ΕΠΙΚΟΙΝΩΝΙΑ ΑΝΘΡΩΠΟΥ ΜΗΧΑΝΗΣ

ΕΡΓΑΣΙΑ 3

Υλοποίηση τμήματος του ιστοχώρου του Υπουργείου Εργασίας και Κοινωνικών Υποθέσεων (https://www.ypakp.gr/)

ΜΕΛΗ:

ΑΝΤΥΠΑΣ ΙΟΡΔΑΝΗΣ(1115201700004)

ΓΙΑΝΝΙΟΣ ΝΙΚΟΛΑΟΣ(1115201700020)

**ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ**

1. ΕΙΣΑΓΩΓΗ

2. ΣΧΕΔΙΑΣΤΙΚΕΣ ΕΠΙΛΟΓΕΣ

3. DATABASE

4. ΕΝΔΕΙΚΤΙΚΟΙ ΧΡΗΣΤΕΣ ΕΦΑΡΜΟΓΗΣ

5. DOCKER

6. ΤΡΕΞΙΜΟ LOCALLY

7. BACKEND

8. FRONTEND

**ΕΙΣΑΓΩΓΗ**

Η εργασία υλοποιήθηκε με React στο frontend, Django στο backend και με βάση δεδομένων Postgresql, ενώ για το deployment χρησιμοποιήθηκε το docker-compose. Η εργασία ακολουθεί σε μεγάλο βαθμό τις προδιαγραφές τις εκφώνησης. Δεν έχει υλοποιηθεί το BONUS ερώτημα.

**ΣΧΕΔΙΑΣΤΙΚΕΣ ΕΠΙΛΟΓΕΣ**

Ο εργαζόμενος μπορεί να υλοποιήσει όλες τις φόρμες τις οποίες όμως πρέπει να εγκρίνει ο εργοδότης. Ο εργοδότης μπορεί να υλοποιήσει όλες τις φόρμες πέρα από την άδεια ειδικού σκοπού, ενώ μπορεί οποιαδήποτε στιγμή να εγκρίνει ή να απορρίψει φόρμες.

Οι χρήστες έχουν τη δυνατότητα να αλλάζουν πέρα από τα άλλα στοιχεία τους εταιρεία αλλά όχι username και ρόλο.

**DATABASE**

Για να γίνει import το αρχείο της ΒΔ που περιλαμβάνουμε αρκούν:

Δημιουργία της βάσης είτε από το terminal(psql) είτε με κάποιο database tool(πχ dbeaver):

CREATE ROLE qfsjgwmf WITH LOGIN PASSWORD 'fkSlNUsGtKhCc-ZGkY8\_oWXZJelaDdgA';

CREATE DATABASE test\_db;

Και στο terminal η εκτέλεση της εντολής:

pg\_restore -d test\_db sdi1700004.sql

Για να τρέξει η local db έναντι τις cloud db, σε περίπτωση που κάτι προκύψει με την online db που χρησιμοποιούμε αρκεί να μπει σε comment η πρώτη γραμμή του αρχείου backend/server/.env(DJANGO\_ENV=production -> // DJANGO\_ENV=production)

Σημείωση: Η χρήση local db στο docker θεωρήθηκε εκτός των πλαισίων της άσκησης. Οπότε το pg\_restore γίνεται μόνο με το να τρέξει το πρόγραμμα locally ενώ το docker γίνεται να συνδεθεί μόνο με τη online db.

**ΕΝΔΕΙΚΤΙΚΟΙ ΧΡΗΣΤΕΣ ΕΦΑΡΜΟΓΗΣ**

EMPLOYEE

{

"username": "ni87",

"password": "hardpassword"

},

EMPLOYER

{

"username": "ni878",

"password": "easypassword"

}

**DOCKER**

Για να τρέξει η εφαρμογή με το docker αρκούν οι εντολές:

docker-compose build --no-cache

docker-compose up --force-recreate --no-deps

Η εφαρμογή ακούει στο localhost:80. Σε περίπτωση που το port χρησιμοποιείται ήδη μπορεί να φανεί χρήσιμη η εντολή sudo /etc/init.d/apache2 stop

**ΤΡΕΞΙΜΟ LOCALLY**

Backend:

In backend/server:

linux: sudo python3 -m venv env / Windows: py -m venv env

linux: source env/bin/activate / Windows: .\env\Scripts\activate

pip install -r ../requirements.txt

python manage.py runserver

Frontend:

**BACKEND**

Για το authenticate των χρηστών χρησιμοποιείται JWT token.

API:

http://localhost:8000:

(register) POST: /authentication/register

example: {

"password": "hardpassword",

"email": "n7k@gmail.com",

"first\_name": "n7is",

"last\_name": "sdhj",

"amka": "1781",

"afm": "1871",

"username": "ni87",

"phone": "1127118",

"role": "employer",(employer/employee)

"has\_child\_under\_12": "True",(boolean)

"company": "UoA"

}

(login) POST: /authentication/api/token/:

{

"username": "nikos",

"password": "hardpassword"

}

(logout) POST: /authentication/logout/blacklist/

GET: /authentication/profile -> gets logged in user's profile

POST: /forms/adeiaeidikoyskopoy:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19",

"children": [{"age": 10, "school": "12o"}, {"age": 12, "school": "13o"}]

}

GET: /forms/adeiaeidikoyskopoy/<int:form\_id>

POST: /forms/adeiaeidikoyskopoy/<int:form\_id>

{

"status": -1 (-1: NOT\_APPROVED, 0: PENDING, 1: APPROVED)

}

POST: /forms/adeiaergasias:

IF USER IS EMPLOYER:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19",

"employee\_username": "admin2"

}

IF USER IS EMPLOYEE:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19"

}

GET: /forms/adeiaergasias/<int:form\_id>

POST: /forms/adeiaergasias/<int:form\_id>

{

"status": -1 (-1: NOT\_APPROVED, 0: PENDING, 1: APPROVED)

}

POST: /forms/anastoliergasias:

IF USER IS EMPLOYER:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19",

"employee\_username": "admin2"

}

IF USER IS EMPLOYEE:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19"

}

GET: /forms/anastoliergasias/<int:form\_id>

POST: /forms/anastoliergasias/<int:form\_id>

{

"status": -1 (-1: NOT\_APPROVED, 0: PENDING, 1: APPROVED)

}

POST: /forms/tilergasia:

IF USER IS EMPLOYER:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19",

"employee\_username": "admin2"

}

IF USER IS EMPLOYEE:

{

"from\_time": "2021-01-17",

"to\_time": "2021-01-19"

}

GET: /forms/tilergasia/<int:form\_id>

POST: /forms/tilergasia/<int:form\_id>

{

"status": -1 (-1: NOT\_APPROVED, 0: PENDING, 1: APPROVED)

}

GET: /forms/myforms -> gets logged in user's forms

POST: /authentication/update\_profile

{

"email": "ok@gmail.com",

"first\_name": "n7is889",

"last\_name": "sdhj880",

"amka": "1781880",

"afm": "1871880",

"phone": "112711888",

}

POST: /authentication/update\_password

{

"old\_password": "hardpassword",

"new\_password": "easypassword",

"new\_rpassword": "easypassword"

}

POST: /appointments/create\_appointment:

IF NOT LOGGED IN

{

"email": "nikos@gmail.com",

"phone": "1",

"date": "2021-01-31",

"time": "18:00"

}

IF LOGGED IN

{

"date": "2021-01-31",

"time": "18:00"

}

GET: /appointments/myappointments

GET: /appointments/appointment\_by\_date

{

"date": "2020-01-11"

}

**FRONTEND:**