑΚΑΛΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 4243

ΠΡΟΓΡΑΜΜΑΤΙΣΤΉΣ (PROGRAMMER)
ΜΙΧΕΛΆΚΗΣ ΣΤΈΦΑΝΟΣ
ΑΕΜ 4149

Τεκμηρίωση

Ξεκινώντας από τον server, έχουμε το αρχείο app.js. Το αρχείο app.js είναι η "βάση" του server και χωρίς αυτό ο server δεν θα μπορεί να απαντήσει στα request που του έρχονται από το front end πρόγραμμα το οποίο είναι γραμμένο σε Java/JavaFX. Ο τρόπος λειτουργείας του είναι απλός, δέχεται request τύπου POST, PATCH και DELETE και αναλόγος με την δομή των request αυτών εκτελεί τα ανάλογα SQL Queries στην βάση που έχει στηθεί. Για να το επιτευχθεί αυτό, το αρχείο app.js κάνει εισαγωγή τα παρακάτω αρχεία και κάθε φορά που λαμβάνει ένα request, το περνάει αυτόματα στο αρχείο το οποίο απευθύνεται το request αυτό.

```
const express = require("express");
const app = express();
const userRouter = require("./api/users/user.router");
const providerRouter = require("./api/providers/provider.router");
const permissionRouter = require("./api/permissions/permission.router");
const productRouter = require("./api/products/product.router");
const orderRouter = require("./api/orders/order.router");
const clientRouter = require("./api/clients/client.router");
const invoiceRouter = require("./api/invoices/invoice.router");
```

Για να κάνει ένας χρήστης είσοδο στο πρόγραμμα στέλνετε το request:
http://localhost:3000/api/users/login.

Το request αυτό περίεχει ένα "σώμα" στο οποίο μέσα βρίσκεται το username και το password που έδωσε ο χρήστης, και εκτελείτε στην βάση δεδομένων η εντολή:

SELECT * FROM users WHERE username = ? AND password = ?

Αν τα στοιχεία που έδωσε ο χρήστης είναι σωστά, το request αυτό επιστρέφει σαν απάντηση το validation token το οποίο δημιουργείτε από τον server και περιέχει μέσα το όνομα χρήστη και την ώρα που στάλθηκε το request "/login", δηλαδή την ώρα που έκανε την σύνδεση. Αν τα στοιχεία που έδωσε ο χρήστης είναι λάθος και δεν αντιστοιχούν σε κάποιον χρήστη τότε δεν επιστρέφεται token. Κάθε token ισχύει για 8 ώρες, αφού περάσουν οι 8 ώρες αυτές ο χρήστης θα πρέπει να κάνει αποσύνδεση και να ξανακάνει log in.

Μόνο δύο request δεν απαιτούν validation token για να επιστρέψουν απάντηση, το "log in" και το "sign up". Για να κάνει ένας χρήστης εγγραφή στέλνετε το request:
http://localhost:3000/api/users/createUser

Όπως και το request του log in, έτσι και το request του sign up έχει ένα "σώμα" το οποίο μέσα έχει τα στοιχεία εγγραφής του χρήστη. Κάτα την εκτέλεση του request αυτού ο server κάνει εισαγωγή των στοιχείων του χρήστη στην βάση δεδομένων και αν η βάση επιστρέχει error(π.χ. Σε περίπτωση που υπάρχει ένας χρήστης με το ίδιο username) ή δεν επιστρέψει καμία απάντηση τότε ο χρήστης δεν δημιουργείται και ο server επιστρέφει σαν απάντηση ένα error το οποίο στην συνέχεια επεξεργάζεται από το front end και εμφανίζεται στον χρήστη. Αν η βάση επιστρέψει "Data inserted successfully" τότε ο χρήστης έχει δημιουργηθεί με επιτυχεία και ο server επιστρέφει ένα json αρχείο το οποίο μέσα έχει

το μύνημα “success: true”.

Όλα τα request που στέλνει ένας χρήστης στον server, πρέπει έχουν μέσα ένα validation token.

Κάθε request απευθύνεται σε μία κατηγορία(π.χ. Το request http://localhost:3000/api/orders/... απευθύνεται στην κατηογορία orders).

Ένα request περνάει από τρία αρχεία πρωτού επιστρέψει απάντηση ο server, το αρχείο router, controller και service.

Όταν στέλνετε ένα request για εισαγωγή, διαγραφή, εμφανίση ή επεξεργασία δεδομένων το αρχείο app.js στέλνει το request αυτό στο αντίστοιχο αρχείο “router” της κατηγορίας στην οποία απευθύνετε το request. Στην συνέχεια, ο “router” της κατηγορίας καλεί την συνάρτηση checkToken η οποία ελέγχει αν ο χρήστης που έστειλε την ετνολή έχει δικαιώματα(δηλαδή έχει πάρει validation token από τον server) ελέγχοντας το token το οποίο έδωσε ο χρήστης μαζί με το request. Αν ο χρήστης δεν έδωσε token ή το token που έδωσε δεν μπορεί να επαληθευθεί από την συνάρτηση checkToken το request δεν εκτελείται και ο server επιστρέφει “Authentication Failed”. Ένα validation token μοιάζει κάπως έτσι:

{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjoiQURNSU4iLCJpYXQiOjE1OTEwMzY3MDksImV4cCI6MTU5MTA2NTUwOX0.dLqhT_RnwfYV429GIimhDQ30vDNVqBxVAagEnag1WGQ"}

Αφού γίνετι η επαλήθευση του token, ο router περνάει την εντολή στον controller της κατηγορίας. Ο controller παίρνει τα στοιχεία του σώματος του request και τα στέλνει σαν “data” στο αρχείο service της κατηγορίας. Το αρχείο service στην συνέχεια περνάει τα στοιχεία που πήρε από τον controller σε ένα SQL query το οποίο εκτελείτε και επιστρέφει μια απάντηση. Η απάντηση αυτή, επιστρέφεται από το αρχείο service στον αρχείο controller. Ο controller στην συνέχεια ελέγχει αν η απάντηση που δόθηκε από την βάση είναι error ή δεδομένα προς εμφάνιση. Αν η απάντηση είναι error, ο controller προσαρμόζει μια απάντηση η οποία θα διαβάζεται ως error από την Java. Αν η απάντηση είναι δεδομένα προς εμφάνιση, τότε ο controller επιστρέφει ένα αρχείο json το οποίο μέσα περιέχει μία λίστα με δεδομένα σε μορφή την οποία μπορεί να αναγνωρίσει η java ως αντικείμενα και να τα επεξεργαστεί.

Για παράδειγμα, αν σταλθεί στον server η εντολή http://localhost:3000/api/clients/createClient το αρχείο app.js περνάει το request και το σώμα του στον clientRouter.

```
app.use("/api/clients", clientRouter);
```

Από εκεί, ο clientRouter ελέγχει το validation token που ήρθε μάζι με το request. Αν το validation token είναι δεκτό, χρησιμοποιόντας μια συνάρτηση η οποία έχει γίνει εισαγωγή από το αρχείο επαλύθευσης. Αν η επαλύθευση πραγματοποιηθεί με επιτυχεία, το request περνιέται στην συνάρτηση getClients του controller.

```
const { checkToken } = require("../../auth/token_validation");
router.post("/createClient", checkToken, createClient);
```

Από εκεί ο controller στέλνει το request και το σώμα, αν υπάρχει, σαν μεταβλητές στο αρχείο client.service

Στην συνέχεια, εκτελείτε ένα SQL Query στην βάση από το αρχείο client.service με τα στοιχεία του σώματος που ήρθαν με το request

```
createClient: (data, callback) => {
    pool.query(
      `insert into clients(pharmacyName, phoneNo, address, numberOfOrders, zipCode)
        values(?,?,?,0,?)`,
      [
        data.pharmacyName,
        data.phoneNo,
        data.address,
        data.zipCode
      ],
      (error, results, fields) => {
        if (error) {
          return callback(error);
        }
        return callback(null, results);
      }
    );
  },
```

Αφού εκτελεστεί το sql query, το αρχείο client.service περνάει την απάντηση στο client.controller και από εκεί επιστρέφεται η απάντηση είτε σαν error είτε σαν δεδομένα προς εμφάνιση

```
createClient: (req, res) => {
    const body = req.body;
    createClient(body, (err, results) =>{
      if (err) {
        console.log(err);
        return res.status(500).json({
          success: false,
          message: "Database connection error"
        });
      }
      return res.status(200).json({
        data: results
      });
    });
  },
```

Όπως αναφέρεται πιο πάνω, για το front end χρησιμοποιείται η γλώσσα προγραμματισμού Java/JavaFX. Στην είσοδο ενώς χρήστη, η Java στέλνει ένα Http Request στον server με τα στοιχεία του χρήστη και αν επιστραφεί token από τον server(δηλαδή τα στοιχεία του χρήστη επαλυθεύτηκαν με επιτυχία) αποθηκεύεται σαν String μεταυλητή. Αν επιστραφεί error, ή δεν επιστραφεί απάντηση καθόλου τότε εμφανίζεται στον χρήστη ένα παράθυρο το οποίο τον ενημερώνει αναλόγος την περίπτωση τι πήγε στραβά

```
static String usrToken;


public void logIn() throws Exception {
	Main.UserName=null;
	Main.usrToken=null;
	if (PasswordTxtField.getText().length() < 4) {System.out.println("Password too short"); return;}
	try {
		new
LogInHttpRequestSynchronous().run(Main.globalIP+"/api/users/login",UsernameTxtField.getText(),PasswordTxtField.getT
ext());
	} catch (Exception e) {
		System.out.println(e.getMessage());
		if (e.getMessage().equals("Failed to connect to localhost/0:0:0:0:0:0:0:1:3000")) {
			Alert alert = new Alert(Alert.AlertType.WARNING);
			alert.setTitle("Error");
			alert.setHeaderText("Internal Server Error");
			alert.setContentText("Cannot connect to server. Error Message:" + "\n" + e.getMessage());
			System.out.println(e.getMessage());
			alert.showAndWait();
			return;
		};
		Alert alert = new Alert(Alert.AlertType.WARNING);
		alert.setTitle("Warning Dialog");
		alert.setHeaderText("Error");
		alert.setContentText("Invalid Username/Password combination: " + e.getMessage());
		System.out.println(e.getMessage());
		alert.showAndWait();
		return;
	}
```

Αφου αποθηκευτεί το token του χρήστη, στέλνετε ένα request στον server το οποίο ζητάει να επιστραφούν τα δικαιώματα του χρήστη τα οποία αποθηκεύοντε σε μεταβλητές true or false(boolean) LogInController code:

```
new
GetPermissionsHttpRequest().run(Main.globalIP+"/api/permissions/getPermissionsByUsername",Us
ernameTxtField.getText());
```

Κώδικα που τρέχει μέσα στην ενέργεια run της κλάσσης GetPermisisonsHttpRequest:

```
String userResponse= response.body().string();

userPermissions userPermissions = gson.fromJson(userResponse, userPermissions.class);
if (userPermissions.getPermission1()==1) LogInController.permission1received=true;
if (userPermissions.getPermission2()==1) LogInController.permission2received=true;
if (userPermissions.getPermission3()==1) LogInController.permission3received=true;
client.dispatcher().executorService().shutdown();
```

Αρχικοποίηση λογικών μεταβλητών boolean στην κλάση LogInController:

```
public class LogInController {
    static Boolean permission1received = false;
    static Boolean permission2received = false;
    static Boolean permission3received = false;
```

Στην συνέχεια εμφανίζεται ένα παράθυρο στον χρήστη το οποίο του δίνει επιλογή προσβάσης στα παράθυρα της εφαρμογής. Με το που πατήσει το κουμπί μιας κατηγορίας του παραθύρου, γίνεται έλεγχος για το αν έχει ο χρήστης δικαίωμα πρόσβασης στο παράθυρο αυτό ελέγχοντας της μεταβλητές boolean που αναφέρονται πιο πάνω. Εξαίρεση αποτελεί το παράθυρο διαχείρισης χρηστών το οποίο, αντί για τις μεταβλητές, ελέγχει αν το όνομα χρήστη ισοδυναμεί με “ADMIN”} else {

```
    Main.UserName=UsernameTxtField.getText();
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle('Log in successful');
    alert.setHeaderText(null);
    alert.setContentText('Παρακαλώ επιλέξτε πως θα συνεχίσετε:');
    ButtonType buttonTypeOne = new ButtonType('Διαχείριση πελατών');
    ButtonType buttonTypeTwo = new ButtonType('Διαχείριση προϊόντων');
    ButtonType buttonTypeThree = new ButtonType('Διαχείριση ιστορικού παραγγελιών');
    ButtonType buttonTypeFour = new ButtonType('Διαχείριση δικαιωμάτων χρηστών');
    ButtonType buttonTypeCancel = new ButtonType('Cancel', ButtonBar.ButtonData.CANCEL_CLOSE);
    alert.getButtonTypes().setAll(buttonTypeOne, buttonTypeTwo, buttonTypeThree, buttonTypeFour, buttonTypeCancel);
    Optional<ButtonType> result = alert.showAndWait();
    if (result.get() == buttonTypeOne) {
        if (!permission1received) {//Αν ο χρήστης δεν έχει δικαιόμα εισόδου σε αυτό το
παράθυρο
            Alert AdminOnly = new Alert(Alert.AlertType.WARNING);
            AdminOnly.setTitle('WARNING');
            AdminOnly.setHeaderText(null);
            AdminOnly.setContentText('Δεν έχετε δικαίωμα εισόδου σε αυτό το παράθυρο.
Προσπαθήστε να μπείτε σε άλλο παράθυρο ή επικοινωνήστε με τον διαχειριστή.');
            AdminOnly.showAndWait();
        } else {
            // ... user chose 'Διαχείριση πελατών'
            Parent root = FXMLLoader.load(getClass().getResource('/sample/ClientManagement.fxml'));
            Stage primaryStage = new Stage();
            primaryStage.setTitle('Διαχείριση πελατών');
            primaryStage.setScene(new Scene(root, 800, 600));
            primaryStage.getIcons().add(new Image(Main.class.getResourceAsStream('vector4-removebg-
preview.png')));
            primaryStage.show();
            SPane.getScene().getWindow().hide();
        }
    } else if (result.get() == buttonTypeTwo) {
        // ... user chose 'Διαχείριση προϊόντων'
        if (!permission2received) {//Αν ο χρήστης δεν έχει δικαιόμα εισόδου σε αυτό το
παράθυρο
            Alert AdminOnly = new Alert(Alert.AlertType.WARNING);
            AdminOnly.setTitle('WARNING');
            AdminOnly.setHeaderText(null);
            AdminOnly.setContentText('Δεν έχετε δικαίωμα εισόδου σε αυτό το παράθυρο.
```

```
Προσπαθήστε να μπείτε σε άλλο παράθυρο ή επικοινωνήστε με τον διαχειριστή.');
                AdminOnly.showAndWait();
            } else {
                Parent root = FXMLLoader.load(getClass().getResource("/sample/ProductManagement.fxml"));
                Stage primaryStage = new Stage();
                primaryStage.setTitle("Διαχείριση προϊόντων");
                primaryStage.setScene(new Scene(root, 800, 600));
                primaryStage.getIcons().add(new Image(Main.class.getResourceAsStream("vector4-removebg-
preview.png")));
                primaryStage.show();
                SPane.getScene().getWindow().hide();
            }
        } else if (result.get() == buttonTypeThree) {
            // ... user chose "Διαχείριση ιστορικού παραγγελιών"
            if (!permission3received) {//Αν ο χρήστης δεν έχει δικαιώματα εισόδου σε αυτό το
παράθυρο
                Alert AdminOnly = new Alert(Alert.AlertType.WARNING);
                AdminOnly.setTitle("WARNING");
                AdminOnly.setHeaderText(null);
                AdminOnly.setContentText("Δεν έχετε δικαίωμα εισόδου σε αυτό το παράθυρο.
Προσπαθήστε να μπείτε σε άλλο παράθυρο ή επικοινωνήστε με τον διαχειριστή.');
                AdminOnly.showAndWait();
            } else {
                Parent root = FXMLLoader.load(getClass().getResource("/sample/OrderManagement.fxml"));
                Stage primaryStage = new Stage();
                primaryStage.setTitle("Διαχείριση Παραγγελιών");
                primaryStage.setScene(new Scene(root, 1152, 800));
                primaryStage.getIcons().add(new Image(Main.class.getResourceAsStream("vector4-removebg-
preview.png")));
                primaryStage.show();
                SPane.getScene().getWindow().hide();
            }
        } else if (result.get() == buttonTypeFour) {
            // ... user chose "Διαχείριση δικαιωμάτων χρηστών"
            if (!(UsernameTxtField.getText().equals("ADMIN"))) {//Αν ο χρήστης δεν είναι ο διαχειριστής
                Alert AdminOnly = new Alert(Alert.AlertType.WARNING);
                AdminOnly.setTitle("WARNING");
                AdminOnly.setHeaderText(null);
                AdminOnly.setContentText('Μόνο ο διαχειριστής έχει πρόσβαση σε αυτό το παράθυρο!');
                AdminOnly.showAndWait();
            } else {//Αν ο χρήστης είναι ο διαχειριστής
                Parent root = FXMLLoader.load(getClass().getResource("/sample/UsersManagement.fxml"));
                Stage primaryStage = new Stage();
                primaryStage.setTitle("Διαχείριση χρηστών");
                primaryStage.setScene(new Scene(root, 800, 600));
                primaryStage.getIcons().add(new Image(Main.class.getResourceAsStream("vector4-removebg-
preview.png")));
                primaryStage.show();
                SPane.getScene().getWindow().hide();
            }
        } else {
            // ... user chose CANCEL or closed the dialog
            return;
        }
```

```
}
```