Szoftverarchitektúrák házi feladat

Rendszerterv

**Készítette:** Takács Rajmund *(Q9U7ZY)*

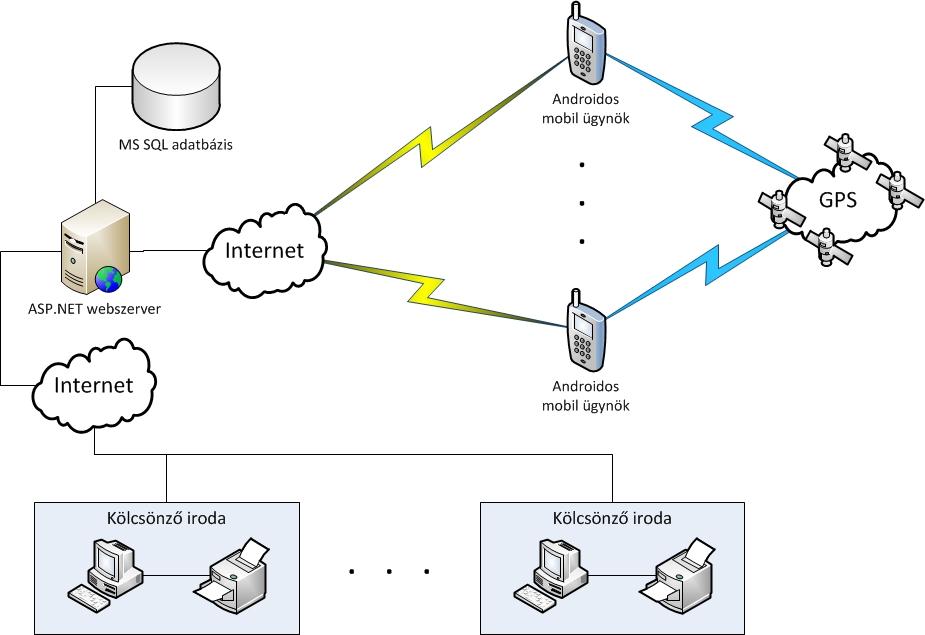
Timár István *(CLL7EW)*  
  
**Feladat:** Biciklikölcsönző rendszer  
  
**Konzulens:** Deák László

# Bevezető

A cél egy olyan biciklikölcsönző rendszer elkészítése volt, amely minden biciklit nyomon követ. A kölcsönzési díj nagyban függ attól, hogy milyen területeken jár a felhasználó. A szolgáltatás igénybevételének végén a számla automatikusan nyomtatásra kerül.

# Rendszerterv

Ahogy az ábrán is látszik a biciklikölcsönző rendszer négy nagyobb egységből épül fel. Mobil ügynök, hogy a bicikliket nyomon tudja követni a rendszer. Web szerver és adatbázis, az adatok tárolására, felvételére és megjelenítésére. Illetve egy nyomtató szerver minden kölcsönzőbe, hogy könnyedén tudjanak nyomtatni. Külön nyomtató szerverre azért van szükség, mert így nem kell minden kölcsönzőbe külön web szervert telepíteni, mégis megoldható a helyben nyomtatás. A web szervert pedig minden kölcsönzőből egy egyszerű web böngésző segítségével el lehet érni. További előny, hogy a nyomtató szervert Java nyelven írtuk meg, így a kölcsönzőkben akár ingyenes Linux disztribúciók is használhatóak, míg az alkalmazásszerver kihasználhatja a zárt, de könnyen karbantartható Microsoft technológiákat. Kliens oldalon azért választottuk az Android platformot, mert hihetetlenül gyors tempóban növekszenek az eladások. Tehát egyre többen használnak Android alapú okostelefonokat. A másik nagy előnye, hogy teljesen nyílt és ingyenes az alkalmazásfejlesztés erre az operációs rendszerre, amely hozzájárult a biciklikölcsönző rendszer költséghatékony elkészítéséhez.



# Architekturális tervezés

A rendszer megvalósítása során háromrétegű architektúrát alkalmaztunk az alábbiak szerint:

* *Megjelenítés:* a társ rendszer felé nyújt interfészt, és az ehhez kapcsolódó eseményeket kezeli le. Ennek a leggyakoribb megvalósítása a weboldal/kliensalkalmazás és az előállító logika. A biciklikölcsönző rendszerünkben ezt a réteget a böngészőben megjelenített weboldal, a telefonon futó alkalmazás, illetve a nyomtató szerver által nyomtatott számla valósítja meg.
* *Üzleti logika:* e réteg feladata az üzleti folyamatok futtatása, hosszú életű tranzakciók kezelése. Itt kerül sor az adatok szélesebb hatókört, több adattípust átölelő szabályainak kikényszerítése is. A mi esetünkben az üzleti logika a felhasználó bicikli használatából ered, amelyet a web alkalmazás kezel. Például, ha veszélyes zónában tartozódik, felárat kell fizetnie.
* *Perzisztencia:* az adatok tartós tárolásával foglalkozik. Itt történik meg a szűkebb hatókörrel rendelkező adatszabályok kikényszerítése és gyakran az objektum és relációs adatmodellek közötti leképezés is. A biciklikölcsönző rendszerünkről számos adatot tárolunk pl.: kölcsönzők listája, bicikli tulajdonságai, adminisztrátorok, kölcsönzéseket, illetve ezek kapcsolatát is. E réteget esetünkben az adatbázisszerver valósítja meg.

# Kommunikáció

A projekt elkészítéséhez többszintű kliens szerver architektúrát választottunk. A biciklikölcsönző rendszer három részből tevődik össze: kliens, alkalmazás szerver illetve nyomtató szerver. A kliens a szerverrel kétirányú kommunikációt végez. Ehhez három információátviteli protokollt valósítottunk meg, a REST architektúra ajánlásainak megfelelően.

A kliens **HTTP GET** kérés segítségével lekérheti az biciklikölcsönzők listáját, amit az alábbi JSON formában specifikáltunk.

KÉRÉS:

HTTP GET /api/Lenders

VÁLASZ:

array of [

{

"id" : int,

"latitude" : float,

"longitude" : float,

"name" : string,

"address" : string

}

]

A felhasználó az egyes biciklikölcsönzőről további adatokat is megtudhat. Ekkor a kliens szintén egy **HTTP GET** kéréssel fordul a szerverhez.

KÉRÉS:

HTTP GET /api/Lenders/{id: int}

VÁLASZ:

{

"latitude" : float,

"longitude" : float,

"name" : string,

"address" : string,

"description" : string

}

Fontos volt, hogy a kliens rendszeresen bejelentse pozícióját, mert a szerver csak így tudja nyomon követni a bicikliket illetve a megfelelő díjat kiszabni. Ezt egy **HTTP POST** kérés válaszban valósítottuk meg. A kliens mindig elküldi a bérlő azonosítóját és koordinátáit, amire a szerver visszaadja a bérlés állapotát illetve a kölcsönzési adatokat (pl: mekkora pénzösszeggel tartozik a kölcsönzőnek.)

KÉRÉS:

HTTP POST /api/Report

KÉRÉS TÖRZSE:

{

"session\_id" : int,

"latitude" : float,

"longitude" : float,

}

VÁLASZ:

{

"status" : [ OK\_NORMAL | OK\_DANGER | ERROR | END\_OF\_SESSION ],

"normal\_time" : int,

"danger\_time" : int,

"total\_balance" : int

}

A nyomtató szerver is kommunikál az alkalmazás szerverrel. Az első kommunikáció a nyomtatásra való feliratkozás, amely a következő JSON formátumban lett megvalósítva:

KÉRÉS:

HTTP PUT /api/PrintingSubscription

KÉRÉS TÖRZSE:

{

"lender\_id" : int,

"printer\_ip" : string,

"printer\_password" : string

}

A nyomtatásról való leiratkozás:

KÉRÉS:

HTTP DELETE /api/PrintingSubscription/{id: int}

Ha nyomtatási parancs érkezik (például beér egy kölcsönzőbe a bérlő), akkor az alkalmazás szerver a következő üzenetet küldi a megadott IP-re, TCP csomagként, JSON formátumban, melyből a nyomtatószerver összeállítja a számlát és kinyomtatja.

ÜZENET:

{

"lender\_name" : string,

"lender\_address" : string,

"bike\_name" : string,

"name" : string,

"address": string,

"items" : array of [

{

"title" : string

"amount" : int

"base\_unit\_price" : int

"vat" : float

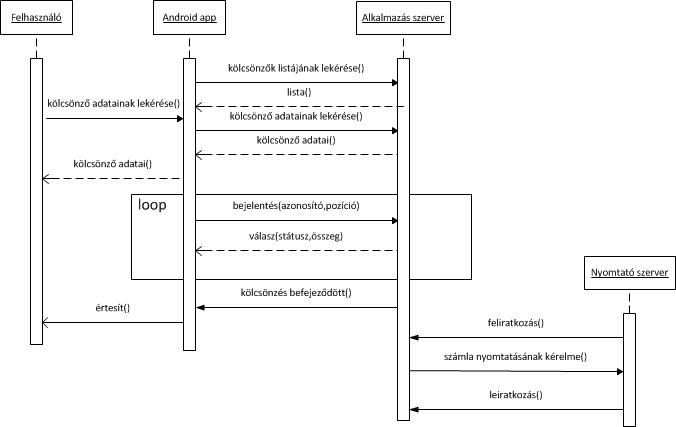
}

],

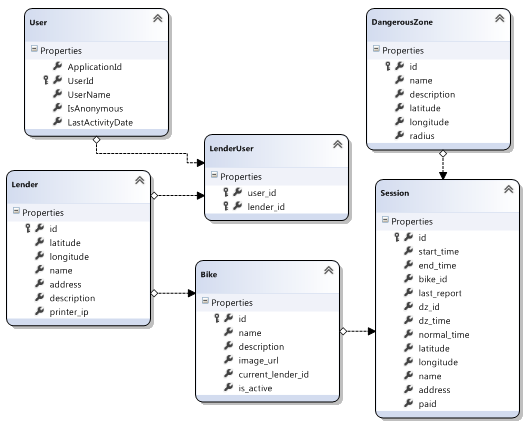
"total\_balance" : int

}

A kommunikációt az alábbi szekvencia diagram is szemlélteti:



# Adatbázis terv



A fenti ábrán látszik, hogy milyen táblák szerepelnek az adatbázisban. A **Bike** nevű tábla tartalmazza a biciklikre vonatkozó információkat. Mivel többféle zónát definiáltunk (normál és veszélyes) ezért ezeket is meg kell különböztetni egymástól. Mivel a veszélyes zóna a speciálisabb ezért erről tárolunk információkat a **DangerousZone** táblában. A **Lender** táblában tároljuk a biciklikölcsönzők tulajdonságait, *(Pl.: név, cím, GPS koordináták, stb…)* A **Session** táblában tároljuk a kölcsönzéssel kapcsolatos információkat. *(Pl.: Mikor kölcsönözte ki a felhasználó a biciklit, a felhasználó neve, címe, aktuális helye, stb…)* A **User** tábla tárolja a rendszer adminisztrátorait, míg a **LenderUser** tábla az adminisztrátorok és kölcsönzők összerendeléséért felelős.

# Összefoglalás

A projekt során megvalósítottunk egy kezdetleges biciklikölcsönző rendszert, amely teljesít minden specifikált igényt. A rendszer képes nyilvántartani a kölcsönzők, biciklik és kölcsönzések aktuális állapotát, továbbá igény szerinti és automatikus számlanyomtatást is tud végezni. Ez utóbbi a nyomtatószervernek hála helyben történhet, a kiválasztott kölcsönzőben, egységes, platform független interfész segítségével. A rendszer a szabványos kommunikációs protokollnak köszönhetően könnyen továbbfejleszthető, új funkciókkal bővíthető. Igény esetén a telefon alapú nyomkövetés a rendszer módosítása nélkül kicserélhető dedikált, titkosított protokollon kommunikáló nyomkövető egységekre is, így minimalizálva a hamis bejelentések kockázatát.

# Továbbfejlesztési lehetőségek

* Bérlők azonosítása egyedi adat *(pl. IMEI szám)* alapján, a bérlési azonosítón túl, hogy rosszindulatú bérlők ne tudjanak más bérlő nevében bejelenteni. A felhasználótól ugyanúgy csak a bérlési azonosítót *(session azonosító)* kérnénk be, de az első bejelentéskor rögzítenénk a telefon valamely egyedi azonosítóját, így más telefonról nem lenne lehetséges az adott bérlő nevében bejelenteni a pozíciót a továbbiakban.
* A kerékpárok élő nyomon követése a térképes nézeten, oldal újratöltés nélkül.
* Titkosítás bevezetése a rendszerbe, hogy a HTTP forgalom lehallgatásával ne kerüljenek ki értékes adatok illetéktelen kezekbe.
* A nyomtatószerver felismerhetné a saját IP címét és automatikusan konfigurálhatná UPNP protokoll segítségével a tűzfalat.
* Kölcsönzőnkként konfigurálható számlakép.
* Kerékpár előfoglalás katalógus alapján.
* A telefonról is lekérdezhető, térképes veszélyes zóna lista, hogy a bérlők is tudják merre ne biciklizzenek.
* Zónánként konfigurálható költségek.
* Alkalmazás fejlesztése iOS, WP7-re is.