Továbbfejlesztési lehetőségek:

Relációs adatbázis használata: Jelenleg az alkalmazás in-memory adatbázist (H2) használ. Termelési környezetben célszerű lehet áttérni egy stabil, skálázható relációs adatbázisra, például MySQL, PostgreSQL stb. Ez javítja az adatok tartósságát és a teljesítményt.

Tranzakciókezelés: A tranzakciók bevezetése fontos adatintegritás és konzisztencia biztosítása szempontjából. Különösen akkor fontos, ha több műveletet kell atomi módon végrehajtani (pl. ügyfél regisztráció és pozíció létrehozása együtt).

Naplózás: Annak érdekében, hogy könnyen lehessen nyomon követni az alkalmazás teljesítményét és hibáit.

Szálkezelés és aszinkron feldolgozás: Ha az alkalmazásnak nagy terhelésnek kell ellenállnia, érdemes lehet bevezetni szálkezelést és aszinkron feldolgozási mechanizmusokat.

Tesztek használata: Hibák korai felfedezése, kódminőség és megbízhatóság növelése.

Docker használata: Az alkalmazás és környezetének izolált futtatása konténerekben. Megkönnyíti az alkalmazás telepítését, hordozhatóságát, és segít a fejlesztőknek, hogy azonos környezetben dolgozzanak.

Frontend megvalósítása: Modern frontend technológiák (pl. Angular, stb.) használata segíthet dinamikus, reszponzív felhasználói élményt biztosítani.

Alkalmazás leírása:

Ez az alkalmazás egy álláskereső rendszer, amely lehetővé teszi ügyfelek regisztrációját, új pozíciók létrehozását, keresését és megtekintését.

Főbb funkciók:

- **Regisztráció**: Új ügyfelek regisztrálása név és email cím alapján. A regisztráció után az ügyfél egy egyedi API kulcsot kap.
- **Pozíció létrehozása**: Új pozíciók hozzáadása megnevezés és helyszín alapján. Csak érvényes API kulccsal rendelkező ügyfelek hozhatnak létre pozíciókat.
- **Pozíció keresése**: Pozíciók keresése cím és helyszín alapján. Csak érvényes API kulccsal rendelkező ügyfelek kereshetnek pozíciókat. Biztosított a részleges egyezés szerinti keresés is.
- **Pozíció megtekintése**: Egy adott pozíció megtekintése azonosító alapján. Csak érvényes API kulccsal rendelkező ügyfelek tekinthetnek meg pozíciókat.

Alkalmazás konfigurálása és futtatása:

1. Maven használata

Az alkalmazás Maven projekt, így szükséges a Maven a függőségek kezeléséhez és a projekt futtatásához. Ha nincs Maven telepítve, telepítse azt: https://maven.apache.org/install.html

2. Függőségek letöltése

A következő parancs letölti a szükséges függőségeket: mvn clean install

3. Alkalmazás futtatása

Az alkalmazás futtatásához használja a következő parancsot: mvn spring-boot:run

Ez a parancs elindítja az alkalmazást a beépített Tomcat szerveren a localhost:8080 címen.

4. H2 Konzol használata

Az alkalmazás H2 adatbázist használ, amelynek konzolja elérhető a következő címen: http://localhost:8080/h2-console

A bejelentkezési adatok az application.properties fájlban találhatók:

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.username=sa

spring.datasource.password=password

5. OpenAPI Dokumentáció

Az API dokumentáció elérhető a következő címen: http://localhost:8080/swagger-ui-custom.html

Ez a dokumentáció interaktív felületet biztosít az API teszteléséhez és használatához.

6. API végpontok

Regisztráció

POST /client

- **Név**: name (String, kötelező)
- **Email**: email (String, kötelező)

Pozíció létrehozása

POST /position

- **Cím**: title (String, kötelező)
- **Helyszín**: location (String, kötelező)
- **API kulcs**: apiKey (String, kötelező)

Pozíció keresése

GET /position/search

- **Cím**: title (String, kötelező)
- **Helyszín**: location (String, kötelező)
- **API kulcs**: apiKey (String, kötelező)

Pozíció megtekintése

GET /position/{id}

- **Azonosító**: id (UUID, kötelező)
- **API kulcs**: apiKey (String, kötelező)

7. Tesztelés

A projekt tartalmaz tesztelési függőségeket is. A teszteket a következő parancs futtatja: mvn test

Ez a parancs lefuttatja az összes egység- és integrációs tesztet.