# Webes alkalmazasásfejlesztés I. beadandó

Bittner Barnabás

April 17, 2017

# Contents

1	Feladat	3
2	Felhasználói dokumentáció         2.1 Felhasználói esetek	<b>4</b>
3	Fejlesztői dokumentáció           3.1 Core            3.2 UI            3.3 Adatbázis	5
4	Tesztelés	6

### 1 Feladat

Készítsük el egy bank ügyfelek kezelését, és az ügyfelekkel kapcsolatos tevékenységek adminisztrálását elősegítő rendszert. A webes felületen keresztül az ügyfelek érik el a bankolási funkciókat.

- 1. A főoldalon lehetőségünk van bejelentkezésre. Bejelentkezéskor meg kell adnunk a felhasználónevünket, jelszavunkat, számlaszámunkat (ha több van, akkor a legelsőt), valamint ellenőrző PIN kódunkat. Ezen felül a felhasználó választhat biztonságos üzemmódot is, ekkor minden művelet (számlatörténet lekérdezés, illetve átutalás) előtt a weblap ismét bekéri a felhasználó jelszavát. A bejelentkezést követően bármikor kijelentkezhet az ügyfél.
- 2. Sikeres bejelentkezés esetén lehetősége nyílik megtekinteni a számlái egyenlegeit (egy ügyfélhez legalább egy, de tetszőlegesen sok számla tartozhat), valamint azok történetét (átutalások, betétek, kivétek listája) egy hónapra visszamenőleg.
- 3. Lehetősége van új átutalást megadni (ha a számla nincs zárolva), ekkor ki kell tölteni az összeget, célszámla tulajdonosát, valamint számlaszámát, majd elküldhetjük az utalást, ekkor az összeg azonnal levonódik az egyenlegből (az átutalandó összeg nem lehet nagyobb a rendelkezésre álló egyenlegnél). Amennyiben a célszámla is a banknál van, akkor ott is meg kell jelennie az összegnek, mint befolyó utalás. A grafikus felületet a banki alkalmazottak használják a tranzakciók (betétek, kivételek és átutalások) kezelésére.
- 4. Az alkalmazottnak be kell jelentkeznie az alkalmazásba, ezt követően kiválaszthatja az ügyfelet, illetve annak bankszámláját.
- 5. Adott bankszámlára végezhet betétet, kivétet, illetve átutalást. Az első két esetben csak az összeget kell megadnia, míg a harmadik esetben (a webes felülethez hasonlóan) a célszámla adatait (számlaszám, tulajdonos) is. A tranzakció elküldésével az összeg azonnal levonódik/hozzáadódik az egyenleghez (átutaláskor, illetve kivételkor az összeg nem lehet nagyobb a rendelkezésre álló egyenlegnél). Amennyiben a célszámla is a banknál van, akkor ott is meg kell jelennie az összegnek, mint befolyó utalás.
- 6. Egy számla zárolható is, ekkor nem lehet semmilyen tevékenységet kezdeményezni a számlán (sem a grafikus, sem a webes felületen), amíg a zárolást fel nem oldja valamely alkalmazott. A zároláshoz a program kérjen megerősítést.

Az adatbázis az alábbi adatokat tárolja (ezek még nem feltétlenül a fizikai adattáblák):

1. ügyfelek (teljes név, felhasználónév, jelszó, PIN kód);

- 2. alkalmazottak (teljes név, felhasználónév, jelszó);
- 3. számlák (felhasználó, számlaszám, létrehozás dátuma);
- 4. tranzakciók (típus, forrás számlaszám, cél számlaszám, cél tulajdonos neve, dátum, összeg).

### 2 Felhasználói dokumentáció

A program használata rendkivül egyszerű csak egy böngészőprogramban meg kell nyitni a SmartBank weboldalát. Ha a felhasználónak még nincs fiókja regisztrálhat egyet. Regisztráció után azonnal be lehet jelentkezni, ahol a felhasználó eléri az összes saját bankszámláját, rajta az összesitett összeggel. Minden számlán le lehet kérdezni a számlatörténetet, mely után a program listázza az elmúlt havi tranzakciók listáját.

### 2.1 Felhasználói esetek

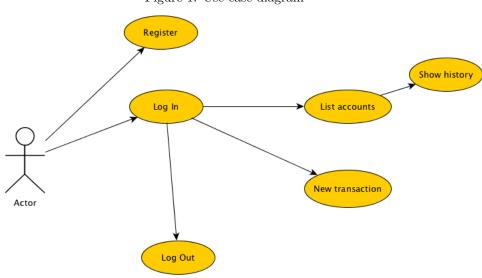


Figure 1: Use case diagram

### 3 Fejlesztői dokumentáció

A program alapvetően három részből áll, melyek: Core, UI, Adatbázis. Ezeket a következő szekciókban részletezem.

#### **3.1** Core

A program funkcionalitásának magja, itt találhatóak az üzleti logika részei. A megvalósitás C# WebAPI segitségével történik, a felhasználói felület HTTP hivásokkal kommunikál az üzleti logikával. A Core alapvetően a Domain Driven Design architectúrában készült, hexagonális architektúrát követve.

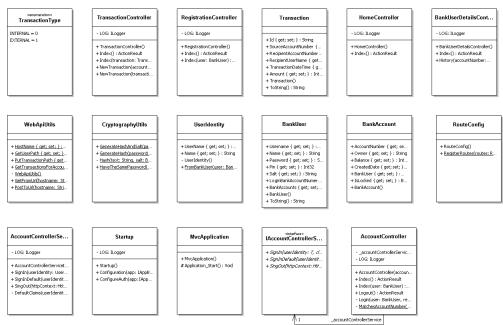
BankUsersController BankUserRepository WebApiConfig BankUser Transaction \_repository: BankUserRe.. - LOG: ILogger + BankUserRepository(dbC. + FindById(id: String) : Ban + Register(config: HttpConf. + Id { get; set; } : String +Username { get; set; } :. + Name { get; set; } : String + Password { get; set; } : 5. + SourceAccountNumber {. + RecipientAccountNumber + GetBankUsersById(id: Str. + Pin { aet; set; } : Int32 + PutBankUser(user: Bank... + TransactionDateTime { g... + Salt { get; set; } : String + Amount { get; set; } : Int. + BankAccounts { get; set; + Transaction() + ToString() : String + BankUser() + ToString() : String AbstractRepositoryl... BankAccount SmartBankDataModel WebApiApplication + AccountNumber { get; se. +BANK\_ACCOUNTS { get;. + WebApiApplication() + AbstractRepositoryImpl(d. + Owner { get; set; } : String + Balance { get; set; } : Int.. +BANK\_USERS { get; set;... + Transactions { get; set; }. # Application\_Start(): Void + FindById(id: TIdentity) :. + FindAll(): IEnumerable < + CreatedDate { get; set; }.. + SmartBankDataModel() # OnModelCreating(modelB + DeleteById(id: TIdentity). + DeleteAll(): Void + IsLocked { get; set; } : B.. + Count() : Int64 + Exists(id: TIdentity) : Bo. + BankAccount() + ToString() : String + Save(entity: TS) : TS + Save(entities: IEnumerab + Commit(): Void «interface»
IRepository<TEntity,... TransactionRepository TransactionController BankAccountReposit... + FindByTdfid: TTd): TEntity hank@ccountRenository + TransactionRepository(db. \_transactionRepository: T + FindAll() : IEnumerable< + DeleteBvId(id: TId): Void + FindTransactionsForAcco. - LOG: ILogger + DeleteAll() : Void bankAccountRepositor + TransactionController(tra. + Count(): Int64 + Exists(id: TId) : Boolean + Save(entity: 15) : 15 + ExecuteTranasction(pend. + Save(entities: IEnumerab.

Figure 2: Core Class Diagram

### 3.2 UI

A felhasználói felület C# ASP MVC technológiával készült el, bármilyen modern böngészőben megjelenithető. A felület HHTP hivasokkal kommunikál a core-al, amitől csak adatokat kap, amiket megfelelő módon megjelenit.

Figure 3: UI Class Diagram



### 3.3 Adatbázis

A program a működéshez használt adatokat fizikai adatbázisban tárolja, melyek tartalmazzák a regisztrált felhasználókat, azok néhány adatát. A felhasználók jelszavát enkriptálva tároljuk, esetleges adatszivárgás esetén ne lehessen (meglehetősen sok időt vegyen igénybe) a jelszavakat megszerezni. A tranzakciókat szintén az adatbázisban tároljuk, a bankszámlákkal együtt melyekhez eltároljuk a rajtuk lévő összeget. Az összeget csakis tranzakciókon keresztül lehet módositani.

BANK\_USERS

| USERNAME | Varchar(16) | | TRANSACTIONS | TRANS\_ID | Varchar(20) | | SRC\_ACC\_NUM | int | Int | SALT | Varchar(32) | | TRANS\_DATE | date time | MANUAL |

Figure 4: Database Diagram

## 4 Tesztelés

A program helyességét Unit testekkel ellenőriztük, melyek nagyban lefedik a program által szolgált lehetőségeket.