Szoftverarchitektúrák   
házi feladat  
  
Közösségi Parkolási portál  
  
*németh Balázs, KSPG27  
Nacsa Gábor, HB6463  
  
konzulens: Sipos márton*

Table of Contents

[2 Specifikáció - 3 -](#_Toc404717954)

[2.1 Fejlesztői csapat: - 3 -](#_Toc404717955)

[2.2 Részletes feladatleírás: - 3 -](#_Toc404717956)

[2.3 Technikai paraméterek - 5 -](#_Toc404717957)

[2.4 Use-case diagram: - 5 -](#_Toc404717958)

[3 A rendszer - 6 -](#_Toc404717959)

[3.1 A rendszer környezete - 6 -](#_Toc404717960)

[4 Adatmodell - 8 -](#_Toc404717961)

[5 Szoftverarchitektúra - 9 -](#_Toc404717962)

[6 Felhasználói felület - 9 -](#_Toc404717963)

[7 References - 11 -](#_Toc404717964)

# Specifikáció

A hallgatók feladata egy olyan térkép alapú online rendszer kifejlesztése, amelyik lehetővé teszi felhasználóinak, hogy megosszák az általuk ismert  ingyenes/előnyös parkolási lehetőségeket a többi felhasználóval. A rendszer tegye lehetővé fényképek feltöltését és a cím alapú keresést. Legyen elérhető mobil készülékekről.

## Fejlesztői csapat:

|  |  |  |
| --- | --- | --- |
| **Név** | **Neptun kód** | **E-mail cím** |
| Németh Balázs | KSPG27 | [nem841425bal@gmail.com](mailto:nem841425bal@gmail.com) |
| Nacsa Gábor | HB6463 | [nacsa.gbr@gmail.com](mailto:nacsa.gbr@gmail.com) |

A csapatban szerepek kiosztását a csapat kis mérete miatt nem tartottuk fontosnak. Az effektív csapatmunka érdekében létrehoztunk GitHubon egy nyilvános projektet, mely segítségével verziókövetés mellett végeztük a fejlesztés egészét [1].

## Részletes feladatleírás:

Célunk egy olyan webes alkalmazás készítése, melynek felhasználói menedzselhetik és megoszthatják egymással parkolóhelyeiket:

* Parkolóhely felvétele: A parkolóhely címének, árának, elérhetőségi sávjainak megadásával.
* Kép felvétele parkolóhelyről: A parkolókhoz opcionálisan fényképek is társíthatók a könnyebb megtalálás érdekében.
* Parkolóhelyek megjelenítése térképen: A parkolóhelyek böngészhetők az adott terület térképén való megjelenítésével, hogy a felhasználó a számára legkedvezőbb helyet találhassa járművének.

Parkolóhelyek szűrése: A parkolóhelyek szűrése az alábbi szempontok alapján végezhető, tetszőleges kombinációban:

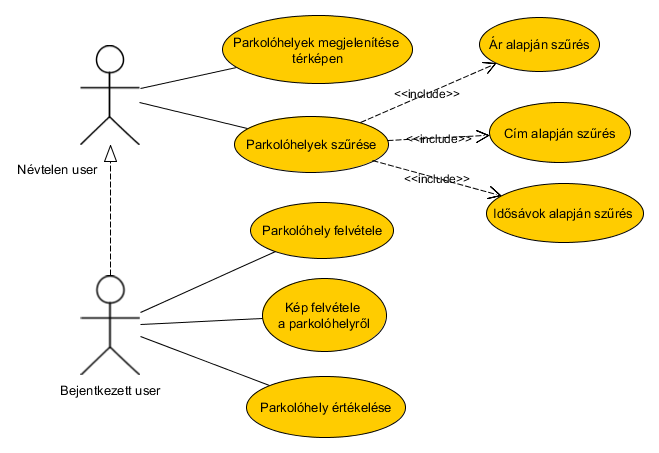
* + Ár alapján: Megadható felső árkorlát, így ingyenesre is szűrhetünk.
  + Cím alapján: Megadható, hogy legfeljebb mekkora távolságra lévő parkolóhelyeket szeretnénk megjeleníteni az adott címtől.
  + Idősávok alapján: Lekérhető, hogy a kívánt idősávban mely parkolók érhetők el.
* Parkolóhely értékelése: A felhasználók minősíthetik az egyes parkolóhelyeket, hogy mennyire találták azt összességében jónak, ezzel segítve a többi felhasználót.

Az alkalmazás implementálásához elengedhetetlen felhasználó kezelés megvalósítása. A felhasználók regisztrálhatnak, majd később ezzel a profillal vehetik igénybe az alkalmazás nyújtotta szolgáltatásokat.

## Technikai paraméterek

Vékonykliens alkalmazásunk egy webszerveren fog futni, így webböngészőből lesz elérhető a felhasználók számára, ezért a klienstől kizárólag egy webbrowser meglétét követeljük meg.

## Use-case diagram:



Ábra 1: A felhasználók által elérhető funkciók

Az Ábra 1-en látható use-case diagram mutatja, hogy az egyes felhasználók milyen funkciókat érhetnek el a programban. A bejelentkezéshez a felhasználónak jelszót és felhasználónevet kelljen megadnia és minden funkció elérhető legyen számára is, mint a névtelen felhasználónak. A funkciók részletes leírása a részletes feladatleírásban található.

# A rendszer

Az elkészített szoftver egy webes alkalmazás, mely lehetővé teszi felhasználóinak, hogy egy nyilvános rendszerbe felvegyenek parkolóhelyeket, és megoszthassák egymással az ezekről alkotott véleményeiket, értékelésüket, tapasztalataikat és a készített fényképeket.

A felhasználók regisztrálhatnak a rendszerbe, szerkeszthetik egymás által felvett parkolóhelyeket, jelölhetik őket Google Maps térképen és tölthetnek fel fényképeket a helyszínekről. Valamint lehetőségük van több szempont alapján szűrni a közösség által felhalmozott tudásbázisban.

## A rendszer környezete

Alkalmazásunk futtatásához csak egy webböngésző kell JavaScript és Web-Kit támogatással, így gyakorlatilag minden operációs rendszerről és minden böngészőből gond nélkül elérhető, mobil készülékről is.

A szoftver fejlesztése Java nyelven történt a Vaadin [2] keretrendszer 7.3.3-as verziójának segítségével, mely biztosítja számunkra a könyvtárakat a rugalmas felhasználói felület elkészítésére.

Az adatbázis rendszerünk megvalósításhoz a 3.8.7-es verziójú SQLite [3] köztulajdonban lévő (public domain), nyílt forráskódú könyvtárat használtuk, mely egy pehelysúlyú, SQL alapú adatbázis kezelő rendszer, mely nem igényel különálló infrastruktúrát az üzemeltetéséhez. Mindazonáltal teljes körűen nyújtja az SQL alapú nyelvek funkcióit, a lekérdezések rugalmas paraméterezhetőségét.

## A rendszer célja

Célkitűzésünk egy olyan közösségi webalkalmazás elkészítése, mely jó alapot nyújthat egy akár valóságban is üzemeltethető, könnyen kiterjeszthető rendszerhez, melyet felhasználói egyszerűen, szívesen használnak parkolóhelyek megosztására. A felhasználók így

## Telepítési útmutató

# Adatmodell

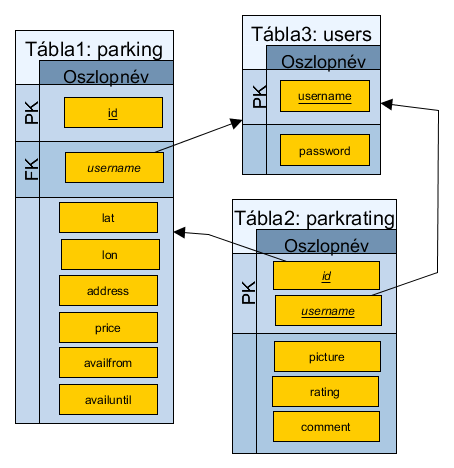
Az adatmodell tervezésénél különös figyelmet fordítottunk a rendszer kiterjeszthetőségére, illetve arra, hogy minél kevesebb redundanciával és hatékonyan tudjuk tárolni az adatokat.

Először megállapítottuk, hogy mit szeretnék gyűjteni a parkolóhelyekről, ahhoz hogy minden követelménynek meg tudjon felelni az alkalmazásunk és sokféle felhasználási lehetőséget biztosítson. Ezeket figyelembe véve az alábbi attribútumokat gyűjtöttük össze (zárójelben az adattáblában, illetve a programban használt elnevezések láthatóak) :

* Parkoló címe (address)
* GPS koordinátái (lat, lon)
* A parkolás ára (price)
* Hány órától vehető igénybe (availfrom)
* Hány óráig vehető igénybe (availuntil)
* Ki vette fel az adatbázisba (username)
* Képek a parkoló környékéről, helyéről (picture)
* Értékelés (rating)
* Kommentár (comment)
* Regisztrált felhasználók felhasználónevei (username)
* A felhasználók jelszavai (password)

## Adatbázis táblák

Az adatbázist három táblába dekomponáltuk. Aláhúzással jelöljük az adott tábla elsődleges kulcsát, dőlt betűvel pedig az idegen kulcsokat. A három táblázat oszlopai a Tábla 1, Tábla 2, Tábla 3 táblázatokban találhatóak. Az adatbázis tábla attribútumait kulcsok szerint csoportosítva láthatjuk (PK – elsődleges kulcs, FK – idegen kulcs, a többi attribútum pedig hagyományos). A táblázatok közötti nyilak jelzik, hogy az egyes idegen kulcsok mely adatbázistáblára mutatnak.



Tábla 1: A parking nevű adatbázistábla oszlopai Tábla 2: A parlrating nevű adatábzistábla oszlopai Tábla 3: A users tábla oszlopai

### A parkolóhelyek alapadatai

A *parking* tábla (Tábla 1) elsődleges kulcsa az azonosító (*id*) oszlop, melyet az adatbázis generál az adatrekord felvételekor. Az automatikus generálást az SQLite *AUTOINCREMENT* kulcsszavával valósítottuk meg, így a kulcs egyedisége is rögtön biztosított.

A *username* oszlop idegen kulcs a *users* táblából, értéke tetszőleges alfanumerikus karakterlánc lehet, melyet a felhasználó a regisztrálás során választ, ebben a táblában azt a felhasználót azonosítja, aki felvette a parkolóhelyet a rendszerbe.

A *lat* és *lon* oszlopok előjeles valós számként tárolják a parkolóhely GPS koordinátáit. A GPS koordináta redundáns adat a parkolóhely címével, azért döntöttünk mégis mindkét adat tárolása mellett, mivel a felhasználók felé a parkolóhely címét szeretnénk mutatni, míg az adatok felhasználásakor inkább a GPS koordinátákkal kényelmesebb és egyértelműbb számolni (távolság alapján szűrés, marker elhelyezés a térképen). Továbbá a két adat közötti konverzió nem túlságosan hibatűrő és nem szerettünk volna túl sűrűn lekérdezést intézni a GoogleMaps szolgáltatáshoz.

Az adatbázis négy fő információját tároló oszlop (id, username, lat, lon) egyike sem lehet NULL, vagyis hiányzó értékű, megadásuk mindig kötelező.

A *price* mezőbe felvehető a parkolóhely óradíja forintban megadva (HUF/h). Megadása nem kötelező, hiányzó adat esetén szűréskor wildcardként kezeljük. Az ingyenes parkolót 0 HUF/h beírásával jelezhetjük.

Az *availfrom* és *availuntil* mezők jelzik, hogy egy nap hány órától hány óráig áll rendelkezésre a parkolóhely. A kötelező beviteli formátum HH:MM, melyet a rendszer ellenőriz. Az üresen hagyott mező szintén wildcardként kezelődik, mindig elérhető parkolót 00:00 és 23:59 időpontok beállításával jelezhetünk.

### Parkolóhelyek értékelése

A *parkrating* (Tábla 2) tartalmazza a parkolók minősítéseit, a felhasználók hozzászólásait, illetve a feltöltött képeket. Itt az elsődleges kulcs egy kompozit kulcs, a *username* és az *id* mezők együttese azonosítja egyértelműen a tábla rekordjait. Így minden felhasználó minden parkolóhelyről pontosan egy értékelés hármast tud tárolni. Ezzel is szeretnénk elkerülni, hogy a felhasználók teljes fotóalbumok tárolására használják a közösségi parkolási portált.

A *picture* oszlop a feltöltött fájl szerver háttértárán lévő elérési útvonalát tárolja stringként, mely egyedien azonosítja a felhasználó fényképét az adott parkolóhelyről.

A *rating* oszlop értéke egy 0 és 5 közötti egész szám, ahol 0 jelenti a nem megadott értékelést, 1 a legrosszabbat, 5 pedig a legjobbat. A felhasználók „csillagokkal” vihetik be az értékelést, így ennek a bevitelnek a validációja automatikusan biztosított.

A *comment* mezőbe a felhasználó szöveges értékelést, illetve hozzáfűzést adhat meg az adott parkolóhelyről.

Az összetett kulcs mindkét komponense idegen kulcs a másik két tába elsődleges kulcsára, így ezek értéke nem lehet NULL, azonban a tábla összes többi oszlopa bármilyen kombinációban lehet.

### Felhasználói információk tárolása

A *users* (Tábla 3) adatbázis tábla tárolja a regisztrált felhasználók adatait. Ebben a táblában az elsődleges kulcs a *username* oszlop, mely egyediségét a rendszer biztosítja, egy felhasználó sem regisztrálhat, amíg nem sikerült neki egyedi alfanumerikus karaktersorozatot beírnia. Regisztrációkor a jelszót kétszer kell beírni, hogy minimalizáljuk a felhasználók gépelési hibáit.

A jelszó tárolása titkosítva történik, az SHA-1 hash függvény segítségével. A felhasználó bejelentkezésekor a beírt jelszó SHA-1 hash értékét hasonlítjuk össze az adatbázisban tárolt értékkel. Sikeres belépés csak akkor történhet, ha a két hash érték megegyezik.

A jelszavak ilyen módú bizalmas kezelésének köszönhetően, a jelszó csak a webszerveren futó alkalmazás memóriájában tárolódik nyílt formában, de ott is csak addig amíg a hash értékét ki nem számítjuk. Így a felhasználók jelszavai a webszerver esetleges feltörése révén sem kerülnek nyilvánosságra.

## Normál forma vizsgálat

Az adatbázison értelmezett funkcionális függések az alábbiak (kapcsos zárójellel jelezzük oszlopok egy halmazát, a halmaz elemeinek felsorolásával a kapcsos zárójelek között):

* id -> {username, lat, lon, address, price, availfrom, availuntil}
* {id, username} -> {picture, rating, comment}
* username -> password
* {lat, lon} -> address
* address -> {lat, lon}

Ez a függőség rendszer a {lat, lon} -> address és az address -> {lat, lon} függőségek miatt nem BCNF és 3NF, mivel sem {lat, lon}, sem address nem része szuperkulcsnak. Így az adatbázis legmagasabb normálformája 2NF.

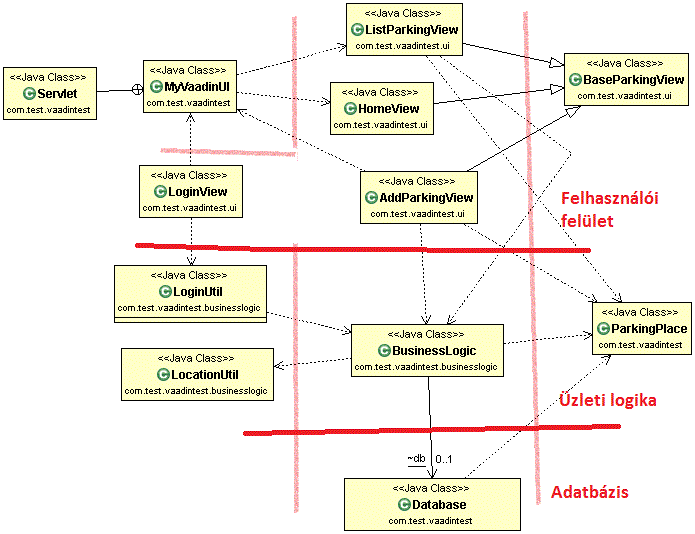
A viszonylag alacsony redundanciát biztosító normálforma ellenére ez egy tűrhető trade-off, amelyet a korábbiakban részletezett könnyű felhasználhatóság (address kijelzés GPS koordináta helyett), és hatékony működés (GoogleMaps kérések minimalizálása) érdekében tudatosan hoztunk.

# Architektúra

Alkalmazásunkat egy három rétegű architektúrával valósítottuk meg, melynek fő részei összefoglalásképp (részletezés később):

* Adatbázis réteg: Az adatmodell megvalósításáért felelős alacsony absztrakciós szintű réteg, mely lehetőséget nyújt a parkolóhelyek perzisztens tárolására, felhasználók kezelésére, adatok szűrésére, illetve módosítására
* Üzleti réteg: Mivel alkalmazásunk megvalósításához nem kell túl sok üzleti logika, ennek a rétegnek a szerepe legfőképpen a kiterjeszthetőség és rugalmas kezelhetőség biztosítása a feladata.
* Felhasználói réteg: A felhasználó felé nyújtott interfész biztosítja, hogy bárki, minimális számítógép használati előképzettséggel képes legyen használni a közösség által tetszőlegesen nagyra felhalmozott adatok összességét.

Az Ábra 2-n látható a program főbb osztályainak áttekintése, melyen berajzoltuk az architektúra három fő rétegét felépítő osztályok csoportjainak határát.



Ábra 2: Osztálydiagram áttekintés

Alul található az adatbázis réteg, mellyel a felhasználói felület az üzleti rétegen keresztül kommunikál. A felhasználói felületet felépítő osztályok közös ős osztálya a BaseParkingView osztály, mely önmagában nem példányosodik sehol.

Az ábrán látható továbbá a program belépési pontja, a MyVaadinUI nevű osztály, mely példányosítja szükséges nézeteket, illetve ezen az osztályon keresztül indul el az alkalmazás a webszerveren. Látható tovább néhány fontosabb segéd osztály, mely egy parkolóhely minden adatát tárolja, az adatbázitól való elkérés után (ParkingPlace), illetve hasznos segéd függvényeket nyújtó segédosztályok a LoginUtil és LocationUtil személyében.

## Adatbázis réteg

Business,

gui

## Objektummodell

Osztálydiagram, architektúra

# Felhasználói felület

Felhasználói felület bemutása, esetleg egykét példa

# Továbbfejlesztés

# Összefoglalás

# References

|  |  |
| --- | --- |
| [1] | Németh Balázs, Nacsa Gábor, "Közösségi Parkolási Portál," https://github.com/nacsa/szoftarch. [Online]. |
| [2] | Joonas Lehtinen, CEO, "Vaadin," vaadin.com. [Online]. |
| [3] | Public Domain, "SQLite - Small. Fast. Reliable. Choose any three.," http://www.sqlite.org/. [Online]. |