Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky

Bc. Dávid Balla, Bc. Ján Branovič, Bc. Adam Kačo, Bc. Erik Takáč

Systém pre lekárov

Zadanie 4

Predmet: Úvod do počítačovej bezpečnosti Ak. Rok: 2023/2024

Contents

1 Základné informácie o projekte

Riešenie sme sa rozhodli postaviť na programovacom jazyku Python verzie 3.8 a vyššie, pričom využívame knižnicu flask a mnohé ďalšie na implementáciu jednotlivých súčastí programu a bezpečnostných aspektov. Aplikácia je postavená na technológií Docker a teda stačí ju rozbehať skrz docker-compose, ktorý vytvorí všetky jej potrebné súčasti a napojenie na databázu. Samotná **aplikácia**/aplikačný server beží na **porte 3000**, **databáza** na štandardnom **porte 5432** pre postgres.

Pre ukladanie informácii o používateľoch a ich aktivite pouívame postgresql databázu, ktorú máme rozdelenú na 3 základné tabuľky:

- Users
- Message
- Pacients
- MedicalRecord

Tabuľka users vyzerá nasledovne:

Atribút	**Typ**
Id	Integer
Username	String (64)
Email	String (128)
Password	String (246)
Salt	String (128)
dateRegistered	Integer
patients	ArrayList

V rámci tabuľky messages si držime iba jej id ako integer a samotný obsah správy ako String.

Do projektu sme v rámci tohto zadania pridali tabuľky Pacients a MedicalRecord. Tabuľka pacients má nasledovné atribúty:

```
id = db.Column(db.Integer, primary_key=True)
user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)
firstName = db.Column(db.String(246), nullable=False)
lastName = db.Column(db.String(246), nullable=False)
birthDate = db.Column(db.Date, nullable=False)
address = db.Column(db.String(246), nullable=False)
phoneNumber = db.Column(db.String(20), nullable=False)
email = db.Column(db.String(246), nullable=False)
gender = db.Column(db.String(246), nullable=False)
height = db.Column(db.Float, nullable=False)
weight = db.Column(db.Float, nullable=False)
bloodType = db.Column(db.String(246), nullable=False)
allergies = db.Column(db.String(246), nullable=False)
healthInsurance = db.Column(db.String(246), nullable=False)
medicalRecords = db.relationship('MedicalRecord', backref='patient')
```

Obrázok 1 - Atribúty tabuľky pacients

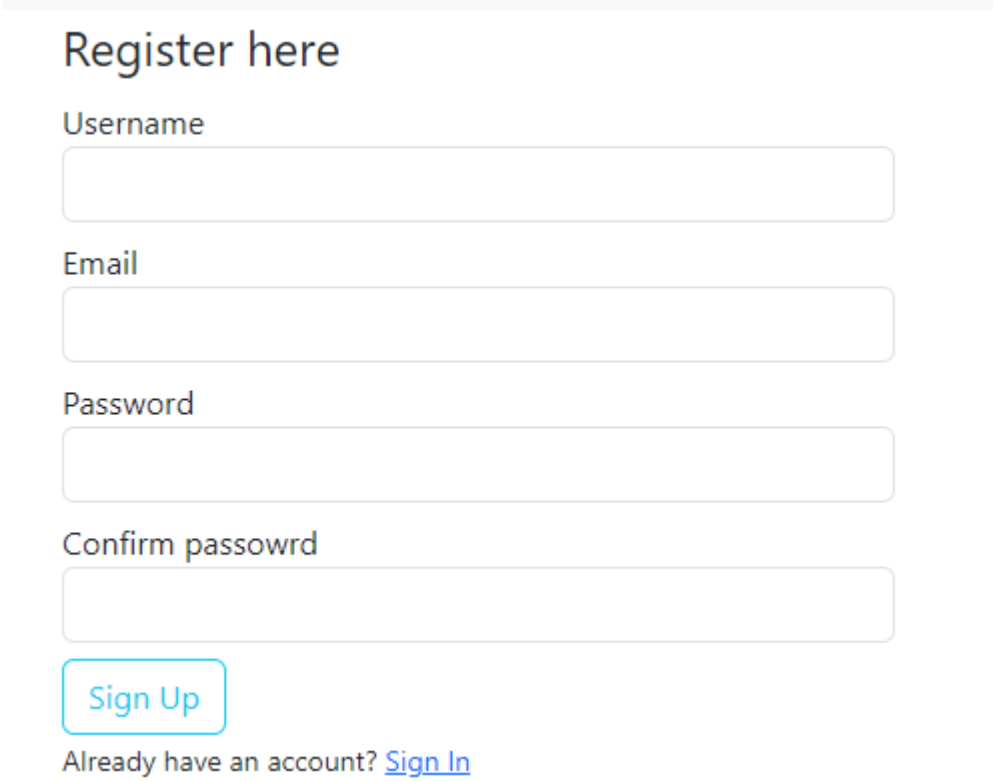
```
class MedicalRecord(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    patient_id = db.Column(db.Integer, db.ForeignKey('patients.id'))
    medical_record = db.Column(db.Text, nullable=False)
    timestamp = db.Column(db.DateTime, default=datetime.utcnow)
```

Obrázok 2 - Atribúty tabuľky MedicalRecord

2 Prihlasovanie a registrácia používateľa

Táto časť sa nám voči **predošlému** odovzdaniu nezmenila a **ostala pôvodná**.

2.1 Registrácia nového používateľa



Registrácia používateľa je aplikovaná samostatným GUI rozhraním ako aj API volaním s nahratím potrebných dát skrz metódu POST vo formáte JSON.

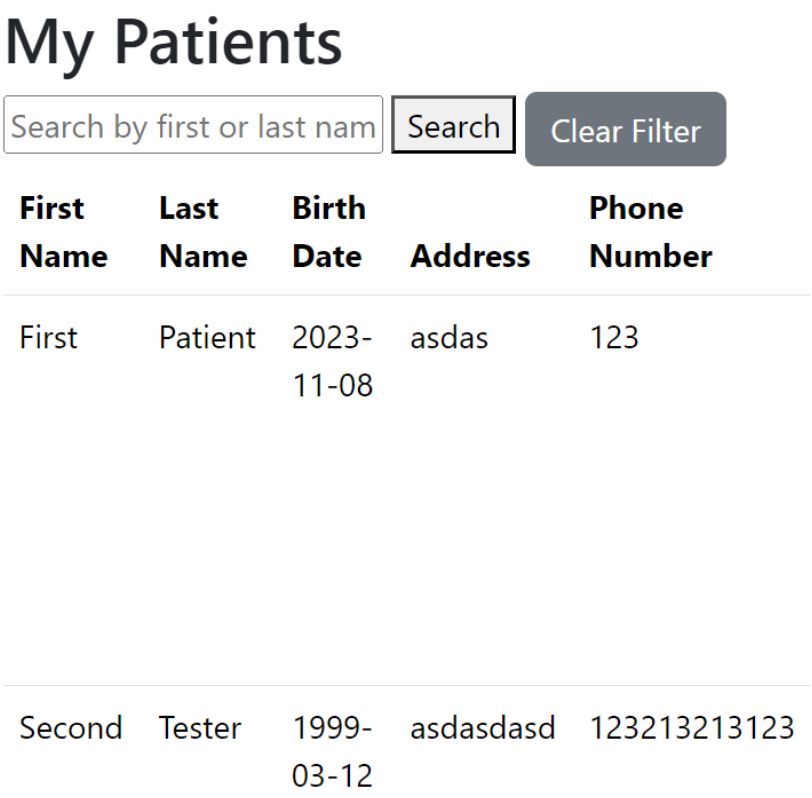
Pri registrácii je od používateľa požadované zadať všetky údaje vo formulári. Následne aplikácia dané dáta skontroluje, overí, či sa v databáze používateľ s rovnakým mailom nenachádza – ten musí byť unikátny.

Po zadaní dát a obzvlášť hesla sa skontroluje, či heslo spĺňa základené **bezpečnostné požiadavky** – musí obsahovať minimálne **8 znakov**, z nich musí byť aspoň jeden znak **číslo** a ďalší **veľké písmeno** abecedy.

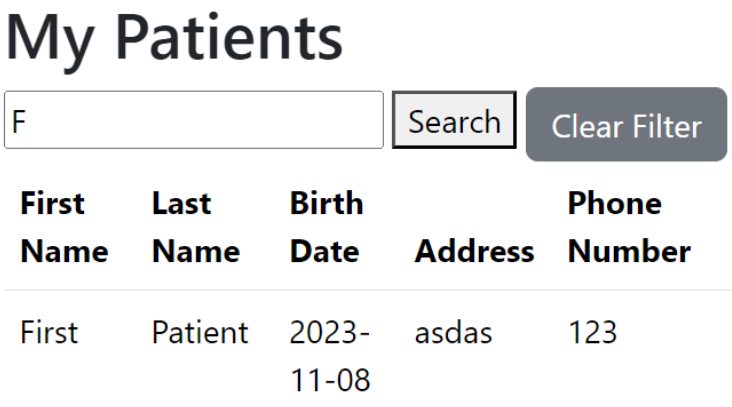
Po kontrole sa dáta zahashujú, pridá sa k heslu salt a uložia sa spoločne do databázy prepoužívateľa.

3 Vyhľadávacie pole

Lekár ma po prihlásaní možnosť vidieť svojich pacientov. V rámci nich dokáže vyhľadávať pomocou vyhľadávacieho poľa.



Obrázok 3 - Vyhľadávacie pole



Obrázok 4 - Aplikácia filtra vo vyhľadávacom poli

Popis fungovania vyhľadávacieho poľa:

1. `search_query = request.args.get('search_query', '')`: Táto časť kódu získa hodnotu `search_query` zo vstupu používateľa. Predpokladá sa, že sa táto hodnota získa cez HTTP GET parameter s názvom `'search_query'`. Ak parameter nie je k dispozícii, nastaví sa predvolená hodnota na prázdny reťazec `''`.

2. `if current_user.is_authenticated:`: Podmienka overuje, či je používateľ prihlásený do aplikácie. Používa sa tu predpokladaná premenná `current_user`, ktorá by mala obsahovať informácie o aktuálne prihlásenom používateľovi.

3. Ak je používateľ prihlásený (`if current_user.is_authenticated`), nasleduje vyhľadávanie pacientov: `patients = Patients.query.filter(...)`: Táto časť kódu používa SQLAlchemy na vyhľadanie pacientov podľa zadaného `search_query`. Používa sa metóda `filter`, ktorá hľadá záznamy v tabuľke `Patients`. Vyhľadávanie prebieha podľa podmienky, ktorá zahŕňa buď `firstName` alebo `lastName` pacienta, pričom musia spĺňať podmienku `ilike` (case-insensitive porovnanie) so zadaným `search_query`. Okrem toho sa tiež overuje, že pacient patrí k aktuálnemu prihlásenému používateľovi pomocou podmienky `Patients.user_id == current_user.id`.

4. `).all()`:Kód načíta všetkých nájdených pacientov zo databázy a uloží ich do premennej `patients`.

5. `return render_template('view_patients.html', patients=patients)`: Ak sú pacienti úspešne nájdení, funkcia vráti vygenerovanú HTML stránku s názvom `'view_patients.html'` a prepojením na dáta pacientov v premennej `patients`.

6. `else: return redirect(url_for('login'))`: Ak používateľ nie je prihlásený, presmeruje sa na stránku prihlásenia (`'login'`).

Vyhľadávanie funguje tak, že prihlasovaný používateľ môže vyhľadávať pacientov na základe ich mena alebo priezviska a výsledky vyhľadávania sú obmedzené na pacientov priradených k aktuálne prihlásenému používateľovi.

4 Používateľská príručka

- Doplníme v ďalšom odovzdaní

Používanie aplikácie je pre lekára jednoduché a intuitívne, všetky potrebné súčasti ako aj vytváranie záznamov vidí po prihlásaní do aplikácie a dokáže ich jednoducho ovládať.