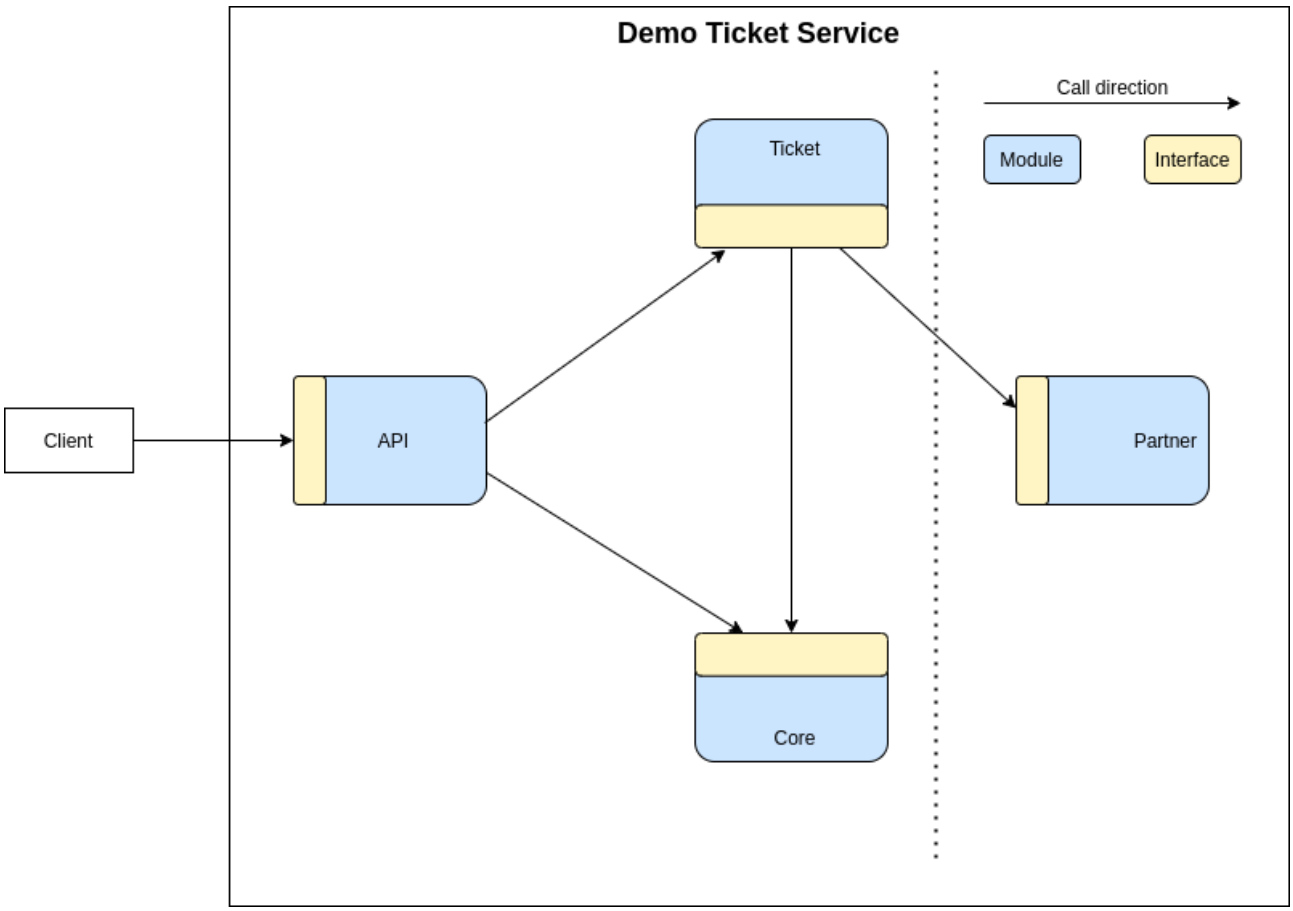
Jegy szolgáltatás

Felhasznált eszközök

A demo applikációt Java-ban, a Spring Boot framework felhasználásával készítettem el. Fejlesztői környezetként az IntelliJ Idea-t használtam, az API-k teszteléséhez pedig a Postman alkalmazást. Az adatbázis MySQL alapon nyugszik, ennek menedzselésére a Dbeaver-t használtam. A dependenciák és a build folyamat kezelését Maven segítségével oldottam meg.

Kialakított modul struktúra

A feladatnak megfelelően 4 modult hoztam létre, a modulok között a kommunikáció HTTP-n folyik, így mindegyik modul természetesen saját HTTP szervert futtat. A Core modul egy egyszerű SpringMVC interfészt valósít meg. A másik három modul interfészét OpenAPI leíróval definiáltam, ezekből a leírókból a modulok generálják maguknak a megfelelő forráskódokat.



1. Core modul

A modul ellátja az adatbázissal kapcsolatos feladatokat, a felhasználók adatainak kezelését. Lehetővé teszi az auth tokenek validálását, valamint a felhasználó bankkártya adatainak lekérését (a kártya a felhasználóhoz tartozik-e, van-e elég fedezet). A kártyaadatok kezelésében eltértem egy kicsit a feladatban meghatározottaktól, és megengedhetőnek találtam, hogy több felhasználóhoz ugyanaz a kártya tartozzon. Ilyen előfordulhat például akkor, ha többen használnak egy céges bankkártyát. Ezt egy kapcsolótábla bevezetésével oldottam meg, amely összeköti a felhasználókat a kártyával.

2. Partner modul

A modul információt szolgáltat az elérhető eseményekről.
A Ticket modul autentikációját egy API kulcs segítségével oldottam meg, ez a kulcs mindkét alkalmazásban konfigurációs opció. Természetesen normál működés mellett ezt a kulcsot a Partner modul kéne hogy szolgáltassa valamilyen adminisztrációs felületen, adatbázisban tárolva az egyes felhasználási esetekhez rendelt kulcsokat.
Az 1-es id-jű esemény start-, illetve endTimeStamp-jét módosítottam, hogy az a jövőben legyen, ezáltal tudjam tesztelni a megfelelő funkcionalitást.
Az egyes eseményekre való szék foglalásokat az egyszerűség miatt nem rögzítettem, természetesen ezeknek az adatoknak is egy adatbázisban lenne a helyük (az események és azok adataival együtt).

3. Ticket modul

Ez a komponens implementálja a szék foglaláshoz szükséges logikát, az eseményeket és azok adatait jóformán egy az egyben csak továbbítja a Partner modultól az API modul felé.
A foglalás során ellenőrzésre kerül, hogy a felhasználó megfelelő adatokat adott-e meg, létezik-e az esemény, létezik-e a szék, foglalt-e már a szék, elkezdődött-e már az esemény, valamint, hogy van-e elég fedezet a kártyáján. Amennyiben előzetesen minden feltétel teljesül, akkor a jegy ára zárolásra kerül és egy kérést küldünk a Partner modul felé, hogy lefoglaljuk az adott helyet. Amennyiben a kérés nem sikerül, akkor a zárolt összeg visszakerül a felhasználó kártyájára. Természetesen ez nem egy tökéletes megoldás tranzakciókezelés gyanánt, viszont egy demo rendszerhez ezt elégségesnek találtam. Egy kifinomultabb verzióban külön kellene tárolni az egyes pénzügyi tranzakciókat, valamint lehetőséget kellene biztosítani a tranzakciók sztornózására.

4. API modul

A kliensek számára elérhetővé teszi a rendszer különböző szolgáltatásait. A Ticket modulhoz intézett hívások előtt validálja, hogy a felhasználó által megadott User-Token érvényes-e- A Ticket modullal csak akkor veszi fel a kapcsolatot, ha a token valósnak bizonyult.

Fordítás és futtatás

Az egyes modulokat az `mvn clean install -pl module` parancs használatával buildeltem, majd az `mvn spring-boot:run -pl module` paranccsal indítottam őket. Az egyes modulok nevei: core, api, ticket, partner
Az alap konfigurációban a MySQL szerver a localhost 3306-os portján futott, a használt adatbázis neve ticketservice-core, az adatbázis szerverre a belépéshez a root felhasználót használtam root jelszóval. A Core modul által szolgáltatott interfész a 8888-as porton elérhető, az API modul a 9999-es porton fut, a Ticket és Partner modulok pedig a 11111 és 12222 porton érhetők el.