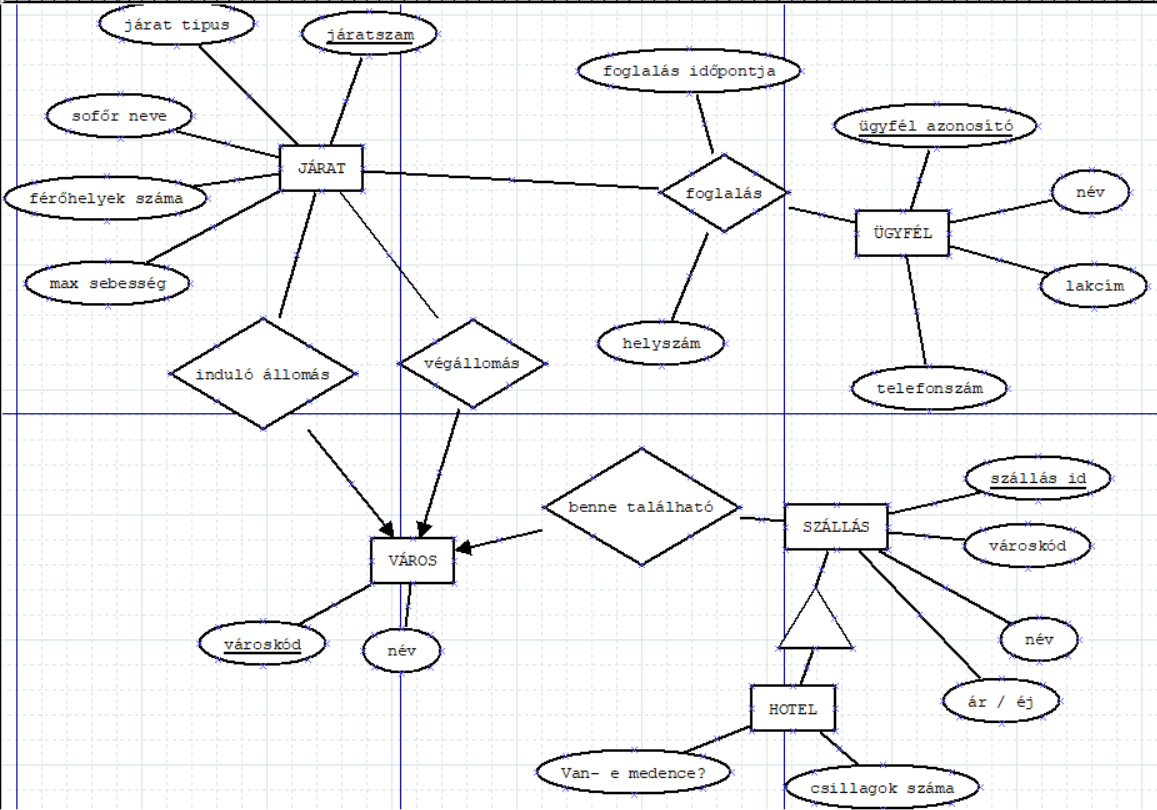
Busz, Vonat, Repülő

Jegyfoglalás

Adatbázisok kötelező feladat

A jegyfoglalás webapplikáció járatok, szállások listázására szolgál. Az adatbázisban szereplő (vagy felvitt) ügyfelek foglalhatnak jegyet az elérhető járatokra. Szállásokat kereshetnek az elérhető városokban.

**Egyed-kapcsolat modell**



Az adatmodellben 5 darab egyed található:

* Járat (5 attribútum, 3 kapcsolat= 2db 1: N, 1db N:M)
* Ügyfél (4 attribútum, 1 kapcsolat N:M)
* Város (2 attribútum, 3 kapcsolat 3db 1: N)
* Szállás (4 attribútum, 2 kapcsolat 1db 1: N, 1 db specializáló kapcsolat)
* Hotel (speciális egyed, 2 db attribútum, 1 db specializáló kapcsolat)
* Az Ügyfél és a Járat egyedet a foglalás kapcsolat köti össze. Ennek két plusz attribútuma van.
* A Város és Járat egyedet az Induló állomás és végállomás kapcsolat köti össze.
* A Szállás és Város egyedet a benne található kapcsolat köti össze.
* A Hotel egyed az egy speciális Szállás.

**Relációs adatbázisséma**

ÜGYFÉL(ügyfél azonosító, név, lakcím, telefonszám)

JÁRAT(járatszám, járat típusa, sofőr neve, férőhelyek száma, max sebesség, *induló állomás.városkód*, *végállomás.városkód*)

VÁROS(városkód, név)

FOGLALÁSOK(*járatszám, ügyfélazonosító*, foglalás időpontja, helyszám)

SZÁLLÁS(szállás id, *városkód,* név, ár/éj)

HOTEL(*szállás id*, csillagok száma, van-e medence)

**Normalizálás**

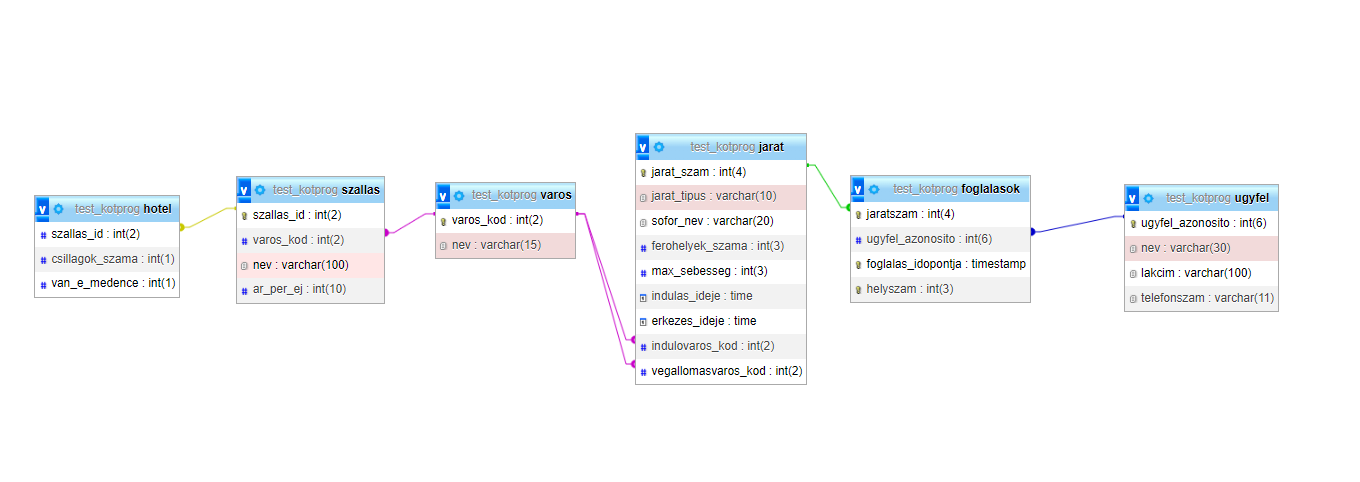
**2NF:**

* Az ÜGYFÉL séma 2NF-ben van, mert egyetlen kulcsa az ügyfél azonosító, és ez ettől minden másodlagos attribútum függ.
* A JÁRAT séma 2NF-ben van, mert egyetlen kulcsa a járatszám, és ez ettől minden másodlagos attribútum függ.
* A VÁROS séma 2NF-ben van, mert egyetlen kulcsa az ügyfél azonosító, és ez ettől minden másodlagos attribútum függ.
* A FOGLALÁSOK séma 2NF-ben van, mert egyetlen másodlagos attribútuma sincs.
* A SZÁLLÁS séma 2NF-ben van, mert a {név, ár/éj} halmaz függ a {szállás id, városkód} kulcsoktól.

**3NF:**

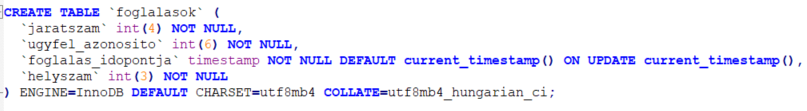
* Az ÜGYFÉL séma 3NF-ben van, mert nincs benne tranzitív függés.
* A JÁRAT séma 3NF-ben van, mert nincs benne tranzitív függés.
* A VÁROS séma 3NF-ben van, mert nincs benne tranzitív függés.
* A FOGLALÁSOK séma 3NF-ben van, mert nincs benne tranzitív függés.
* A SZÁLLÁS séma 3NF-ben van, mert nincs benne tranzitív függés.

**Táblatervek**

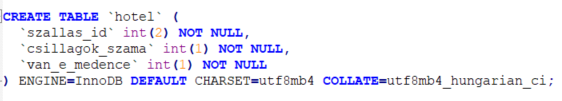


SQL:

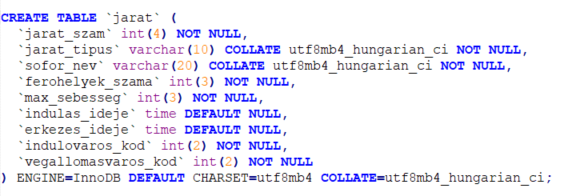
Foglalások:



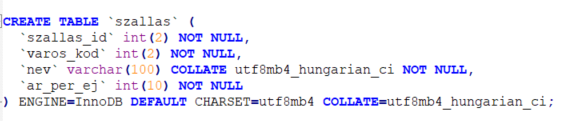
Hotel:



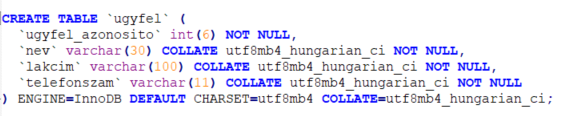
Járat:



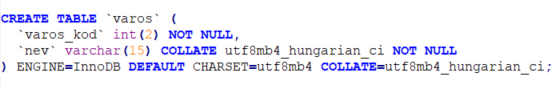
Szállás:



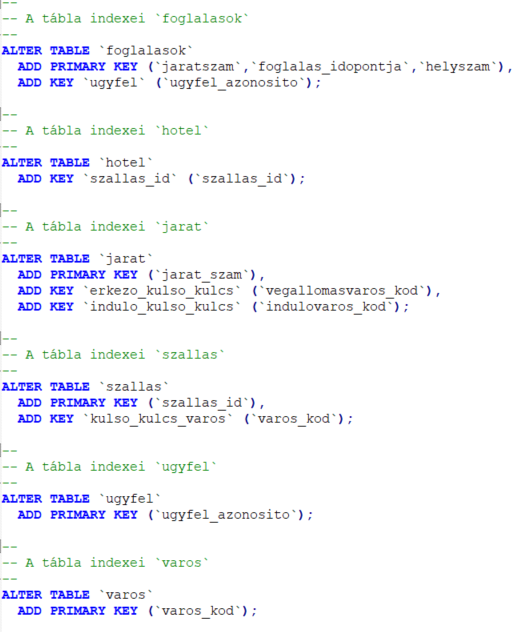
Ügyfél:



Város:



Kulcsok:

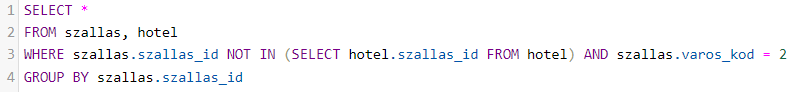




**Összetett lekérdezések**

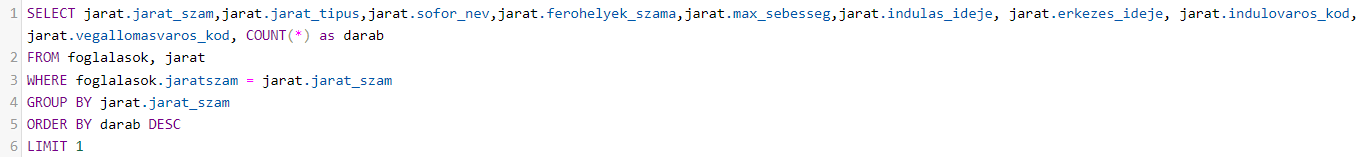
1. lekérdezés:

A szállás oldalon ki lehet választani, hogy apartmant vagy hotelt keresünk. Mivel csak egy szállás tábla van, és annak egy hotel speciális egyede, így a hotel táblában nem tárolom el csak azoknak a szállásoknak az azonosítóját, amik hotelek is egyben. Ezek mellett még tárolom a hotelekről a csillagok számát és azt, hogy van-e medence. A Szállások oldalon ezzel a lekérdezéssel tudom visszaadni azokat a szállásokat, amik hotelek, vagy nem.



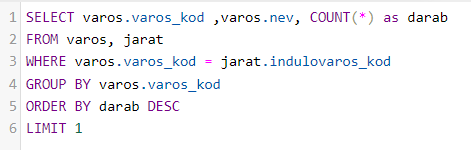
(Most a kettes város kóddal rendelkező városban keresi csak az apartmanokat)

1. lekérdezés:

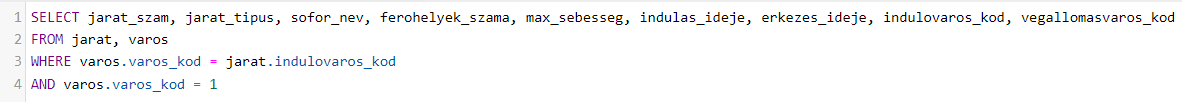
Ennek a lekérdezés segítségével vissza tudom adni azt a járatot amire a legtöbb ember foglalt jegyet. Majd ennek a járatnak az adataik kiíratom a Foglalások oldalon.

1. lekérdezés:

Ez a lekérdezés megnézi, hogy melyik városból indul a legtöbb járat.

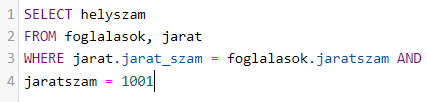


1. lekérdezés:

Ezzel meg tudom nézni, hogy specifikus városból mely járatok indulnak. Ezt a Járatok oldalon használom.

1. lekérdezés:

Ezzel meg tudom nézni, hogy egy adott járaton mely helyek foglaltak, így mikor új foglalás érkezik az adott járatra, azt a helyet nem osztom ki addig amíg foglalt.



**Megvalósítás**

A program megvalósításához a Java nyelvet használtam (JDK 17). A JetBrains IDE-jét használtam az IDEA Intellij-t. Spring boot volt az adatbázis kapcsolatot segítő eszköz. Az adatbázis és a program összekötését a DAO osztályokban valósítottam meg, itt a Spring Boot-ba beépített JdbcDaoSupport volt segítségemre. Az SQL kódokat nem generálta le előre, így a programba nekem kellet felvinni a megfelelő SQL kódokat. A megjelenítést HTML+ CSS segítségével oldottam meg. A programot és a HTML oldalakat a Thymeleaf sablonmotor segítségével kötöttem össze. Néhány formázást online Bootstrap css segítségével oldottam meg.

**Funkciók**

Főoldal:

A főoldalon csak egy egyszerű üdvözlő szöveg van. DE a „weboldalon” szövegre kattintva az admin oldalra jutunk. Az admin oldalon vannak azok a dolgok, amik egy alap felhasználó nem férhet hozzá. Pl.: Járatok hozzáadása, törlése…

Ügyfél oldal:

Az ügyfél oldalon tudunk hozzáadni egy ügyfelet az adatbázishoz. Mikor hozzáadjuk az ügyfelet, akkor megjelennek az adatai a lentebb található táblázatban. Ebben a táblázatban tudjuk törölni az adott felhasználót a „Törlés” gombbal, a „Szerkesztés” gombbal meg az adott felhasználó adatai belekerülnek a szerkesztés oldalon található form-ba. Itt tudjuk szerkeszteni az ügyfelünket.

Járatok oldal:

A járatok oldalon ki tudjuk listázni az adott városból induló járatokat. A válassz egy várost legördülő menüben ki tudjuk választani az adatbázisban szereplő városokat. (Ha az admin új várost ad az adatbázishoz, akkor ez a lista is frissül). Egy várost kiválasztva majd az „Elküld” gombra kattintva, megjelennek az adott városból induló járatok adatai. A „Legtöbb járat innen indul:” szöveg alatt annak a városnak a neve jelenik meg amiből a legtöbb járat indul. (Ha új járatokat adunk hozzá az adatbázishoz, ami más városból indul akkor mindig az jelenik meg ahonnan a legtöbb indul).

Foglalások oldal:

A foglalások oldalon két legördülő listát látunk. Az egyikben az adatbázisban szereplő ügyfelek vannak, a másikban meg az adatbázisban szereplő járatok. A listákból kiválasztva az adott ügyfelet és járatot, majd megnyomva az „Elküld” gombot tudunk foglalást leadni. Ez megjelenik a lentebb található listában. Ha egy ügyfél több járatot is foglalt akkor a foglalásai egymás alatt jelennek meg, név szerint csoportosítva. Az ügyfel tudja törölni a foglalását a foglalása sorában levő „Törlés” gombal. (Az oldalon szintén frissülnek a lenyíló fülek az adatbázis frissítése esetén)

Szállások oldal:

A szállások oldalon a „Szállások listázása” szöveg alatt ki tudjuk választani azt a várost, ahol ki szeretnénk listázni a szállásokat. A lenyíló fül alatt rádió gombokkal ki tudjuk választani, hogy apartmant vagy hotelt akarunk keresni az adott városban. Ha az apartmant választjuk akkor az adott városból kilistázza csak az apartmanokat. (Azokat a szállásokat, amik hotelek is egyben, azokat nem). Ha a hotel rádió gombot válasszuk akkor az alatta található lenyíló menüben ki kell választanunk, hogy hány csillagos hotelt keresünk, majd alatt a rádió gombokkal meg kell adnunk, hogy olyan szállást keressen, ahol van medence. Ha találat van akkor a jobbra található táblázatban megjelennek a szállás/szállások adatai. Ha nincs találat, akkor nem jelenik meg semmi. (Mivel a hotelek keresésénél nagyon sok kombináció lehet így pár példa: Gyula, Hotel, csillagok száma=4, van e medence = van; Kecskemét, Hotel, csillagok száma=2, van e medence= nincs; Barcelona, Hotel, csillagok száma = 5, van e medence = van) Ha apartmant keresünk akkor nem kell ezeket a dolgokat elvégeznünk, csak kiválasztani a várost, majd az „Elküld” gomb megnyomása után kilistázza a városban található apartmanokat.

Admin oldal:

Mint már említettem a főoldalon a „weboldalon” szövegre kattintva érhetjük el.  
Funkciók:

* Járat törlés: Járatszám alapján tudjuk törölni a járatokat.
* Járat hozzáadás: Meg kell adnunk a feltüntetett adatokat:
  + Járat szám: int, az adott járat száma, egyedi, 1-9999 között
  + járat típus: string
  + sofőr neve: string
  + férőhelyek száma: int
  + max sebesség: int
  + indulás ideje: time, formátum: 00:00:00
  + érkezés ideje: time, formátum: 00:00:00
  + induló város: legördülő lista
  + végállomás: legördülő lista
* Járat szerkesztése:
  + járat szám: legördülő lista
  + új járat típus: string
  + új sofőr neve: string
  + új férőhelyek szama: int
  + új max sebesség: int
  + új indulás ideje: time, formátum: 00:00:00
  + új érkezés ideje: time, formátum: 00:00:00
  + új induló város: legördülő lista
  + új végállomás: legördülő lista
* Foglalás szerkesztése:
  + ügyfél: lista
  + foglalás időpontja: mikor történt a foglalás (ez a foglalások oldalon látszódik, ugyan abban a formátumban):
  + volt helyszám (ez a foglalások oldalon látszódik): int
  + új helyszám: int
* Szállás hozzáadása:
  + város: mely városban található: lista
  + név: string
  + ár: int
* Szállás törlése:
  + a szállás neve alapján tudjuk törölni a szállást
* Szállás frissítése:
  + szállás név: melyik szállást akarjuk frissíteni: lista
  + új név: string
  + új ár: int
* Hotelek listázása:
  + külön kilistáztam az adatbázisban található olyan szállásokat, amik hotelek is egyben, ha véletlen a sok lehetséges kombináció miatt nem lenne találat a Szállások oldalon, a hotelek keresésénél.
* Hotel hozzáadása:
  + név: azok a szállások, amik nem hotelek: lista
  + van e medence: 0-1 között, 0 ha nincs, 1 ha van: int
  + csillagok száma: 1-5 között, int
* Hotel törlése:
  + szállás törlése a hotelek közül név alapján: lista
* Hotelek szerkesztése:
  + hotel neve: lista
  + van e medence (új): 0-1 között, 0 ha nincs, 1 ha van: int
  + csillagok száma (új): 1-5 között, int
* Város hozzáadása:
  + név: string
* Város törlése:
  + város törlése név alapján: lista
* Város szerkesztése:
  + melyik város, név szerint: lista
  + város új neve: string

A külső kulcs kapcsolatok miatt egyes táblák módosítása befolyásolnak más táblákat is. PL.: Ha törlünk egy járatot, akkor a járatra érkezett foglalások is törlődnek. Vagy ha törlünk egy várost akkor a oda érkező és onnan induló járatok is törlődnek. Tábla Update-nál is működnek ezek a funkciók.