EÖTVÖS LORÁND TUDOMÁNYEGYETEM



Informatikai Kar

Média- és Oktatásinformatikai Tanszék

**Hiteligénylési folyamatot megvalósító webes alkalmazás**

|  |  |
| --- | --- |
| *Témavezető:*  Dr. Menyhárt László Gábor | Piller Csaba  Programtervező informatikus Bsc |
| adjunktus, PhD |

Budapest, 2022

Tartalom

[2 Bevezetés 3](#_Toc123245226)

[2.1 Az alkalmazás témája 3](#_Toc123245227)

[2.2 Az alkalmazás mögötti technológia 4](#_Toc123245228)

[3 Felhasználói dokumentáció 6](#_Toc123245229)

[3.1 Bejelentkezés 6](#_Toc123245230)

[3.2 Menü 7](#_Toc123245231)

[3.3 Kezdőképernyő 8](#_Toc123245232)

[3.4 Új hiteligénylés 8](#_Toc123245233)

[3.4.1 Ügyfél kiválasztása 8](#_Toc123245234)

[3.4.2 Az igénylés alapadatainak a megadása 11](#_Toc123245235)

[3.4.3 Jövedelmek és munkáltatók rögzítése 12](#_Toc123245236)

[3.4.4 Igénylő további adatai 13](#_Toc123245237)

[3.4.5 Bírálat és a bírálat eredménye 14](#_Toc123245238)

[3.4.6 Ajánlatok kiválasztása 14](#_Toc123245239)

[3.4.7 Szerződéskötés 15](#_Toc123245240)

[3.5 Hiteligénylés folytatása 15](#_Toc123245241)

[4 Fejlesztői dokumentáció 17](#_Toc123245242)

[4.1 Use case-ek és folyamatábrák 17](#_Toc123245243)

[4.1.1 Use case-ek 17](#_Toc123245244)

[4.1.2 Folyamatábrák 18](#_Toc123245245)

[4.2 Az Alkalmazásban használt keretrendszerek 22](#_Toc123245246)

[4.2.1 Vaadin 22](#_Toc123245247)

[4.2.2 Spring 24](#_Toc123245248)

[4.3 Alkalmazás felépítése 26](#_Toc123245249)

[4.3.1 Képernyők, komponensek 26](#_Toc123245250)

[4.3.2 Egyedi stílus 31](#_Toc123245251)

[4.3.3 Workflow modul 32](#_Toc123245252)

[4.3.4 Bírálati modul 33](#_Toc123245253)

[4.4 Függőségek 36](#_Toc123245254)

[4.5 Adatbázis szerkezet 39](#_Toc123245255)

[4.5.1 Relációk rövid ismertetése 40](#_Toc123245256)

[4.5.2 Az adatbázis használata 41](#_Toc123245257)

[4.5.3 Adatbázis rendszer 42](#_Toc123245258)

[4.6 Build automatizálás 42](#_Toc123245259)

[4.7 Üzemeltetés 43](#_Toc123245260)

[4.7.1 Fordítás 43](#_Toc123245261)

[4.7.2 Futtatáshoz szükséges eszközök 43](#_Toc123245262)

[4.8 Tesztelés 44](#_Toc123245263)

[4.8.1 Programozott tesztek 44](#_Toc123245264)

[4.8.2 Manuális tesztek 45](#_Toc123245265)

[5 Összefoglalás 46](#_Toc123245266)

[6 További fejlesztési lehetőségek 48](#_Toc123245267)

[7 Irodalomjegyzék 49](#_Toc123245268)

[7.1 Egyéb források 49](#_Toc123245269)

[8 Ábrajegyzék 50](#_Toc123245270)

[9 Mellékletek 52](#_Toc123245271)

[9.1 Képernyő tervek (wireframes) 52](#_Toc123245272)

[9.2 Manuális tesztesetek 57](#_Toc123245273)

# 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 folyamataival, akár meglévő rendszerek üzemeltetéséről, továbbfejlesztéséről, 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 és 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 felhasználói 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 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 folyamatok is hasonló komplexitásúak. Az előzőleg tárgyalt okok miatt az általam készített alkalmazás is ezt az irányt képviseli.

## Az alkalmazás mögötti technológia

Az egyetemi képzés során 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-endet is magába foglaló Vaadin-t választottam.

A Vaadin egy nyílt forráskódú platform Java back-enddel rendelkező modern webes alkalmazások készítésére. Egyetlen „opinionated” webfejlesztési stack-be integrálva tartalmazza a UI komponenseket, keretrendszereket és eszközöket. Szemben például az Angularral vagy Reacttel, a Vaadin bizonyos tekintetben a back-endet 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.

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

# Felhasználói dokumentáció

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

Az alkalmazás célja, hogy a személyi kölcsön hiteligénylési folyamatát a fiókhálózatban szabályozott keretek között támogassa. Az alkalmazás használata során a felhasználók két főbb tevékenységet végezhetnek el:

* új hiteligénylés indítása: lehetőség van egy új hiteligénylés indítására, melynek során a később kifejtett lépéseken vezet keresztül az alkalmazás
* hiteligénylés folytatása: amennyiben egy korábban megkezdett hiteligénylés nem került befejezésre, lehetőség van annak folytatására

Az alkalmazás felhasználói a bankfiókban dolgozó, fogyasztásihitel-értékesítéssel foglalkozó ügyintézők.

Az alkalmazás böngészőn keresztül érhető el.

|  |  |
| --- | --- |
| Megjegyzés | Az alkalmazás konkrét címe függ az élesítést végző szervezet domain címétől és annak üzemeltetési gyakorlatának függvényében alakul ki.  Az alkalmazást elindítva az jelenleg a localhost:8080 címen érhető el. |

## Bejelentkezés

Az alkalmazás használatához felhasználói jogosultság szükséges, ami központilag kerül beállításra. A felhasználó a beállított azonosítót és jelszót email-ben kapja meg.

Timeline

Description automatically generated

1. ábra Bejelentkezés az alkalmazásba

|  |  |
| --- | --- |
| Megjegyzés | Éles banki környezetben az alkalmazás a bank által használt felhasználó- vagy hálózatkezelő rendszerhez lenne kötve és a felhasználók egy egységes felhasználónevet és jelszót használnának vagy nem is lenne szükség bejelentkezésre.  Jelen esetben az alkalmazás előre definiált felhasználói (felhasználó név/jelszó): user/pass és admin/pass. |

## Menü

Az alkalmazás jelenleg csak a személyi kölcsön hiteligénylési folyamatát támogatja, ennek megfelelően egy egyszerű menüvel rendelkezik, ami lehetővé teszi a nyitóképernyőre történő navigációt. Az oldalsó menü elrejthető.

Graphical user interface

Description automatically generated

2. ábra Az alkalmazás menüje

## Kezdőképernyő

Bejelentkezést követően a felhasználót egy nyitóképernyő fogadja, ahol két opció közül választhat:

* új hiteligénylési folyamat indítása és
* igénylés folytatása

Graphical user interface, text, application, chat or text message

Description automatically generated

3. ábra Kezdőképernyő

## 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.

Graphical user interface, text, application, chat or text message

Description automatically generated

4. ábra Ügyfél keresés

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

Graphical user interface, application

Description automatically generated

5. ábra Ügyfél találati lista

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

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

Graphical user interface, application, Word

Description automatically generated

6. ábra Új ügyfél rögzítése gomb

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

Graphical user interface, application

Description automatically generated

7. ábra Ügyfél személyes adatok

A piros csillag jelöli a kötelezően kitöltendő mezőket. Amennyiben a kitöltés nem megfelelő, akkor ezt a mező alatt egy piros hibaüzenet jelzi. Ez a működés a folyamat összes rögzítendő mezőjére igaz.



8. ábra Hibásan kitöltött mező

Ügyelni kell arra, hogy egy ügyfél csak egyszer rögzíthető. Amennyiben a rögzíteni kívánt ügyfél már létezik (név, születési dátum és anyja neve megegyezik) egy felugró ablak figyelmeztet.

Graphical user interface, text, application, chat or text message

Description automatically generated

9. ábra Létező ügyfél hibaüzenet

Ilyenkor két lehetőség közül választhatunk:

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

### Az igénylés alapadatainak 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.

Graphical user interface, text, application, chat or text message

Description automatically generated

10. ábra Hitelösszeg megadása

### 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.

Graphical user interface, text, application, chat or text message

Description automatically generated

11. ábra Munkáltató rögzítése

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

Olyan rögzített munkáltató, ami már egy jövedelemhez kapcsolódik, nem törölhető. Erre egy hibaüzenet figyelmeztet.

Graphical user interface, application

Description automatically generated

12. ábra Munkáltató törlése hibaüzenet

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

Graphical user interface, text, application, chat or text message

Description automatically generated

13. ábra Jövedelem rögzítése

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

Graphical user interface, text, application, chat or text message

Description automatically generated

14. ábra Igénylő további adatainak a rögzítése

Ahol szükséges, az alkalmazás súgót (tooltip-et) jelenít meg a mező fölé állva.

A picture containing application

Description automatically generated

15. ábra Súgó példa

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, így a felhasználónak nincs további 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 az igénylés mely része nem volt megfelelő.

Graphical user interface, text, application, email

Description automatically generated

16. ábra Bírálat részletes eredménye

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

### Ajánlatok kiválasztása

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

Rectangle

Description automatically generated with medium confidence

17. ábra Ajánlatok választása

Az ajánlat kiválasztását követően a szerződéskötés történik.

### Szerződéskötés

Az ajánlat elfogadását követően az ügyfélnek alá kell írnia a szerződést. Ehhez első lépésként szükséges a szerződést legenerálni. Generálást követően megjelenik a letöltés gomb. A szerződés az elfogadott ajánlat paramétereit tartalmazza (hitelösszeg, futamidő). Nyomtatás után a szerződés a Bank és az ügyfél részéről is aláírható.

Graphical user interface, text, application

Description automatically generated

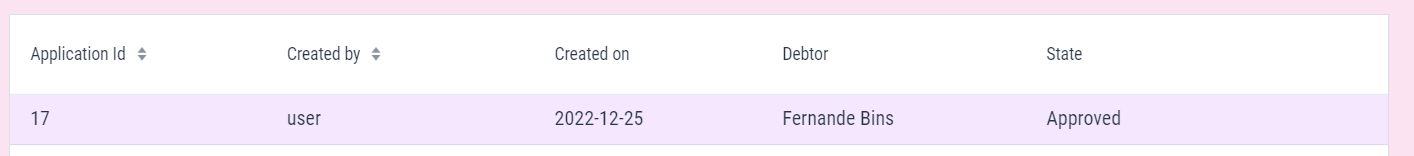
18. ábra Szerződés nyomtatása

A szerződés aláírását követően a folyamat véget ér.

|  |  |
| --- | --- |
| Megjegyzés | A PDF fájl jelenleg a futtatott jar állomány könyvtárába kerül mentésre. Éles banki környezetben a szerződés a dokumentumok archiválásáért felelős alkalmazásnak kerülne átadásra. |

## Hiteligénylés folytatása

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



19. ábra Folytatható igénylés kiválasztása

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

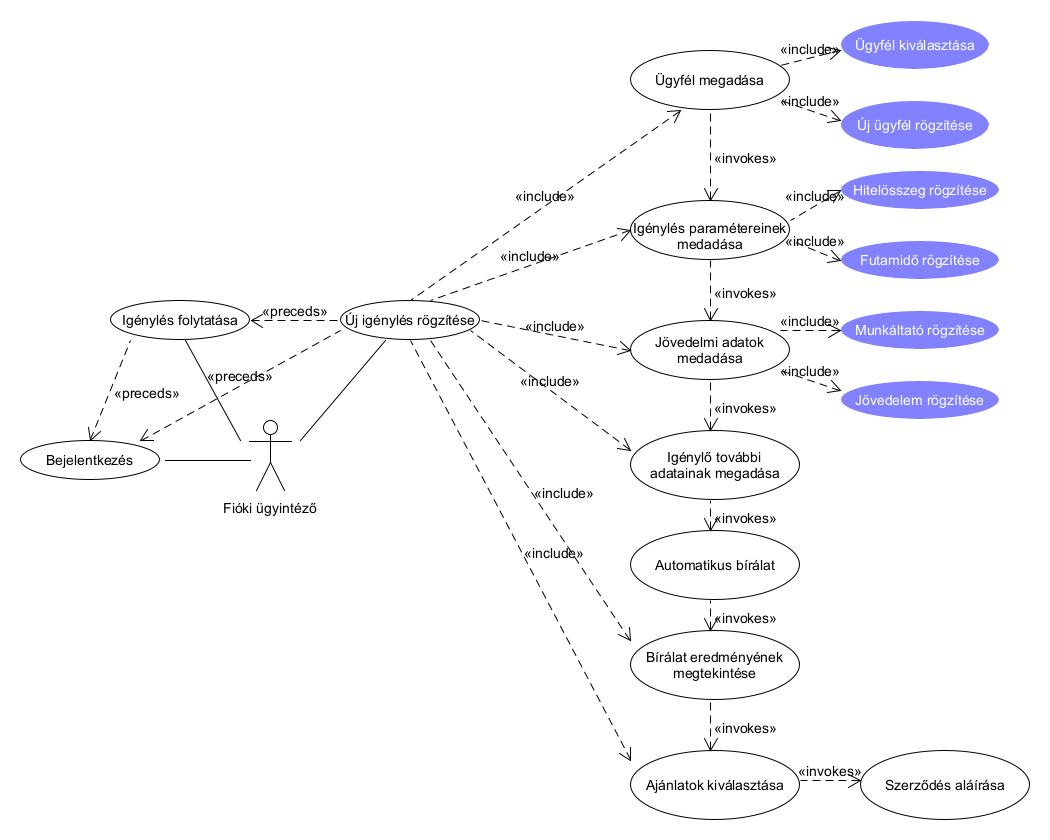
### Use case-ek

Az alkalmazás jelenleg két alapvető use case-szel rendelkezik:

* új igénylés rögzítése
  + a felhasználó először keres a rendszerben már rögzített ügyfelek között név alapján. Amennyiben az ügyfél megtalálható, kiválasztja, máskülönben új ügyfelet rögzít. Az alkalmazás ellenőrzi, hogy egy ügyfél csak egyszer kerüljön rögzítésre.
  + 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.
  + a felhasználó generálja, majd nyomtatja a szerződést
* 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 üzemeltető, mint aktor nem került feltüntetésre.

Mind a két use case-t megelőzi az alkalmazásba történő bejelentkezés.



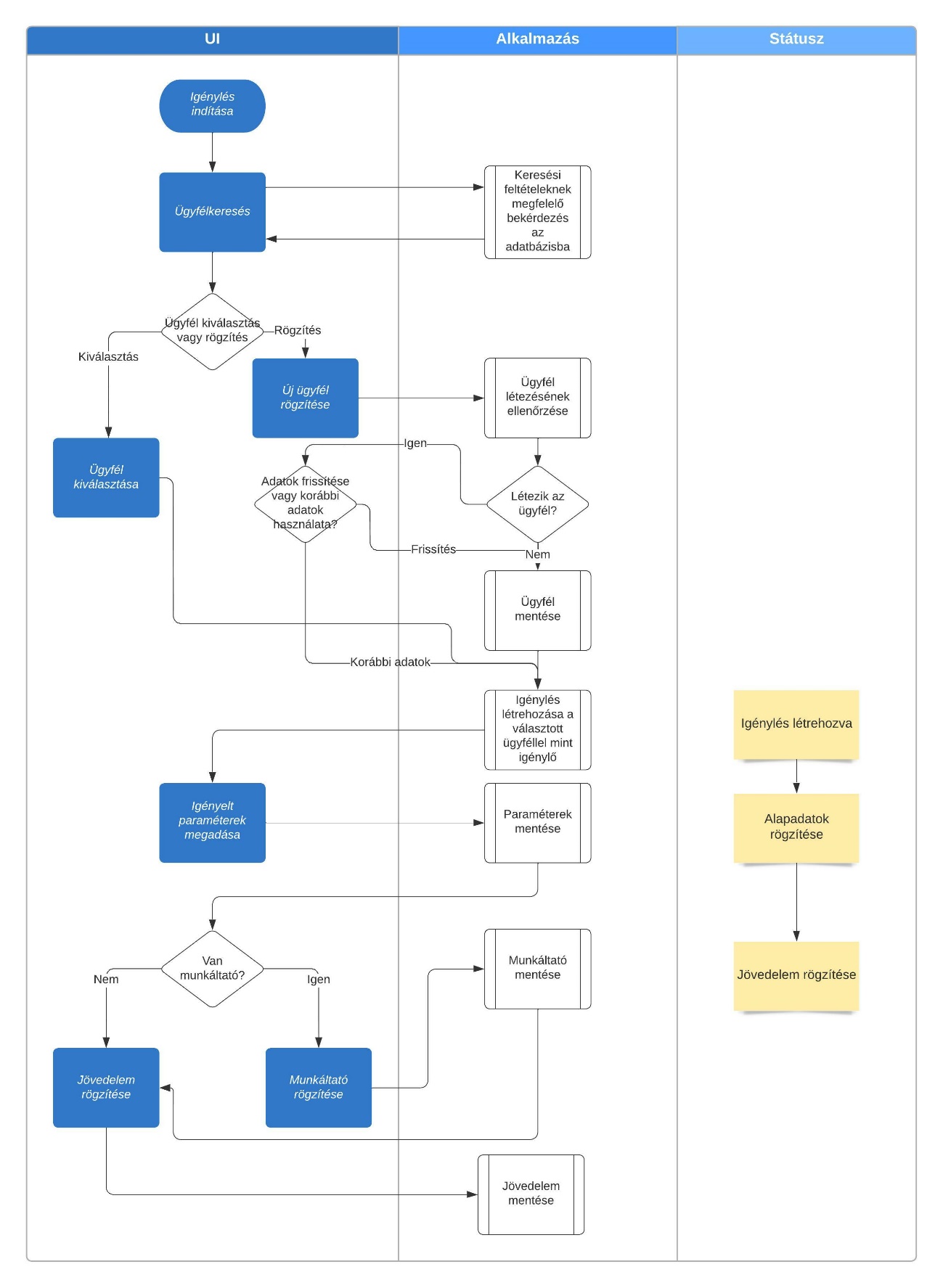
20. ábra Use case diagram

### Folyamatábrák

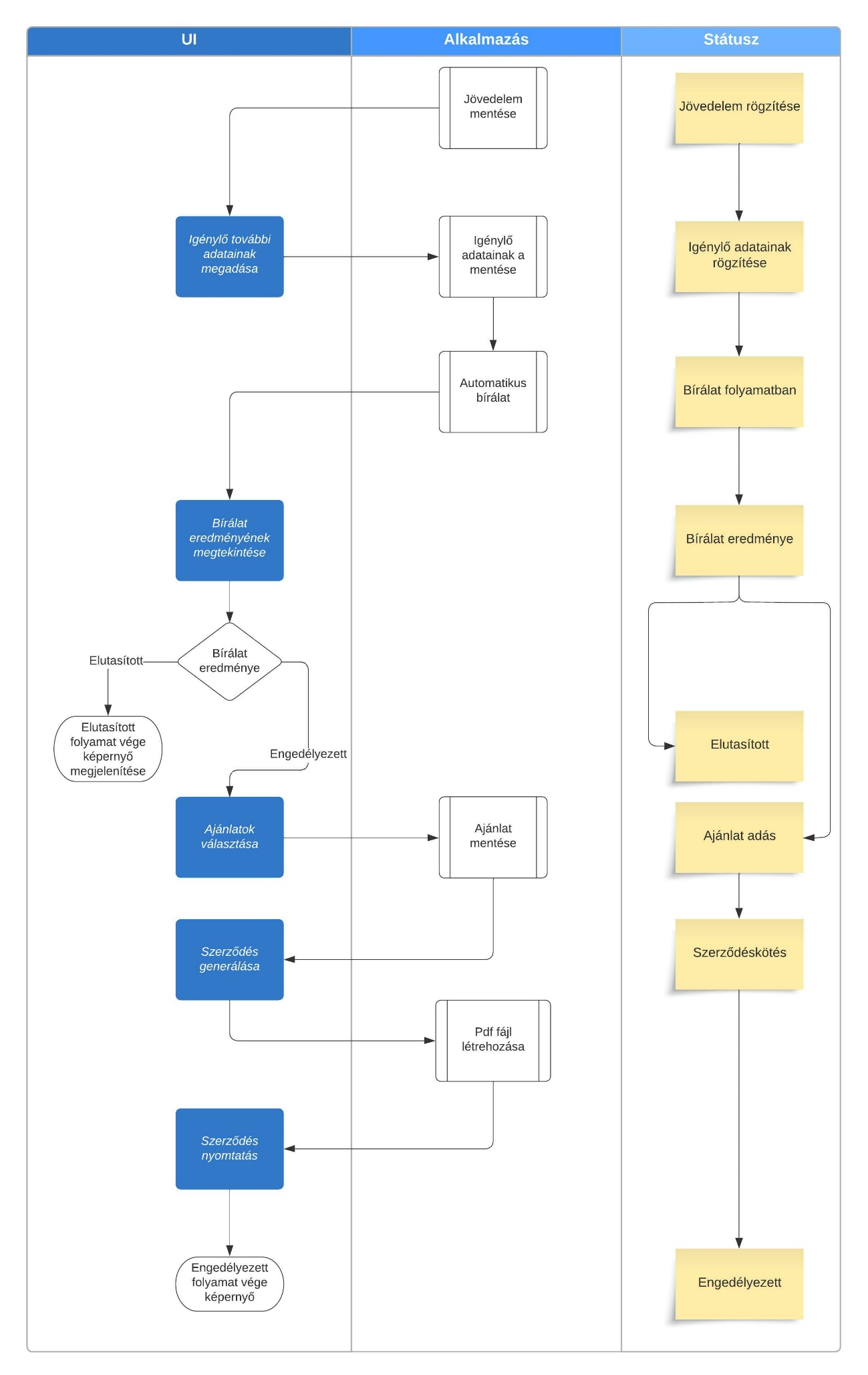
A két use case közül az új igénylés rögzítése több lépésből is áll. Ezek leegyszerűsítve szekvenciális lépések, azonban bizonyos pontokon az igénylés sajátosságaitól függően döntenie kell a felhasználónak.

A folyamatot az alábbi uszodadiagram szemlélteti. 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 tükrözése, így technikai értelemben nem feltétlenül helytálló a megkülönböztetés.

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



21. ábra Új igénylés folyamatábrája 1.



22. ábra Új igénylés folyamatábrája 2.

## Az Alkalmazásban használt keretrendszerek

### Vaadin

Az alkalmazást leginkább meghatározó keretrendszer a Vaadin. A Vaadin egy nyílt forráskódú platform, Java back-enddel 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 a back-enden tartja nyilván, a böngészőben megjelenített weboldal ennek a leképezése. Az alkalmazás view-kból épül fel. Ahhoz, hogy egy view elérhető legyen, szükséges egy elérési útvonallal rendelkeznie, ami a *@Route* annotációval érhető el.



23. 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 drawerrel rendelkezik. |
| Vertical-/Horizontal Layout | A Vaadin két alapelrendezést 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 az adatokat. Szövegen túl egyéb elemek és komponensek (pl. gomb) is megjeleníthetők. |
| Form Layout | A Form Layout segítségével több oszlopos, címkézett, reszponzív űrlap készíthető. |
| 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. |

24. ábra Főbb komponensek

#### Események

A Vaadin Flow esemény vezérelt programozást használ a felhasználói interakciók kezelésére. A felhasználó 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[[4]](#footnote-4).

#### 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

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.

A Spring Boot a Spring kiterjesztéseként is felfogható, tehát úgy is fogalmazhatunk, hogy az alkalmazás a Spring Boot keretrendszert használja.

#### 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 a következők:

* org.springframework.beans
* 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. adatbáziskezelés, autentikáció). Ezeken felül kiemelhető még 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, ez részletesebben az

*Adatbázis* szerkezetfejezetben került kifejtésre.

#### Spring Security

A felhasználók és jogosultságok kezelése a Spring Security keretrendszer segítségével történik, alkalmazva a Vaadin keretrendszer miatt szükséges egyedi beállításokat.

Ennek megfelelően a konfigurációs osztály a VaadinWebSecurity osztály kiterjesztése. A felhasználók és jelszavak (titkosított formában történő) tárolása adatbázisban történik, emiatt a keretrendszer JdbcUserDetailsManager osztálya került használatra.

Fontos különbség az általános végpont alapú jogosultságkezeléshez képest, hogy az egyes view-k elérhetősége csak annotációk elhelyezésével kerül korlátozásra. Erre egy példa alább látható, ahol a *@PermitAll* annotáció lehetővé teszi, hogy minden bejelentkezett felhasználó elérje az oldalt.



25. ábra Jogosultság kezelés annotációkkal

## 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 találhatók. Az egyes képernyők ezen belül is külön package-ekben találhatók, a képernyők mellett a szorosan kapcsolódó, máshol nem használt komponensek mindig egy components nevű package-ben kerültek elhelyezésre.

Jelenleg az alkalmazás összes képernyője 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

26. ábra HomeView UML

A HomeView az alkalmazás jelenlegi kezdőoldala, ahol a személyikölcsön-igénylésekkel kapcsolatos funkciók elérhetők. 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

27. ábra SelectClientView UML

A képernyőn lehetőség van keresni a már korábban rögzített ügyfelek között 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

28. ábra NewClientView UML

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. Amennyiben igen, akkor egy dialógus ablak jelenik meg, majd a választól függően frissítésre került az ügyfél rekord vagy elvetésre kerülnek az új adatok.

#### ApplicationBasicView

Text

Description automatically generated with medium confidence

29. ábra ApplicationBasicView UML

A képernyőn az igénylés paramétereit lehet megadni egy-egy egyedi FormattedIntegerLayout komponensen keresztül. A Vaadin számokat elfogadó input mezője nem teszi lehetővé a megjelenés formázását, ezért ez a szöveges input mező kiterjesztésével került megvalósításra, kiegészítve az érték növelését/csökkentését lehetővé tevő gombokkal. A mezők módosulásához kapcsolódó esemény hatására (helyes érték esetén) automatikusan újraszámításra kerül a törlesztőrészlet.

#### IncomeView

Text

Description automatically generated

30. ábra IncomeView UML

Az IncomeView 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

31. ábra UnderwritingResultsView UML

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 jelennek meg.

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

#### OffersView

Text

Description automatically generated

32. ábra OffersView UML

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.

#### ContractView

Table

Description automatically generated with medium confidence

33. ábra ContractView UML

A képernyőn lehetőség van egy szerződés generálására, valamint annak letöltésére. A PDF dokumentum előállítása az iText pdfHTML könyvtár segítségével történik, ami képes egy HTML sablonból PDF fájlt generálni. A HTML sablon a resources-on belül a contract könyvtárban található. Tekintve, hogy jelen esetben csak egy darab egyoldalas dokumentum kerül generálásra, a változók (pl. dátum, elfogadott hitelösszeg) behelyettesítése memóriában, a sablon beolvasása közben történik.

A PDF fájl generálását a PdfService szerviz osztály támogatja.

### Egyedi stílus

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

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



34. ábra Theme annotáció használata

A CSS fájlok a Vaadin specifikációja szerint 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 elemeit 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 találhatók.

### Workflow modul

Table

Description automatically generated with medium confidence

35. ábra WorkflowService UML

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

36. ábra WorkflowStep UML

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 szerviz osztály konstruktorában kerülnek regisztrálása. Az osztály főbb metódusai:

|  |  |
| --- | --- |
| *WorkflowService.currentStep()* | |
| bemeneti adatok | * source: Component – a navigációs kérés forrása, általában az a layout, ahol a felhasználó éppen tartózkodik * applicationId: Long – az igénylés azonosítója |
| kimeneti adat | a metódus visszatérési értéke void |
| tevékenység | A metódus feladata az igénylés állapotától függő képernyőre történő navigálás, ami lényegében az állapothoz kapcsolódó toCurrentStep BiConsumer meghívását jelenti. |

|  |  |
| --- | --- |
| *WorkflowService.nextStep()* | |
| bemeneti adatok | * source: Component – a navigációs kérés forrása, általában az a layout, ahol a felhasználó éppen tartózkodik * applicationId: Long – az igénylés azonosítója |
| kimeneti adat | a metódus visszatérési értéke void |
| tevékenység | A metódus lépteti az igénylés állapotát az aktuális állapothoz rendelt logika alapján (toNextState Consumer meghívása). Ezt követően az aktuális képernyőre navigálás a currentStep() metódus meghívásával történik, ami már az új állapot alapján dolgozik. |

### Bírálati modul

Text

Description automatically generated

37. ábra UnderwritingService UML

Bírálat futtatása a bírálatot végző szerviz *runUnderwriting()* metódusának meghívásával történik. A metóduson lévő *@Async* annotáció miatt a végrehajtás külön szálon történik.

|  |  |
| --- | --- |
| *UnderwritingService.runUnderwriting()* | |
| bemeneti adatok | applicationId: Long – az igénylés azonosítója |
| kimeneti adat | ListenableFuture<Void>: a metódus visszatérési értéke egy visszatérés érték nélküli ListenableFuture. Tehát a bírálat futtatása aszinkron módon történik. |
| tevékenység | Első lépésben összeállításra kerül a bírálati folyamat, ami több bírálati lépcsőből (UnderwritingStage) állhat. Egy lépcső több bírálati elemből is állhat, melyek párhuzamosan kerülnek kiértékelésre.  Ezt követően történik meg a bírálati lépcsők kiértékelése a bírálati elemek run() metódusának aszinkron módon történő futtatásával (CompletableFuture.runAsync() metódus). A lépcsők kiértékelése egymást követően, nem aszinkron módon történik, így adott elemek építhetnek egy korábbi elem eredményére. |

A bírálati elemek az UnderwritingStep absztrakt osztály kiterjesztései, a bírálati logikát a run() absztrakt metódus implementációja, valamint az egyedi bírálti szabályok valósítják meg.

Text

Description automatically generated with low confidence

38. ábra UnderwritingStep UML

|  |  |
| --- | --- |
| *UnderwritingStep.run()* | |
| bemeneti adatok | * applicationId: long – az igénylés azonosítója * underwritingId: long – a bírálat azonosítója |
| kimeneti adat | a metódus visszatérési értéke void |
| tevékenység | Tekintve, hogy a run egy absztrakt metódus, így általánosítva mutatom be a tevékenységét. Első lépésben lekérdezésre, kiszámításra kerülnek azok az adatok, amik több bírálati szabályban is felhasználásra kerülnek. A bírálati szabályok bírálati elemenként egyedi privát metódusok.  A bírálati szabályok kezelése a handleResult() metódus segítségével történik, ami az eredménytől és szabály típusától függően létrehoz egy dto osztályt a használandó üzenet és eredmény tárolására.  Ezt követően ezen dto-k felhasználásával a saveResults() metódus:   * megállapítja a bírálati elem eredményét * létrehozza és elmenti a bírálati elem egy előfordulását * a bírálati elemhez kapcsoltan elmenti a bírálati szabályok üzeneteit |

#### Bírálati elemek és szabályok összefoglalása

Az alkalmazás jelenleg három bírálati elemet értékel ki.

Fizetőképesség (Budget):

* Minimum törlesztőrészlet: annak ellenőrzése, hogy figyelembe véve az igénylő összesített jövedelmét, fennálló tartozásainak összegét, az elméleti legkisebb törlesztőrészletet, valamint a jövedelem maximális terhelhetőségét (DTI – debt to income ratio, Magyarországon JTM a „32/2014. (IX. 10.) MNB rendelet a jövedelemarányos törlesztőrészlet és a hitelfedezeti arányok szabályozásáról” alapján) az ügyfél jogosult-e ajánlatra.
* Ajánlat kalkuláció: az előző pontban említett adatok alapján kerül ellenőrzésre a konkrét ajánlatok elérhetősége.

Termékfeltételek (Product rules):

* Munkaviszony kezdete: amennyiben kapcsolódik munkáltató valamelyik rögzített jövedelemhez, akkor a munkaviszonynak minden munkáltató esetében el kell érnie a minimum előírt időt (jelenleg ez az idő a bírálatot megelőző 3 hónap).
* Munkaviszony vége: amennyiben kapcsolódik munkáltató valamelyik rögzített jövedelemhez, akkor legalább egy esetében a munkaviszony határozatlan idejű vagy legalább még az előírt időpontig tart (jelenleg ez az időtartam a bírálat plusz 6 hónap).
* Minimum jövedelem: az igénylő összesített jövedelme el kell, hogy érjen egy minimum előírt összeget (jelenleg ez az összeg 100.000 Ft).

Visszaélés ellenőrzése (Fraud check):

* Korábban visszaélést elkövető ügyfél: amennyiben az ügyfél korábban bizonyítottan visszaélést követett el a bankkal szemben, akkor újabb hitel felvételére nem jogosult, az ilyen ügyfelek kiszűrésre kerülnek
* Igénylések száma: amennyiben az ügyfél rövid időn belül több igénylést is benyújt, akkor fennáll a veszélye, hogy próbálja kiismerni az igénylési folyamatot és nem valós adatokat ad meg az igénylés során. Ebben az esetben az igénylés elutasításra kerül.

## Függőségek

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

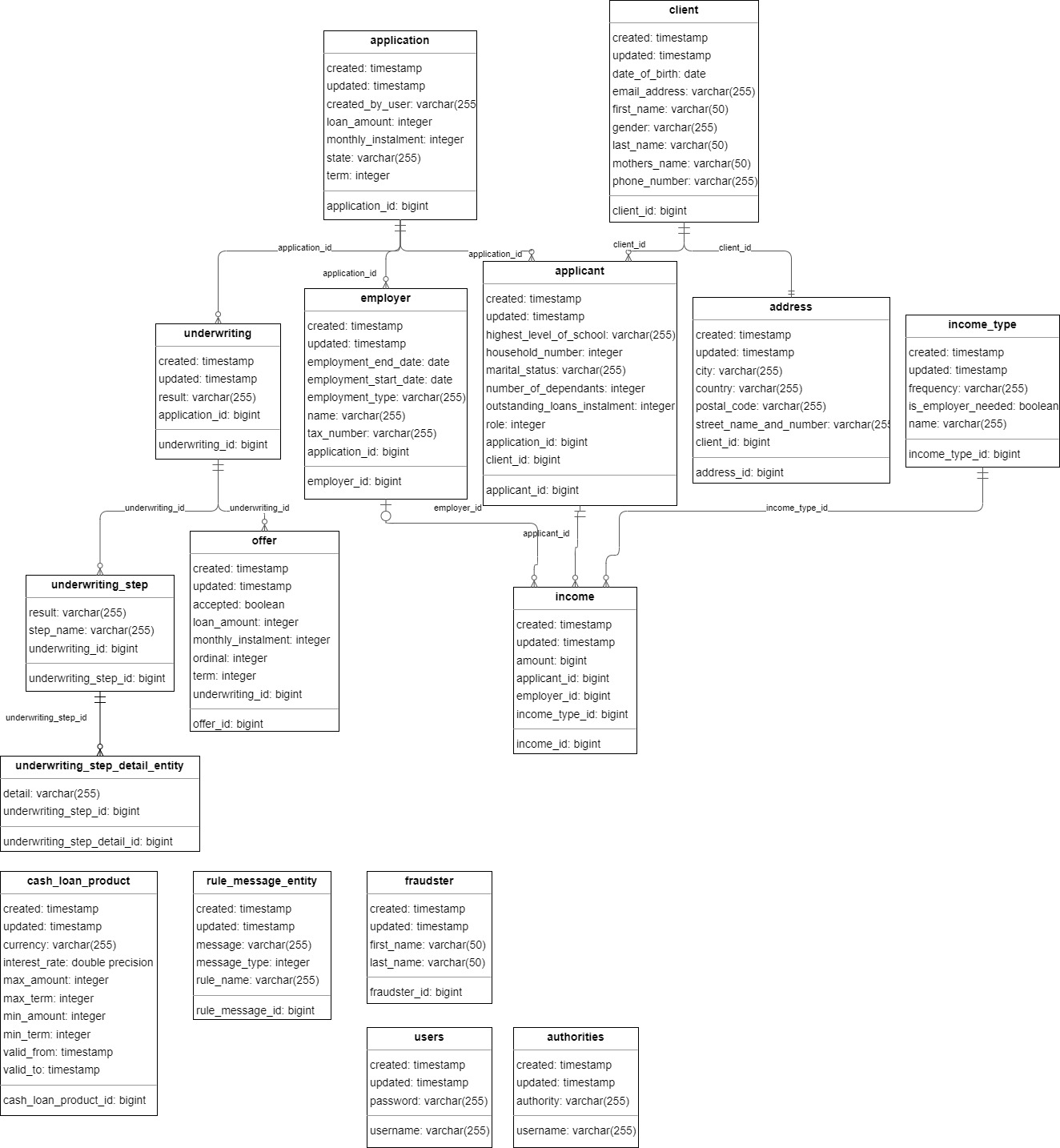
Ahogy az korábban bemutatásra került, az alkalmazás egy Spring Boot projekt, így a szülő pom fájl a *spring-boot-starter-parent*. Ebben a fejezetben csak a közvetlenül importált függőségek kerülnek felsorolásra.

|  |  |  |
| --- | --- | --- |
| Group ID | Artifact ID | Leírás |
| com.vaadin | vaadin | A Vaadin keretrendszer. |
| com.vaadin | vaadin-spring-boot-starter | A Vaadin Spring Boot-tal történő futtatásához szükséges modulok és konfigurációk. |
| 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 alprojektje |
| org.springframework.boot | spring-boot-starter-validation | Bean szintű validációk futtatásához szükséges függőség. Ez a funkció főképp az űrlapok kitöltésekor végzett validációk során kerül használatra. |
| 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.seleniumhq.selenium | selenium-java | Böngészőben végzett automatikus tesztek futtatását lehetővé tevő könyvtár |
| io.github.bonigarcia | webdrivermanager | A Selenium tesztek futtatásához szükséges böngésző driverek automatikus menedzselését segítő könyvtár |
| com.itextpdf | html2pdf | Ez a könyvtár a PDF formátumú szerződések előállítását támogatja. |
| org.vaadin.olli | file-download-wrapper | Vaadin add-on a fájlok böngészőből történő letöltéséhez |

39. ábra Függőségek

## Adatbázis szerkezet

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



40. ábra Relációk és kapcsolataik

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

### Relációk rövid ismertetése

|  |  |
| --- | --- |
| Reláció neve | Leírás |
| Application | A hiteligénylés, 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 jövedelemelőfordulás is kapcsolódhat. |
| Income type | A jövedelem típusát leíró reláció, ami meghatározza, hogy a jövedelemhez kapcsolódik-e munkáltató, illetve annak gyakoriságát is megadja. |
| Employer | Munkáltató, igénylés szinten egyedi, adatbázis szinten viszont 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. |

|  |  |
| --- | --- |
| Reláció neve | Leírás |
| Rule message | Az egyes szabályok engedélyezés vagy elutasítás esetén megjelenítendő üzenetei. |
| Offer | Egy bírálat során kalkulált ajánlatok. |
| Cash loan product | A személyi kölcsön termék paraméterei, konstrukció. Az alkalmazás jelenleg egy adott időpontban egy érvényes konstrukciót feltételez. |
| Fraudster | Korábban visszaélésben részt vett ügyfelek listája. |
| Users | Az alkalmazás által kezelt felhasználók. Jelen esetben csak előre definiált felhasználókat tartalmaz, tekintve, hogy éles környezetben a felhasználók nyilvántartása nem az alkalmazás feladata lenne. |
| Authorities | A felhasználókhoz kapcsolódó jogosultságok. |

41. ábra Relációk

### 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.



42. 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.

Az adatbázis sémája az Entity osztályok alapján a Hibernate által kerül létrehozásra.

### 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.

## 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, a build folyamatokat és a dokumentációt.

Mind a Vaadin és Spring Boot rendelkezik a 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 folyamat lépései az *Üzemeltetés* fejezetben kerültek kifejtésre.

## 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, ami a target könyvtárban állva a következő paranccsal indítható: java -jar bank-hojd-1.0-RELEASE.jar

### Futtatáshoz szükséges eszközök

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

A Vaadin által egy minta alkalmazáson végzett terheléses tesztek[[5]](#footnote-5) alapján az alábbi hardver követelmények fogalmazhatók meg:

* 1000 konkurens felhasználónkként célszerű legalább 3,7 GB memória lefoglalása.
* Durva becslésként 400 konkurens felhasználónkként javasolt legalább 1 logikai CPU megléte.

## Tesztelés

Az alkalmazás tesztelése jelenleg három különböző módon történik:

* programozott tesztek:
  + unit tesztek
  + automatizált tesztek
* manuális tesztek

### Programozott tesztek

Az adott funkció működésétől függően az alkalmazás jelenleg két típusú programozott tesztet tartalmaz.

A tisztán back-end feladatokat ellátó osztályok unit tesztekkel rendelkeznek. A unit tesztek nem veszik igénybe a Spring alkalmazás kontextust a futtatáshoz, így az nem kerül inicializálásra. A függőségek befecskendezése a Mockito könyvtár segítségével történik. Emiatt a tesztek adatbázis kapcsolatot sem igényelnek.

A UI-hoz kapcsolódó osztályok automatizált tesztje a Selenium könyvtár segítségével történik. A Selenium segítségével lehetőség van már futó alkalmazás mellett egy igénylés automatizált rögzítésére, így biztosítva, hogy az alkalmazás továbbra is működképes legyen.

Maven projekt lévén a tesztek a src\test könyvtárban találhatók.

A tesztek futtatása a következő Maven parancs kiadásával történik:

* mvn verify -Pit,production

Ez a parancs az *it* Maven profilban futtatja a teszteket, amely elindítja a Spring Boot szervert a Selenium tesztek futtatása előtt, majd leállítja azt utána.

### Manuális tesztek

A manuális tesztek szintén a teljes igénylési folyamatot lefedik a felhasználó szempontjából. A tesztesetek és azok lépései az elvárt eredményekkel a *Mellékletek* fejezetben találhatók.

# Összefoglalás

A szakdolgozatom keretében egy webes alkalmazást készítettem. Habár a közelmúltban már sikerült fejlesztőként elhelyezkednem, az inspirációt a szakdolgozatom témájához a korábbi szakmai múltam alapján bankszektorból, azon belül is a lakossági hitelezésből merítettem.

Több hiteligénylési rendszerrel is dolgoztam és érdekelt, hogy modern, nyílt forráskódú keretrendszerekkel egy hiteligénylési folyamat milyen formában valósítható meg. Ezért esett a választásom egy személyikölcsön-hiteligénylés rögzítését támogató folyamatra, ami jelen esetben a fiókhálózatban történő rögzítést jelenti.

A folyamat során bejelentkezést követően az ügyintéző beazonosítja vagy rögzíti az ügyfelet, majd különböző információkat ad meg az igényléssel kapcsolatban. A háttérben megtörténik a bírálat, majd annak pozitív kimenetele esetén lehetőség van választani a kalkulált ajánlatok közül.

A program intenzíven kommunikál egy adatbázissal és a folyamat több pontján is tartalmaz ellenőrzéseket.

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

Emellett 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 kódrészletekből 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 megjelenített képernyők építőkövei a Vaadin által biztosított komponensek, de természetesen a feladat elvégzéséhez nagy mértékben szükséges volt ezek kiterjesztése és kompozíciója.

Szakdolgozatom elkészítéséhez szükség volt behatóbban megismerkednem a Vaadinnal, és bár természetesen a jövőben még tovább bővíthető ez a tudás, de úgy érzem, hogy mind a konkrét keretrendszerrel, mind általánosan a webfejlesztéssel kapcsolatban is hasznos ismereteket szereztem.

Szintén elmondható, hogy az alkalmazás a **Spring Boot** keretrendszert használja. A Spring 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 szintén jobban el kellett merülnöm a Spring keretrendszer használatában. A Spring által biztosított funkciók közül a jelentőségük tekintetében mindenképp kiemelendő a függőségek befecskendezése, valamint az adatbázis réteg absztrakciója.

Ahogy fentebb említettem, a munka megkezdése előtt nagy mértékben motivált az a kérdés, hogy vajon az elvárásaimnak megfelelnek-e ezek az eszközök, a banki környezetben javasolt-e ezek használatával alkalmazást építeni. Az elkészült alkalmazást vizsgálva arra a következtetésre jutottam, hogy ezen eszközök egyértelműen megfelelnek a banki környezet által támasztott elvárásoknak. Természetesen szükséges energiát fektetni a keretrendszer megértésébe, de úgy gondolom, hogy az alapvető ismeretek megszerzését követően a Vaadin alkalmas mind egyszerűbb, mind akár bonyolultabb alkalmazások készítésére is.

# 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 mobilbarát. Figyelembevéve, 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 érhető el, így a következő lépéseknél tartalmazniuk kell az i18n bevezetését.
* 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[[6]](#footnote-6).

# Irodalomjegyzék

## Egyéb források

1. Vaadin-nal kapcsolatban gyakran feltett kérdések, <https://vaadin.com/faq>, 2022.11.05.
2. Vaadin-nal kapcsolatban gyakran feltett kérdések, <https://vaadin.com/faq>, 2022.11.05.
3. Vaadin dokumentáció, komponensek <https://vaadin.com/docs/latest/components>, 2022.11.07.
4. Vaadin dokumentáció, események kezelése <https://vaadin.com/docs/latest/application/events>, 2022.11.07.
5. Vaadin: Scalability Report, 2019, letölthető: <https://vaadin.com/scalability>
6. Hivatkozás a Camundára, <https://camunda.com/>, 2022.11.15.

# Ábrajegyzék

[1. ábra Bejelentkezés az alkalmazásba 7](#_Toc123245274)

[2. ábra Az alkalmazás menüje 8](#_Toc123245275)

[3. ábra Kezdőképernyő 8](#_Toc123245276)

[4. ábra Ügyfél keresés 9](#_Toc123245277)

[5. ábra Ügyfél találati lista 9](#_Toc123245278)

[6. ábra Új ügyfél rögzítése gomb 10](#_Toc123245279)

[7. ábra Ügyfél személyes adatok 10](#_Toc123245280)

[8. ábra Hibásan kitöltött mező 10](#_Toc123245281)

[9. ábra Létező ügyfél hibaüzenet 11](#_Toc123245282)

[10. ábra Hitelösszeg megadása 11](#_Toc123245283)

[11. ábra Munkáltató rögzítése 12](#_Toc123245284)

[12. ábra Munkáltató törlése hibaüzenet 12](#_Toc123245285)

[13. ábra Jövedelem rögzítése 13](#_Toc123245286)

[14. ábra Igénylő további adatainak a rögzítése 13](#_Toc123245287)

[15. ábra Súgó példa 13](#_Toc123245288)

[16. ábra Bírálat részletes eredménye 14](#_Toc123245289)

[17. ábra Ajánlatok választása 14](#_Toc123245290)

[18. ábra Szerződés nyomtatása 15](#_Toc123245291)

[19. ábra Folytatható igénylés kiválasztása 16](#_Toc123245292)

[20. ábra Use case diagram 18](#_Toc123245293)

[21. ábra Új igénylés folyamatábrája 1. 20](#_Toc123245294)

[22. ábra Új igénylés folyamatábrája 2. 21](#_Toc123245295)

[23. ábra Egy view-ként funkcionáló osztály deklarációja 22](#_Toc123245296)

[24. ábra Főbb komponensek 23](#_Toc123245297)

[25. ábra Jogosultság kezelés annotációkkal 26](#_Toc123245298)

[26. ábra HomeView UML 27](#_Toc123245299)

[27. ábra SelectClientView UML 27](#_Toc123245300)

[28. ábra NewClientView UML 28](#_Toc123245301)

[29. ábra ApplicationBasicView UML 28](#_Toc123245302)

[30. ábra IncomeView UML 29](#_Toc123245303)

[31. ábra UnderwritingResultsView UML 29](#_Toc123245304)

[32. ábra OffersView UML 30](#_Toc123245305)

[33. ábra ContractView UML 30](#_Toc123245306)

[34. ábra Theme annotáció használata 31](#_Toc123245307)

[35. ábra WorkflowService UML 32](#_Toc123245308)

[36. ábra WorkflowStep UML 32](#_Toc123245309)

[37. ábra UnderwritingService UML 33](#_Toc123245310)

[38. ábra UnderwritingStep UML 34](#_Toc123245311)

[39. ábra Függőségek 38](#_Toc123245312)

[40. ábra Relációk és kapcsolataik 39](#_Toc123245313)

[41. ábra Relációk 41](#_Toc123245314)

[42. ábra Példa az objektum-reláció leképezésre annotációk használatával 41](#_Toc123245315)

# Mellékletek

## Képernyő tervek (wireframes)

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, chat or text message

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface

Description automatically generatedText

Description automatically generatedA picture containing text

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generated

## Manuális tesztesetek

|  |  |  |
| --- | --- | --- |
| Teszteset: | Nyitó képernyő és igénylés indítása | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Alkalmazás elindítása a böngészőben | A bejelentkező képernyő jelenik meg |
| 2 | Bejelentkezés az alkalmazásba | Bejelentkezés sikeresen elvégezhető, a nyitóképernyő jelenik meg két gombbal: "Create new application" és "Continue application" |
| 3 | Új igénylés indítása a "Create new application" gomb megnyomásával | A gomb megnyomható, az alkalmazás a következő képernyőre navigál |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Új ügyfél rögzítése | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni az ügyfél képernyőre | A rögzítés rendben elvégezhető |
| 2 | Ügyfél képernyőn a "Create new client gomb megnyomása" | A gomb megnyomható, az alkalmazás a következő, ügyfél rögzítés képernyőre navigál |
| 3 | Új igénylés indítása a "Create new application" gomb megnyomásával | A gomb megnyomható, az alkalmazás a következő képernyőre navigál |
| 4 | Keresztnév rögzítése | Rögzíthető, minimum 3 karakter megadása szükséges |
| 5 | Vezetéknév rögzítése | Rögzíthető, minimum 3 karakter megadása szükséges |
| 6 | Születési dátum rögzítése | Rögzíthető, megjelenik a dátum választó komponens, csak múltbeli dátum rögzíthető |
| 7 | Anyja neve rögzítése | Rögzíthető, minimum 3 karakter megadása szükséges |
| 8 | Nem rögzítése | Csak az értékkészletnek megfelelő válasz rögzíthető |
| 9 | Cím ország elemének rögzítése | Rögzíthető, minimum 3 karakter megadása szükséges |
| 10 | Cím város elemének rögzítése | Rögzíthető, minimum 2 karakter megadása szükséges |
| 11 | Cím irányítószám elemének rögzítése | Rögzíthető, minimum 4 karakter megadása szükséges |
| 12 | Cím utca, házszám elemének rögzítése | Rögzíthető, minimum 3 karakter megadása szükséges |
| 13 | Telefonszám rögzítése | A telefonszám rögzíthető |
| 14 | Email cím rögzítése | Csak megfelelő formátumú email cím rögzíthető |
| 15 | Kitöltés kötelezőség ellenőrzése minden mezőre | A továbblépéshez minden mező megfelelő kitöltése kötelező |
| 16 | Minden mező kitöltése | Az "Add client" gomb aktív lesz |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Ügyfél keresés | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni az ügyfél képernyőre | A rögzítés rendben elvégezhető |
| 2 | Keresztnév megadásával keresés | A megadott karakter sorozatot tartalmazó keresztnevű ügyfelek jelennek meg a táblázatban |
| 3 | Vezetéknév megadásával keresés | A megadott karakter sorozatot tartalmazó vezetéknevű ügyfelek jelennek meg a táblázatban |
| 4 | Keresztnév és vezetéknév megadásával keresés | A megadott karakter sorozatot tartalmazó kereszt- és vezetéknevű ügyfelek jelennek meg a táblázatban |
| 5 | A táblázatban egy sorra kattintás | Az ügyfél kijelölhető, a "Select client" gomb aktív lesz |
| 6 | "Select client" gomb megnyomás | A gomb megnyomható, az alkalmazás a következő képernyőre navigál |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Alapadatok megadása | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni az alapadat képernyőre | A rögzítés rendben elvégezhető |
| 2 | Hitelösszeg megadása | Hitelösszeg megadható az aktuális termék minimum és maximum értékek között (zárt intervallum) |
| 3 | Futamidő megadása | Futamidő megadható az aktuális termék minimum és maximum értékek között (zárt intervallum) |
| 4 | Hitelösszeg módosítása | A törlesztőrészlet értéke frissül annuitásos képlet szerint |
| 5 | Érvénytelen hitelösszeg megadása | "Continue" gomb inaktív |
| 6 | Érvényes paraméterek megadása | "Continue" gomb aktív, megnyomva az alkalmazás a következő képernyőre navigál |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Munkáltató rögzítése | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni a jövedelem képernyőre | A rögzítés rendben elvégezhető |
| 2 | "Add new employer" gomb megnyomása | Megjelenik az új munkáltató űrlap |
| 3 | Cégnév rögzítése | A cégnév rögzíthető |
| 4 | Adószám rögzítése | Az adószám rögzíthető |
| 5 | Munkaviszony típusának rögzítése | Csak az értékkészletnek megfelelő válasz rögzíthető |
| 6 | Munkaviszony kezdete rögzítése | Rögzíthető mindkét munkaviszony típus mellett, megjelenik a dátum választó komponens, csak múltbeli dátum rögzíthető |
| 7 | Munkaviszony vége rögzítése | Csak határozott munkaviszony típus mellett rögzíthető, megjelenik a dátum választó komponens, csak jövőbeli dátum rögzíthető |
| 8 | Kitöltés kötelezőség ellenőrzése minden mezőre | A továbblépéshez minden mező megfelelő kitöltése kötelező |
| 9 | Minden mező kitöltése | Az "Add employer" gomb aktív, megnyomva a rögzített munkáltató bekerül a táblázatba |
| 10 | Újabb munkáltató rögzítése | További munkáltató is rögzíthető |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Jövedelem rögzítése | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni a jövedelem képernyőre | A rögzítés rendben elvégezhető |
| 2 | "Add new income" gomb megnyomása | Megjelenik az új jövedelem űrlap |
| 3 | Jövedelem típusának rögzítése | Csak az értékkészletnek megfelelő válasz rögzíthető |
| 4 | Munkáltató rögzítése munkabér mellett | A korábban rögzített munkáltatók közül egy értékkészletből választható |
| 5 | Munkáltató rögzítése nyugdíj mellett | Munkáltató nem rögzíthető |
| 6 | Jövedelem összegének rögzítése | A jövedelem összege rögzíthető, csak pozitív szám adható meg |
| 7 | Kitöltés kötelezőség ellenőrzése minden mezőre | A továbblépéshez minden mező megfelelő kitöltése kötelező |
| 8 | Minden mező kitöltése | Az "Add income" gomb aktív, megnyomva a rögzített jövedelem bekerül a táblázatba |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Igénylő részletes adatok rögzítése | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni az igénylő részletes adatok képernyőre | A rögzítés rendben elvégezhető |
| 2 | Iskolai végzettség rögzítése | Csak az értékkészletnek megfelelő válasz rögzíthető |
| 3 | Családi állapot rögzítése | Csak az értékkészletnek megfelelő válasz rögzíthető |
| 4 | Háztartás létszám rögzítése | Csak pozitív szám rögzíthető |
| 5 | Eltartottak számának rögzítése | Csak nulla vagy pozitív szám rögzíthető |
| 6 | Fennálló tartozás rögzítése | Csak nulla vagy pozitív szám rögzíthető |
| 7 | Kitöltés kötelezőség ellenőrzése minden mezőre | A továbblépéshez minden mező megfelelő kitöltése kötelező |
| 8 | Minden mező kitöltése | Az "Next" gomb aktív, megnyomva az alkalmazás a következő képernyőre navigál |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Bírálat eredménye képernyő | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni a bírálat eredménye képernyőre | A rögzítés rendben elvégezhető |
| 2 | Bírálati elemek megtekintése | A képernyőn listázásra kerülnek a bírálati elemek, engedélyezés esetén zöld színnel jelölve, elutasítás esetén piros színnel jelölve |
| 3 | Bírálati szabályok megtekintése | A bírálati elemek lenyithatóak, felsorolásra kerülnek a szabályok eredményei |
| 4 | "Next" gomb megnyomása | A gomb megnyomható |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Elutasítás képernyő | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés rögzítése, a bírálat eredménye elutasítás, tovább lépés a bírálat eredménye képernyőről | Elutasító képernyő jelenik meg |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Ajánlat választás | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | Új igénylés folyamatban eljutni az ajánlatválasztás képernyőre | A rögzítés rendben elvégezhető |
| 2 | Ajánlatra kattintás | Az ajánlat kijelölésre kerül, a "Select offer" gomb aktív lesz |
| 3 | Kijelölt ajánlatra kattintás | Az ajánlat kijelölése megszűnik, a "Select offer" gomb inaktív lesz |
| 4 | "Select offer" gomb megnyomása | A gomb megnyomható, az engedélyezés képernyő jelenik meg |

|  |  |  |
| --- | --- | --- |
| Teszteset: | Igénylés folytatása | |
| # | Tesztlépés | Elvárt eredmény |
| 1 | A nyitó képernyőn a "Continue application" gomb megnyomása | A gomb megnyomható, az igényléseket listázó táblázat jelenik meg |
| 2 | Egy igénylésre kattintás | Az igénylés kijelölhető, a "Continue existing application" gomb aktív lesz |
| 3 | "Continue existing application" gomb megnyomása | Az igénylés az aktuális státusznak megfelelő képernyőn folytatódik |

1. <https://vaadin.com/faq>, 2022.11.05. [↑](#footnote-ref-1)
2. <https://vaadin.com/faq>, 2022.11.05. [↑](#footnote-ref-2)
3. <https://vaadin.com/docs/latest/components>, 2022.11.07. [↑](#footnote-ref-3)
4. <https://vaadin.com/docs/latest/application/events>, 2022.11.07. [↑](#footnote-ref-4)
5. Vaadin: Scalability Report, 2019, letölthető: <https://vaadin.com/scalability> [↑](#footnote-ref-5)
6. <https://camunda.com/>, 2022.11.15. [↑](#footnote-ref-6)