**EÖTVÖS LORÁND TUDOMÁNYEGYETEM**



Informatikai Kar

Média- és Oktatásinformatikai Tanszék

**Hiteligénylési folyamatot megvalósító webes alkalmazás**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dr. Menyhárt László Gábo** |  |  |  |  |  | **Piller Csaba** |
| **adjunktus, PhD** |  |  |  |  |  | **Programtervező informatikus Bsc** |

**Budapest, 2022**

Tartalom

[1 Bevezetés 3](#_Toc122385476)

[1.1 Az alkalmazás témája 3](#_Toc122385477)

[1.2 Az alkalmazás mögötti technológia 4](#_Toc122385478)

[2 Felhasználói dokumentáció 6](#_Toc122385479)

[2.1 Új hiteligénylés 6](#_Toc122385480)

[2.1.1 Ügyfél kiválasztása 6](#_Toc122385481)

[2.1.2 Az igénylés alapadatinak a megadása 7](#_Toc122385482)

[2.1.3 Jövedelmek és munkáltatók rögzítése 7](#_Toc122385483)

[2.1.4 Igénylő további adatai 8](#_Toc122385484)

[2.1.5 Bírálat és a bírálat eredménye 8](#_Toc122385485)

[2.1.6 Ajánlatok kiválasztása 9](#_Toc122385486)

[2.2 Hiteligénylés folytatása 9](#_Toc122385487)

[3 Fejlesztői dokumentáció 10](#_Toc122385488)

[3.1 Use case-ek és folyamatábrák 10](#_Toc122385489)

[3.1.1 Use case-ek 10](#_Toc122385490)

[3.1.2 Folyamatábrák 11](#_Toc122385491)

[3.2 Adatbázis szerkezet 13](#_Toc122385492)

[3.2.1 Relációk rövid ismertetése 14](#_Toc122385493)

[3.2.2 Az adatbázis használata 16](#_Toc122385494)

[3.2.3 Adatbázis rendszer 16](#_Toc122385495)

[3.3 Az Alkalmazásban használt keretrendszerek 17](#_Toc122385496)

[3.3.1 Vaadin 17](#_Toc122385497)

[3.3.2 Spring 19](#_Toc122385498)

[3.4 Build automatizálás 20](#_Toc122385499)

[3.5 Függőségek 21](#_Toc122385500)

[3.6 Alkalmazás felépítése 22](#_Toc122385501)

[3.6.1 Képernyők, komponensek 22](#_Toc122385502)

[3.6.2 Egyedi stílus 26](#_Toc122385503)

[3.6.3 Workflow modul 27](#_Toc122385504)

[3.6.4 Bírálati modul 28](#_Toc122385505)

[3.7 Üzemeltetés 28](#_Toc122385506)

[3.7.1 Fordítás 28](#_Toc122385507)

[3.7.2 Futtatáshoz szükséges eszközök 29](#_Toc122385508)

[3.8 Tesztelés 29](#_Toc122385509)

[4 Összefoglalás 30](#_Toc122385510)

[5 További fejlesztési lehetőségek 32](#_Toc122385511)

[6 Irodalomjegyzék 33](#_Toc122385512)

[7 Mellékletek 34](#_Toc122385513)

# Bevezetés

A szakdolgozatom keretében egy hiteligénylési folyamatot támogató webes alkalmazás elkészítését tűztem ki célul. Több éves tapasztalattal rendelkezem kockázatkezelési területen, amit egy hazai hitelintézet alkalmazásában szereztem. A tanulmányaim megkezdésekor még üzleti, megrendelői oldalon dolgoztam, így ilyen megközelítésből találkoztam a lakossági hitelek igénylési folyamatival, akár meglévő rendszerek üzemeltetéséről, tovább fejlesztéséről, akár új rendszerek bevezetéséről volt szó.

Az alkalmazásokat tekintve több lehetséges technológiai megvalósítással is találkoztam. Ezek jellemzően - bár nem kizárólag - dobozos termékek voltak és mindig is érdekelt, hogy modern és nyílt forráskódú keretrendszerekre, platformokra építve megvalósíthatóak-e az ilyen folyamatok.

## Az alkalmazás témája

Egy bank jellemzően több szegmensben, többféle terméket értékesít eltérő csatornákon. Ezek közül a szakdolgozatom témájaként a lakossági szegmens egy termékét, a személyi kölcsönt választottam. A két legforgalmasabb és leginkább ismert csatorna a bankfiók és az online hiteligénylés. Bár az online igénylések súlya évről évre nő, én mégis a jelenleg még mindig legnagyobb forgalmat lebonyolító és talán kevesebbszer bemutatott bankfióki csatornát választottam. Ebből adódóan az alkalmazás felhasználó a banki alkalmazottak.

A hiteligénylési folyamat tervezése során az elsődleges feladatukból adódóan az értékesítési és kockázatkezelési területek között valamilyen szintű érdekellentét áll fenn. Míg értékesítési szemszögből a cél egy minél rövidebb, túlzással élve akár „három gombos” folyamatot követő folyósítás lenne az ideális, addig a hiteligénylési döntés meghozatalához lehetőleg minél több adatra van szükség. Ugyanakkor a piaci trendek alapján egyre inkább a rövidebb folyamatok felé tolódik a hangsúly. Ma már egy hiteligénylés felhasználó szemmel sokkal közelebb áll az online vásárlási folyamatokhoz, mint mondjuk egy gépjármű biztosítás megkötéséhez. Ami persze nem azt jelenti, hogy a back-end oldali folyamatokat is hasonló komplexitásúak. Az előzőek miatt az általam készített alkalmazás is ezt az irányt képviseli.

## Az alkalmazás mögötti technológia

A webfejlesztés témaköre érthető módon szinte minden félévben jelen volt legalább egy kurzus keretében. Ennek köszönhetően az alapokon túlmenően több programozási nyelvvel, front- és back-end keretrendszerrel is megismerkedtünk. Ugyanakkor fontosnak tartottam, hogy a szakdolgozat elkészítése a tanulmányaim részeként további ismeretekkel gazdagítsa a tudásomat, valamint olyan területeken segítsen fejlődni, amit a jelenlegi munkahelyemen is hasznosítani tudok. Ez utóbbi okból az alkalmazásban elsőlegesen használt programozási nyelv a Java, míg konkrét keretrendszerként a front- és back-end-et is magába foglaló Vaadin-t választottam.

A Vaadin egy nyílt forráskódú platform Java back-end-el rendelkező modern webes alkalmazások készítésére. Egyetlen „opinionated” webfejlesztési stack-be integrálva tartalmazza a UI komponenseket, keretrendszereket és eszközöket. Szemben például az Angular-ral vagy React-el, a Vaadin a back-end-et is magába foglalja. A fentiek miatt úgy gondolom, hogy nagyon nagy forgalmú webes alkalmazások fejlesztésére nem feltétlenül, de a célnak választott banki alkalmazásnak kifejezetten megfelel. A banki környezetben elterjedt a Java használata és a Spring keretrendszerrel párosítva lehetővé teszi az alkalmazások gyors fejlesztését. A szerver oldali architektúra csökkenti az alkalmazás sebezhetőségét, ami szintén fontos szempont banki környezetben. A Vaadin az azonos nevű cég ajánlása[[1]](#footnote-1) alapján egyetlen szerveren is képes 2.000 – 15.000 konkurens felhasználó kiszolgálására, ami a hazai bankszektorban elegendő, akár az ügyfél általi online igénylések kiszolgálására is. Mivel az állapot kezelése a szerver oldalon történik, ezért horizontális skálázódás csak sticky session használatával lehetséges.

Nem utolsó sorban szintén személyes érv volt a Vaadin mellett, hogy így ismét egy újabb keretrendszert ismerhettem meg.

# Felhasználói dokumentáció

|  |  |
| --- | --- |
| **Megjegyzés** | Az alkalmazás egy elképzelt hitelintézet, a Bank Höjd hiteligénylési rendszere. A továbbiakban a programra csak „alkalmazás”-ként hivatkozom. |

Az alkalmazás célja, hogy a személyi kölcsön hiteligénylési folyamatot szabályozott keretek között támogassa. Az alkalmazás elindításakor a felhasználó két opció közül választhat:

* új hiteligénylési folyamat indítása
* igénylés folytatása

Kép

## Új hiteligénylés

### Ügyfél kiválasztása

Új hiteligénylés esetén első lépésként szükséges a hitelt igénylő ügyfél megadása. Ez két módon történhet.

Amennyiben az ügyfél korábban már igénybe vett valamilyen szolgáltatást, akkor az ügyfelet, illetve személyes adatait az alkalmazás már tárolja. Ebben az esetben az ügyfél név alapján kereshető. Kép

A keresés eredménye egy táblázatban jelenik meg, ahol az ügyfél személyes adatai (születési dátum, anyja neve) alapján kell a megjelent ügyfelet azonosítani és kiválasztani. Kép

Mivel előfordulhat, hogy az ügyfél nem emlékszik a korábbi kapcsolatra, ezért minden esetben szükséges az ügyfél keresést elvégezni.

Új ügyfél létrehozására van szükség, amennyiben az ügyfél még nem rendelkezik bankkapcsolattal. Ezt a képernyő alján lévő gomb megnyomásával lehet kezdeményezni. Kép

Ilyenkor egy új képernyő szükséges megadni az ügyfél személyes adatait, címét, valamint telefonos és elektronikus elérhetőségét (e-mail cím). Minden adat megadása kötelező. Kép

Amennyiben a rögzíteni kívánt ügyfél már létezik (név, születési dátum és anyja neve megegyezik) egy felugró ablak figyelmeztet. Ilyenkor két lehetőség közül választhatunk:

* az igénylés folytatása a rendszerben már korábban rögzített ügyfél adatokkal vagy
* az igénylés folytatása a korábban már rögzített ügyféllel, frissítve a korábban mentett adatokat a most megadottakkal (pl. cím).

### Az igénylés alapadatinak a megadása

Az ügyfél rögzítését követően szükséges megadni az igényelt hitelösszeget és futamidőt. Ezek minimális és maximális értéke megjegyzésként feltüntetésre kerül. A megadott hitelösszeg és futamidő függvényében automatikusan kiszámításra kerül a törlesztőrészlet. Amennyiben az igényelt paraméterek megfelelnek az ügyfélnek az igénylés folytatható.

### Jövedelmek és munkáltatók rögzítése

Ezután következik az ügyfél jövedelmének rögzítése. Ennek első lépése a munkáltatók rögzítése, amennyiben az ügyfél alkalmazott vagy vállalkozó. Munkáltató rögzítése során csak a munkáltató beazonosításához szükséges minimális adatkör (cég neve, adószám), valamint a munkaviszony időtartamára vonatkozó adatok megadása kötelező. Kép

Munkáltató rögzítése nem kötelező, de annak hiányában a későbbiekben nem rögzíthető olyan jövedelem, ami munkáltatóhoz kapcsolódik (pl. munkabér). Több munkáltató is megadható.

A munkáltatók rögzítését a jövedelmek megadása követi, amihez csupán a jövedelem típusát, összegét és az esetlegesen kapcsolódó munkáltatót kell megadni. Kép

Az alkalmazás csak abban az esetben engedi a munkáltató megadását, ha ez értelmezett (pl. nyugdíjnál nem), de ezekben az esetekben ez kötelező. Természetesen amennyiben szükséges a munkáltatók rögzítésére még ezen a ponton is sor kerülhet.

### Igénylő további adatai

A következő képernyőn további, az ügyféltörzs részét nem képező információkat kell megadni az ügyféllel kapcsolatban:

* legmagasabb iskolai végzettség
* családi állapot
* háztartás létszáma
* eltartottak száma
* fennálló hitelek törlesztőrészlete

A fennálló hitelek törlesztőrészlete mezőbe az ügyfél hiteleinek összesített törlesztőrészletét szükséges rögzíteni. Minden mező kitöltése kötelező.

### Bírálat és a bírálat eredménye

Ezt követően az alkalmazás elvégzi az automatikus bírálatot, ami pár másodpercet vesz igénybe, a felhasználónak nincs teendője. A bírálatot követően megjelenik a bírálat eredménye, ami lehet engedélyezés vagy elutasítás. Mind a két esetben felsorolásra kerülnek a bírálati elemek, melyeket kibontva látható, hogy az igénylés milyen feltételeket teljesített, illetve mi nem volt megfelelő. Kép

Elutasítás esetén az igénylési folyamat véget ér, további teendő nincs.

Engedélyezés esetén az igénylés folytatható.

### Ajánlatok kiválasztása

Az igénylés engedélyezése esetén szükséges kiválasztani a hitel végleges paramétereit (hitelösszeg, futamidő). Amennyiben az ügyfél jövedelmi helyzete alapján elérhető, akkor az első ajánlat minden esetben az igényelt hitelösszeget és futamidőt tartalmazza. Az alkalmazás további ajánlatokat is kalkulál, szintén az ügyfél jövedelmi helyzetét figyelembe véve. Kép

Az ajánlat kiválasztását követően a folyamat véget ér

## Hiteligénylés folytatása

Új igénylés indításán kívül lehetősége van igénylés folytatására is. Ekkor az igénylések egy táblázatban kerülnek megjelenítésre, majd a megfelelő igénylést kiválasztva az igénylés annak státuszától függő képernyőn folytatható. Kép

Az igénylés státusza minden lépést követően automatikusan frissül, külön mentésre nincs szükség. Az alkalmazás a folyamaton kisebb lépésekre bontva vezeti végig a felhasználót, így a képernyőn berögzített adatok mentésére csak a következő folyamatlépésre történő navigálással van lehetőség, részleges kitöltés nem menthető.

# Fejlesztői dokumentáció

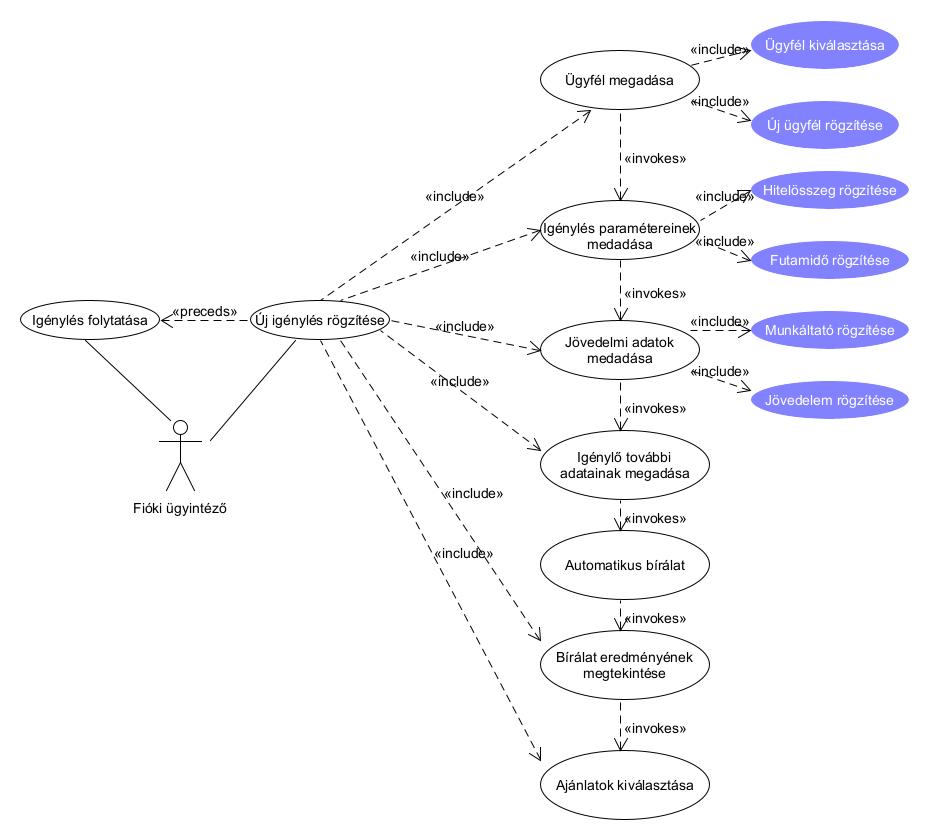
## Use case-ek és folyamatábrák

### Use case-ek

Az alkalmazás jelenleg két alapvető use case-el rendelkezik:

* új igénylés rögzítés
  + a felhasználó először keres a rendszerbe már rögzített ügyfelek között név alapján. Amennyiben az ügyfél megtalálható kiválasztja, különben új ügyfelet rögzít. Egy ügyfél többször nem rögzíthető, ezt az alkalmazás ellenőrzi.
  + a felhasználó megadja az igényelt hitelösszeget és futamidőt
  + a felhasználó rögzíti az igénylő munkáltatóját (amennyiben rendelkezik vele vagy vállalkozó) és az ügyfél jövedelmét. Több munkáltató és jövedelem is rögzíthető.
  + a felhasználó megad további adatokat az ügyfélről, mint igénylő: legmagasabb iskolai végzettség, családi állapot, háztartás létszáma, eltartottak száma, fennálló hitelek törlesztőrészlete.
  + a háttérben megtörténik az automatikus bírálat és ennek eredménye megjelenítésre kerül a felhasználónak. Megjelenítésre kerül az összesített eredmény, valamint bírálati elemenként a szabályok eredménye. Elutasítás esetén az igénylés nem folytatható.
  + a felhasználó választhat a kalkulált ajánlatok közül. Csak egy ajánlat választható.
* igénylés folytatása
  + a felhasználó választhat az igénylések listájából. Az igénylés státuszának megfelelő ponton folytatódik a folyamat.

Az alkalmazást jelenleg egy aktor használja, a fióki ügyintéző. Az alkalmazás bizonyos mértékben módosítható az adatbázis módosításával (pl. új jövedelem rögzítése), de mivel ez az alkalmazástól mondhatni függetlenül történik, ezért üzemeltetőt nem tüntetünk fel aktorként.



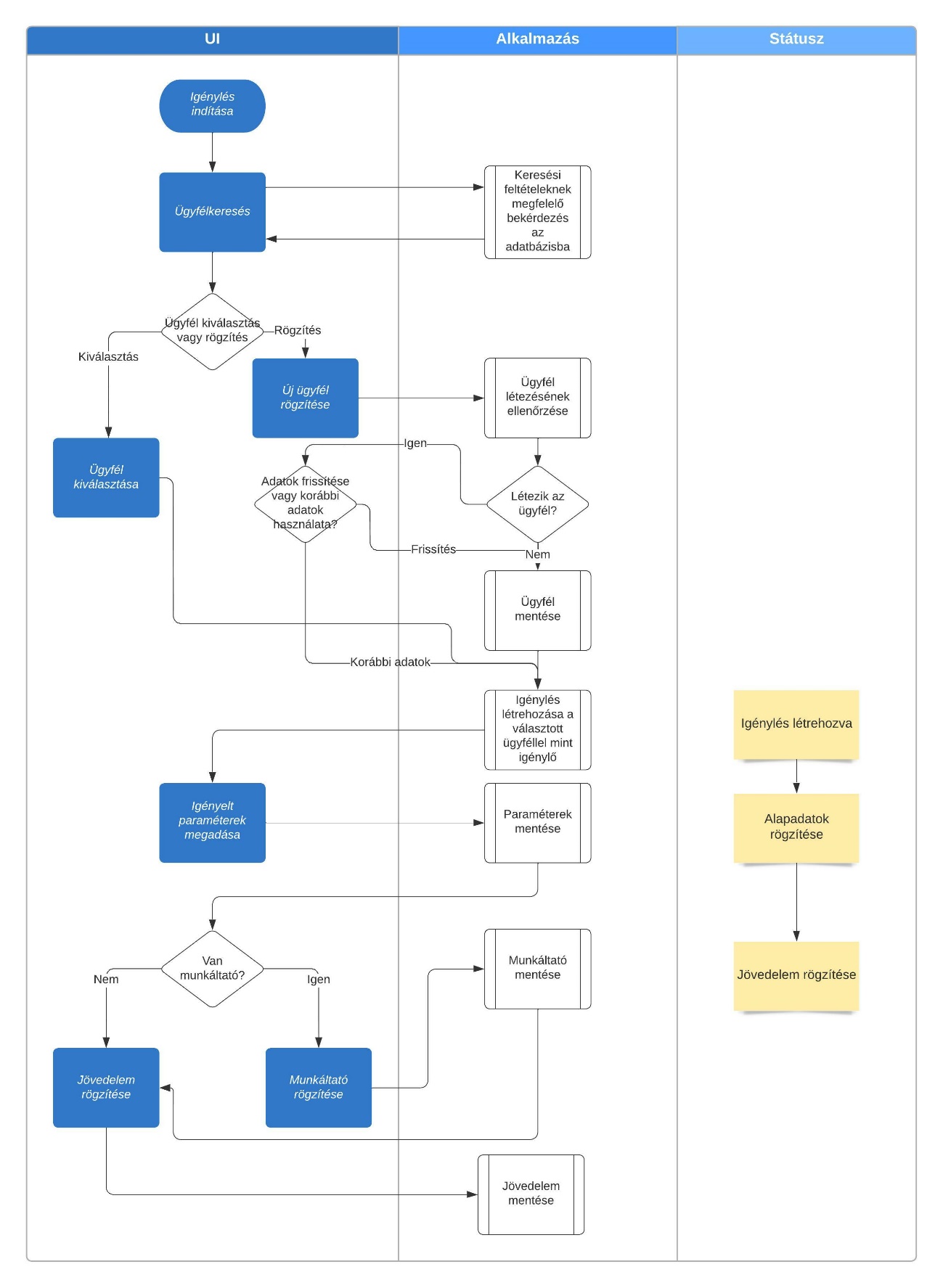
1. ábra Use case diagram

### Folyamatábrák

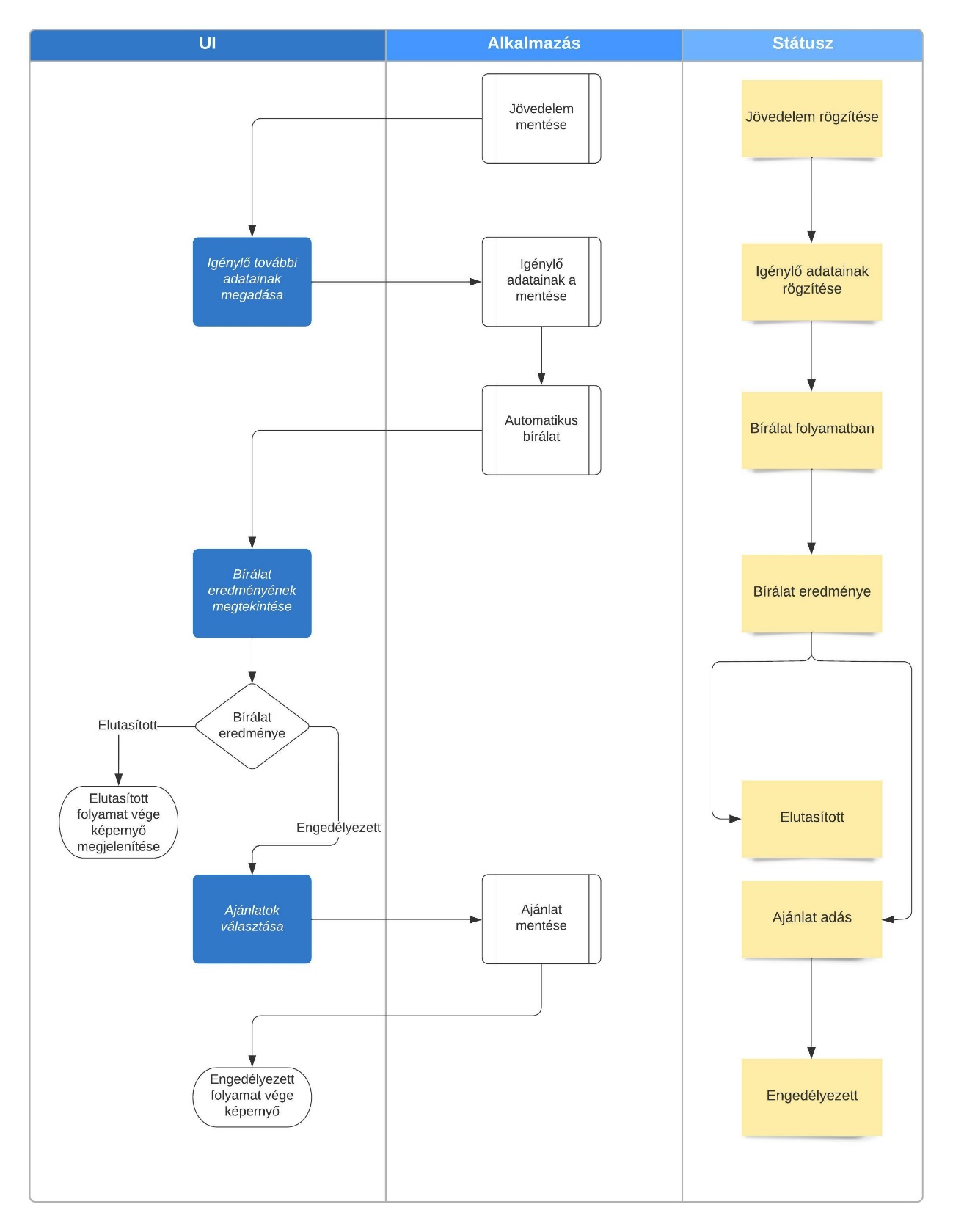
Több lépésből az új igénylés rögzítése use case áll. Ezek magasról nézve szekvenciális lépések, azonban bizonyos pontokon az igénylés sajátosságaitól függően döntése van a felhasználónak.

Ezt a folyamatot az alábbi uszodadiagram szemléltetni. Az ábrán a felhasználói felület és az alkalmazás különválasztásra került, azonban fontos megjegyezni, hogy a használt keretrendszer (Vaadin) működése miatt a felhasználói felület a szerveroldali állapot tükröződése, így technikai értelemben nem feltétlenül helytálló a megkülönböztetés.

A harmadik sávban az aktuális ponton érvénybe lépő új igénylési állapotok kerültek feltüntetésre.



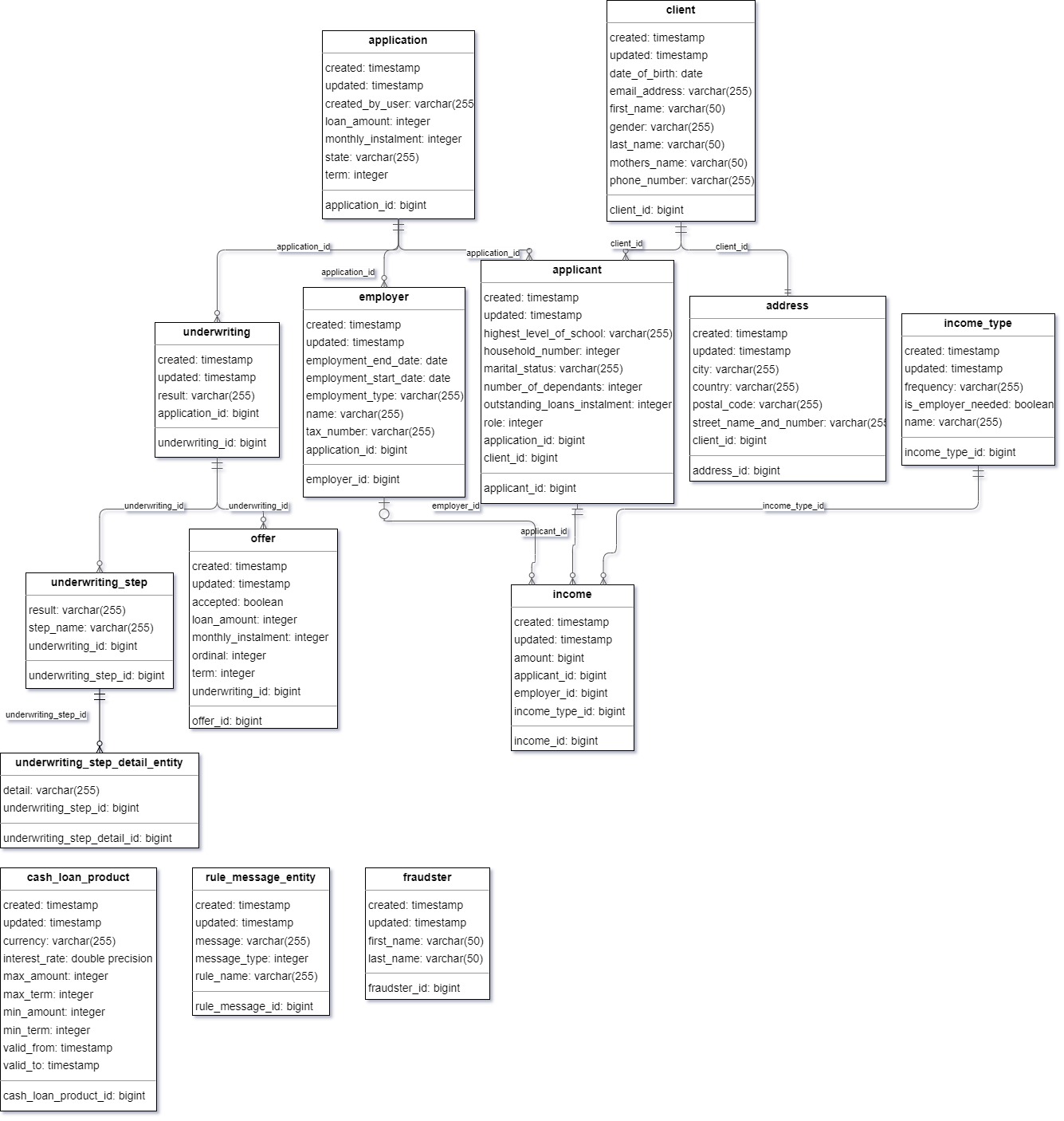
2. ábra Új igénylés folyamatábrája 1.



3. ábra Új igénylés folyamatábrája 2.

## Adatbázis szerkezet

Az alkalmazás által használt relációkat és azok kapcsolatait az alábbi ábra szemlélteti.



4. ábra Relációk és kapcsolataik

A relációk közötti kapcsolatok Crow’s Foot jelöléssel kerültek megadásra.

### Relációk rövid ismertetése

|  |  |
| --- | --- |
| Reláció neve | Leírás |
| Application | A hitel igénylés, egy folyamatot az igénylés tábla egy előfordulása reprezentál. Minden nem konfigurációs (a működést befolyásoló) adat az igénylésből vezethető le. |
| Client | A reláció az ügyfeleket tartalmazza, jelen esetben a bank ügyféltörzsét reprezentálja. Ebből kifolyólag minden magánszemély csak egyszer szerepelhet, de egy ügyfél több igénylésben is lehet igénylő. Egy ügyfél az igényléshez az Applicant reláción keresztül kapcsolódik. |
| Applicant | Igénylő, az ügyfeleket és igénylést összekapcsoló reláció. Bizonyos attribútumai az ügyfelet leíró, de igénylésenként változó adatok, tehát nem az ügyfél reláció részei. |
| Address | Az ügyfél címét leíró adatok. |
| Income | A jövedelmeket tartalmazó reláció. Egy igénylőhöz több jövedelem előfordulás is kapcsolódhat. |
| Income type | A jövedelem típusát leíró reláció, ami meghatározza, hogy a jövedelemhez kapcsolódik-e munkáltató, illetve annak gyakoriságát. |
| Employer | Munkáltató, igénylés szinten egyedi, adatbázis szinten nem, ezért ezt a megkötést az alkalmazás biztosítja. Egy munkáltatóhoz több jövedelem is kapcsolódhat. |
| Underwriting | Az automatikus bírálatot és annak eredményét reprezentáló reláció. |
| Underwriting step | Bírálati lépés, egy bírálat több lépésből áll, melyek külön-külön rendelkeznek eredménnyel. |
| Underwriting step detail | A bírálati lépésben kiértékelt szabályok eredményét reprezentáló reláció. |
| Rule message | Az egyes szabályok engedélyezés vagy elutasítás esetén megjelenítendő üzenetei. |
| Offer | Egy bírálat során kalkulált ajánlatok. |
| Cash loan product | Személyi kölcsön termék paraméterei, konstrukció. Az alkalmazás jelenleg egy adott időpontban egy érvényes konstrukciót feltételez. |
| Fraudster | Korábban visszaélésben részt vett ügyfelek listája. |

### Az adatbázis használata

Az alkalmazás az adatbázist több absztrakción keresztül kezeli.

Ha magát a JDBC-t nem tekintjük absztrakciónak, akkor az első ilyen szint a Java Persistence API (JPA). Az objektum-reláció leképezés annotációk használatával történt, ennek egy példája látható az alábbi ábrán.



5. ábra Példa az objektum-reláció leképezésre annotációk használatával

A JPA specifikáció konkrét megvalósítása az alkalmazásban Hibernate ORM használatával történik, ami a „spring-boot-starter-data-jpa” függőség által alapértelmezetten használt JPA implementáció.

A második absztrakciós szint a Spring Data JPA keretrendszer, aminek használatával csökken a boilerplate kód mennyisége.

A fentieknek megfelelően a relációknak megfelelő Entity osztályokat a *pillercs.app.vaadin.data.entity* package, míg az entitások eléréséhez használt repository osztályokat a *pillercs.app.vaadin.data.repository* package tartalmazza.

### Adatbázis rendszer

Az alkalmazás jelenleg egy in-memory adatbázist (H2) használ, de az előzőekben kifejtett JPA és Spring Data használat miatt, ez könnyen módosítható más adatbázis rendszerre is. Például PostgreSQL használatához csak a postgresql függőség felvétele szükséges a pom fájlba, illetve az application.properties fájlt kell módosítani az adatbázis címével és hitelesítő adatokkal.

## Az Alkalmazásban használt keretrendszerek

### Vaadin

Az alkalmazást leginkább meghatározó keretrendszer a Vaadin. A Vaadin egy nyílt forráskódú platform, Java back-end-el rendelkező modern webes alkalmazások készítésére[[2]](#footnote-2). Egyetlen „opinionated” webfejlesztési stack-be integrálva tartalmazza a UI komponenseket, keretrendszereket és eszközöket. A Vaadin keretrendszer részletes ismertetése nem célja a dokumentációnak, azonban az alábbiakban összegzésre kerülnek az alkalmazás működésének megértéséhez leginkább szükséges funkciók.

A Vaadin egy single page alkalmazás, ami a UI állapotát teljes mértékben a back-end-en tartja nyilván, a böngészőben megjelenített weboldal ennek a leképezése. Az alkalmazás view-kból épül fel. Ahhoz egy view elérhető legyen szükséges egy elérési útvonallal rendelkeznie, ami a *@Route* annotációval érhető el.



6. ábra Egy view-ként funkcionáló osztály deklarációja

Egy view Vaadin komponensekből épül fel. A komponensek a W3C szabványnak megfelelő egyedi HTML elemek. Az alkalmazás építőkövei ezek a keretrendszer által definiált komponensek és az azokból kiterjesztéssel és kompozícióval készített egyedi komponensek.

#### Komponensek

Az alkalmazásban használt főbb Vaadin komponensek[[3]](#footnote-3):

|  |  |
| --- | --- |
| Komponens | Rövid leírás |
| App Layout | Egységesített elrendezések építéséhez használt komponens. Három szekcióból áll: horizontális navbar, összecsukható drawer és a tartalom. Az alkalmazás egy layout-ot használ (MainLayout), ami egy drawer-er rendelkezik. |
| Vertical-/Horizontal Layout | A Vaadin két alap elrendezést tartalmaz, egy vertikális és egy horizontális elrendezést. Ezek rendre vertikálisan és horizontálisan jelenítik meg a bennük elhelyezett komponenseket. A komponensek a hozzáadás sorrendjében jelennek meg, de lehetőség van új komponens beszúrására is. |
| Grid | A grid egy táblázat, ami sorokban és oszlopokban jeleníti meg adatokat. Szövegen túl egyéb elemek és komponensek (pl. gomb) is megjeleníthető. |
| Form Layout | A Form Layout segítségével több oszlopos, címkézett, reszponzív űrlapok készíthetőek. |
| Input Fields | Szövegek és számok bevitelére szolgáló komponensek. |
| Button | A Button (Gomb) komponens lehetővé teszi a felhasználók számára, hogy műveleteket hajtsanak végre. Ikonok és szövegfeliratok megjelenítését is támogatja. |

#### Események

A Vaadin Flow esemény vezérelt programozást használ a felhasználói interakciók kezelésére. A felhasználó a UI-val történő interakciója a böngészőben eseményeket vált ki. Az események továbbításra kerülnek a böngészőből a szerveroldalra, ahol az alkalmazás kódja kezelheti ezeket. Minden olyan komponenshez, amivel a felhasználó interakcióba léphet egy eseményfigyelőt kell implementálni.

#### Vaadin Pro

A Vaadin egy nyílt forráskódú keretrendszer, de több előfizetési modellel is rendelkezik. A Pro szint tartalmaz ingyenesen nem elérhető komponenseket is, de az alkalmazás ezeket nem használja.

### Spring

A Vaadin mellett szintén hasonló mértékben meghatározó keretrendszer az alkalmazás működésében a Spring. A Spring egy nyílt forráskódú keretrendszer Java alkalmazások készítéséhez. Az alkalmazás a Spring több alapvető funkcióját és alprojektjét vagy modulját is használja.

#### Spring boot

Azon túl, hogy az alkalmazás a Spring keretrendszert használja elmondható, hogy az egy Spring Boot alkalmazás.

A Spring Boot a Spring csapat által fejlesztett keretrendszer, ami megkönnyíti az új Spring alkalmazások létrehozását és fejlesztését. Több alapértelmezett kód és annotáció konfigurációt biztosít, ami lehetővé teszi az új Spring-projektek gyors indítását. Az "Opinionated Defaults Configuration" megközelítést követve elkerüli a sok sablonkód és konfiguráció használatát, így javítva a fejlesztési és tesztelési folyamatot.

#### IOC container (dependency injection)

Az IoC (Inversion of Control) container a Spring egyik alapvető funkciója, amely egyszerű módot biztosít a Java objektumok konfigurálására és kezelésére. Ez felelős a Java objektum életciklusának kezeléséért.

Az IoC a dependency injection vagy dependency lookup mintákat használja az objektum referenciájának futásidőben történő biztosításához. A Spring által biztosított, IoC-hez használható csomagok: org.springframework.beans és org.springframework.context.

Az alkalmazásban a Spring bean-ek a Vaadin által biztosított *@SpringComponent* annotáció használatával kerültek jelölésre (szemben az általánosan használt *@Component* annotációval), mivel a Vaadin a HTML komponensek jelölésére már használja a *@Component* elnevezést. A bean-ek injektálása konstruktor alapú injektálással történik.

#### AOP (Aspect Oriented Programming)

Az alkalmazás a Spring AOP funkciói közül többet is igénybe vesz, de ezek jellemzően egyéb funkciókhoz kapcsolódnak (pl. adatbázis kezelés, autentikáció). Ezeket leszámítva kiemelhető a logolás.

#### Webszerver

A microservice architektúra erős támogatása miatt a Spring Boot alkalmazások egy beágyazott web szervert tartalmaznak (alapértelmezetten Tomcat), amit ez az alkalmazás is használ.

#### Adatbázis

Az alkalmazás használja a Spring Data projektet, részletesebben az link fejezetben kerül kifejtésre.

#### Spring Security

Később megírni

## Build automatizálás

Az alkalmazás build automatizáláshoz a Maven-t használja. A Maven egy széles körben elterjedt nyílt forráskódú eszköz, ami segíti a projektek menedzselését és megértését azáltal, hogy egy sztenderdizált build rendszert biztosít. Lehetővé teszi, hogy a fejlesztők könnyedén kezeljék a függőségeket, build folyamatokat és dokumentációt.

Mind a Vaadin és Spring Boot rendelkezik egy projekt gyors indítását segítő bootstrap weboldalakkal (<https://start.spring.io/> és <https://start.vaadin.com/app>). A projektet leíró pom.xml fájl az ezen oldalak által generált fájlok egyedi kombinációja.

A build folyamatról bővebben a link fejezetben esik szó.

## Függőségek

Ebben a fejezetben az alkalmazás által használt főbb függőségek kerülnek bemutatásra.

Ahogy az korábban bemutatásra került, az alkalmazás egy Spring Boot projekt, így a szülő pom fájl a *spring-boot-starter-parent*, ami természetesen sok egyéb függőséget is tartalmaz. Ez a dokumentáció ezeket nem mutatja be.

|  |  |  |
| --- | --- | --- |
|  | Függőség | Leírás |
| com.vaadin | vaadin |  |
| com.vaadin | vaadin-spring-boot-starter | A Vaadin Spring Boot-tal történő futtatásához szükséges modulok és konfigurációk. |
| com.vaadin | vaadin-testbench |  |
| com.h2database | h2 | Az alkalmazás által használt in-memory adatbázis használatához szükséges függőség. |
| com.github.javafaker | javafaker | Tesztadatok előállításhoz használt függőség. |
| org.projectlombok | lombok | A Project Lombok egy Java könyvtár, ami segít csökkenteni a Java projektekben használt boilerplate kód mennyiségét annotációk és automatikusan generált kód használatával. |
| org.springframework.boot | spring-boot-starter-data-jpa | A Sping JPA használatát segítő al-projektje |
| org.springframework.boot | spring-boot-starter-validation | Bean szintű validációk futtatásához szükséges függőség. Ez a funkció az űrlapok kitöltésekor nagy mértékben igénybe van véve. |
| org.springframework.boot | spring-boot-devtools | A fejlesztést segítő különböző eszközök, illetve konfigurációk. Pontosabban fogalmazva nem az alkalmazás függősége, mivel éles környezetben nem kerül alkalmazásra (optinal flag). |
| org.springframework.boot | spring-boot-starter-test | Tesztek írásához szükséges könyvtárak és konfigurációk összessége. |
| org.junit.vintage | junit-vintage-engine | Bizonyos Vaadin tesztekhez |
| io.github.bonigarcia | webdrivermanager |  |

## Alkalmazás felépítése

Ebben a fejezetben az alkalmazás főbb osztályai kerülnek bemutatásra.

### Képernyők, komponensek

Ahogy korábban kifejtésre került egy Vaadin alkalmazásban egy-egy képernyőt egy view osztály reprezentál. A képernyők a *pillercs.app.vaadin.views package*-ben találhatóak. Az egyes képernyők ezen belül is külön package-ekben találhatóak, a képernyők mellett a szorosan kapcsolódó, máshol nem használt komponensek mindig egy components nevű package-ben kerültek elhelyezésre.

Az alkalmazás jelenleg összes képernyője alap layout-ként a MainLayout osztályt használja.

Az űrlapok esetében egyedi esetektől eltekintve a validáció az entitások mezőin elhelyezett megszorító annotációk (*javax.validation.constraints*) segítségével történik.

Az összetettebb view-k UML ábrája és leírása a következő fejezetekben található.

#### HomeView

Text

Description automatically generated

Az alkalmazás jelenlegi kezdőoldala, ahol a személyi kölcsön igénylésekkel kapcsolatos funkciók elérhetőek. A képernyő két blokkból áll, az elsőben új igénylés indítható, a második pedig a folytatás opció esetén kerül megjelenítésre. Ez utóbbi a SelectApplicationGrid egyedi komponenst jeleníti meg, aminek feladata az igénylések lekérdezése az adatbázisból, egy grid-ben való megjelenítése és a lapozhatóság megvalósítása.

#### SelectClientView

Table

Description automatically generated with medium confidence

A képernyőn lehetőség van keresni a már korábban rögzített ügyfelek közül vagy jelezni, ha új ügyfél rögzítése szükséges. A keresést az egyedi SingleSelectClientGrid komponens valósítja meg, az adatbázisban történő keresés pedig a Spring Data QueryByExample interfésze által biztosított metódust használja.

#### NewClientView

A picture containing text

Description automatically generated

A képernyőn új ügyfél rögzítésére van lehetőség. Ezt egy egyedi, a Form Layout-ot kiterjesztő komponens a NewClientForm biztosítja. A NewClientForm komponens feladata annak ellenőrzése is, hogy a rögzíteni kívánt ügyfél létezik-e már az adatbázisban.

#### ApplicationBasicView

Text

Description automatically generated with medium confidence

A képernyőn az igénylés paramétereit lehet megadni egy-egy egyedi FormattedIntegerLayout komponensen keresztül. A Vaadin számokat elfogadó input mezője nem teszi lehetővé a megjelenés formázást, ezért ez a szöveges input mező kiterjesztésével került megvalósításra, kiegészítve az érték növelését/csökkentését lehetővé tevő gombokkal.

#### IncomeView

Text

Description automatically generated

A munkáltatók és jövedelmek rögzítését lehetővé tevő képernyő. Ezen említett tevékenységekhez egy-egy layout kapcsolódik, ami tartalmaz egy űrlapot új elem rögzítéséhez, valamint egy grid-et a már rögzített elemek megjelenítéséhez.

A munkáltatók esetében a form feladata ellenőrizni, hogy egy munkáltató egy igényléshez csak egyszer kerülhet mentésre. Lehetőség van rögzített munkáltató törlésére is, de csak abban az esetben, ha az még nem kapcsolódik egy jövedelemhez sem.

#### UnderwritingResultsView

Text

Description automatically generated

A bírálati eredmények megjelenítése ezen a képernyőn történik. A view felhasználja a bírálat eredményét, valamint a bírálati elemek részleteit is, amik a Vaadin Accordion komponens segítségével kerülnek megjelenítésre.

A képernyő adatokkal történő feltöltése routing-ot követően a processResults metódus meghívásával történik.

#### OffersView

Text

Description automatically generated

Az OffersView képernyőn kerülnek megjelenítésre az ügyfélnek kalkulált ajánlatok. A view ajánlatonként egy-egy OfferDiv komponenst hoz létre, valamint CSS attribútumok használatával kezeli a kiválasztás logikáját.

### Egyedi stílus

Az alkalmazás megjelenése CSS fájlokkal kerül módosításra. A Vaadin alapértelmezett stílusának elnevezése Lumo, de lehetőséget ad egyedi stílus csomagok készítésére is, amik ilyenkor a Lumo kiterjesztéseként értelmezhetőek.

Az alkalmazás egyedi stílusának kulcsszava „bankhojd”, ami a main metódust tartalmazó Application osztályon elhelyezett annotációval kerül jelölésre:



A CSS fájlok a frontend/themes/bankhojd könyvtárban kerültek elhelyezésre.

A Vaadin által biztosított komponensek bizonyos elemei a komponensek shadow DOM-jában helyezkednek el. Attól függően, hogy az ilyen „belső” elemeket akarjuk-e formázni két módja van a CSS fájlok importálásának.

Ha a Vaadin komponensek shadow DOM-jában lévő elemekeit szeretnék módosítani, akkor CSS szabályokat a már említett könyvtáron belül a components nevű mappában kell elhelyeznünk és a CSS fájl elnevezése meg kell, hogy egyezzen a komponens HTML megnevezésével, pl. vaadin-button.css. Mivel a shadow DOM-ban lévő elemekre hivatkozunk ezért általában szükség van a *:host* vagy *:host-context* selectorok alkalmazására.

Egyébként pedig a Vaadin automatikusan az említett könyvtárban található style.css fájlt olvassa fel. Ebben az alkalmazás view neveiről elnevezett CSS fájlok kerültek importálásra.

Megemlítendő még, hogy a Vaadin alapértelmezett stílusa több CSS változót is definiál, ezek felülírásával egyszerű módon érhető el a teljes alkalmazásra vonatkozó változtatás. A style.css fájlban ilyen módosított változók is találhatóak.

### Workflow modul

Table

Description automatically generated with medium confidence

Annak érdekében, hogy egy igénylés folytatható legyen szükséges volt az igénylést egy állapottal ellátni (ApplicationState enum, az application reláció state nevű attribútuma).

Az állapot léptetéséhez, valamint az adott állapotnak megfelelő címre történő navigálásához egy WorkflowService nevű szerviz osztály került létrehozásra.

Az osztály tartalmaz egy WorkflowStep nevű privát osztályt, melynek egy-egy példánya egy állapotot reprezentál.

A picture containing table

Description automatically generated

Egy WorkflowStep példány tartalmazza az állapotra való hivatkozást (ApplicationState), az állapotnak megfelelő view-ra történő navigálás módját (toCurrentStep), és a következő státuszra léptetést logikáját (toNextState). Az állapotok a szerzív osztály konstruktorában kerülnek regisztrálása.

Egy igénylés aktuális állapotának megfelelő képernyőre való navigálása a WorkflowService currentStep(), míg a következő állapotra való léptetés a nextStep() metódusának meghívásával történik.

### Bírálati modul

Text

Description automatically generated

Bírálat futtatása a bírálatot végző szerviz *runUnderwriting()* metódosunának meghívásával történik. A metóduson lévő *@Async* annotáció miatt az külön szálon kerül futtatásra.

Első lépésben összeállításra kerül a bírálati folyamat, ami több bírálati lépcsőből (UnderwritingStage) állhat. Egy lépcső több bírálati elemből is állhat, amik párhuzamosan kerülnek kiértékelésre. A lépcsők kiértékelése viszont egymást követően történik, így egy adott elemek építhetnek egy korábbi elem eredményére.

A bírálati elemek az UnderwritingStep absztrakt osztály kiterjesztései.

## Üzemeltetés

### Fordítás

Az alkalmazás fordításához az alábbi programok, csomagok telepítése szükséges:

* az alkalmazás egy Maven projekt ezért a fordításhoz a Maven telepítése szükséges: <https://maven.apache.org/download.cgi>
* Az alkalmazás a Java 17-es verziójával készült, emiatt JDK 17 telepítése szükséges: <https://www.oracle.com/java/technologies/downloads/>

Az alkalmazás éles környezetre történő fordításához a következő Maven parancsot kell használni: mvn clean package -Pproduction

A parancs futtatását követően a target könyvtárban létrejön a futtatható jar file.

### Futtatáshoz szükséges eszközök

Az alkalmazás Java 17-es verziót használ, annak működéséhez Java 17 futtására alkalmas környezet szükséges.

## Tesztelés

# Összefoglalás

A szakdolgozatom keretében egy webes alkalmazást készítettem. Az alkalmazás lehetővé teszi egy személyi kölcsön hiteligénylés rögzítését egy bank fiókhálózatban dolgozó alkalmazottai számára.

Az alkalmazás készítése során nagy mértékben támaszkodtam a webfejlesztés témakörében tartott kurzusok során szerzett ismereteimre. Szintén nagy mértékben szükségem volt az általános programozás elméleti, illetve Java nyelven oktatott kurzusok során tanultakra. Harmadrészt kiemelendő a Szoftvertechnológia kurzus, ahol a modern alkalmazásfejlesztés eszközeivel (VCS rendszerek, tervminták, build rendszerek) ismerkedtem meg.

Ugyanakkor az alkalmazás elkészítéséhez szükségem volt új ismeretekre is, főképp az alkalmazás által használt keretrendszerek tekintetében.

Az alkalmazás a **Vaadin** keretrendszert használja. Habár a Vaadin általában nem kerül fel a front-end keretrendszerek használati vagy népszerűségi toplistájára, több olyan tulajdonsággal is rendelkezik, ami alkalmassá teszi egy banki alkalmazás elkészítéséhez. Az alkalmazás oldalai nem HTML sablonokból vagy kódrészeletekből épülnek fel, hanem a HTML elemeket reprezentáló osztályokból. Az alkalmazás a szerveren fut, ami lehetőséget ad bármilyen JVM által biztosított szolgáltatás elérésére, így növelve annak biztonságát. A szakdolgozat elkészítéséhez meg kellett ismerkednem a Vaadin-nal.

Szintén elmondható, hogy az alkalmazás a **Spring Boot** keretrendszert használja, ami bár népszerűségben már felveszi a versenyt más back-end keretrendszerekkel, de a tanulmányaim során nem találkoztam vele. Ugyan előzetesen már rendelkeztem ismeretekkel, de az alkalmazás elkészítéséhez jobban el kellett merülnöm a Spring keretrendszer használatában.

A munka megkezdése előtt nagy mértékben motivált az a kérdés, hogy vajon az elvárásaimnak megfelelnek-e majd ezek az eszközök. Banki környezetben javasolt-e ezek használatával alkalmazást építeni. Az alkalmazás elkészültét követően a válasz erre egyértelműen igen. Természetesen szükséges energiát belefektetni a keretrendszer megértésébe, de úgy gondolom, hogy az alapvető ismeretek megszerzését követően a Vaadin alkalmas mind egyszerűbb, mind akár bonyolultabb alkalmazások készítésére.

alkalmazás rövid jellemzése

mit kellett megtanulnom

hogy használtam a tanultakat, kurzusok

tanulságok

nehézségek

értékelése a munkának

# További fejlesztési lehetőségek

Az alkalmazás készítése során igyekeztem minél több oldalról megismerni a keretrendszerek nyújtotta lehetőségeket, de természetesen így sem jutott idő minden funkció lefejlesztésére.

Az alkalmazás bankfióki használatra, asztali gépen történő megtekintésre készült. Habár természetesen döntő többségében reszponzív elemekből épül fel, de jelenleg a megjelenítés nem minden esetben mobil barát. Figyelembe véve, hogy az igénylési folyamat kis módosítással akár egy internetbanki csatornán is elképzelhető lenne javasolt a képernyők kisebb képernyőre történő optimalizálása is.

Az alkalmazás jelenleg csak angol nyelven elérhető, a következő lépések egyike kell, hogy legyen az i18n bevezetése.

A megvalósított hiteligénylési folyamat nem tette szükségessé külső workflow motor használatát. A jövőben azonban, kiváltképp további termékek esetén, érdemes lenne a projektbe ennek külső szervizként történő bevonása. Ilyen alkalmazás lehet például a Camunda[[4]](#footnote-4).

# Irodalomjegyzék

# Mellékletek

1. https://vaadin.com/faq [↑](#footnote-ref-1)
2. https://vaadin.com/faq [↑](#footnote-ref-2)
3. https://vaadin.com/docs/latest/components [↑](#footnote-ref-3)
4. <https://camunda.com/> [↑](#footnote-ref-4)