ZÁRÓDOLGOZAT

Vince Patrik

2021

SZÁMALK-Szalézi
Technikum és Szakgimnázium

Szállodai szobafoglalás

Vince Patrik

SZÁMALK-Szalézi Technikum és Szakgimnázium

Szoftverfejlesztő

Konzulens: Kupcsikné Fitus Ilona

SZÁMALK-Szalézi
Technikum és Szakgimnázium

Nyilatkozat

a záródolgozat eredetiségéről

Alulírott Szöveg beírásához kattintson ide. (név) {Szöveg beírásához kattintson ide. (anyja neve) Szöveg beírásához kattintson ide. (szem. ig. szám)} büntetőjogi és fegyelmi felelősségem tudatában kijelentem és aláírásommal igazolom, hogy a záródolgozat saját munkám eredménye. A felhasznált irodalmi és egyéb információs forrásokat az előírásoknak megfelelően kezeltem, a záródolgozat készítésre vonatkozó szabályokat betartottam.

Kijelentem, hogy ahol mások eredményeit, szavait vagy gondolatait idéztem, azt a záródolgozatomban minden esetben, beazonosítható módon feltüntettem, a dolgozatban található fotók és ábrák közlésével pedig mások szerzői jogait nem sértem.

Kijelentem, hogy a záródolgozatom elektronikus változata teljes egészében megegyezik a nyomtatott formával.
Hozzájárulok ahhoz, hogy az érvényben lévő jogszabályok és a Számalk-Szalézi Szakgimnázium belső szabályzata alapján az iskola saját könyvtárában megtekinthető (olvasható) legyen a záródolgozatom.

A záródolgozat titkos/<u>nem titkos.</u>

Budapest, 2021. április 05.

.....

Tanuló aláírása

Tartalomjegyzék

1. Bevezetés 3
2. Felhasználói dokumentáció 4
2.1 Rendszerkövetelmények 4
2.2 Menürendszer 4
2.3 Kezdőlap 5
2.4 Regisztráció 6
2.5 Bejelentkezés 6
2.6 Foglalásaim 7
2.7 Foglalás lemondása 8
2.8 Kijelentkezés 8
2.9 Admin 8
2.10 Foglalás törlése 10
3. Fejlesztői dokumentáció 11
3.1 Követelmények 11
3.2 Adatbázis 11
3.2.1 Tervezés 11
3.2.2 Implementáció 13
3.3 Weboldal 16
3.3.1 Felület terv 16
3.3.2 Implementáció 17
3.4 Tesztelés 26
4. Továbbfejlesztés 29
5. Összefoglalás 30
Irodalomjegyzék 31

Ábrajegyzék

1. ábra Bejelentkezés nélkül 4
2. ábra: Felhasználóként bejelentkezve 5
3. ábra: Adminkét bejelentkezve 5
4. ábra: Foglalás 6
5. ábra: Regisztráció 6
6. ábra Belépés 7
7. ábra: Foglalásaim adattal 7
8. ábra: Foglalás lemondása adat nélkül 8
9. ábra: Mai érkezők 9
10. ábra: Legtöbbet fizetett megrendelő 9
11. ábra: Éves bevétel 10
12. ábra: Foglalás törlése 10
13. ábra: Fizetendők nézet és mai szobafizetések 25
14. ábra: Legtöbbet fizetett megrendelő nézet 25
15. ábra: Árbevételek évente 25

1. Bevezetés

Manapság már elengedhetetlen, hogy egy cég ne rendelkezzen online felülettel bármilyen szakterületen is van jelen. Egy szállodának elengedhetetlen, hogy ne legyen egy olyan internetes felülete, ahol előzetesen lehessen szobát foglalni. Fontos szempont az is, hogy mindenki számára elérhető legyen, ezért egy weboldal kézenfekvő választás lehet. Szakdolgozatom alapja ezért egy olyan adatbázis elérésű dinamikus weboldal, mely segítségével a regisztrált látogatók előre tudnak foglalni maguknak szobát, szobákat több szobatípus közül. A megrendelő számára a foglalások követhetők, esetleg lemondhatók. Foglalás törlését csak adminisztrátori (admin) szerepben lehet végezni. Az admin feladata a szobák és szobatípusok kezelése, valamint a szálloda számára hasznos kimutatások megjelenítése, valamint a tervezett árváltozások ütemezése. A vendégek kiköltözése/beköltözése és a foglalás kifizetése nincs megvalósítva.

A megvalósításra PHP 7.0 nyelvet választottam, hiszen ez az egyik legnépszerűbb szerveroldali programozási nyelv hazánkban. Az adatbázishoz MySQL adatbázisszervert, mivel ez nagyon jól összedolgozik a választott programozási nyelvvel. Webszerver megvalósításra Apache webszervert használtam. PHPSTORM integrált fejlesztői környezettel készítem. Szakdolgozatom készítése során a hangsúlyt a funkcionalitás bővítésére helyeztem a kinézettel és reszponzivitással szemben.

Azért választottam ezt a témát, mert szerettem volna olyan adatbázist tervezni majd megvalósítani, ami mindennapi kihívásokra próbal megoldást nyújtani. A weboldalt azért hoztam létre, hogy lássam, hogyan használható fel az adatbázisom, amivel egy újabb szeletét ismerhettem meg az informatika világának.

2. Felhasználói dokumentáció

A weboldal célja, hogy a felhasználók megtekinthessék a szállodában lévő szobatípusok árát és képeit, majd regisztrációt követően lehetőségük legyen egy a számukra megfelelő szobatípusból szobát lefoglalni. A foglalást követően lehetősége van a megrendelőknek megtekinteni az adott foglalásait, valamint lemondani azokat. Admin felhasználók esetén lehetőség van kimutatások készítésére, foglalás törlésére és nem utolsó sorban a szobatípusok árainak változtatására.

Akkor hívok egy foglalást aktuálisnak, ha nincs kifizetve, vagy lemondva az. A régi foglalásoka nem szeretném megjeleníteni.

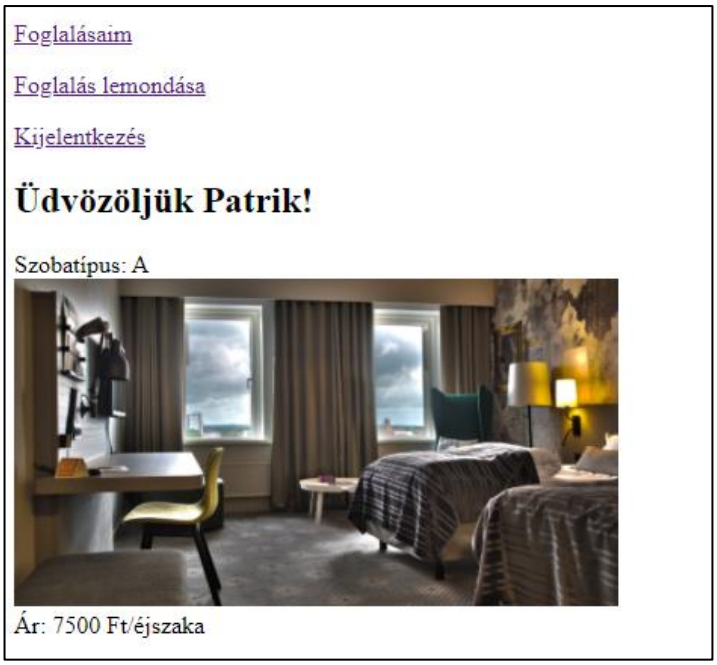
2.1 Rendszerkövetelmények

A weboldal megnyitásához bármilyen eszköz megfelel, amelyen egy naprakész böngésző áll rendelkezésre. Számítógépen az ajánlott böngésző a Google Chrome, de Opera GX és Safari-n is funkcionalitásában helyesen működik. Továbbá mobilon és tableten is használható. Folyamatos internetkapcsolat elengedhetetlen a weboldal használata közben.

2.2 Menürendszer

A képernyő bal felső sarkában található. Aszerint változik, hogy éppen be/ki van e jelentkezve a felhasználó esetleg adminnal van belépve az oldalra.

1. ábra Bejelentkezés nélkül

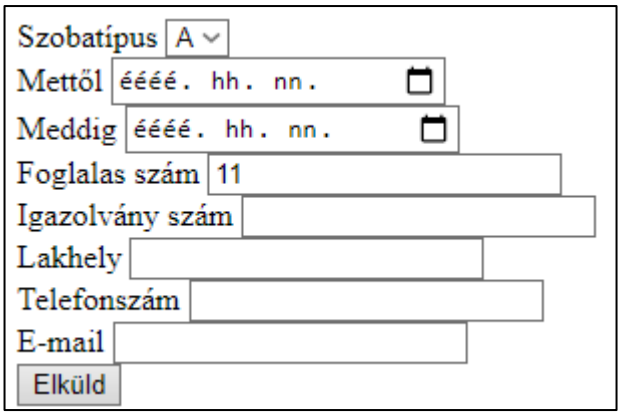


2. ábra: Felhasználóként bejelentkezve

3. ábra: Adminkét bejelentkezve

2.3 Kezdőlap

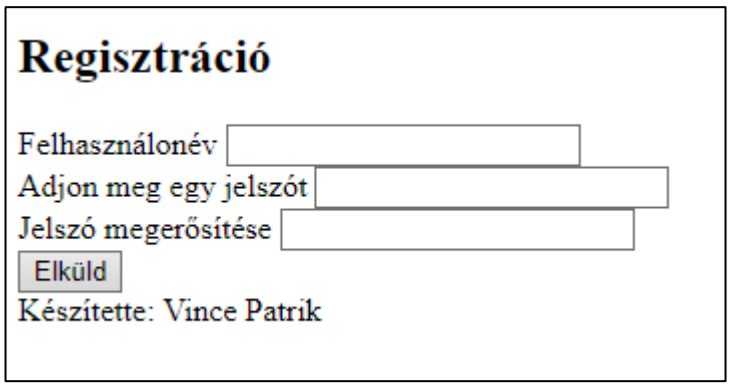
A kezdőlapon bejelentkezés vagy regisztráció nélkül megtekinthetővé válnak az aktuális szobatípusok és a hozzájuk tartozó aktuális árak és a szobák képei. A menüsorban elérhetővé válik a regisztráció, valamint a bejelentkezés menüpont. Itt még nincs lehetőség a foglalásra. Amint sikeres volt a regisztráció vagy a bejelentkezés lehetőséget kap a felhasználó a szobafoglalásra, ahol meg kell adnia egy általa foglalni kívánt szobatípust, az érkezésének a dátumát, a távozásának a dátumát, az igazolványszámát amivel tudja magát azonosítani, a lakhelyét ez opcionális, a telefonszámát, amin el lehet érni ezen kívül az email címét, amely szintén opcionális. A felhasználó foglaláskor kap egy generált foglalásszámot, amely azonosítani fogja az ő foglalását.



4. ábra: Foglalás

2.4 Regisztráció

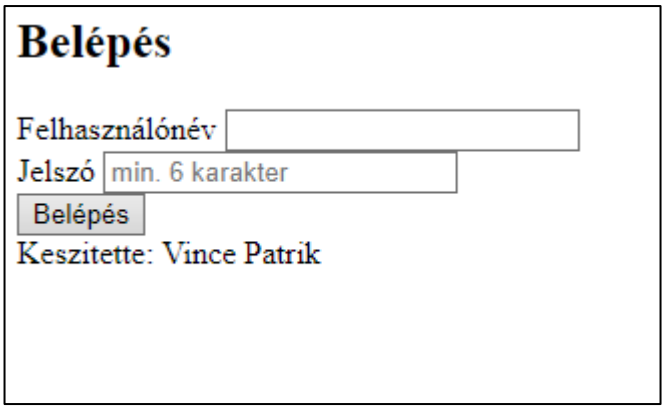
A regisztráció oldalon lehetőséget kapunk, a regisztrálásra, valamint, ha már rendelkezünk regisztrált felhasználóval, átkattinthatunk a bejelentkezés oldalra, vagy ha meggondolnánk magunkat, visszamehetünk a kezdőlapra. A sikeres regisztrációhoz meg kell adnunk egy felhasználónevet, jelszót, valamint meg kell erősítenünk a már megadott jelszavunkat. A jelszónak minimum 6 karaktert kell tartalmaznia. Sikeres regisztráció után egyből a kezdőlapra kerülünk, ahol már lehetőséget kapunk, a szobafoglalásra, valamint megjelennek plusz menüpontok, mint a foglalásaim, foglalásaim törlése és a kijelentkezés.



5. ábra: Regisztráció

2.5 Bejelentkezés

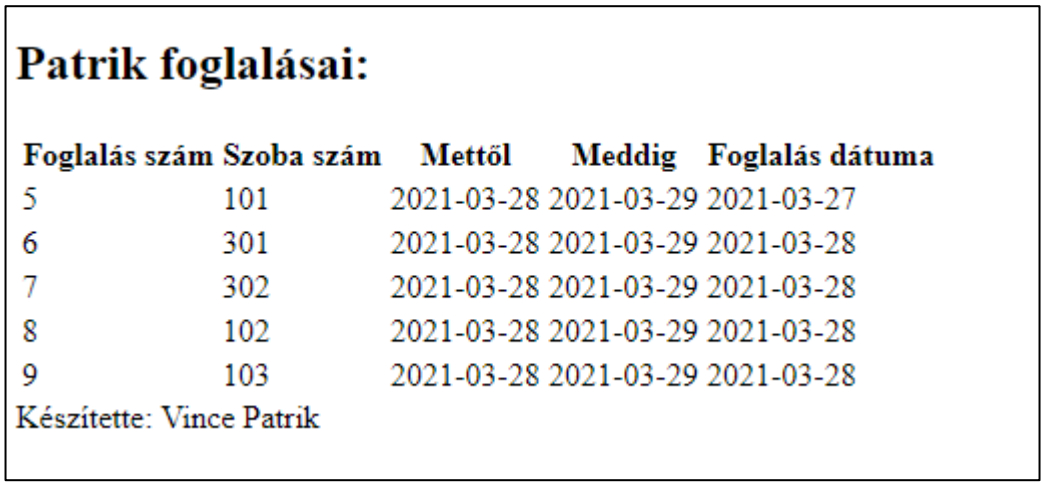
A sikeres bejelentkezéshez meg kell adni a regisztrációkor megadott felhasználónevet és a hozzá tartozó jelszót. Ez után átnavigálunk a kezdőlapra, ahol már lehetőséget kapunk a szobafoglalásra, valamint megjelennek plusz menüpontok, mint a foglalásaim, foglalásaim törlése és a kijelentkezés.



6. ábra Belépés

2.6 Foglalásaim

A foglalásaim oldalon megjelennek a foglalásaim táblázatos formában amennyiben van aktuális foglalásom. Itt lehetőségünk van visszamenni a kezdőlapra, a foglalás lemondása lapra, valamint a kijelentkezés oldalra.

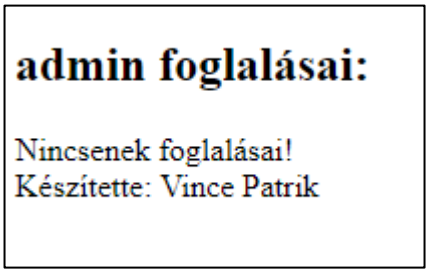


Patrik foglalásai:

Foglalás szám	Szoba szám	Mettől	Meddig	Foglalás dátuma
5	101	2021-03-28	2021-03-29	2021-03-27
6	301	2021-03-28	2021-03-29	2021-03-28
7	302	2021-03-28	2021-03-29	2021-03-28
8	102	2021-03-28	2021-03-29	2021-03-28
9	103	2021-03-28	2021-03-29	2021-03-28

Készítette: Vince Patrik

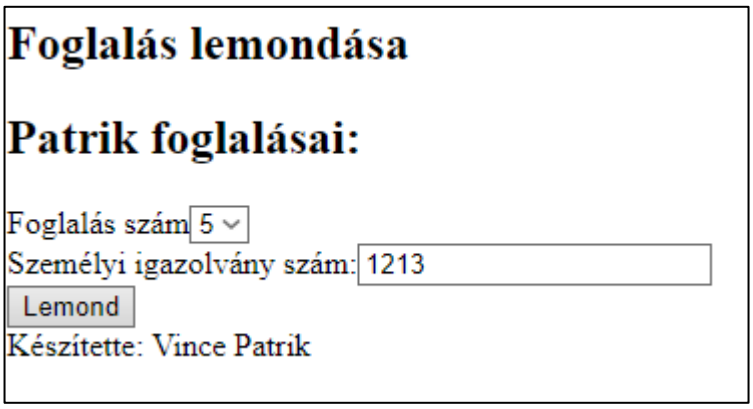
7. ábra: Foglalásaim adattal



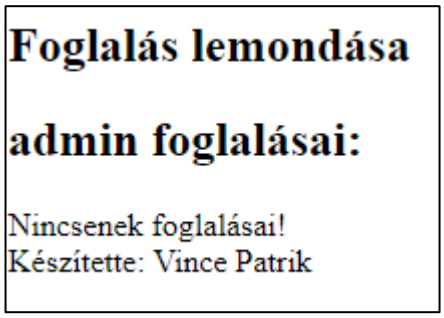
6. ábra.: Foglalásaim adat nélül

2.7 Foglalás lemondása

A foglalás lemondása oldalon lehetőségünk van lemondani a már lefoglalt szobánkat. Ha nem rendelkezünk aktuális foglalással, akkor addig nem tudjuk a lemondás funkciót használni, míg újra nem rendelkezünk akuális foglalással.



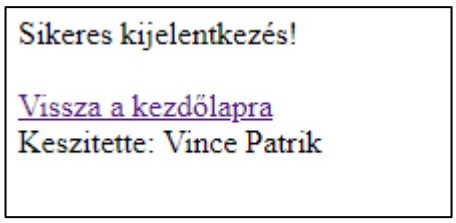
7. ábra: Foglalás lemondása adattal



8. ábra: Foglalás lemondása adat nélkül

2.8 Kijelentkezés

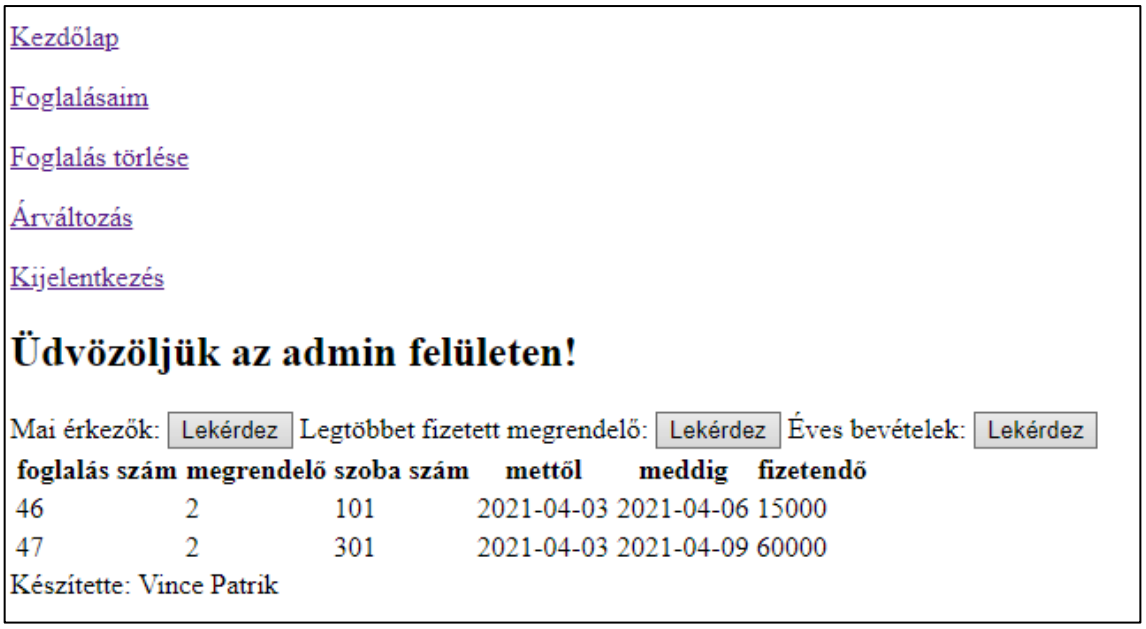
A kijelentkezéskor megszüntetjük a bejelentkezett accountot, majd innen visszamehetünk a kezdőlapra.



7. ábra: Kijelentkezés

2.9 Admin

Az admin oldalon különféle kimutatásokat láthatunk a szállodával kapcsolatban. Erre az oldalra csak az ahhoz jogosultak tudnak belépni.



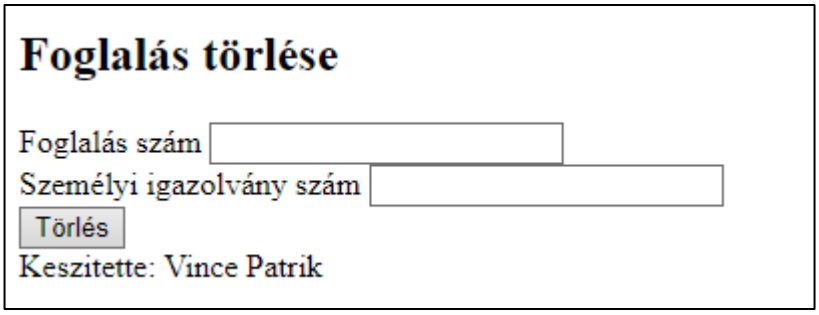
9. ábra: Mai érkezők

10. ábra: Legtöbbet fizetett megrendelő

11. ábra: Éves bevétel

2.10 Foglalás törlése

A foglalás törlése oldalon lehetőségünk van törölni foglalást az adatbázisból, ehhez tudnunk kell a foglalásszámot és az igazolványszámot.



12. ábra: Foglalás törlése

3. Fejlesztői dokumentáció

3.1 Követelmények

A jelszavakat titkosítva kell tárolni a GPR-nak megfelelően. A kliensnek kommunikálnia kell az adatbázissal és nem megengedhető az adatvesztés.

Lehetőséget kell biztosítani a bejelentkezésre és kijelentkezésre, foglalások megvalósítása és megtekintése, esetleg lemondására, a vendégek beköltözés és kiköltözés lehetőségére. Admin felületen lehetőséget kell adni az árak változására a foglalások törlésére és kimutatások készítésére.

3.2 Adatbázis

3.2.1 Tervezés

A bevezetőben ismertetett ügyvitel alapján a következőkben felsorolt funkcionalitás megvalósítását tervezem.

Ügyviteli funkciók

- Szobatípusok kezelése
- Árak meghatározása
- Szobák adatai
- Szobafoglalás
- Ügyfelek/vendégek kezelése
- Foglalás befizetése
- Beköltözés
- Kiköltözés

Táblák kialakítása

A szállodában szobák vannak és ezeket típusokba sorolom.

SZOBATÍPUS (szobatípus, hány_ágyas, felszereltség, komfort, kép)

A szállodában szobatípus alapján lehet szobát foglalni, ezért tárolom a szobának a típusát, azt, hogy hány ágyas, a felszereltségét és a komfortérzetét. A felszereltég és a komfort 1-től 5-ös skálán értendő: 1 minimális, 5 luxus. A szobákról a típusnak megfelelő vonzó képet szeretném megjeleníteni a felületen, mivel a megrendelő szobatípust választ a foglaláskor. Egy típushoz több szoba is tartozhat, ezért szükség van egy SZOBA táblára.

SZOBA {szoba_szám, szoba_típus}

Mivel foglalás után a vendég nem szobatípusba, hanem szobába fog beköltözni, ezért tárolom a szobának a számát, és típusát.

SZOBAÁRAK: {szobatípus, mettől, nappal_előtte, előtte_ár, utána_ár}

A szobák árát a szobatípushoz állapítja meg az admin. Mivel az árak időszakosan megváltozhatnak, ezért azokat sávosan tárolom el az időszak kezdetétől. Azt is figyelembe veszem, mennyivel előbb foglalják le a szobát, ezért két ár közül fogom kivenni a foglaláskor aktuális árat, amit a megrendelőnek kell fizetni.

FOGLALÁSOK: {foglalás_szám, szoba_szám, mettől, meddig, megrendelő_azonosító, foglalás_dátum, állapot}

Foglaláskor a megrendelő szobatípust ad meg egy időintervallummal, melyre én egy annak megfelelő szobát adok, amely a szoba előző foglalásai egyikével sem fog ütközni. A foglalás időpontja fontos az árszabás miatt. A foglalás változó állapotát 1-től 5-ig határoztam meg: 1 lefoglalva, 2 fizetve, 3 beköltözve, 4 kiköltözve, 5 lemondva.

LAKIK (foglalás, vendég_azonosító, érkezés, távozás)

Az adatbázist úgy terveztem meg, hogy a vendégek beköltözhessenek a lefoglalt szobákba. A vendégek érkezésének és a távozásának a dátumát tárolom és figyelem, hogy egy szobában maximum annyian lakhatnak ahány ágyas az adott szoba.

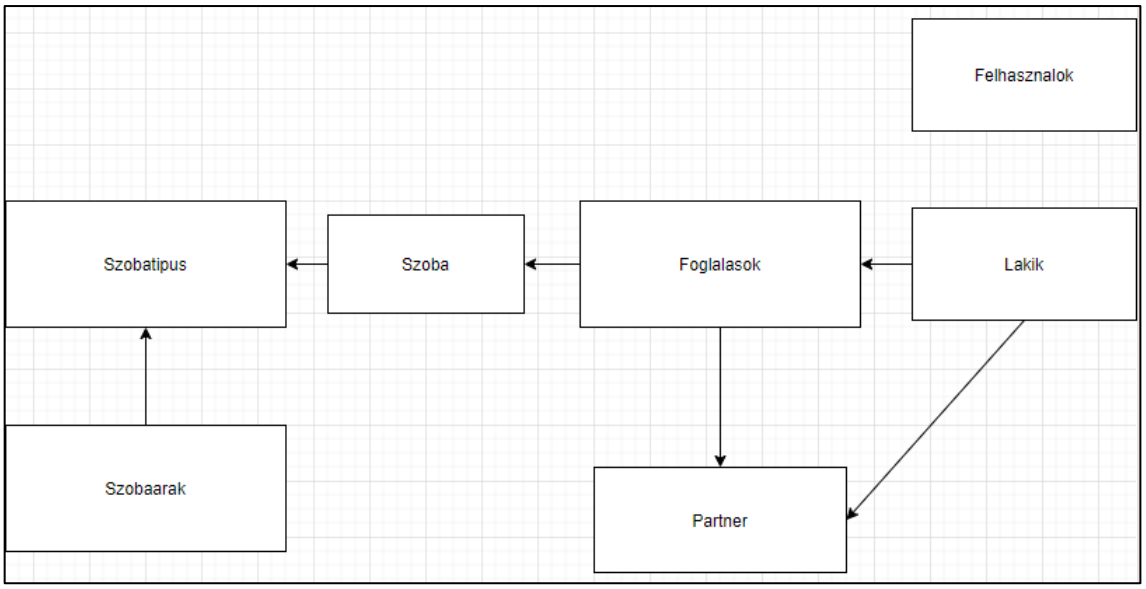
PARTNER (azonosító, igazolványszám, név, lakhely, telefonszám, email)

A megrendelők és a vendégek törzsadatait egyben fogom tartani. A foglaláskor a megrendelő adatait tárolom ebben a táblában, akinek regisztrált felhasználónak kell lenni. A megrendelő nem feltétlenül lesz vendég. (A vendégek felvitele, ezért a felületen más űrlapot igényelne.) A vendégeknek viszont nem kell egyesével regisztrálni beköltözéskor, ezért nem fogok hivatkozni a felhasználó táblára. Egy megrendelő többször is foglalhat, de egy foglalás csak egy megrendelőhöz tartozik.

FELHASZNÁLÓK (id, felhasználónév, jelszó)

Ahhoz, hogy valaki foglalhasson felhasználónak kell lennie, ezért szükség volt egy FELHASZNÁLÓK táblára, ahol tárolom a felhasználónevüket és jelszavukat. Ez a tábla független a többi táblától hiszen aki szerepel benne nem feltétlenül lesz megrendelő, megmaradhat látogató.

Egyed-kapcsolat diagram



8. ábra: Egyedkapcsolat diagram

3.2.2 Implementáció

Táblák szerkezete

A megvalósítás során a phpMyAdmin-ban MySql nyelven azt tapasztaltam, hogy sajnos nem úgy lehet megvalósítani a megszorításokat, ahogy azt MS SQL Serverben lehet, ezért nem mind lett megvalósítva a következőkben felsoroltakból.

A táblákhoz tartozó megszorítások, függvények és/vagy tárolt eljárások a mellékletben megtalálhatók.

SZOBATÍPUS

mező	**adattípus**	**szerep**	**korlátozás**
sz_tipus	varchar(1)	Kulcs	
agyak_szama	tinyint(4)	kötelező	agyak_szama > 0
felszereltseg	tinyint(4)	alapértelmezett 1	1<=felszereltseg<=5
komfort	tinyint(4)	alapértelmezett 1	1<=komfort<=5
kep	varchar(50)	alapértelmezett 0	

SZOBAÁRAK

mező	**adattípus**	**szerep**	**korlátozás**
sz_tipus	varchar(1)	Kulcs része, KK(Szobatípus)	
mettol	date	Kulcs része	
nappal_elotte	tinyint(4)	alapértelmezett 0	1<=nappal_elotte<=90

elotte_ar	int(11)	alapértelmezett 0	elotte_ar<>utana_ar
utana_ar	int(11)	alapértelmezett 0	elotte_ar<>utana_ar

SZOBA

mező	**adattípus**	**szerep**	**korlátozás**
sz_szam	int(11)	Kulcs	
sz_tipus	varchar(1)	KK(Szobatípus), kötelező	

FOGLALÁSOK

mező	**adattípus**	**szerep**	**korlátozás**
fogl_szam	int(11)	Kulcs	
szoba	int(11)	KK(Szoba), kötelező	
mettol	date	kötelező	mettol<meddig
meddig	date	kötelező	
megrendelo	int(11)	KK(Partner), kötelező	
fogl_datum	date	kötelező	fogl_datum <= mettol
allapot	tinyint(4)	alapértelmezett 1	1<=allapot<=5

Csak akkor vihető fel sor, ha az aktuális szoba előző foglalásainak egyikével sem ütközik időben.

LAKIK

mező	**adattípus**	**szerep**	**korlátozás**
fogl	int(11)	Kulcs része	fogl.allapot = 2 VAGY fogl.allapot = 3
vendeg	int(11)	Kulcs része,KK(Partner)	
erkezes	date	alapértelmezett 0	erkezes < tavozas erkezes >= fogl.mettol ÉS

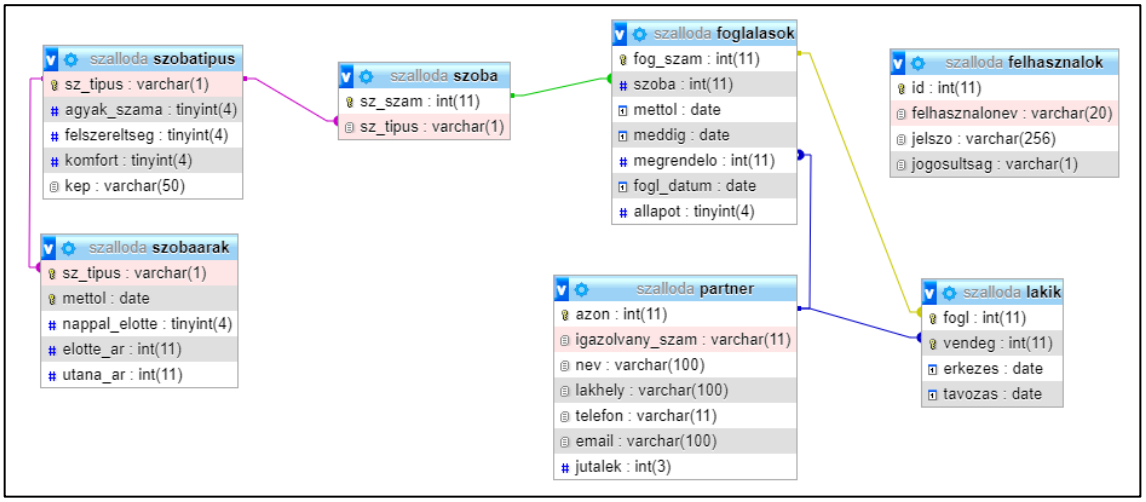
			erkezes <= fogl.meddig
tavozas	date	alapértelmezett 0	tavozas >= fogl.mettol ÉS tavozas <= fogl.meddig

Csak annyian lakhatnak egy szobában maximum, ahány ágyas az adott szobatípushoz tartozó szoba.

PARTNER

mező	**adattípus**	**szerep**	**korlátozás**
azon	int(11)	Kulcs	
igazolvany_szam	int(11)	alapértelmezett 0	
nev	nchar(100)	alapértelmezett 0	
lakhely	varchar(100)	alapértelmezett 0	
telefon	varchar(11)	kötelező	
email	varchar(100)	kötelező	
jutalék	tinyint(3)	alapértelmezett 0	100 <= jutalék >= 0

Kapcsolati ábra



8. ábra: Egyed-kapcsolat diagram

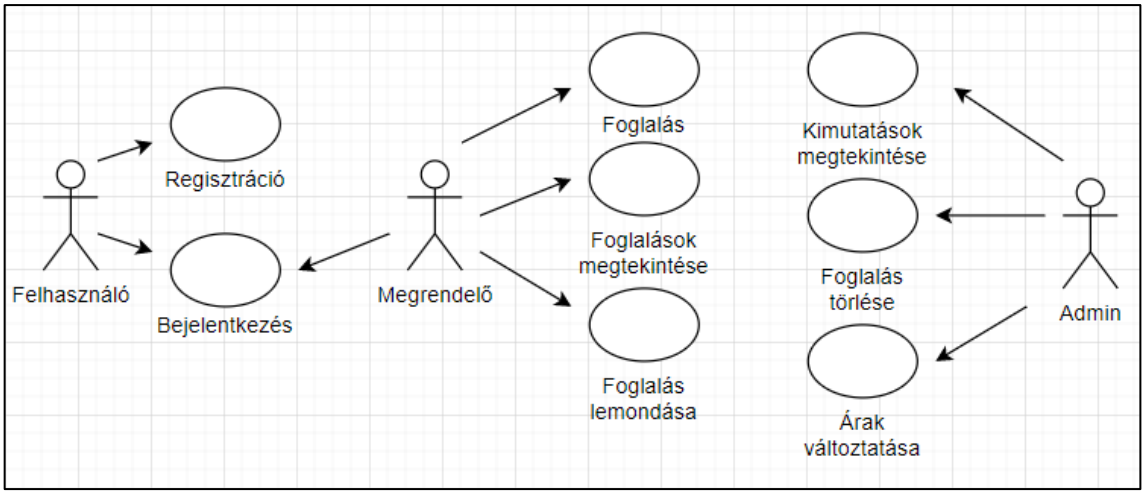
3.3 Weboldal

3.3.1 Felület terv

A felhasználóknak bejelentkezés vagy regisztráció után van csak lehetőségük használni az oldal főbb funkcióit, mint a szobafoglalást. Bejelentkezés nélkül csupán a szobákat tekinthetik csak meg. A bejelentkezésre a címsor is felhívja a figyelmet, és csak olyan menü látható, ami valamilyen formában belépteti a felhasználót. Ezután a felhasználó jogosult a további funkciók használatára, azonban aktív foglalás hiányában így sem rendelkeznek a teljes funkcionalitással. Ezek a funkciók rendelkezésre állnak az adminisztrátoroknak is, de kiegészítve egyéb hasznos funkcióval.

Használati esetek

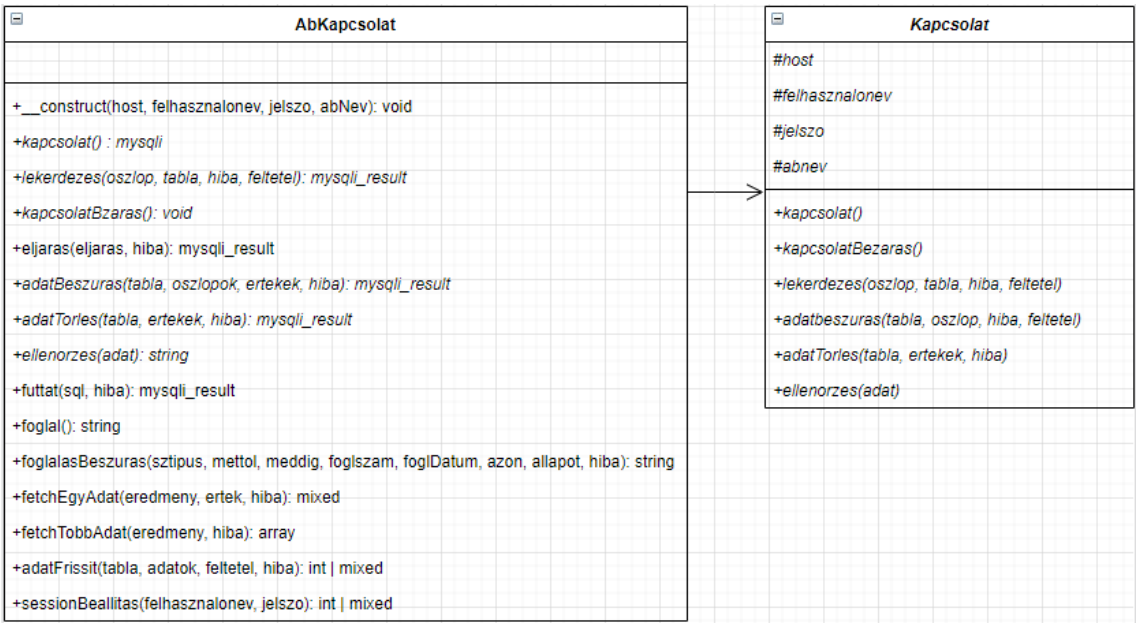
Minden felhasználó megtekintheti a szobákat és a szoba árait. Az ezen felüli funkciókhoz különböző jogosultságokra van szükség. A felhasználó regisztrációt vagy ha már regisztrált bejelentkezést követően tud szobát foglalni. Ha még nem foglalt szobát akkor is lehetősége van megtekinteni a foglalásait vagy lemondani azokat, ám értelemszerűen ezeket a funkciókat addig nem képes érdemben használni amíg nincs egy aktuális foglalása.



9. ábra: Használati eset diagram

A kapcsolat osztály úgy lett tervezve, hogy bármikor lehet használni más adatbázis kapcsoláshoz, amennyiben szükséges. A fő szempont az újrahasználhatóság volt. Az *abKapcsolat* osztálynak meg kell valósítani a meghatározott metódusokat és függvényeket, melyek segítik és meg is könnyítik az adatbázissal való együttműködést.

UML diagram



10. ábra: UML diagram

3.3.2 Implementáció

Az elején inkább úgynevezett sablon osztályokról és függvényekről lesz szó, ezért általánosabban fogalmazom meg a megvalósításukat, később viszont egy adott probléma megoldására létrehozott függvényekről és szkriptekről lesz szó, ezért azokat specifikusabban fogalmazom meg. A továbbiakban *dőlt betű*vel a függvények, tárolt eljárások és változók neveit különbözetem meg.

Osztályok és függvények:

Kapcsolat absztrakt osztály: Az adatbázishoz való kapcsolatot és műveleteket megvalósítandó függvényeket tartalmazza. Adattagjai *protected*-ek, azaz csak az osztályban, vagy a belőle örökölt osztályokban érhetők el. Függvényei publikusak, azaz a projecten belül bárki hívhatja őket. Öröklődés céljából lett létrehozva.

AbKapcsolat osztály: Megvalósítja az adatbázis kapcsolatot és az örökölt metódusokat, valamint definiál újabb hasznos metódusokat. Tizenöt publikus függvényt tartalmaz. Segítségükkel történik az adatbázis műveletek megvalósítása.

__construct (): Konstruktor. Példányosítás során az első, aki lefut. Paramétereiben meghatározzuk az adatbáziskapcsolatot. Beállítja a kapott változók értékét, majd meghívja a *kapcsolat ()* függvényt.

kapcsolat (): Kapcsolatot teremt az adatbázis szerverrel. Ha nem sikerül kapcsolódni az adatbázis szerverhez, akkor hibaüzenettel tér vissza, egyebként egy *mysqli* típussal.

kapcsolatBezar (): Bezárja az adatbázis kapcsolatot amennyiben van már nyitva.

lekerdezes (): Bemeneti paraméterek:

- *oszlop*: ettől a változótól függ, hogy az SQL lekérdezésnek melyik oszlopával tér vissza,
- *tabla*: ettől a változótól függ, hogy az SQL lekérdezés melyik táblából nyerje ki az adatokat,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a lekérdezés, ha hibára futott,
- *feltetel:* ettől a változótól függ, hogy az SQL lekérdezünk milyen feltételek mellett adjon eredményt, nem kötelező megadni

A bemenő paraméterek segítségével meghatároz és SQL parancsot, amit majd tovább küld a *futtat ()* függvénynek, majd meghívja azt. Az eredmény értékével tér vissza.

futtat (): bemeneti paramérterek*:*

- *sql:* ez a változó tartalmazza az SQL lekérdezést,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a lekérdezés, ha hibára futott

Meghívja a *kapcsolat ()* függvényt, amely meghívja a *query ()* függvényt, amely megvalósítja a lekérdezést. Sikeres futtatás esetén mysqli_resut-tal tér vissza, hiba esetén pedig az átadott hibaüzenettel.

adatBeszuras (): bemeneti paraméterek:

- *tabla:* ettől a változótól függ, melyik táblába szúrja be az adatokat,
- *oszlopok:* ettől a változótól függ, hogy az adatbázisba való beszúráskor mely oszlopokba szúrjunk be érteket,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a beszúrás, ha hibára futott

A bemeneti paraméterek segítségével definiál egy SQL beszúró parancsot, majd meghívja ezzel és a *hiba* változóval a *futtat ()* metódust. Sikeres futtatás esetén mysqli_resut-tal tér vissza, hiba esetén pedig az átadott hibaüzenettel.

adatTorles (): bemeneti paraméterek:

- *tabla:* ettől a változótól függ, hogy melyik tábából törölje az adatokat,
- *ertekek:* ettől a változótól függ, hogy milyen feltételnek kell megfelelni az adatnak, hogy törlésre kerüljön,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a törlés, ha hibára futott

A bemeneti paraméterek segítségével definiál egy SQL parancsot, melyet átad a *futtat ()* függvénynek, a hiba változóval együtt. Sikeres futtatás esetén mysqli_resut-tal tér vissza, hiba esetén pedig az átadott hibaüzenettel.

ellenorzes (): bemeneti paraméter:

- *adat:* ezt a változót fogja megtisztítani

A bemeneti paraméterben átadott szövegből levágja a „*white space*” -eket, a „\” karaktert, majd a speciális HTML karaktereket. Visszatér a tisztított adattal.

adatFrissit (): bemeneti paraméterek:

- *tabla:* ettől a változótól függ, hogy melyik tábában végezze el a frissítést,
- *adatok:* ettől a változótól függ, hogy melyik adatot, mire frissítse,
- *feltetel:* ettől a változótól függ, hogy milyen feltételek alapján végezze el a frissítést,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza az adat frissítés, ha hibára futott

A bemeneti paraméterek segítségével definiál egy SQL parancsot, melyet átad a *futtat ()* függvénynek, a hiba változóval együtt. Sikeres futtatás esetén mysqli_resut-tal tér vissza, hiba esetén pedig az átadott hibaüzenettel.

eljaras (): bemeneti paraméterek:

- *eljaras:* ettől a változótól függ, hogy az adatbázisban előre létrehozott tárolt eljárást hívja meg,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a tárolt eljárás, ha hibára futott

A bemeneti paraméterek segítségével definiál egy SQL parancsot, melyet átad a *futtat ()* függvénynek, a hiba változóval együtt. Sikeres futtatás esetén mysqli_resut-tal tér vissza, hiba esetén pedig az átadott hibaüzenettel.

fetchEgyAdat (): bemeneti paraméterek:

- *eredmeny:* ettől a változótól függ, hogy milyen asszociatív tömböt dolgozza fel,
- *ertek:* ettől a változótól függ, hogy melyik oszlopban lévő adattal tér vissza,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a fetch-elés, ha hibára futott, nem kötelező megadni

Végigmegy a kapott asszociatív tömbön, majd sorrá alakítja azt, végül a paraméterként megadott értéket adja vissza. Ha nem kap vissza eredményt, akkor a megadott hibaüzenettel tér vissza.

fetchTobbAdat (): bemeneti paraméterek:

- *eredmeny:* ettől a változótól függ, hogy milyen asszociatív tömböt dolgozza fel,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a fetch-elés, ha hibára futott, nem kötelező megadni

Végigmegy a kapott asszociatív tömbön, majd a definiált tömbbe beletölti a sorokat, majd azzal a tömbbel visszatér. Hiba esetén visszatér az általunk definiált hibával, amennyiben definiálásra került az a változó.

sessionIdBeallitas (): bemeneti paraméterek:

- *felhasznalonev:* ettől a változótól függ, hogy melyik felhasználónévre állítson session-t,
- *jelszo:* ettől a változótól függ, hogy melyik jelszóra állítson session-t

Meghívja a *lekerdez ():* függvényt és definiál a bemenő paramétereknek értéket, majd átadja azokat. Ez visszatér egy *eredmeny* változóval, Meghívja a *fetchEgyAdat ()* függvényt, aminek átadja az *eredmeny változót* és egy a lekérdezéshez definiált *oszlop* nevű változót. Majd visszaadja a kívánt érteket. Hiba esetén a rendszer által definiált hibát kapunk.

foglal (): A POST szuperglobális tömbben lévő adatokkal dolgozik, ha *_POST['kuld']* – nek van értéke. A kapott értékeket átadja az *ellenorzes ()* függvénynek, ami visszaadja az adatokat már tisztított formában. Amennyiben a kapott értékek ki vannak töltve, beállít segédváltozókat, majd meghívja a *lekerdezes ()* függvényt, majd a kapott eredménnyel meghívja a *fetchEgyAdat ()* függvényt. Ennek az értéke visszaadja, hogy szerepel-e a foglalni kívánó felhasználó a partnerek között. Ha igen, akkor meghívja a *foglalBeszuras ()* függvényt és átadja a paramétereket. Amennyiben nem szerepel a felhasználó a partnerek között, beállítok segédváltozókat, majd meghívja az *adatBeszuras ()* függvényt és átadja neki azokat. Ez után meghívja a *foglalBeszuras ()* függvényt a kapott változókkal. Amennyiben nincsen értéke valamelyik kapott változónak hibát ír ki.

foglalBeszuras (): bemeneti paraméterek:

- *sztipus:* ettől a változótól függ, hogy milyen szobatípusra foglal,
- *mettol:* ez a változó állítja be az érkezés dátumát,
- *meddig:* ez a változó állítja be a távozás dátumát,
- *foglszam* ez a változó állítja be a foglalás számát,
- *foglDatum* ez a változó állítja be a foglalásnak a dátumát,

- *azon* ettől a változótól függ, hogy melyik megrendelő foglal,
- *allapot:* ez a változó állítja be a foglalás állapotát,
- *hiba:* ettől a változótól függ, hogy milyen hibaüzenettel tér vissza a foglálás beszúrásakor, ha hibára futott, nem kötelező megadni

A paramérterként kapott szobatípussal és hibával meghívja a *lekerdezes ()* függvényt, amely visszaadja azokat a szobaszámokat, amelyek szobatípusa a megadott szobatípus. Létrehoz egy segédtömböt, amibe beletölti a kapott szobaszámokat, majd egy while segítségével megnézi, hogy egy adott szobaszám a kapott dátumok között le van e már foglalva, azaz szerepel-e a foglalásban. Ehhez szükség van egy olyan tárolt eljárás meghívására, amely visszaadja, hogy az adott szoba mennyi szobával ütközik.

```
CREATE PROCEDURE mennyivelUtkozik
(
    IN szoba1 INT, IN mettol1 DATE, IN meddig1 DATE
)
BEGIN
    SELECT COUNT(*) as ossz
    FROM foglalasok
    WHERE szoba = szoba1 AND (mettol BETWEEN mettol1 AND DATE_ADD(meddig1,
INTERVAL -1 DAY)
          OR mettol1 BETWEEN mettol AND DATE_ADD(meddig, INTERVAL -1 DAY));
END
```

11. ábra Mennyivel ütközik tárolt eljárás

Amennyiben az SQL szkriptem 0-val tér vissza, nincs ütközésben egy foglalással sem, ezért felvihető az adatbázisba. Amennyiben nem 0-val tér vissza, van ütközés, megnézi, hogy a következő szobaszámra is van-e, ha minden szobaszámra az aktuálisan kiválasztott szobatípusról ütközés van, azaz le vannak foglalva a szobák, akkor hibaüzenettel tér vissza. Amennyiben sikeresen foglaltunk, átirányít minket a kezdőlapra.

Oldalak

Kezdőlap

Elindítok egy „SESSION” -t a *session_start ()* függvény meghívásával, azért, hogy az eddig használt vagy beállításra került „SESSION” -öket tudjam használni az oldalon. Amennyiben a *$_SESSION['fnev']* szuperglobális tömb „fnev” kulcsértékén be lett állítva érték, akkor 2 féle lehetséges kinézete lehet az oldalnak:

1. Ha be van jelentkezve/regisztrálva van a felhasználó
2. Ha adminként van bejelentkezve a felhasználó

Első esetben: regisztráció/bejelentkezés megjelenik a beállított SESSION['fnev'] értéke, mint felhasználónak szóló üdvözlés, valamint a szobatípusok, árral és képpel és a foglalás űrlap, ahol dinamikusan jelenik meg, hogy mely szobatípusok állnak rendelkezésre az adatbázisban. A foglalás szám is dinamikusan kerül a „value" attributumba, ami automatikusan a következő foglalásszám, azaz az maximum foglalásszám + 1. A többi szimpla beviteli mező. Az elküld gombra kattintva elküldi POST -al az „abFoglal.php" oldalra. A menü ebben az esetben áll egy foglalásaim, egy foglalás lemondása és egy kijelentkezésből.

Második esteben: szinte ugyan azok a funkciók jelennek meg a különbség az üdvözlésben lévő SESSION['fnev'] értéke, valamint a menük. Ez esetben a menük a következők: admin, foglalásaim, foglalások törlése és kijelentkezés.

Abban az esetben mikor nincs élő SESSION, azaz sem regisztrálva sem bejelentkezve nincs a felhasználóm csak a szobák típusai jelennek meg, képekkel és a hozzájuk tartozó aktuális árral, valamint kétféle menü, a regisztráció és bejelentkezés.

Regisztráció

A mezők kitöltése után a gomb megnyomásakor a beírt értékek a *$_POST* szuperglobális tömbben elküldi feldolgozásra. A „regisztral.php" fájl fogadja a tömböt, amennyiben a tömbben vannak adatok akkor felhasználja az értékeiket és definiál változókat. Mindenekelőtt behívom a „szallodaKapcsolat.php" fájt, ami példányosítja az „abKapcsolat" osztályt, ezzel lehetőségem van minden benne megvalósított függvény használatára. Majd a kapott változókkal meghívja az *ellenoriz ()* függvényt adattisztítás céljából, elkerülve egy esetleges SQL injection-t. Ez után titkosítja a jelszót, amit a felhasználó állított be az *md5 ()* függvény segítségével. Ezután lekérdezésre kerül a maximális „id" a felhasználók táblából és ezt eltárolja egy tömbben. Amennyiben a felhasználó ugyan azt a két jelszót írja be akkor az eltárolt „id" +1 és a kapott értékekkel meghívja az *adatBeszuras ()* függvényt, amivel bekerülnek az adatok az adatbázisunkba. Alapértelemzett állapot 0-val kerül fel, hiszen csak az admin rendelkezik 1-es állapottal. Ha sikeresen felkerültek az adatok akkor indítok egy SESSION-t *$_SESSION['fnev']* néven, ami tárolni fogja a beírt felhasználónevet, valamint ezzel a felhasználónévvel és jelszóval meghívjuk a *sessionIdBeallitas ()* függvényt, majd bezárom a kapcsolatot. Amennyiben nem kapott értéket a *$_POST* vagy a két jelszó nem egyezik meg, akkor hibaüzenetet írok ki az oldalra.

Bejelentkezés

Az űrlap kitöltése után az „Belépés” gomb elküldi a beírt adatokat a *$_POST* szuperglobális tömbbe tárolva a „bejelentkezesFeldolgoz.php” fájlnak. Amennyiben a tömb üres, hibaüzenetet írok az oldalra. Helyes kitöltés esetén ugyan úgy, mint a Regisztrációban (ezért nem fejtem ki hosszabban), megtisztítjuk a kapott adatokat, majd a jelszót titkosítom. Meghívom a *lekérdez ()* függvényt, hogy visszakapjam eredménynek azt, hogy a kapott jelszó benne van e a felhasználók táblában, ha nincs hibaüzenettel tér vissza. Ha benne van, akkor lekérdezem a *lekerdezes ()* függvénnyel az előbbi felhasználóhoz tartozó jogosultságot. Ha a jogosultság 1, azaz admin, akkor elindítok egy SESSION-t, beállítok egy „admin”, egy fnev és egy „id” SESSION-t, majd átirányítom az admin oldalra. Ha a jogosultság nem 0, azaz felhasználó, akkor szintén elindítok egy SESSION-t majd beállítok egy fnev és egy „id” SESSION-t. Ezután átirányítom a kezdőlapra, majd bezárom az adatbázis kapcsolatot az *kapcsolatBezaras ()* függvénnyel, hisz nincs rá többet szükség.

Foglalásaim

Kezdek egy új SESSION-t a *session_start ()* függvénnyel, hogy hozzáférjek a „fnev” és „id” SESSION-ökhöz. Majd az előbbi segítségével, mint már a kezdőlapon is,felhasználóra szóló üzenetet írok ki az oldalra. Behívom a „szallodaKapcsolat.php” fájlt, amiben példányosítom az „abKapcsolat” osztályomat. Aztán meghívok egy tárolt eljárást az *eljaras ()* függvénnyel, ami visszaadja nekem, hogy az adott felhasználó benne van e már a partner táblában. A visszaadott értéket eltárolom egy változóban. Ezután egy lekérdezés segítségével megszámolom, hogy a partner táblában szerepel e az eltárolt változó értékével megegyező azonosítóval. Amennyiben nem, kiírom a megfelelő üzenetet, egyébként pedig lekérdezem a foglalásait. Majd a kapott adatokból egy asszociatív tömböt készítek a *fetchTobbAdat ()* függvénnyel, majd egy foreach-el végig megyek a kapott tömbön és kiírom a benne tárolt adatokat táblázatos formában. Lezárom az adatbáziskapcsolatot, hiszen nincs már szükségem rá.

Foglalás lemondás

Az űrlapot dinamikusan állítom elő, amit a „lemondasokKiiras.php” fájlban valósítok meg. Újra használom a „foglalasok.php” fájlban leírtakat, aztán ha vannak foglalásai az adott felhasználóknak, akkor vizsgálom, hogy a visszaadott érték nem „String” (azaz szöveg), akkor egy üzenet ír ki az oldalra, egyébként, egy asszociatív tömb. Vizsgálom, hogy a kapott eredményem nem „String” -e, ha nem az akkor „<select>” tag- ek közé foreach -el „<option>” tag -ekbe beleírom a kapott foglalások számát, majd kimentem a foglalás számát egy változóba. Ez után meghívom az *eljaras ()* függvényt az előbbi

változó felhasználásával. Majd ennek az eredményét átadom egy általam definiált hibával a *fetchEgyAdat ()* függvényt. Ennek az eredményét tárolom egy változóban. Lezárom az adatbáziskapcsolatot a *kapcsolatBezaras ()* függvénnyel. Majd kiírom az űrlap többi részét és a kapott eredményt egy „value" attribútumba helyezem, amely kiírja a felhasználónak, hogy mi volt az igazolvány száma.

A „Lemond" gomb megnyomásakor az űrlapon beírt értékeket elküldi a „lemondasFeldolgoz.php" fájlnak. Behívom a „szallodaKapcsolat.php" -t, amiben példányosítom az „abKapcsolat" osztályomat. Vizsgálom, hogy van e érték a *$_POST['lemond']* szuperglobális tömbben. Létrehozok változókat a kapott értékeknek, hogy később is tudjam használni őket. Meghívom az értékekre az *ellenoriz ()* függvényt adattisztítás céljából. Majd meghívom az *adatFrissit ()* függvényt az értékekkel. Ezt a műveletet 3x ismétlem, mivel egyszerre csak egy értéket tudok frissíteni. Ez után bezárom az adatbáziskapcsolatot a *kapcsolatBezaras ()* függvénnyel, majd visszanavigálom „lemondas" oldalra, hogy frissítse az adatokat, ezért nincs lehetőségem üzenetet írni, ha sikeres volt a lemondás.

Kijelentkezés

Az oldal betöltésekor meghívom a *session_start ()* függvényt, hogy aztán használni tudjam a *session_uset ()* függvényt, ami lehetővé teszi, hogy minden eddig beállított SESSION értékét kivegye a *$_SESSION* szuperglobális tömbből. Ez után kiírok egy üzenetet a felhasználónak.

Admin

Indítok egy SESSION-t a *session_start ()* függvénnyel, hogy ellenőrizni tudjam, hogy a *$_SESSION['admin']* szuperglobális tömb értéke „true" -e, ha ez nem igaz, akkor nem admin jogosultágú a felhasználó, ezért visszairányítom a kezdőlapra. (Ezt minden admin jogosultságú oldal esetén megteszem, ezért a továbbiakban nem ismétlem.) Ha viszont admin, akkor lehetőséget adok neki, hogy láthassa az itt lévő tartalmat. Három gombot hoztam létre, amivel az adminnak lehetősége van az adott kimutatás megtekintésére. Vizsgálom, hogy melyik gombon lévő kimutatást szeretné megtekinteni. Majd a *lekerdez ()* függvénnyel lekérdezem és a *fetchTobbAdat ()* függvénnyel kinyerem egy asszociatív tömbbe az eredményeket, ezeket egy „foreach" segítségével megjelenítem az oldalon.

```
CREATE VIEW fizetendok  AS
select f.fog_szam AS fog_szam,f.szoba AS szoba,f.mettol AS mettol,f.meddig AS meddig,f.megrendelo AS
megrendelo,f.fogl_datum AS fogl_datum,f.allapot AS allapot,sz.sz_szam AS sz_szam,sz.sz_tipus AS
sz_tipus,sz.kep AS kep,fizetendofgv(sz.sz_tipus,f.fogl_datum,f.mettol) * (to_days(f.meddig) -
to_days(f.mettol)) AS fizetendo
from (foglalasok f join szoba sz on(f.szoba = sz.sz_szam));

CREATE VIEW maiszobafizetesek  AS
select fizetendok.fog_szam AS fog_szam,fizetendok.megrendelo AS megrendelo,fizetendok.szoba AS
szoba,fizetendok.mettol AS mettol,fizetendok.meddig AS meddig,fizetendok.fizetendo AS fizetendo
from fizetendok
where fizetendok.mettol = curdate() ;
```

13. ábra: Fizetendők nézet és mai szobafizetések

```
CREATE VIEW legtobbetfizetettmegrendelo  AS
select fizetendok.megrendelo AS megrendelo,sum(fizetendok.fizetendo) AS arbevetel
from fizetendok
where fizetendok.mettol <= curdate()
group by fizetendok.megrendelo order by 2 desc limit 1;
```

14. ábra: Legtöbbet fizetett megrendelő nézet

```
CREATE VIEW arbevetelekevente  AS
select year(fizetendok.mettol) AS evben,sum(fizetendok.fizetendo) AS arbevetel
from fizetendok
where fizetendok.allapot > 1 and fizetendok.allapot < 5
group by year(fizetendok.mettol) ;
```

15. ábra: Árbevételek évente

Foglalás törlése

Két input text mezős űrlapot jelenítek meg. A „Küld” gomb megnyomása után meghívom a „torlesFeldolgoz.php” fájlt. Vizsgálom, hogy a *$_POST['kuld']* szuperglobális tömb „kuld” kulcsában van-e érték, ha van, akkor behívom a „szallodaKApcsolat.php” fájlt, hogy példányosítsam az „abKapcsolat” osztályomat. Majd a kapott értékeket eltárolom változókba és meghívom az *ellenoriz ()* függvényt adattisztítás céljából. Vizsgálom, hogy van e tényleges érték a kapott változóimba, ha valamelyikben nincs, akkor hibaüzenettel térek vissza. Ha van, akkor a *lekerdez ()* függvénnyel lekérdezem, majd a *fetchEgyadat ()* függvénnyel visszakapom, majd egy változóba eltárolom azt a foglalás számot, aminek az igazolványszáma megegyezik a kapott igazolványszámmal. Vizsgálom, hogy az eltárolt igazolványszám megegyezik-e a kapott igazolványszámmal. Ha nem, akkor hibaüzenettel térek vissza. Ha igen akkor meghívom az *adatTorles ()* függvényt, majd visszaigazoló üzenetet írok az oldalra.

Árváltozás

Űrlapot hozok létre, amiben inputokat helyezek el. A „Változtat” gomb megnyomásakor az „arvaltozasFeldolgoz.php” fájlba küldi el az adatokat. Vizsgálom, hogy a *$_POST['kuld']* szuperglobális tömb „kuld” kulcsában vannak-e értékek. Ha vannak, akkor az itt kapott adatokat változókba mentem. Behívom a „szallodaKapcsolat.php”

fájlt, hogy példányosítsam az „abKapcsolat” osztályomat. Majd meghívom az *adatBeszur ()* függvényemet és paraméterül adom nekik a változóimat. Vizsgálom, hogy a kapott eredményem igaz volt-e, azaz sikeres volt a beszúrás vagy hamis, azaz hibába ütközött. Ha sikeres volt, akkor visszajelző üzenetet írok az oldalra és sikertelen esetben is.

3.4 Tesztelés

A folyamatos teszten túl múltbeli és jövőbe szóló foglalások adatainak szkripjét is biztosítom a kódok és sql szkriptek mappában.

Teszteset	Elvárt eredmény	Kapott eredmény	Elfogadás
Regisztráció esetén két tárolandó titkosított jelszó összehasonlítása	Összehasonlításkor ugyan az a hashelt kód jön ki, ami szerint össze lehet hasonlítani a két szöveget.	Összehasonlításkor ugyan az a hashelt kód jön ki, ami szerint össze lehet hasonlítani a két szöveget.	Elfogadva
Bejelentkezéskor rossz adat megadása	Kiírja a képernyőre, hogy melyik volt a rossz adat.	Kiírja a képernyőre a rosszul megadott adat beviteli mezőjének a nevét.	Elfogadva
Szoba ajánlatok megjelenítése	Dinamikusan írja ki a szobatípusokat, a hozzájuk tartozó megnevezéssel, aktuális árral és képpel.	Helyesen írja ki a típust, az aktuális árat és a hozzá tartozó képet.	Elfogadva
Foglalás	Keres egy olyan szobát, amely időben nem ütközik egy már foglalt szobával a felhasználó által megadott típusból.	A felhasználó által megadott szobatípusra keres egy szobát, mely nem ütközik az előző foglalásra.	Localhoston elfogadva, Szerveren javítás alatt
Aktuális foglalások megjelenítése	A megrendelő minden olyan foglalásának kiírása, amely még nincs kifizetve.	Kiírja a megrendelő összes olyan foglalását, amelyet még nem fizetett ki.	Elfogadva

Foglalás lemondás	Adatbázisban megváltoztatja az állapotot.	Adatbázisban megváltoztatja az állapotot és a dátumot is.	Elfogadva
Mai érkezések megjelenítése	Adatbázisból megjeleníti gombra kattintáskor, hogy a mindenkori mai napon milyen foglalás számmal melyik szobát mettől meddig foglalják le és mennyi lesz a kifizetendő összeg.	Gombra kattintáskor megjeleníti a mindenkori mai napon a foglalást, a foglalás számot, a megrendelő azonosítóját, a lefoglalt szoba számát, a beköltözés és kiköltözés napját és a foglálás fizetendő összegét	Elfogadva
Legtöbbet fizetett vendég megjelenítése	Gombra kattintáskor megjeleníti az adatbázisból annak a megrendelőnek az adatait, aki összesen a legtöbbet fizette.	Gombra kattintva megjeleníti a megrendelő azonosítóját, nevét és az eddig összesen fizetett összeget.	Elfogadva
Évenkénti bevétel	Gombra kattintáskor megjeleníti az adott évre, hogy mennyi volt a bevétel.	Gombra kattintáskor megjeleníti évenkénti lebontásban az árbevételt.	Elfogadva
Kötelező mezők kitöltése	Kötelező mezők kitöltése esetén jelenjen meg hibaüzenet az adott mezővel.	Ha nem töltik ki a kötelező mezőket hibaüzenetet ír az oldalra.	Elfogadva az összes űrlap esetén
Foglalás törlése	Adatbázisból való végleges törlés.	Törli az adott foglalást az adatbázisból.	Elfogadva

Ár módosítás	Megadott szobatípusra beállítja az adatbázisban az árat a kezdeti dátummal.	Beállítja a szobatípusra az előtte és utána árat a kezdeti dátummal.	Elfogadva
Kijelentkezés	Megszűnteti a regisztrációkor vagy bejelentkezéskor beállított session-t, majd lehetőséget ad a kezdőlapra lépésre. Visszajelző üzenet megjelenítése a felületen.	Visszajelző üzenetet jelenít meg, majd megszűnteti az aktív session-t. Megjelenít egy linket, amivel átlépést tesz lehetővé a kezdőlapra	Elfogadva

4. Továbbfejlesztés

Továbbfejlesztési lehetőség lehet a vendégek ki és beköltöztetése, amely azért nem tudott megvalósulni, mert vendégek törzsadatihoz más űrlapra lenne szükség, amire már nem volt elég időm. Az adatbázisban meg van tervezve a tábla és a hozzá kapcsolódó megszorítások. A kezdőlapot hasznos kimutatással lehetne bővíteni, mint például a legtöbbször foglalt szobatípus. Az admin oldalt hasznos kimutatásokkal lehetne bővíteni, mint például melyik vendég lakott a legtöbb éjszakát vagy melyik 3 vendég tartózkodott legtöbbet. A dizájnt lehetne fejleszteni.

5. Összefoglalás

A szakdolgozatom adatbázis része érdekelt a legjobban, ami abban is látszik, hogy sokkal jobban meg lett tervezve és valósítva ez a rész. Szerintem legértékesebb funkció az árváltozás, ahol két módon is adok lehetőséget az árváltozásra: egyrészt időszak függően, másrészt, hogy hány nappal előtte foglaltak. Ez a funkció a jóval előre foglalás ára és a last minute árak alakítására is hasznos. Emellett kifejezetten élveztem a weboldal programozását is, főleg mikor nem triviális dolgokat kellett megvalósítanom, azonban maradtak olyan funkciók melyeket még nem sikerült megvalósítanom. A php nyelvvel sokat kellett küzdenem, mire teljesen helyreállt, hogy hogyan tudnám kihozni belőle azt, ami számomra megfelelő lenne a weboldalam dinamikussá tétele során. MySql -ben sokkal nehezebb dolgom volt a sokszor túlbonyolított szintaktika miatt ezért is van olyan, ahol kénytelen voltam a felületen validálni az adatbázis helyett. Sok segítséget kaptam tanáraimtól és az adott nyelvek dokumentációiból. Összességében a vállalt funkcionalitást lefedi az alkalmazásom, az adatbázis megvalósításom pedig igényes.

Az iskolába való jelentkezéskor még semmi tapasztalatom nem volt a szoftverfejlesztéssel kapcsolatban. Köszönöm a szakmai és a rengeteg segítséget Kupcsikné Fitus Ilona tanárnőnek, aki nélkül nem lett volna értékes a szakdolgozatom. Továbbá külön köszönet a szaktársaimnak, akik segítettek amikor reménytelenül elakadtam.

Irodalomjegyzék

2021-03-10 https://etananyag.szamalk-szalezi.hu/file.php/13/adatbazis_alap/index.html

2021-03-10 https://etananyag.szamalk-szalezi.hu/file.php/13/adatbazis_alap/83_szlloda.html

2021-03-11 https://etananyag.szamalk-szalezi.hu/pluginfile.php/7128/mod_resource/content/0/adatb_sql/index.html

2021-03-11 https://etananyag.szamalk-szalezi.hu/mod/folder/view.php?id=2120

2021-03-15 https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html

2021-03-16 https://dev.mysql.com/doc/refman/8.0/en/create-procedure.html

2021-03-17 https://etananyag.szamalk-szalezi.hu/mod/folder/view.php?id=8525

2021-03-20 https://www.w3schools.com/php/php_oop_classes_abstract.asp

2021-03-20 https://www.php.net/manual/en/language.oop5.abstract.php

2021-03-21 https://www.w3schools.com/php/func_date_date.asp

2021-03-24 https://etananyag.szamalk-szalezi.hu/mod/assign/view.php?id=16115

Mellékletek

```
CREATE TABLE szobatipus (
        sz_tipus varchar(1) COLLATE utf8_hungarian_ci NOT NULL,
        agyak_szama tinyint(4)  NOT NULL,
        felszereltseg tinyint(4)  NOT NULL DEFAULT 1,
        komfort tinyint(4)  DEFAULT 1
);
ALTER TABLE szobatipus
        ADD PRIMARY KEY (sz_tipus);
ALTER TABLE szobatipus  ADD CONSTRAINT CHK_AgyakSzama
        CHECK (agyak_szama > 0);
ALTER TABLE szobatipus  ADD CONSTRAINT CHK_Felszereltseg
        CHECK (felszereltseg >= 1 and felszereltseg <= 5);
```

```
ALTER TABLE szobatipus  ADD CONSTRAINT CHK_Komfort
        CHECK (komfort >= 1 and komfort <= 5);
ALTER TABLE szoba
        ADD PRIMARY KEY (sz_szam), ADD KEY kapcs1 (sz_tipus);
ALTER TABLE szoba
        ADD CONSTRAINT kapcs1 FOREIGN KEY (sz_tipus)
        REFERENCES szobatipus (sz_tipus);
ALTER TABLE szobaarak
        ADD PRIMARY KEY (sz_tipus,mettol);
ALTER TABLE szobaarak
        ADD CONSTRAINT kapcs2 FOREIGN KEY (sz_tipus)
        REFERENCES szobatipus (sz_tipus);
ALTER TABLE szobaarak ADD CONSTRAINT CHK_FogalasNappalElotte
        CHECK (nappal_elotte >= 1 AND nappal_elotte <= 90);
ALTER TABLE szobaarak ADD CONSTRAINT CHK_SzobaAr
        CHECK (elotte_ar <> utana_ar);:
ALTER TABLE foglalasok
        ADD PRIMARY KEY (fog_szam),
        ADD KEY szoba (szoba),
        ADD KEY megrendelo (megrendelo
);
ALTER TABLE foglalasok
        ADD CONSTRAINT foglalasok_ibfk_1 FOREIGN KEY (szoba) REFERENCES szoba
        (sz_szam),
        ADD CONSTRAINT foglalasok_ibfk_2 FOREIGN KEY (megrendelo) REFERENCES
        partner (azon);
ALTER TABLE foglalasok  ADD CONSTRAINT CHK_Allapot
        CHECK (allapot <= 5 AND allapot >= 1);
ALTER TABLE foglalasok  ADD CONSTRAINT CHK_Tartozkodas
        CHECK (mettol < meddig);
ALTER TABLE lakik
        ADD PRIMARY KEY (fogl,vendeg), ADD KEY vendeg (vendeg);
ALTER TABLE lakik
        ADD CONSTRAINT lakik_ibfk_1 FOREIGN KEY (vendeg)
        REFERENCES partner (azon),
        ADD CONSTRAINT lakik_ibfk_2 FOREIGN KEY (fogl)
        REFERENCES foglalasok (fog_szam);
```

```
ALTER TABLE lakik ADD CONSTRAINT CHK_Lakik
    CHECK (erkezes < tavozas);
ALTER TABLE partner
    ADD PRIMARY KEY (azon);
ALTER TABLE partner ADD CONSTRAINT CHK_Jutalek
    CHECK (jutalek <= 100 AND jutalek >= 0);
ALTER TABLE felhasznalok
    ADD PRIMARY KEY (id);
ALTER TABLE felhasznalok
    MODIFY id int(11) NOT NULL AUTO_INCREMENT;
CREATE VIEW arbevetelek AS
    SELECT CASE WHEN mettol <= curdate() THEN 'Befolyt' ELSE 'Varhato' END AS
    megjegyzes, SUM(fizetendo) AS arbevetel
    FROM fizetendok
    GROUP BY CASE WHEN mettol <= curdate() THEN 'Befolyt' ELSE 'Varhato' END;
CREATE VIEW arbevetelekevente AS
    SELECT year(mettol) AS evben, SUM(fizetendo) AS arbevetel
    FROM fizetendok
    WHERE allapot > 1 AND allapot < 5
    GROUP BY YEAR(mettol) ;
    CREATE VIEW fizetendok AS
        SELECT f.fog_szam, f.szoba, f.mettol, f.meddig, f.megrendelo, f.fogl_datum,
        f.allapot, sz.sz_szam, sz_tipus, sz.kep,
        fizetendofgv(sz.sz_tipus,f.fogl_datum,f.mettol) * (to_days(f.meddig) -
        to_days(f.mettol)) AS fizetendo
        FROM (foglalasok f INNER JOIN szoba sz ON (f.szoba = sz.sz_szam)) ;
CREATE VIEW legtobbetfizetettmegrendelo AS
    SELECT megrendelo, SUM(fizetendo) AS arbevetel
    FROM fizetendok
    WHERE mettol <= curdate()
    GROUP BY megrendelo
    ORDER BY 2 DESC limit 1;
CREATE VIEW maiszobafizetesek AS
    SELECT fog_szam, megrendelo, szoba, mettol, meddig, fizetendo
    FROM fizetendok
    WHERE mettol = curdate() ;
CREATE VIEW top3legtobbetfizetettmegrendelo AS
```

```
        SELECT megrendelo, SUM(fizetendo) AS arbevetel
        FROM fizetendok
        WHERE mettol <= curdate()
        GROUP BY megrendelo
        ORDER BY 2 DESC limit 3;
CREATE FUNCTION allapotMegszoritasFg
(
foglszam INT(11)
)
RETURNS INT(11)
RETURN (
        SELECT allapot
        FROM foglalasok
        WHERE fog_szam = foglszam
)
CREATE FUNCTION fizetendofgv
(
        sztip VARCHAR(1),
        foglDat DATE,
        foglMettol DATE
)
RETURNS INT(11)
 RETURN
 (
        SELECT CASE WHEN DATEDIFF(foglMettol, foglDat) < nappal_elotte THEN
        utana_ar ELSE elotte_ar END
        FROM szobaarak
        WHERE sz_tipus = sztip AND mettol = ( SELECT MAX(mettol) FROM szobaarak
        WHERE sz_tipus = sztip AND mettol <= foglDat )
)
CREATE PROCEDURE allapotMegszoritas
(
        IN foglszam INT
)
BEGIN
        SELECT allapot
        FROM foglalasok
```

```
        WHERE fog_szam = foglszam;
END
CREATE PROCEDURE erkezesMegszoritas
(
        IN foglszam INT
)
BEGIN
        SELECT mettol
        FROM foglalasok
        WHERE fog_szam=foglszam;
END
CREATE PROCEDURE felhasznaloAPartnerben
(
        IN id INT
)
BEGIN
        SELECT p.azon
        FROM partner p INNER JOIN felhasznalok f on p.azon = f.id
        WHERE f.id = id;
END
CREATE PROCEDURE fizetendo
(
        IN foglszam INT
)
BEGIN
        DECLARE sztip varchar(1);
        DECLARE foglDat date;
        DECLARE foglMettol date;
        DECLARE foglMeddig date;
        DECLARE ara int;
        SELECT sz.sz_tipus, f.fogl_datum, f.mettol, f.meddig INTO sztip, foglDat, foglMettol,
        foglMeddig
        FROM foglalasok f INNER JOIN szoba sz ON f.szoba = sz.sz_szam
        WHERE fog_szam = foglszam;
        #SELECT sztip, foglDat, foglMettol, foglMeddig, DATEDIFF(foglMeddig, foglMettol);
        SELECT CASE WHEN DATEDIFF(foglMettol, foglDat) < nappal_elotte THEN
        utana_ar ELSE elotte_ar END INTO ara
```

```
        FROM szobaarak
        WHERE sz_tipus = sztip AND mettol = ( SELECT MAX(mettol) FROM szobaarak
        where sz_tipus = sztip and mettol <= foglDat );
        SELECT ara * DATEDIFF(foglMeddig, foglMettol) as fizetendo;
END
CREATE PROCEDURE hanyAgyas
(
        IN fogl INT
 )
BEGIN
SELECT szt.agyak_szama
FROM foglalasok f INNER JOIN szoba sz ON f.szoba = sz.sz_szam
        INNER JOIN szobatipus szt ON sz.sz_tipus = szt.sz_tipus
        WHERE fog_szam = fogl;
END
CREATE PROCEDURE igazolvanySzam
(
IN foglszam INT
)
BEGIN
        SELECT p.igazolvany_szam
        FROM partner p INNER JOIN foglalasok f ON p.azon = f.megrendelo
        WHERE f.fog_szam = foglszam;
END
CREATE PROCEDURE mennyivelUtkozik
(
        IN szoba1 INT,
        IN mettol1 DATE,
        IN meddig1 DATE
)
BEGIN
        SELECT COUNT(*) as ossz
        FROM foglalasok
        WHERE szoba = szoba1 AND (mettol BETWEEN mettol1 AND
        DATE_ADD(meddig1, INTERVAL -1 DAY) OR mettol1 BETWEEN mettol AND
        DATE_ADD(meddig, INTERVAL -1 DAY));
END
```