

DEBRECENI EGYETEM
INFORMATIKAI KAR

SZENTESI VIKTOR

**WEBES ALKALMAZÁSFEJLESZTÉS:
EGY BANKI ÜGYFÉLKAPCSOLAT TÁMOGATÓ
RENDSZER**

SZAKDOLGOZAT

KONZULENS: KUKI ATTILA

DEBRECEN, 2007

TARTALOMJEGYZÉK

BEVEZETÉS	4
ALKALMAZOTT ESZKÖZÖK, TECHNOLOGIÁK	5
1. 1. A PHP	5
1. 1. 1. MI A PHP?	5
1. 1. 2. MIÉRT ÉPPEN PHP?	7
1. 2. A MYSQL	8
1. 2. 1. MI A MYSQL?	8
1. 2. 2. A MYSQL LEGFONTOSABB TULAJDONSÁGAI	9
1. 3. AZ UML	10
1. 4. EGYÉB	13
RENDSZERFEJLESZTÉS, RENDSZERSZERVERZÉS	14
2. 1. A FEJLESZTÉS KLASSZIKUS FOLYAMATA	14
2. 2. SZOFTVERFEJLESZTÉSI MODELLEK	16
2. 2. 1. A VÍZESÉS MODELL	16
2. 2. 2. A SPIRÁLIS MODELL	17
2. 2. 3. AZ ITERATÍV MODELL	18
2. 2. 4. AZ INKREMENTÁLIS MODELL	19
2. 3. RENDSZERKÖVETELMÉNYEK ÉS ELEMZÉS	20
2. 3. 1. A USE CASE-EKRŐL ÁLTALÁBAN	20
2. 3. 2. A RENDSZER HASZNÁLATI ESETEI	22
2. 3. 2. 1. SZEREPLŐK	23
2. 3. 2. 2. FELHASZNÁLÓI ADMINISZTRÁCIÓ	24
2. 3. 2. 3. ÜGYFÉL ADMINISZTRÁCIÓ	31
2. 3. 2. 4. ÜGYFÉL KOMMUNIKÁCIÓ	33
2. 3. 2. 5. FELADATOK	39
2. 3. 2. 6. LEKÉRDEZÉSEK	44
2. 4. TERVEZÉS ÉS SPECIFIKÁCIÓ	47
2. 4. 1. RENDSZERFOLYAMATOK MODELLEZÉSE	47
2. 4. 1. 1. AZ AKTIVITÁS DIAGRAMOKRÓL	47
2. 4. 1. 2. EGY AKTIVITÁS DIAGRAM PÉLDA	48

2. 4. 2. OSZTÁLYOK KIALAKÍTÁSA	50
2. 4. 2. 1. AZ OSZTÁLYDIAGRAMOKRÓL	50
2. 4. 2. 2. A RENDSZER OSZTÁLYAI	51
2. 4. 3. AZ ADATBÁZIS MEGTERVEZÉSE	53
2. 4. 4. AZ ALKALMAZÁS KÜLSŐ MEGJELENÉSE	56
2. 4. 5. A HARDVER	57
2. 5. VERIFIKÁCIÓ ÉS VALIDÁCIÓ	57
2. 6. IMPLEMENTÁCIÓ	58
2. 6. 1. AZ ALKALMAZÁS FELÉPÍTÉSE	58
2. 6. 2. BIZTONSÁG	60
2. 7. TESZTELÉS ÉS TELEPÍTÉS	60
A RENDSZER BEMUTATÁSA	62
ÖSSZEFOGLALÁS	68
MELLÉKLETEK	69
IRODALOMJEGYZÉK	75

BEVEZETÉS

Az egyetemen végzett tanulmányaim mellett részletesebben a webprogramozás, honlapfejlesztés területével foglalkoztam, ezért választottam a szakdolgozat témájaként webes alkalmazásfejlesztést. Két évet dolgoztam közgazdászként a vállalati hitelezés területén az OTP Bank Nyrt.-nél, ezért egy banki ügyfélkapcsolat támogató rendszer fejlesztését tűztem ki célul. A webes, intranetre tervezett rendszer a hitelezés területén a napi, ügyfelekkel történő ügyintézését segítheti. A fejlesztés során a rendszerkövetelményeket saját tapasztalataim során állapítottam meg, a fejlesztésbe a bank nem került bevonásra.

A dolgozat a fejlesztés menetét mutatja be. Az első fejezet a fejlesztéshez felhasznált eszközöket taglalja. Webes alkalmazás fejlesztésekor szükség van egy programozási nyelvre, amellyel lehetőség van a dinamikus oldalak kezelésére. Választásom a PHP-re esett. Ebben a fejezetben részletezem, hogy miért. A webes alkalmazások másik követelménye, hogy az adatokat valahol tárolni lehessen. Adatbázis-kezelőként a MySQL-t használtam, amely szintén bemutatásra kerül. A fejezet további részében pedig a tervezéshez használt UML-ről esik szó.

A dolgozat második fejezete a fejlesztés konkrét szakaszait taglalja, rövid rendszerfejlesztési bevezető után, a klasszikus fejlesztési fázisok alapján. Bemutatásra kerülnek a tervezés során elkészített UML diagramok és dokumentációk, a használati esetektől az osztálydiagramokig. Majd az adatbázis és hardver terveket követően a modell verifikációját és validációját végeztem el. A fejezet utolsó részei az implementációval, a teszteléssel és telepítéssel foglalkoznak. Az implementációs részben bemutatom az alkalmazás fájlok felépítését, és az alkalmazási biztonságáért felelős funkciókat.

Végül, a dolgozat utolsó, harmadik fejezete a kész rendszer funkcióinak áttekintésére szolgál. A rendszer bemutatását képernyőképek segédletével, felhasználói fiókként haladva végzem el.

1

ALKALMAZOTT ESZKÖZÖK, TECHNOLÓGIÁK

1. 1. A PHP

1. 1. 1. Mi a PHP?

A PHP egy webes alkalmazás fejlesztő nyelv, melyet webfejlesztők írtak webfejlesztők számára. A PHP szó a PHP: Hypertext Preprocessor-t („hipertext előfeldolgozó”) jelenti. Eredeti neve Personal Home Page Tools („személy honlap eszközök”) volt, ebből jön a PHP rövidítés. Később ezt terjesztették ki az újabb, megfelelőbb elnevezésre. (Converse – Park – Morgan, 2004)

A nyelv fő célja, hogy elősegítse a webfejlesztők számára a dinamikusan generált oldalak gyors elkészítését. (Vaswani, 2005)

A PHP megalkotása Rasmus Lerdorf nevéhez fűződik, aki 1994-ben, személyes használatra írt egy kódot, amellyel nyomon követte az oldalának látogatottságát. A következő évben kiadta első csomagját, a Personal Home Page Tools-t.

A PHP jelenleg az ötödik verziójánál tart, melyet PHP5-nek neveznek. Az ötödik verziónál a legnagyobb változást az jelentette, hogy jobban koncentrált az objektumorientált programozásra (object oriented programming – OOP). A PHP már nagyon jó volt a procedurális programozásban, azonban komplexebb, nagyobb alkalmazásokat jobban szolgálja az OOP. Pl. a PHP4 még érték szerinti paraméterátadással dolgozott, referenciák használata helyett. A PHP5 ezen is változtatott. A PHP5-ben kibővítették a hibakezelést és lehetőséget biztosítottak a külső forrásból (pl. Java) származó objektumok integrálására. (Glass – Le Scouarnec – Naramore – Mailer – Stolz – Gerner, 2004)

A PHP egy szerveroldali szkript nyelv, amely beágyazható HTML-be vagy futtatható különálló bináris alkalmazásként. A szerveroldali programozás egy webes alkalmazásnál szkriptek futtatását jelenti, amely után az outputot elküldjük a böngészőnek. A PHP kód a szerveren fut, és nem a kliensen, ezért a PHP kód

teljesen független a felhasználó böngészőjétől (Vaswani, 2005). Amit a PHP csinál, tulajdonképpen láthatatlan a végfelhasználó számára. Ugyanis, ha valaki megnéz egy PHP oldalt, nem tudja megmondani, hogy PHP-ben vagy csak HTML-ben íródott, mert a PHP is HTML-t állít elő. (Converse – Park – Morgan, 2004)

A PHP főleg szerver-oldali szkriptek írására készült, azaz bármire képes, amit más CGI programok el tudnak végezni, ilyen funkciók az űrlap adatok feldolgozása, dinamikus tartalom generálása vagy süti küldése és fogadása. De a PHP ennél sokkal többet tud.

Három fő területen használnak PHP programokat:

- Szerver oldali programozás. Ez a hagyományos, és fő használati formája a PHP-nek. Három komponens szükséges ahhoz, hogy ezt a formát használhassuk. Az első a PHP értelmező (CGI vagy szerver modul formájában), egy webszerver és egy webböngésző. Egy webszerverrel mindenképpen rendelkezni kell, megfelelően telepített és beállított PHP-vel. A PHP program kimenetét a webböngészővel lehet megtekinteni, a szerveren keresztül elérve a szkriptet.
- Parancssori programozás. PHP programok szerver és böngésző nélkül is futtathatóak. Ilyen környezetben csak a PHP értelmezőre van szükség. Ebben a formában gyakran cron-al (ütemező Windows esetén) futtatott programokat írnak, vagy egyszerű szövegfeldolgozó szkripteket készítenek.
- Kliens-oldali ablakozós alkalmazások írása. A PHP valószínűleg nem a legjobb nyelv ablakozós alkalmazások írásához, de a PHP-GTK használható ilyen programok írásához. Ezt használva lehetőség van operációs rendszerfüggetlen programok írására is. A PHP-GTK a PHP egy kiterjesztése, nem érhető el a hivatalos PHP csomagban. (Bakken – Schmid, 2002)

A PHP-t helyettesítő termékek lehetnek ebben a szegmensben a Microsoft Active Server Pages (ASP), a Macromedia (Adobe) Coldfusion és a Sun Java Server Pages (JSP). Szokták azt mondani, hogy a PHP a nyílt forráskódú ASP, mert a két nyelv ugyanazt tudja. Azonban a megfogalmazás nem helytálló, mert az ASP később született.

A PHP az Apache HTTP Server hivatalos modulja, amely a világ piacvezető ingyenes webservere – a World Wide Web (www) 67%-án fut. Ez azt jelenti, hogy a PHP szkript motorja beépíthető a webserverbe, amely gyorsabb futást, jobb memóriaallokációt és könnyebb karbantartást eredményez. Ahogy az Apache, a PHP is platformfüggetlen és nyílt forráskódú. (Converse – Park – Morgan, 2004)

1. 1. 2. MIÉRT ÉPPEN PHP?

A PHP használata mellett sok érv szól, összegyűjtöttem a legfontosabbakat (Converse – Park – Morgan, 2004 és Vaswani, 2005 alapján):

Ingyenes

A PHP semmibe nem kerül. Ez jelentősen csökkenti a fejlesztési költségeket. Továbbá az Apache-csal és MySQL-lel együtt a legolcsóbb webserver – adatbázis – fejlesztőeszköz kombinációt alkotja, alacsony gépigénnyel.

Nyílt forráskódú

A fejlesztőket segíti, mert nyelv hibái hamarabb kerülnek javításra, és az újabb technológiák gyorsabban kerülnek integrálásra az érintett fejlesztők nagy száma miatt.

Gyors

A PHP szkriptek független mérések által bizonyítottan gyorsabban futnak, mint más szkriptek (JSP, ASP.NET, Perl). A PHP5-tel a sebesség csak fokozódott.

Egyszerű

A PHP konzisztens és logikus szintaktikával rendelkezik, kezdők számára is viszonylag könnyen tanulható.

Hordozható, platform független

A PHP majdnem teljesen platform független, azaz kompatibilis szinte mindegyik webserverrel és operációs rendszerrel. Egy Windows alatt megírt kódot bármilyen

nehézség nélkül lehet alkalmazni UNIX-nál is. Ugyanis a PHP kódhoz csak egy interpreter szükséges, fordítani nem kell.

Kiterjeszthető

A PHP alkotói a jövőre gondolva egy kiterjeszthető architektúrát alakítottak ki a nyelvben, amely lehetővé teszi új technológiák bevezetését moduláris kiterjesztéseken keresztül. Ez segít mindig „frissen tartani” a PHP-t. Ilyen modulok pl.: képek, PDF, SWF fájlok dinamikus létrehozása, elektronikus fizetések kezelése, Perl, Java, COM kódok futtatása, stb.

1. 2. A MySQL

1. 2. 1. Mi a MySQL?

A MySQL egy nyílt forráskódú, SQL relációs adatbázis kezelő rendszer (Relational Database Management System, RDBMS). A MySQL világ legnépszerűbb nyílt forráskódú RDBMS-e, gyorsasága, stabilitása, egyszerű kezelhetősége és ingyenes licencfeltételei (GNU General Public License) miatt. Hivatalos adatok szerint több mint 5 millió oldal használja a világon a MySQL-t, a legnagyobb felhasználók között olyan cégek szerepelnek, mint a Yahoo!, Google, Cisco, NASA, HP. (Vaswani, 2005) Az adatbázisok talán a leghasznosabb szoftvercsaládba tartoznak. A MySQL – mint sok más ingyenes és fizetős versenytársa – nem alkot addig adatbázist, amíg nem adunk neki szerkezetet és formát. Ez az adatbázis és a RDBMS közötti különbségre vezethető vissza: az adatbázist az RDBMS és a felhasználói igények alkotják. (Converse – Park – Morgan, 2004)

A MySQL története 1979-ig vezethető vissza, amikor Michael „Monthy” Widenius létrehozott egy UNIREG nevű adatbázis rendszert a TcX nevű svéd informatikai cég számára. A rendszer nem működött megfelelően, ezért a cég keresett egy másik szoftvert. Amit kipróbáltak, az David Hughes msql-e volt, amely szintén kudarcba fulladt. Ezek után kezdett neki Widenius egy új rendszer, adatbázis szerver fejlesztéséhez, melyhez az msql API-ját is felhasználta. A rendszer, amelyet egy kisebb csoport részére adtak ki 1996-ban, a MySQL 1.0 nevet kapta. A MySQL

névben az SQL struktúrált lekérdező nyelvet (Structured Query Language) jelent. Az SQL egy standardizált nyelv, melyet adatbázisok elérésére használnak, és az ANSI/ISO SQL Standard definiálja. A „My” előtag eredete az elnevezésben nem tisztázott. Egyrészt sok könyvtár elnevezése „my” előtaggal kezdődött, másrészt Monthy lányát My-nak hívták. (Vaswani, 2005)

1. 2. 2. A MYSQL LEGFONTOSABB TULAJDONSÁGAI

Sebesség

Egy relációs adatbázis kezelő rendszerénél a sebesség – ami egy lekérdezés végrehajtását és az eredmény visszaadását jelenti – a legfontosabb. A MySQL ezen a téren elég magasra tette a mércét, megelőzve szinte az összes versenytársát, mint pl. a Microsoft SQL Servert vagy az IBM DB2-t. A MySQL gyorsaságát az alábbiak segítik: architektúrája teljes mértékben többszálú; speciális optimalizációja van komplex feladatokra, pl. a join-okra vagy indexelésre; rendelkezik egy cache-sel a lekérdezésekre, amely fokozza a teljesítményt, mindenféle speciális programozás nélkül. (Vaswani, 2005)

Megbízhatóság

A MySQL-nek mély gyökerei vannak a nyílt forráskódú közösségben, ezért minden egyes új verziót rengeteg új felhasználó tesztl. Vannak kimondottan olyan tesztek is, amelyeknek az célja, hogy kiakasszák a rendszert. A MySQL nagy szervezeteknél is bevált, mint a NASA, ahol igazán kritikus a rendszer működése. (Vaswani, 2005)

Biztonság

Ha több felhasználós adatbázisról van szó, akkor a biztonság nagyon fontos. Ezért a MySQL olyan biztonságos, amennyire csak lehet. A fejlesztők egy kifinomult hozzáférés és jogosultság kezelést alakítottak ki. A rendszer 5 rétegű jogosultság hierarchiával rendelkezik, amely lehetővé teszi az adatbázis adminisztrátorok számára az adatok megfelelő védelmét. Továbbá minden felhasználó részére külön be lehet állítani, mely táblákhoz férhet hozzá, milyen parancsokat futtathat. (MySQL AB, 2005)

Kapacitás

A MySQL meglehetősen nagy és bonyolult adatbázisokat tud kezelni, viszonylag kevés teljesítménycsökkenéssel. Maga a MySQL weboldala olyan adatbázist használ, amely 50 millió rekordot tartalmaz. De ismernek olyan felhasználót is, aki 60 ezer táblával és 5 milliárd sorral dolgozik egy MySQL szerveren. (MySQL AB, 2005)

Platform függetlenség

Nemcsak a kliens alkalmazásai, hanem maga a MySQL szerver is nagyon sokféle operációs rendszeren fut. Ennek következtében az adatbázis könnyen hordozható egyik platformról a másikra. (Kofler, 2004)

Széleskörű API

A MySQL sok API-val rendelkezik, melyek lehetővé teszik, hogy a felhasználó az adatbázis-vezérelt alkalmazást az általa választott programozási nyelven írja meg (C, C++, Java, Perl, PHP, Python, Tcl.) (Kofler, 2004)

1. 3. Az UML

Az UML (Unified Modeling Language – „egységes modellező nyelv”) a szoftver- és rendszerfejlesztés standard modellező nyelve. Miért van erre szükség? A legegyszerűbb alkalmazástól a többretegű vállalati rendszerekig minden alkalmazás több 100 vagy akár több ezer szoftver- és hardverelemből állhat. A rendszerfejlesztésben a modellezés a komplexitás menedzselését szolgálja. A modellezés segít abban, hogy a rendszer legfontosabb elemeire koncentráljunk.

A modell a valóság egy absztrakciója, valós rendszer egy leegyszerűsített változata, ahol a rendszer kivitelezése, megvalósíthatósága, életképessége sokkal egyszerűbben vizsgálható, megérthető, átlátható, mintha magát teljes rendszert kellene egyből átlátni. Egy modellező nyelv állhat pseudo-kódokból, képekből, diagramokból, leírásokból, ill. bármiből, amely segít leírni a rendszert. (Hamilton – Miles, 2006)

Az UML egy formális modellező nyelv, mely olyan precíz, mint egy programozási nyelv. Rendelkezik szintaktikával és szemantikával, tehát, szabályok vannak arra,

hogy a nyelv elemeit hogyan kell egymáshoz illeszteni, és azok milyen jelentéssel bírnak. (Pilone – Pitman, 2005)

Az UML-t sok mindenre lehet használni, a legfontosabbak a következők:

- szoftvertervezés
- szoftver- vagy üzleti folyamatok összekapcsolása
- a leendő rendszer részleteinek kiemelése, elemzése
- meglévő rendszer, folyamat, szervezet dokumentálása.

Az UML alapvető építőeleme a diagram, amelyekből sokféle típus áll rendelkezésre, az általánosabbaktól a speciálisabbakig. Az UML diagramjairól mutat egy összesítést az 1. táblázat. Az UML legfontosabb eszközei a 2. fejezetben részletesen bemutatásra kerülnek.

1. táblázat
Az UML 2.0 diagramjai

Diagram neve	Mit modellez?
Használati eset (Use Case) diagram	A rendszer, a felhasználók és/vagy külső rendszer közötti interakciókat írja le. Segít a rendszer funkcióinak kialakításában is.
Aktivitás diagram	Egymás utáni és párhuzamos tevékenységek a rendszerben.
Osztály diagram	Osztályok, típusok, interfészek és a közöttük lévő kapcsolatok
Objektum diagram	Az osztálydiagramok definiált osztályok példányai, a rendszer számára fontos konfigurációban.
Szekvencia diagram	Objektumok közötti interakciók, ahol az interakciók sorrendje fontos.
Kommunikációs diagram	Az objektumok interakciójának módja, és az ehhez szükséges kapcsolatok
Idő diagram	Objektumok közötti interakciók, ahol az idő fontos tényező.
Interakció áttekintő diagram	Az szekvencia-, kommunikációs-, és idő diagramok összegzése, ahol nyomon követhető egy egész rendszeren keresztül menő folyamat.
Összetett rendszer diagram	Osztály vagy komponens tartalma, osztályok közötti kapcsolatok leírása
Komponens diagram	A rendszer lényeges komponensei, és az interfészek, melyeket az interakciókra használnak.
Csomag diagram	Az osztályok és komponensek csoportjainak hierarchiába szerveződése.
Állapot diagram	Egy objektum állapota a létezése alatt, és események, amelyek meg tudják változtatni az állapotát.
Bevetés diagram	A rendszer működése egy valós szituációban.

Forrás: Hamilton – Miles, 2006

1. 4. EGYÉB

Az alkalmazás fejlesztéséhez használt egyéb eszközök, segédsoftverek:

- HTML: Mivel a PHP maga is HTML-t generál, ezért a HTML ismerete elengedhetetlen.
- CSS: A grafikai megjelenítés egyszerű, átlátható formázását teszi lehetővé.
- PHP Designer 2007 és Notepad++: A PHP-kód szerkesztéséhez
- phpMyAdmin - 2.11.0: Az adatbázis szerkesztéséhez
- Visual Paradigm for UML 6.0 és 6.1: az UML diagramok és dokumentáció elkészítéséhez
- AllWebMenus PRO: Az alkalmazás menüjének létrehozásához
- Photoshop: a header kép elkészítéséhez.

2

RENDSZERFEJLESZTÉS, RENDSZERSZERVERZÉS

2. 1. A FEJLESZTÉS KLASSZIKUS FOLYAMATA

A szoftverfejlesztésnek sok fázisa lehet az alkalmazott tervezési módszertől, modelltől függően. A következőkben a fejlesztés általános szakaszai kerülnek röviden bemutatásra (O'Docherty, 2005 alapján):

Követelmények feltárása

A követelmények vizsgálatakor meg kell határoznunk, mit szeretnénk elérni a fejlesztendő szoftverrel. Ennek két nézőpontja van. Az egyik az üzleti modellezés, ahol a működési környezet, kontextus kerül meghatározásra. A másik a rendszerkövetelmények modellezése, ahol eldöntjük, milyen funkciókkal fog rendelkezni az alkalmazás. Világosan meg kell fogalmazni, hogy a rendszerben mit lehet majd csinálni, és mit nem - a későbbiekben ebből is tudjuk megítélni, hogy a fejlesztés befejeződött-e, sikeres volt-e.

Elemzés

Az elemzés azt a fázist jelenti, amelyben megértjük, mivel van dolgunk. Mielőtt a követelmények alapján elkezdenénk a tervezést, szükség van arra, hogy tisztában legyünk a releváns egyedekkel, tulajdonságaikkal és egymás közötti kapcsolataikkal.

Tervezés

A tervezés fázisban kidolgozzuk, hogyan oldjuk meg a problémát. Eldöntjük, milyen szoftver fogunk írni és hogyan kivitelezzük. A rendszertervezés a rendszert logikai és fizikai alrendszerekre bontja, azaz folyamatokra és hardverre, hálózatokra. Az alrendszerek tervezésénél az döntjük el, hogy a logikai alrendszereket hogyan formáljuk hatékony forráskóddá, ill. pl. hogy a számítógépek hogyan fognak egymással kommunikálni.

Specifikáció

A specifikáció a leggyakrabban használt értelemben a programozási elemek elvárt viselkedésének meghatározását jelenti. Az osztályspezifikáció például egy világos, félreérthetetlen leírása annak, hogy a szoftverkomponenseket hogyan kell használni, és hogyan fognak viselkedni, ha megfelelően használják őket.

A specifikáció felhasználható:

- tesztalkalmazás tervezésének alapjaként,
- annak bizonyítására, hogy a szoftver korrektül működik,
- bizonyos mértékig a szoftverkomponensek dokumentálására, az implementáció megkönnyítése érdekében,
- annak leírására, hogy a forráskódot hogyan lehet biztonságosan felhasználni más alkalmazásokhoz.

Implementáció

Az implementáció a kódírást jelenti, ahol a kódokból összeállnak az alrendszerek és maga a rendszer. Bár a tervezés szakaszában a nehéz kódolási döntések meghozatala már megtörtént, azaz a szoftverkomponensek megtervezésre, dokumentálásra kerültek, az implementációs fázisra is jut kreativitás: a programozó szabadon eldöntheti a belső működéseket.

Tesztelés

Amikor a szoftver elkészült, tesztelni szükséges, hogy a rendszer működése, funkciói megfelelnek-e az eredeti céloknak.

Jó ötlet, ha a programozók egy-egy elkészült kisebb programrészletet is tesztelnek, ahogy implementálnak, mert ezzel a végső forráskód minőségét javítják. A fő rendszertesztet ugyanis nem a programozók hajtják végre.

Telepítés

A telepítés a rendszer működésének elindítását jelenti a végfelhasználók részére, a szükséges szoftver és hardverelemekkel, továbbá felhasználói kézikönyvek és oktatóanyagok átadásával.

Karbantartás

A rendszer telepítése után történik az alkalmazás valós tesztje. Ekkor jöhetnek elő újabb, eddig nem észlelt hibák (bug), melyeket minél hamarabb ki kell javítani. A felhasználók észrevehetnek hiányosságokat, (amelyeket tudnia kellene a rendszernek, de nem tud) és javasolhatnak funkciókat, amelyek jobbá teszik a rendszert.

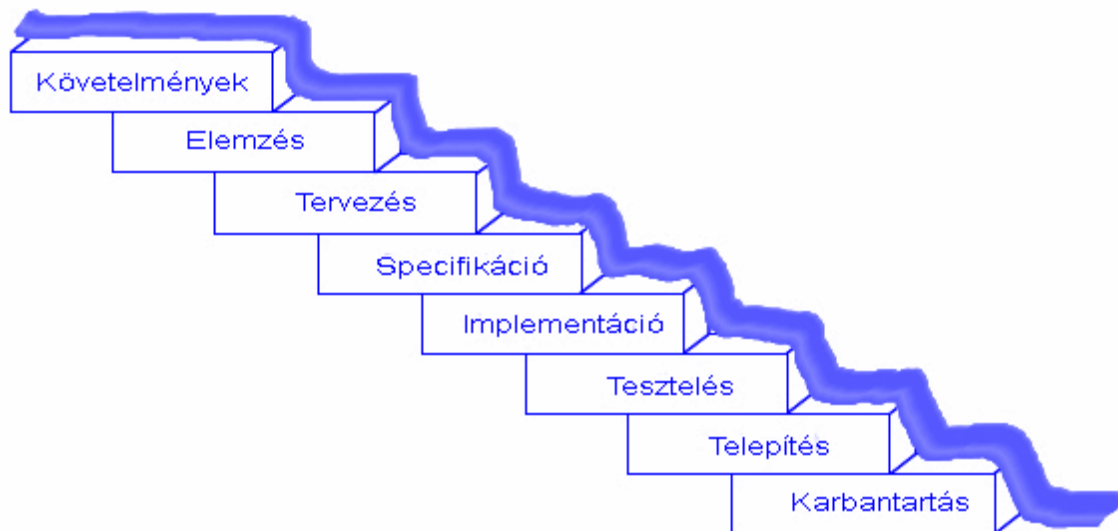
2. 2. SZOFTVERFEJLESZTÉSI MODELLEK

2. 2. 1. A VÍZESÉS MODELLE

A vízesés modell volt az első modell, amely széleskörűen elterjedt a szoftverfejlesztésben. A lényege, hogy a fejlesztés szépen halad a klasszikus - korábban részletezett - szakaszok mentén, és minden egyes szakasz sikeresen lezárul, mielőtt a következő elkezdődik.

1. ábra

A vízesés modell



Forrás: O'Docherty, 2005

A vízesés modell alapján a fejlesztést könnyű megtervezni, könnyű ütemezni. A módszer kitalálása jó ötlet volt, azonban nem közelíti megfelelően a valóságot. Még

ha meg is tudjuk viszonylag pontosan tervezni, hogy az egyes szakaszok elvégzése mennyi időbe kerül, amíg részletesen nem vizsgáljuk meg a problémát, addig nem fogjuk tudni, milyen nehézségekkel kerülhetünk szembe a folyamat során. Éppen ezért bármelyik fázis több idő alatt érhet véget, mint terveztük, amely kumulálódva az egész rendszer késedelmes szállítását eredményezi. Az időbeli csúszás alatt a megrendelő igényei is változhatnak. A módszer nem igazán engedi meg a menet közben felvetett változások miatti korrekciókat.

Az egyes szakaszokért felelős fejlesztőcsapat felfedezhet hibákat, amiket korábbi fázisokban követtek el. Kérdés, hogy ilyenkor mit lehet tenni? Visszaadni az előző szakasz felelőseinek? Nem, mert az a vizesés megmászását jelenti, és annak előrefelé kell haladnia. Vagy feljegyezni a hibákat, és a projekt végén kijavítani a dokumentációt? Ami, ha nem történik meg, a végső dokumentáció eltérhet a végső rendszertől.

A fenti problémák mellett a módszer képtelen arra, hogy újrafelhasználható kódot eredményezzen.

A fentiek ellenére, néhány esetben mégis érdemes alkalmazni a vizesés modellt:

- Ha ugyanazt a fejlesztést kis változtatásokkal ismételjük.
- Az iteratív modell segítségével
- Kis projektek kevés programozóval történő gyors fejlesztésére.

2. 2. 2. A SPIRÁLIS MODELL

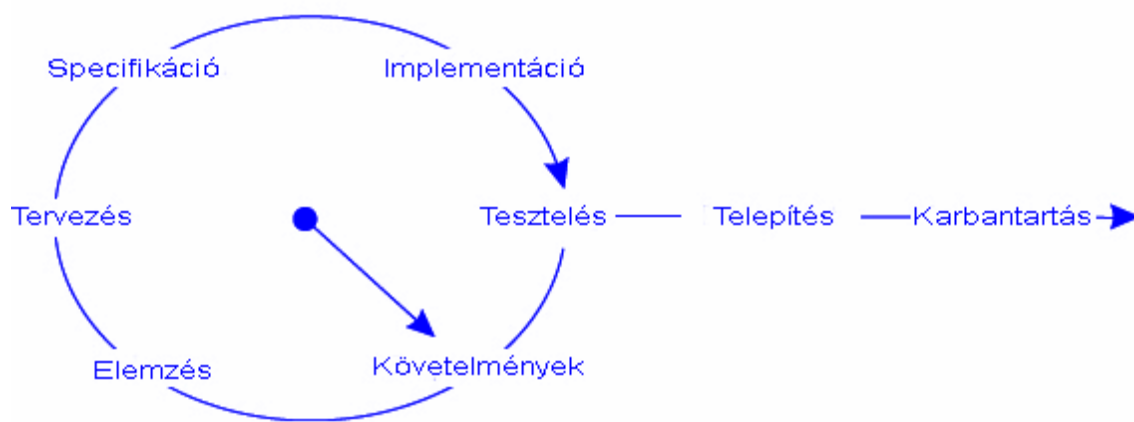
A spirális módszernél szintén a követelményekkel kezdjük, ahogy mindig. Folytatjuk a klasszikus fázisokkal, eljutunk az implementációig, kódolunk, még ha az előző fázisok által végzett munka nem is tökéletes. Aztán csinálunk egy gyors tesztet a felhasználók bevonásával. Így lényegében már végig is mentünk egyszer a folyamaton, megértettük valamilyen szinten a problémát, bevontuk a felhasználókat, akik még most tudnak korrigálni.

Aztán az új és korábbi ismeretekkel felvértezve végigmegyünk ismét a folyamaton, és javítjuk, amit szükséges. Aztán a folyamatot annyiszor ismételjük, ameddig el nem értük célunkat. Menet közben a fejlesztők számára is egyre világosabb lesz, mikorra

Látszik, hogy ez a módszer sokkal jobb az elődjénél, de nem tökéletes, nem elég rugalmas. Bármilyen kis probléma is derül ki menet közben, meg kell várni a tesztelést, és újra végig kell menni az összes fázison, hogy javítani lehessen.

2. ábra

A spirális modell

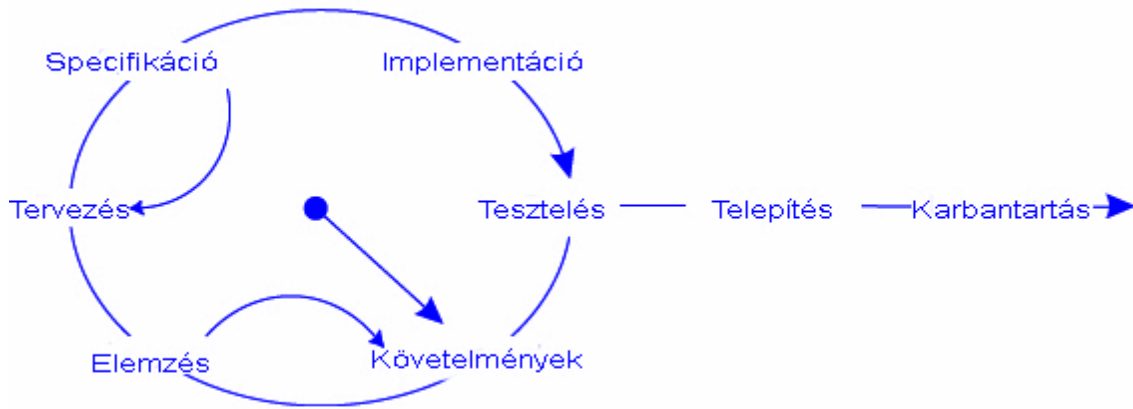


Forrás: O'Docherty, 2005

2.2.3. AZ ITERATÍV MODELL

Az iteratív modell (3. ábra) a spirális modell tovább fejlesztése. Az iteratív modellben többször van lehetőségünk a visszalépésre és a folytatásra, ezáltal egy fázis outputja könnyebben tökéletesíthető, mintha végig kellene menni a teljes cikluson előtte. A szoftvert már a fejlesztés korai fázisaiban egy kerek egészzé formálhatjuk, a megrendelők elégedettségére.

3. ábra
Az iteratív modell



Forrás: O'Docherty, 2005

2. 2. 4. AZ INKREMENTÁLIS MODELL

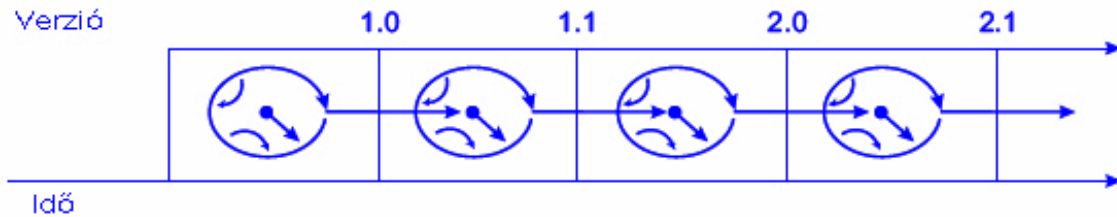
A modell szerint a fejlesztés fokozatosan, lépésenként halad. A lépés során végbement fejlesztés kap egy verziószámot. Az újabb lépés egy újabb verzió, a rendszer egy bővített változata, amely lecseréli az előzőt, és így tovább.

Az eddig felsorolt modellek kombinálásával talán az egyik leghatékonyabb fejlesztési módszer hozható létre (4.ábra):

A módszerben a vízesés modell mutatja meg a fázisok sorrendiségét. A spirális modell biztosít lehetőséget a folyamat nagy részének többszöri bejárhatóságára. Az újra bejárás új verziót hoz létre az inkrementális modell szerint. Az iteratív modell alapján az észrevett kis hibák visszalépéssel korrigálhatók.

4. ábra

A kombinált módszer



Forrás: O'Docherty, 2005

A következőkben a fejlesztés menetét mutatom be a klasszikus módszer szakaszai sorrendjében, bár maga a fejlesztés módja leginkább az iteratív modellhez hasonlított. Többször kellett visszalépnem, és a tervezés egy-két hiányossága csak a kódíráskor derült ki.

2. 3. RENDSZERKÖVETELMÉNYEK ÉS ELEMZÉS

A rendszerkövetelmények és az elemzés leírását egyszerre végzem el, mert a funkciókkal együtt talán könnyebb megérteni a felhasználási környezetet, az esetleges fogalmakat, és hogy melyik funkcióra miért van szükség.

A fejlesztett szoftver tehát egy vállalati intranetre tervezett webes alkalmazás, PHP és MySQL használatával. Célja a hitelezésben dolgozó munkatársak ügyfélkapcsolati munkájának támogatása.

2. 3. 1. A USE CASE-EKRŐL ÁLTALÁBAN

A Use Case (használati eset) a rendszer funkcionalitásának és követelményeinek meghatározását jelenti az UML-ben. A Use Case a rendszerkövetelmények lefordítását jelenti a felhasználó által használt természetes nyelvről a fejlesztők számára egyértelmű UML-re. A használati eset egy eset vagy szituáció, amely során a rendszer egy vagy több felhasználói követelményt teljesít. A Use Case a modell szíve, a rendszertervezés többi elemére is hatással van, ahogy ezt az 5. ábra is mutatja. (Hamilton – Miles, 2006)

5. ábra

A rendszer 4+1 nézőpontja.



Forrás: Hamilton – Miles, 2006

A rendszer nézőpontjai koncepció nem szigorúan az UML része, de mindenképpen segít a megfelelő UML diagramok kiválasztásában. A Use Case tulajdonképpen a felhasználó nézőpontját is jelenti.

A rendszerkövetelmények leírása Use Case-ekkel két fő részből áll: a diagram megrajzolásából és a hozzá tartozó dokumentáció megírásából. A dokumentációhoz léteznek template-ek, amelyek megmutatják, milyen információkat kell leírunk. A leglényesebbek a következők: a használati eset neve, leírása, alkalmazásának feltételei, szereplői és az interakciók.

A használati eset diagram legfőbb célja, hogy bemutassa, a felhasználó (amit a diagramon a szereplő jelöl) hogyan kerül kapcsolatba a rendszerrel, és hogyan használja azt. Ezt úgy lehet kivitelezni, hogy a szereplőt egy vagy több használati esethez kapcsoljuk, vagy több diagramot rajzolunk.

A használati eset legfőbb célja (összevetve a diagrammal) egy rendszerbeli folyamat dokumentálása. Így a használati esetek a rendszer funkcionális követelményeinek dokumentációját képezik.

A használati esetek segítik a fejlesztés ütemezését. Egy megfelelő használati eset lista segít – a felhasználó, az üzleti elemző és a projektvezető közreműködésével – abban, hogy használati eseteket szükséges elindítani a fejlesztési ciklusban. Ezért segít a fejlesztési idő és költség megbecslésében is.

A használati eset diagramok legfőbb előnye a felhasználók, a szereplők modellezése. A szereplők fontos szerepet játszanak abban, hogy az üzleti elemzők megértsék a rendszerkövetelményeket. Továbbá, mivel a felhasználók láthatják magukat a használati eset diagramon, könnyebben és részletesebben elő tudják adni, mit szeretnének a rendszertől, és hogyan akarják használni.

A használati esetek kidolgozásánál arra azonban vigyázni kell, hogy a rendszer összes funkcióját nem szükséges, és nem is lehet dokumentálni velük (pl. sebesség, biztonság, méret, teljesítmény). (Unhelkar, 2005)

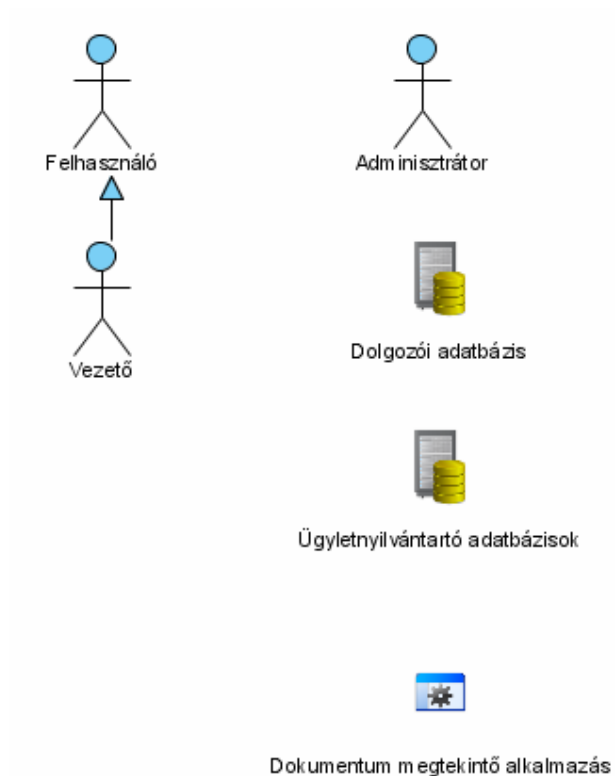
2. 3. 2. A RENDSZER HASZNÁLATI ESETEI

Ahogy a bevezetésben már szó volt róla, egy banki ügyfélkapcsolati támogató rendszert fejlesztettem. Mivel a vállalati hitelezés területén dolgoztam 2 évet, ezért tisztában vagyok azzal, mi az, ami az ügyfélkapcsolat-tartásban egy ilyen pozícióban segíteni tudja a munkát. Ennek megfelelően a rendszer funkcióit, a „felhasználók igényeit” teljes mértékben saját megfigyeléseim, tapasztalataim alapján alakítottam ki. A következőkben bemutatásra kerülnek ez elkészített használati eset diagramok és dokumentációk. A szükséges funkciókat csoportosítottam jellegük szerint, ezért egy diagramon több használati eset is szerepel.

2.3.2.1. SZEREPLŐK

1. UML diagram

Szereplők



A legfontosabb szereplő a Felhasználó. Ő a hitelezéssel foglalkozó munkatárs, akinek a munkáját hivatott segíteni a rendszer. Látható, hogy létezik egy speciális felhasználó, a Vezető. A vezetőnek plusz jogosultságai lesznek a rendszerben.

Az Adminisztrátor egy külön szereplő, más jogosultságokkal rendelkezik (pl. sok mindenhez nem férhet majd hozzá, amihez a Felhasználó igen), és ezért nem tekinthető a Felhasználó leszármazottjának.

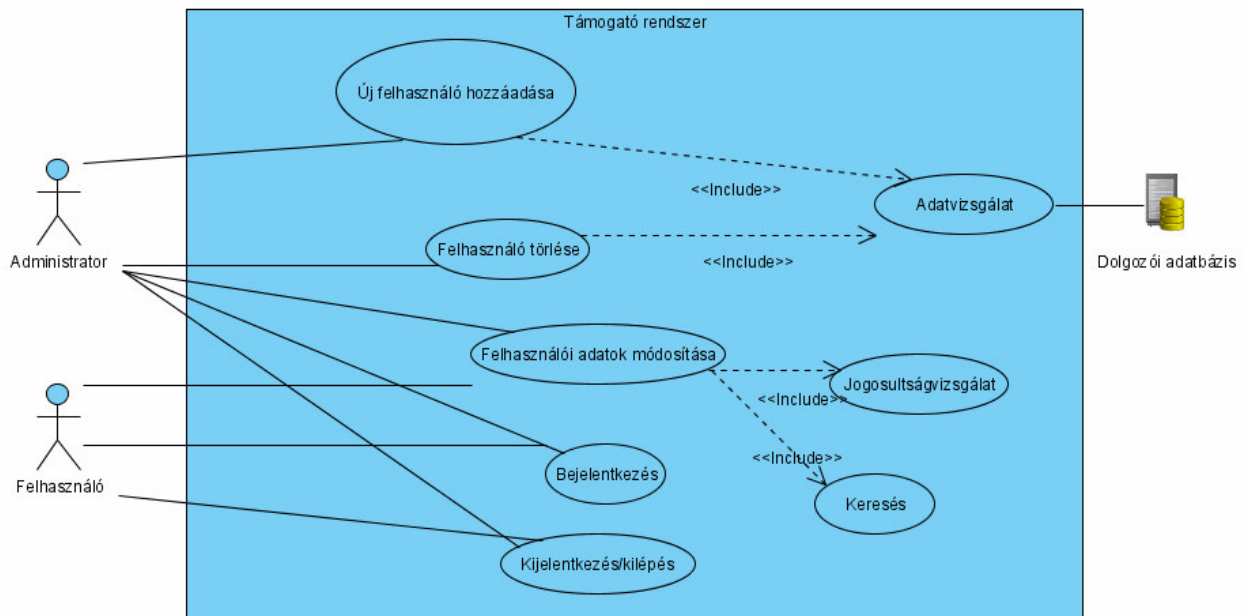
Szükség van külső rendszerek bevonására is. Itt két adatbázisról is van szó, amely a rendszer számára külső. Jelen rendszer elsősorban az ügyfélkapcsolat támogatására szolgál, és feltételezi, hogy a felhasználó szervezet (pénzügyintézet) már rendelkezik bizonyos rendszerekkel. Ezért a dolgozók, ügyletek, ügyfelek adatait a rendszer nem tudja megváltoztatni, csak olvasni egy-egy külső adatbázisból.

A rendszer a felvitt dokumentumok megtekintésére külső alkalmazást vesz igénybe (Word, Excel, Adobe Reader).

2. 3. 2. 2. FELHASZNÁLÓI ADMINISZTRÁCIÓ

2. UML diagram

Felhasználói adminisztráció



A 2. UML diagram a felhasználói adminisztrációval kapcsolatos használati eseteket mutatja be. A kék téglalap jelöli a rendszert, így jól látható, hogy a felhasználó, adminisztrátor és az adatbázis külső szereplők, amelyekre a rendszernek nincs hatása. Az ellipszisek jelentik a használati eseteket. Az ábrából látszik, hogy a Felhasználónak bejelentkezésre, kijelentkezésre és adatainak módosítására van lehetősége. Az adatok módosítása magában foglal (<<include>>) egy jogosultságvizsgálatot és kereső funkciót.

Az Adminisztrátornak ennek több lehetősége van, felvihet és törölhet Felhasználót, amelyhez a külső adatbázisból történő adatvizsgálat szükséges. A következőkben bemutatom a felhasználói adminisztrációval kapcsolatos használati eseteket.

2. táblázat

Bejelentkezés

Rövid leírás	A felhasználó bejelentkezése a rendszerbe.		
Előfeltétel	A felhasználó rendelkezzen azonosítóval és jelszóval.		
Sikeres teljesülés	Sikeres bejelentkezés		
Sikertelenség	A bejelentkezés nem sikerült.		
Elsődleges szereplők	Felhasználó, Administrator		
Másodlagos szereplők	-		
Beágyazott használati esetek	-		
Folyamat		Szereplő	Rendszer
	1	A szereplő bejelentkezést kezdeményez a rendszerbe, és megadja felhasználói azonosítóját és jelszavát.	
	2		A rendszer a megadott információkat elemzi, és ha rendben találta az azonosító és jelszó párost, belépteti a felhasználót.
Elágazás 2.1		Szereplő	Rendszer
	1		A rendszer szerint nem helyes a megadott azonosító és/vagy jelszó.
	2		A rendszer kijelzi a hibát a szereplő részére.

A 2. táblázat a Bejelentkezés használati eset dokumentációját tartalmazza. Az alkalmazott template a felhasznált irodalom alapján került kialakításra. Látható, hogyan zajlik a szereplő és a rendszer közötti interakció. Az egyszerű folyamatba belekerült egy elágazás is.

3. táblázat

Kijelentkezés

Rövid leírás	A felhasználó kilép, kijelentkezik a rendszerből, mert nem kívánja tovább használni azt.
Előfeltétel	A felhasználó be van jelentkezve a rendszerbe.
Sikeres teljesülés	A felhasználó kijelentkezett.
Sikertelenség	-
Elsődleges szereplők	Felhasználó, Administrator
Másodlagos szereplők	-

Beágyazott használati esetek		-	
Folyamat		Szereplő	Rendszer
	1	A szereplő kezdeményezi a rendszerből való kilépést.	
	2		A rendszer lezárja a felhasználó futó folyamatait, bontja a kapcsolatot a felhasználó számítógépével, és a visszatér a bejelentkező képernyőhöz.

4. táblázat

Felhasználói adatok módosítása

Rövid leírás	A rendszernek biztosítania kell lehetőséget arra, hogy a felhasználói adatokat módosítani lehessen. A módosítás során alapadatok (pl. dolgozói törzsszám) nem kerülhetnek módosításra. A módosítással az alábbi információk változtathatók meg: • a felhasználó elérhetősége (munkavégzés helye, telefon, email, stb.) • a felhasználó jelszava • a felhasználó mely szervezeti egységhez tartozik • a felhasználó beosztása és ennek megfelelően rendszerbeli jogosultságai.		
Előfeltétel	Létező felhasználói fiók a rendszerben.		
Sikeres teljesülés	A felhasználói adatok módosításra kerültek.		
Sikertelenség	A felhasználói adatok nem módosíthatók.		
Elsődleges szereplők	Adminisztrátor, Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A szereplő kéri a felhasználói adatok módosítását a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A szereplő kiválasztja a módosítani kívánt adatokat, és beírja az új értékeket.	
	4		A rendszer visszaigazolja a változtatást.

	5	Opcionális: ha az adminisztrátor hajtott végre módosítást, akkor értesíti a felhasználót az eredményről.	
Elágazás 2.1		Szereplő	Rendszer
	1	Ha a szereplő adminisztrátor, kiválasztja a módosítani kívánt felhasználót.	

A felhasználói adatok módosítása már egy bonyolultabb használati eset. Tartalmaz egy másik használati esetet, a jogosultságvizsgálatot. Erre azért van szükség, mert a Felhasználó csak a saját adatait módosíthatja, az Adminisztrátor pedig bármely Felhasználóét.

A következő két funkció csak az Adminisztrátor részére engedélyezett:

5. táblázat

Új felhasználó hozzáadása

Rövid leírás	A rendszernek lehetővé kell tennie, hogy egy adminisztrátor hozzá tudjon adni egy új felhasználót.		
Előfeltétel	A dolgozó az adott cégnél megfelelő területen dolgozik, és engedélyt kap vezetőitől a rendszer használatára. Kérelem a dolgozótól/vezetőtől a hozzáférés igénylésére.		
Sikeres teljesülés	A felhasználói fiók létrehozásra került a felhasználó részére.		
Sikertelenség	A felhasználói fiók hozzáadása visszautasítva		
Elsődleges szereplők	Adminisztrátor		
Másodlagos szereplők	-		
Beágyazott használati esetek	Adatvizsgálat		
Folyamat		Szereplő	Rendszer
	1	Az adminisztrátor megadja a dolgozó törzsszámát.	
	2	Az adminisztrátor kiválasztja a dolgozó részére beállítandó felhasználói fiók típusát.	
	3		include:: Adatvizsgálat
	4	Az adminisztrátor kéri a visszaadott adatokkal új felhasználó hozzáadását.	
	5		A rendszer menti az adatokat, hozzáadja a felhasználót, amelynek eredményét visszaigazolja.

	6	Az adminisztrátor értesíti a felhasználót az eredményről.	
Elágazás 3.1		Szereplő	Rendszer
	1		A rendszer azt az információt kapja, hogy a megadott dolgozói számon dolgozó nem található, és visszaadja az adminisztrátornak.
	2	Az adminisztrátor az eredeti folyamat 1. vagy 6. lépésétől folytatja.	
Elágazás 3.2		Szereplő	Rendszer
	1		A rendszer azt az információt kapja, hogy a megadott típusú fiók nem felel meg a dolgozó hatáskörének, és visszaadja az adminisztrátornak.
	2	Az adminisztrátor az eredeti folyamat 2. vagy 6. lépésétől folytatja.	

6. táblázat

Felhasználó törlése

Rövid leírás	A rendszernek lehetővé kell tennie, hogy az adminisztrátor törölni tudjon egy felhasználót a rendszerből. Azonban minden tevékenység, amit a felhasználó végzett, látható marad a rendszerben.		
Előfeltétel	Az illetékes vezető kéri az adminisztrátortól a felhasználó törlését a rendszerből. Ennek oka a következő lehet: a dolgozó már nem dolgozik a cégnél vagy olyan területen dolgozik, amelyhez nincs szükség a rendszer használatára. A törölni kívánt felhasználóhoz nem lehet hozzárendelve ügyfél.		
Sikeres teljesülés	A felhasználói fiók nem használható tovább a rendszerben.		
Sikertelenség	A felhasználói fiók törlése visszautasítva.		
Elsődleges szereplők	Adminisztrátor		
Másodlagos szereplők	-		
Beágyazott használati esetek	Adatvizsgálat		
Folyamat		Szereplő	Rendszer
	1	Az adminisztrátor kiválasztja a törlendő felhasználó dolgozói azonosítóját.	
	2		include:: Adatvizsgálat
	3		A rendszer megvizsgálja, hogy az adott felhasználóhoz van-e rendelve ügyfél.
	4		A rendszer a 2. és 3. lépés alapján

			visszaadja, hogy törölhető a felhasználó.
	5	Az adminisztrátor kéri a rendszertől a beazonosított felhasználói fiók törlését, ha a rendszer szerint törölhető a felhasználó.	
	6		A rendszer törli a felhasználói fiókot., amely a továbbiakban már nem használható.
	7	Az adminisztrátor értesíti az illetékes vezetőt az eredményről.	
Elágazás 4.1		Szereplő	Rendszer
	1		A rendszer visszaadja, hogy nem törölhető a felhasználó.
	2	Ugrás a folyamat 7. pontjára.	

Végül következzenek a beágyazott használati esetek, amelyek önmagukban külön funkciót alkotnak, és többször felhasználhatóak más használati esetekben is.

7. táblázat

Adatvizsgálat

Rövid leírás	A támogató rendszer kommunikál a dolgozói adatbázissal, dolgozói törzsszám és/vagy beosztás (és egyéb adatok) alapján. Lekérdezés, amely arra irányul, hogy az adott dolgozó szerepel-e a dolgozói adatbázisban.		
Előfeltétel	Dolgozói törzsszám és/vagy beosztás (esetleg egyéb adat) megadása a támogató rendszer részéről.		
Sikeres teljesülés	Az adott dolgozó megtalálható dolgozói adatbázisban.		
Sikertelenség	Az adott dolgozó nem található dolgozói adatbázisban vagy más adatokkal található.		
Elsődleges szereplők	Dolgozói adatbázis		
Másodlagos szereplők	-		
Beágyazott használati esetek	-		
Folyamat		Szereplő	Rendszer
	1		A rendszer kéri a dolgozói törzsszám és/vagy hatáskör alapján a dolgozói adatok visszaigazolását a dolgozói adatbázistól.
	2	A dolgozói adatbázis visszaadja a megadott törzsszámú dolgozó adatait. És visszaigazolja, hogy az adott típusú fiók megfelel a dolgozó hatáskörének.	

	3		A rendszer visszaadja az adatvizsgálat eredményét.
Elágazás 2.1		Szereplő	Rendszer
	1	A dolgozói adatbázisban a megadott törzsszámú dolgozó nem szerepel.	
	2		Ugrás a 3. pontra.
Elágazás 2.2		Szereplő	Rendszer
	1	A dolgozói adatbázisban a megadott törzsszámú dolgozó más hatáskörrel szerepel, mint amit a rendszer továbbított.	
	2		Ugrás a 3. pontra.

8. táblázat

Jogosultságvizsgálat

Rövid leírás	A rendszernek meg kell vizsgálnia, hogy egy szereplő, mikor elindít egy folyamatot, jogosult-e rá. A vizsgálat bizonyos tekintetben szűrőként is működik, azaz a vizsgálat elvégzése után már csak azok a funkciók kerülnek megjelenítésre, amelyekre az adott szereplő jogosult.		
Előfeltétel	A szereplő elindít egy folyamatot, a folyamat kéri a jogosultságvizsgálatot.		
Sikeres teljesülés	A szereplő jogosult az adott folyamat végrehajtására.		
Sikertelenség	A szereplő nem jogosult az adott folyamat végrehajtására.		
Elsődleges szereplők	-		
Másodlagos szereplők	-		
Beágyazott használati esetek	-		
Folyamat		Szereplő	Rendszer
	1		A rendszer az elindított folyamat egy lépéseként kéri a jogosultságvizsgálatot az adott felhasználóval kapcsolatban.
	2		A rendszer visszaadja az adott folyamatnak, hogy a felhasználó mire jogosult vagy jogosult-e a kijelölt feladatot elvégezni.

2.3.2.3. ÜGYFÉL ADMINISZTRÁCIÓ

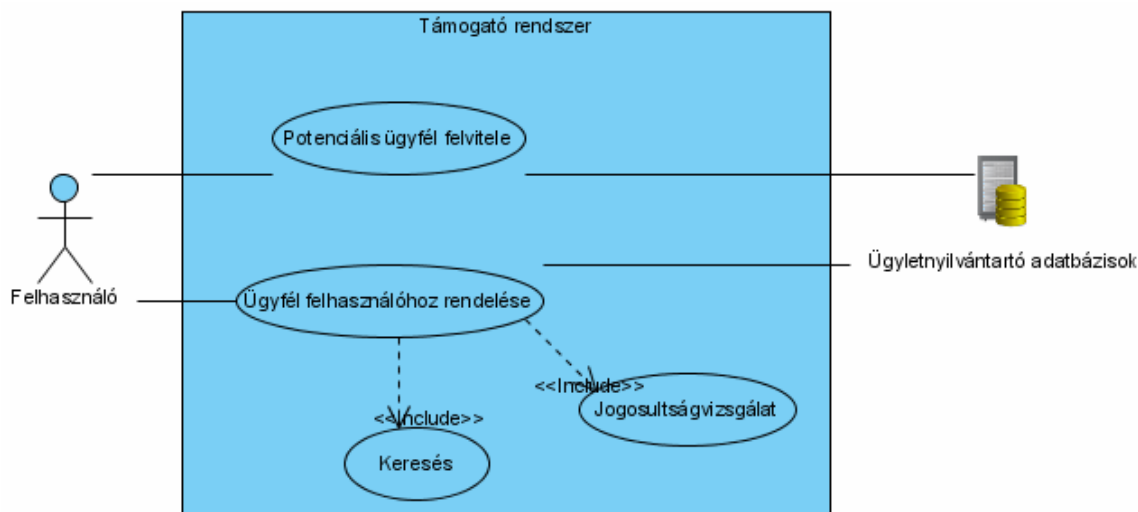
A rendszerben szükség van minimális ügyfél adminisztrációs funkciókra is. Az ügyfeleket és ügyleteiket egy külső rendszer/adatbázis tárolja, a fejlesztendő rendszer csak olvassa azokat az adatokat, módosítani nem tudja őket.

Minden hitelezési munkatárshoz, azaz Felhasználóhoz tartoznak ügyfelek, akikkel foglalkozik, akiknek az ügyeit intézi. Ezért a rendszerben is minden Felhasználóhoz ügyfelet kell rendelni, amelyet a Felhasználó végez el, amennyiben jogosultsága van hozzá. A szervezet hatáskör beosztásától függ, hogy ezt mely Felhasználóknak engedélyezi, valószínűleg csak vezetőknek.

A potenciális ügyfél felvitelére azért van szükség, mert a Felhasználó számos olyan ügyféllel állhat kapcsolatban, akivel még nem kötött üzletet, ezért az Ügyletnyilvántartó adatbázisokban nem szerepel. Ezek az új, leendő ügyfelek, akikkel való kapcsolattartást szintén kezelnie kell a rendszernek.

3. UML diagram

Ügyfél adminisztráció



9. táblázat

Ügyfél felhasználóhoz rendelése

Rövid leírás	A bank valamennyi ügyfelét hozzá kell rendelni egy dolgozóhoz, aki az ügyeit intézi. Ezeket a kapcsolatokat a rendszer tárolja.
Előfeltétel	Az ügyfél és a felhasználó megtalálható a rendszerben.

Sikeres teljesülés	Az ügyfél hozzárendelésre került a felhasználóhoz.		
Sikertelenség	A hozzárendelés nem sikerült.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	Ügyletnyilvántartó adatbázisok		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kéri egy adott ügyfél adott felhasználóhoz rendelését, amelyhez kiválasztja az ügyfelet és a felhasználót.	
	2		include:: Jogosultságvizsgálat
	3		A rendszer visszaigazolja, hogy a hozzárendelés sikeres volt.
Elágazás 2.1		Szereplő	Rendszer
	1		A rendszer szerint a felhasználó nem jogosult a hozzárendelés elvégzésére.
	2		A rendszer értesíti a felhasználót a sikertelenségről.

10. táblázat

Potenciális ügyfél felvitele

Rövid leírás	A potenciális ügyfél még nem ügyfele a banknak, ezért egyéb rendszerekben nem szerepel. Ez a funkció lehetővé teszi azt, hogy egy megszerzendő ügyféllel kapcsolatos ügyintézés is rögzítésre kerüljön a rendszerben.		
Előfeltétel	Az ügyfél még nem ügyfele a banknak.		
Sikeres teljesülés	A potenciális ügyfél rögzítésre került a rendszerben.		
Sikertelenség	A potenciális ügyfél nem rögzíthető.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	Ügyletnyilvántartó adatbázisok		
Beágyazott használati esetek	Ügyfél felhasználóhoz rendelése		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kéri a rendszertől új potenciális ügyfél felvételét, megadja az ügyfél (cég) nevét és adószámát.	
	2		A rendszer kommunikál az ügyletnyilvántartó adatbázisokkal, hogy a megadott adószámon található-e ügyfél.

	3		A rendszer eltárolja az új potenciális ügyfelet.
	4		include:: Ügyfél felhasználóhoz rendelése
Elágazás 2.1		Szereplő	Rendszer
	1		A rendszer azt kapja vissza, hogy az adott adószámon már létezik ügyfél.
	2		A rendszer jelzi a felhasználónak a hibát.

2. 3. 2. 4. ÜGYFÉL KOMMUNIKÁCIÓ

Több olyan funkció is van, amellyel a rendszernek rendelkeznie kell az ügyfél kommunikációval kapcsolatban.

Az első a tárgyalásjelentéssel kapcsolatos funkciók köre. Az ügyfélkapcsolat ápolásának egyik legfontosabb eleme a bankszektorban az ügyféltárgyalás. A rendszerben rögzíthetők az ügyféltárgyalás adatai: időpont, helyszín, résztvevők, témák, stb. Ezek rögzítése azért jó, mert egyszerűen nyomon követhető, ki mikor, kivel, milyen ügyben tárgyalt. Ez hasznos a hitelezési munkatársnak, mert rendszerezetten vissza tudja keresni, szűrni tárgyalásait. Másrészt hasznos a vezetőnek is, aki képbe kerülhet a folyó ügyféltárgyalásokat illetően, ill. ellenőrizheti pl., hogy ki melyik ügyfelével milyen gyakran tárgyal, milyen ügyletek várhatóak, stb. A tárgyalásjelentések kezeléséhez szükséges használati esetek: tárgyalásjelentés írása, törlése, módosítása, lekérdezése. Mindegyiknél szükség van két beágyazott használati esetre, a keresésre és a jogosultságvizsgálatra. A keresés elengedhetetlen, hiszen pl. már a tárgyalásjelentés íráshoz is ki kell választanunk az ügyfelet, akivel tárgyaltunk. A jogosultságvizsgálat pedig azért lényeges, mert minden Felhasználó csak a saját tárgyalásjelentéseit láthatja, módosíthatja, törölheti. Kivéve a vezetőt, aki az alá beosztott felhasználók tárgyalásjelentéseit láthatja, de nem törölheti, módosíthatja azokat.

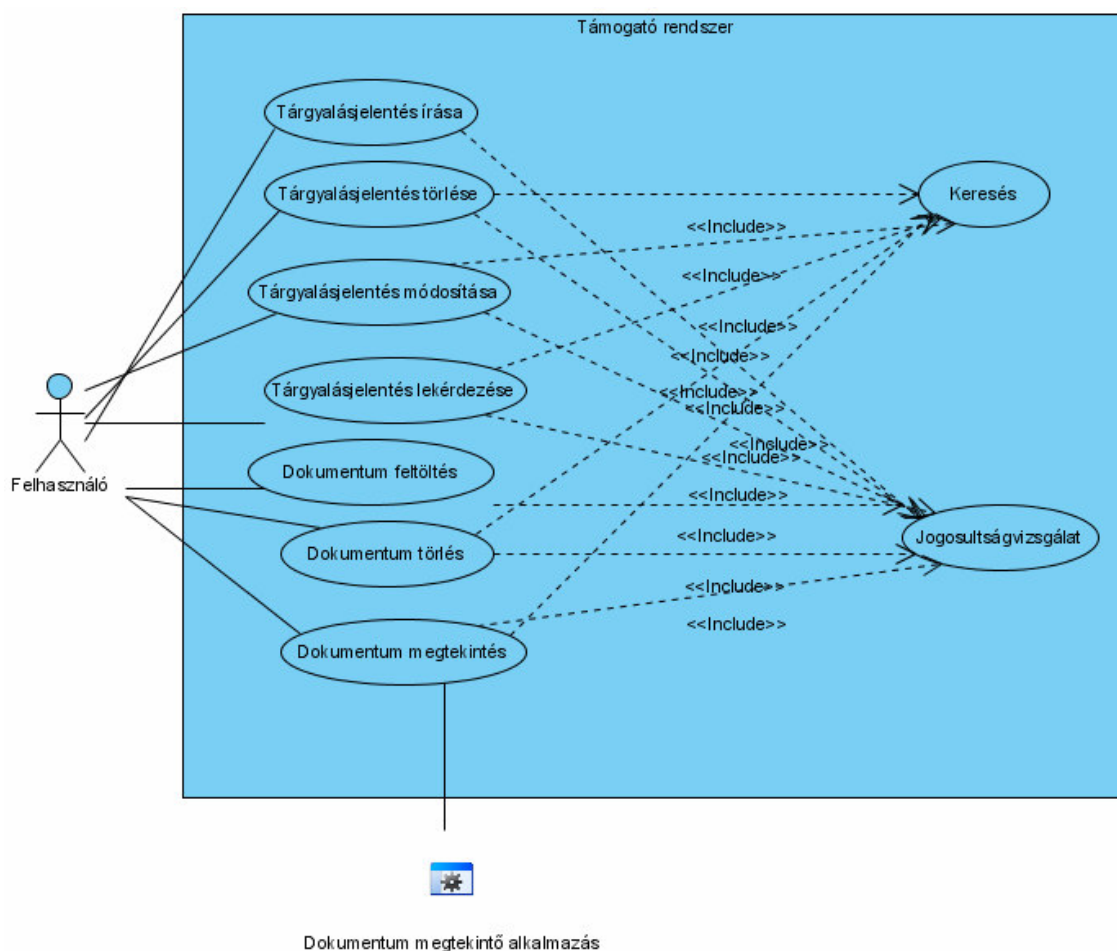
Az ügyfél kommunikációval kapcsolatos funkciók másik körét az ügyfelekkel kapcsolatos dokumentumok képezik. Ilyen dokumentumok lehetnek: az ügyféllel történt levelezés (pl. nyomtatott levél Word-beli változata), ügyféltől kapott elektronikus anyagok, belső banki dokumentumok (pl. előterjesztés, hitel

engedélyező határozat), ügyféllel aláírt szerződés, ill. bármilyen rendelkezésre álló, ügyféllel kapcsolatos elektronikus dokumentum. A jogosultságvizsgálat a tárgyalásjelentésekkel azonos módon működik.

Az ügyfél kommunikációval kapcsolatos használati eseteket a 4. UML diagram és a 11-17. táblázatok tartalmazzák.

4. UML diagram

Ügyfélkommunikáció



11. táblázat

Tárgyalásjelentés írása

Rövid leírás	A felhasználó számára lehetővé teszi, hogy egy ügyféllel folytatott tárgyalásokat összefoglalva felvigye a rendszerbe.
Előfeltétel	Az ügyféllel folytatott tárgyalás. Az ügyfél benne legyen a rendszerben.
Sikeres teljesülés	A tárgyalásjelentés hozzáadásra került.
Sikertelenség	A tárgyalásjelentés nem adható hozzá.

Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó tárgyalásjelentés felvitelét kezdeményezi a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja, melyik ügyfélhez rögzíti a tárgyalásjelentést.	
	4	A felhasználó megadja a tárgyalásjelentés rögzítéséhez szükséges adatokat.	
	5		A rendszer tárolja, és visszaigazolja azt.

12. táblázat

Tárgyalásjelentés törlése

Rövid leírás	A tárgyalásjelentés törlését teszi lehetővé. Ez akkor kerülhet használatba, ha egy tárgyalásjelentés rosszul lett rögzítve. Törlésre csak a tárgyalásjelentés írója jogosult.		
Előfeltétel	Rögzített tárgyalásjelentés.		
Sikeres teljesülés	A tárgyalásjelentés törlésre került.		
Sikertelenség	A tárgyalásjelentés nem törölhető.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó tárgyalásjelentés törlését kezdeményezi.	
	2		include:: Jogosultságvizsgálat
	3		A Jogosultságvizsgálat alapján a rendszer visszaadja a törölhető tárgyalásjelentések listáját
	4	A felhasználó kiválasztja a törlendő tárgyalásjelentést.	
	5		A rendszer végrehajtja a törlést, és visszaigazolja az eredményt.

13. táblázat

Tárgyalásjelentés módosítása

Rövid leírás	Rögzített tárgyalásjelentés módosítása a rendszerben. A tárgyalásjelentés rögzítője hajthatja végre.		
Előfeltétel	A felhasználó által rögzített tárgyalásjelentés a rendszerben.		
Sikeres teljesülés	A tárgyalásjelentés módosításra került.		
Sikertelenség	A tárgyalásjelentés nem módosítható.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a tárgyalásjelentés módosítást a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3		A Jogosultságvizsgálat alapján a rendszer visszaadja a módosítható tárgyalásjelentések listáját.
	4	A felhasználó kiválasztja a módosítandó tárgyalásjelentést.	
	5	A felhasználó módosítja a szükséges adatokat, és kezdeményezi a módosítás mentését.	
	6		A rendszer menti az adatokat, és visszaigazolja a módosítást.

14. táblázat

Tárgyalásjelentés lekérdezése

Rövid leírás	A felhasználó számára lehetővé teszi a rendszerben tárolt tárgyalásjelentések megtekintését.
Előfeltétel	
Sikeres teljesülés	A tárgyalásjelentés(ek) megjelenítése sikeres.
Sikertelenség	Nincs megjeleníthető tárgyalásjelentés.
Elsődleges szereplők	Felhasználó
Másodlagos szereplők	-
Beágyazott használati esetek	Jogosultságvizsgálat

Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a tárgyalásjelentés lekérdezést a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3		A rendszer eltárolja a memóriában, mely tárgyalásjelentések kérdezhetők le a felhasználó által.
	4	A felhasználó kiválasztja a tárgyalásjelentést (keresés, szűrés ügyfél, felhasználó, időpont, stb. alapján)	
	5		A rendszer megjeleníti a kiválasztott tárgyalásjelentést.

15. táblázat

Dokumentum feltöltése

Rövid leírás	Ügyfélkommunikációval kapcsolatos dokumentumok feltöltését teszi lehetővé a rendszerbe a felhasználó részére.		
Előfeltétel	Az ügyfél rögzített a rendszerben		
Sikeres teljesülés	A fájl feltöltése sikeres.		
Sikertelenség	A fájl feltöltése nem sikerült.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi dokumentum feltöltését a rendszerbe.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja, mely ügyfélhez kapcsolódik a dokumentum.	
	4	A felhasználó kiválasztja a feltöltendő dokumentumot, megadja a leírását, és elindítja a feltöltést.	
	5		A rendszer feltölti a fájlt, és visszaigazolja az eredményt.

16. táblázat

Dokumentum törlése

Rövid leírás	Lehetővé teszi a rendszerbe feltöltött dokumentumok törlését.		
Előfeltétel	A rendszerbe feltöltött dokumentum.		
Sikeres teljesülés	A dokumentum törlésre került.		
Sikertelenség	A dokumentum nem törölhető.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a dokumentum törlést a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja a törlendő dokumentumot.	
	4		A rendszer törli a dokumentumot.

17. táblázat

Dokumentum megtekintése

Rövid leírás	A rendszerbe feltöltött dokumentumok megtekintése		
Előfeltétel	A rendszerbe feltöltött dokumentum		
Sikeres teljesülés	A feltöltött dokumentum megjelenítésre került.		
Sikertelenség	A dokumentum nem megjeleníthető vagy nincs megjeleníthető dokumentum.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	Dokumentum megtekintő alkalmazás (pl. Word, Excel, Adobe Reader, stb.)		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a rendszerben a dokumentum megtekintést.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja (keresés, szűrés) a megtekintendő fájlt.	

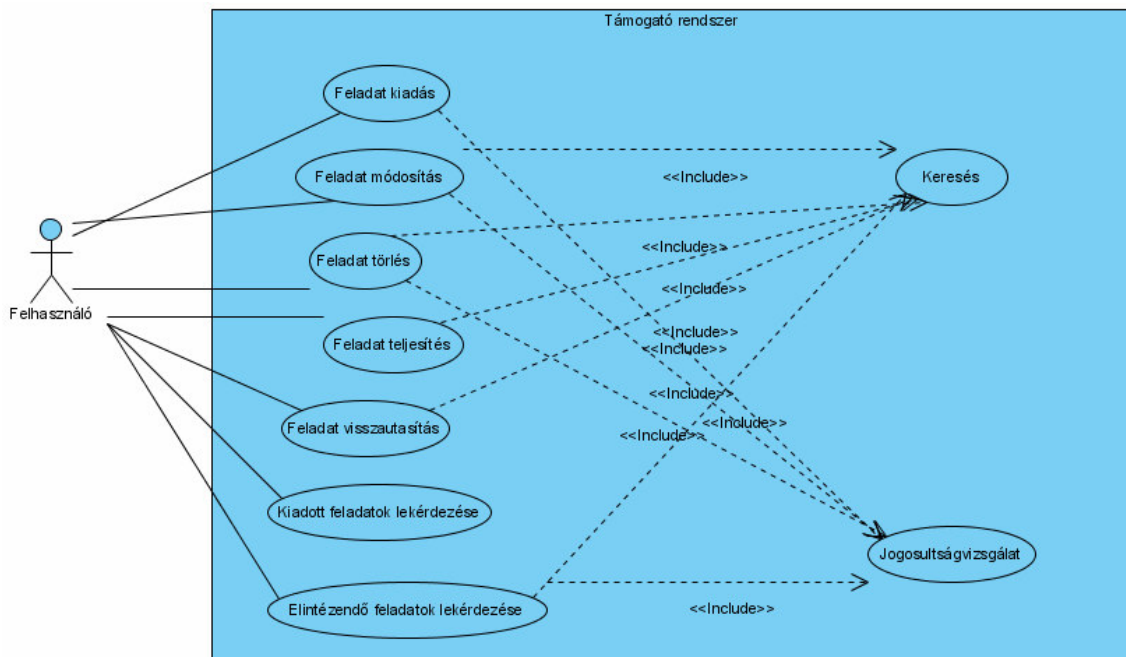
	4		A rendszer a megfelelő alkalmazás segítségével megnyitja a dokumentumot.
--	---	--	--

2.3.2.5. FELADATOK

Rendszerkövetelmény, hogy a rendszerben lehessen kiadni feladatokat. Ezek a feladatok ügyfelekkel kapcsolatos teendők. Elsősorban a monitoring tevékenységet hivatott segíteni, de bármilyen feladról szó lehet, pl. az ügyfél által határidőre becsatolandó dokumentumok benyújtásának ellenőrzése, törlesztés, hiteldíj-fizetés ellenőrzés, hitel, betét lejáratának előrejelzése, ügyfél megkeresési feladat kampányokkal kapcsolatban, stb. A funkció elsősorban a vezetők feladatkiadására, azok végrehajtásának ellenőrzésére szolgál. Egy Felhasználó saját magának is kiadhat feladatot, és a rendszer automatikusan generálhat feladatokat emlékeztetőül pl. a hitel lejáratára.

5. UML diagram

Feladatok



A feladatokkal kapcsolatos funkciók igénylik a jogosultságvizsgálatot. Feladat kiadására más Felhasználó részére csak vezető jogosult. Kiadott feladat módosítására, törlésére, lekérdezésére csak a kiadó jogosult. Elintézendő feladat

teljesítésére, visszautasítására, lekérdezésére csak az a Felhasználó jogosult, akinek a részére kiadták a feladatot.

Keresésre, szűrésre itt is szükség van, hiszen egy feladat adott Felhasználóhoz és ügyfélhez kapcsolódik.

18. táblázat

Feladat kiadás

Rövid leírás	A rendszernek lehetővé kell tennie, hogy egy felhasználó kiadjon egy ügyféllel kapcsolatos feladatot egy felhasználónak. A jogosultságok döntenek el, hogy ki kinek adhat feladatot.		
Előfeltétel			
Sikeres teljesülés	A feladat kiadásra került.		
Sikertelenség			
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi feladat kiadását a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja melyik ügyfélhez kapcsolódik a feladat, és kiválasztja, melyik felhasználónak adja ki a feladatot. És megadja a feladat további részleteit (leírás, határidő, stb.)	
	4		A rendszer eltárolja a feladatot, a feladat ezáltal megjelenik a felhasználó elvégzendő feladatai között. A rendszer visszaigazolja a feladat kiadását.

19. táblázat

Feladat módosítás

Rövid leírás	Kiadott feladat módosítása. A módosítást csak a feladat kiadója végezheti el.
Előfeltétel	Egy kiadott, el nem végzett feladat.
Sikeres teljesülés	A feladat módosításra került.
Sikertelenség	

Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a rendszerben a feladat módosítását.	
	2		include:: Jogosultságvizsgálat
	3		A Jogosultságvizsgálat alapján a rendszer visszaadja a módosítható feladatok listáját.
	4	A felhasználó kiválasztja a módosítandó feladatot.	
	5	A felhasználó megadja a módosításokat.	
	6		A rendszer elmenti a módosításokat, és értesíti a feladat felelősét a módosításról.
	7		A rendszer visszaigazolja a módosítást a feladat kiadójának.

20. táblázat

Feladat törlés

Rövid leírás	Kiadott feladat törlése, visszavonása. Csak a feladat kiadója jogosult elvégezni.		
Előfeltétel	Létező, kiadott feladat		
Sikeres teljesülés	A feladat törlésre került.		
Sikertelenség			
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó egy kiadott feladat törlését kezdeményezi a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3		A Jogosultságvizsgálat alapján a rendszer visszaadja a törölhető feladatok listáját.

	4	A felhasználó kiválasztja a törlendő feladatot.	
	5		A feladat törlésre kerül, amelyet visszaigazol a rendszer.

21. táblázat

Feladat teljesítés

Rövid leírás	A felhasználó rögzíti a rendszerben, hogy az adott feladatot teljesítette.		
Előfeltétel	A felhasználó részére kiadott feladat.		
Sikeres teljesülés	A teljesítés rögzítésre került.		
Sikertelenség			
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek			
Folyamat		Szereplő	Rendszer
	1	A felhasználó kiválasztja a teljesítendő feladatot, és kezdeményezi a rendszerben a feladat teljesítését.	
	2		A rendszer rögzíti a teljesítést, és visszaigazolja a felhasználónak.

22. táblázat

Feladat visszautasítás

Rövid leírás	A felhasználó rögzíti a rendszerben, hogy az adott feladatot nem fogadja el.		
Előfeltétel	A felhasználó részére kiadott feladat.		
Sikeres teljesülés	A visszautasítás rögzítésre került.		
Sikertelenség			
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek			
Folyamat		Szereplő	Rendszer
	1	A felhasználó kiválasztja a visszautasítandó feladatot, és kezdeményezi a rendszerben a	

		feladat visszautasítását.	
	2		A rendszer rögzíti a visszautasítást, és visszaigazolja a felhasználónak.

23. táblázat

Kiadott feladatok lekérdezése

Rövid leírás	A felhasználó számára lehetővé teszi, hogy lekérdezze az általa kiadott feladatokat.		
Előfeltétel			
Sikeres teljesülés	A lekérdezett feladatok megjelennek a képernyőn.		
Sikertelenség	Nincs megjeleníthető feladat.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek			
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi az általa kiadott feladatok lekérdezését.	
	2		A rendszer megkeresi és kilistázza a kiadott feladatokat.

24. táblázat

Elintézendő feladatok lekérdezése

Rövid leírás	A felhasználó részére lekérdezhetők a részére kiadott feladatok. A magasabb beosztású felhasználók lekérdezhetik a beosztottak feladatait.		
Előfeltétel			
Sikeres teljesülés	A feladatok lekérdezése megtörtént.		
Sikertelenség	Nincs a felhasználó részére kiadott feladat.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi a feladat lekérdezést a rendszerben (a feladat lekérdezés menüpontra kattint)	

	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja, mely felhasználó feladatait akarja lekérdezni.	
	4		A rendszer kilistázza a feladatokat.

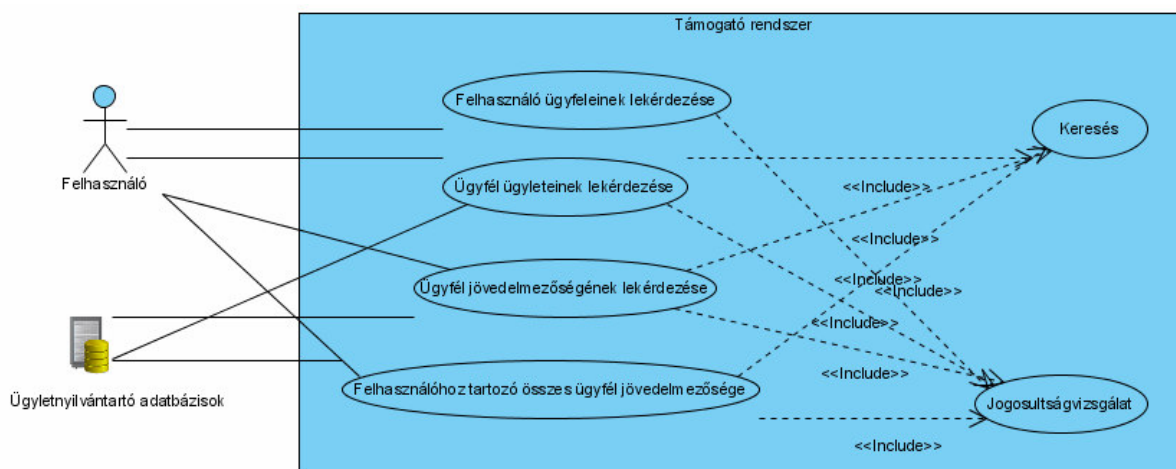
2. 3. 2. 6. LEKÉRDEZÉSEK

Szükség van néhány lekérdező funkcióra, amelyek eddig még nem szerepeltek, de hasznosak lehetnek. Ilyen az egy adott felhasználó ügyfeleinek lekérdezése, adott ügyfél ügyleteinek lekérdezése, és jövedelmezőség lekérdezések.

Az ügyletek és a jövedelmezőség lekérdezéséhez igénybe kell venni a külső, ügyletnyilvántartó adatbázist. A Felhasználó ügyfeleinek lekérdezéséhez ez nem szükséges, mert a Felhasználó – ügyfél hozzárendelést csak a támogató rendszer tárolja.

6. UML diagram

Lekérdezések



25. táblázat

Felhasználó ügyfeleinek lekérdezése

Rövid leírás	A rendszerben lekérdezhetőnek kell lennie az adott felhasználóhoz tartozó ügyfeleknek. A felhasználói jogosultságok szabályozzák, ki kinek az ügyfeleit kérheti le.
---------------------	---

Előfeltétel	A felhasználó rendelkezik felhasználói fiókkal a rendszerben.		
Sikeres teljesülés	A felhasználóhoz tartozó ügyfelek megjelenítésre kerültek.		
Sikertelenség	Nincs megjeleníthető ügyfél.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	-		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi az ügyfelek lekérdezését.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja, mely felhasználó ügyfeleit kívánja megtekinteni.	
	4		A rendszer megjeleníti a kiválasztott felhasználóhoz tartozó ügyfeleket.

26. táblázat

Ügyfél ügyleteinek lekérdezése

Rövid leírás	Lehetővé teszi egy ügyfél ügyleteinek/használt termékeinek megtekintését		
Előfeltétel			
Sikeres teljesülés	Az ügylet lekérdezés sikeres		
Sikertelenség	Nincs megjeleníthető ügylet		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	Ügylet nyilvántartó adatbázisok		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó kezdeményezi az ügylet lekérdezést.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja az ügyletet, amelynek ügyletére kíváncsi.	
	4		A rendszer az Ügylet nyilvántartó adatbázisoktól lekéri az ügyfél adatait, és megjeleníti az adott ügyfél ügyleteit.
Elágazás 4.1		Szereplő	Rendszer

	1	A felhasználó kiválasztja a megjelenített ügyletek közül, amelyikre részletesebben kíváncsi.	
	2		A rendszer megjeleníti a kiválasztott ügylet részleteit.

27. táblázat

Ügyfél jövedelmezőségének lekérdezése

Rövid leírás	A funkció lehetővé teszi egy adott ügyfél jövedelmezőségének kiszámítását és megjelenítését.		
Előfeltétel	Az ügyfél megtalálható a rendszerben		
Sikeres teljesülés	A jövedelmezőség lekérdezés sikeres volt.		
Sikertelenség	A lekérdezés nem teljesíthető.		
Elsődleges szereplők	Felhasználó		
Másodlagos szereplők	Ügyletnyilvántartó adatbázisok		
Beágyazott használati esetek	Jogosultságvizsgálat		
Folyamat		Szereplő	Rendszer
	1	A felhasználó a jövedelmezőség lekérdezést kezdeményezi a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja az ügyfelet, melynek a jövedelmezőségét szeretné lekérdezni.	
	4		A rendszer az Ügyletnyilvántartó adatbázisoktól megkéri a szükséges adatokat, és kiszámolja a jövedelmezőséget, majd megjeleníti a felhasználó részére

28. táblázat

Felhasználóhoz tartozó összes ügyfél jövedelem lekérdezése

Rövid leírás	A funkció lehetővé teszi egy adott felhasználó ügyfelei jövedelmezőségének kiszámítását és megjelenítését.
Előfeltétel	A felhasználó megtalálható a rendszerben
Sikeres teljesülés	A jövedelmezőség lekérdezés sikeres volt.
Sikertelenség	A lekérdezés nem teljesíthető.
Elsődleges szereplők	Felhasználó

Másodlagos szereplők		Ügyletnyilvántartó adatbázisok	
Beágyazott használati esetek		Jogosultságvizsgálat	
Folyamat		Szereplő	Rendszer
	1	A felhasználó a jövedelmezőség lekérdezést kezdeményezi a rendszerben.	
	2		include:: Jogosultságvizsgálat
	3	A felhasználó kiválasztja a felhasználót, akihez tartozó ügyfelek jövedelmezőségét szeretné lekérdezni.	
	4		A rendszer az Ügyletnyilvántartó adatbázisoktól megkéri a szükséges adatokat, és kiszámolja a jövedelmezőséget, majd megjeleníti a felhasználó részére

2. 4. TERVEZÉS ÉS SPECIFIKÁCIÓ

A rendszerkövetelmények összeállítására és elemzése után áttérhetünk a következő fázisokra, azaz hogy hogyan teljesítjük a követelményeket. Ennek érdekében a használati esetek dokumentációja alapján elkészítettem az aktivitás diagramokat, megterveztem az osztályokat és végül a szükséges adatbázist.

A rendszer sok funkcióval rendelkezik, de nem túl komplex. Ezért ilyen mélységű tervezés és kódolás előtti specifikáció elegendőnek bizonyult.

2. 4. 1. RENDSZERFOLYAMATOK MODELLEZÉSE

2. 4. 1. 1. AZ AKTIVITÁS DIAGRAMOKRÓL

Miután kidolgoztuk a rendszer használati eseteit és dokumentáltuk azokat, érdemes elkészíteni az aktivitás diagramokat is, mert azok elég szorosan kapcsolódnak a használati esetekhez. Úgy is lehetne mondani, hogy az aktivitás diagramok a használati esetek működését mutatják be, képszerűen.

A használati esetek, ahogy láttuk, azt írják le, hogy a rendszer mit fog csinálni. Az aktivitás diagramok ezzel szemben azt mutatják be, hogy a rendszer hogyan végzi el a feladatait. Az aktivitás diagramok képesek egy folyamat dokumentálására egy használati eseten vagy teljes rendszeren belül. Fontos tulajdonságuk, hogy meg

tudják mutatni a függőséget a tevékenységek, funkciók között. Az aktivitás diagramok képesek több szálon futó folyamatok ábrázolására, így láthatóvá válik, hogy egyszerre mi történik a rendszerben. Ezért kiváló módszert biztosítanak a folyamatok modellezésére, amely lehet üzleti folyamat, egy használati eseten belüli folyamat vagy használati esetek közötti folyamat. (Unhelkar, 2005)

Saját tapasztalatom szerint az aktivitás diagramok nagy segítséget nyújtanak később a kódoláskor, az algoritmusok kialakításakor is.

Az aktivitás diagram az egyetlen UML diagram, amely a korábban említett rendszer nézőpontjai koncepcióban a folyamat nézőpontot reprezentálja.

6. ábra

A folyamat nézőpont.



Forrás: Hamilton – Miles, 2006

2. 4. 1. 2. EGY AKTIVITÁS DIAGRAM PÉLDA

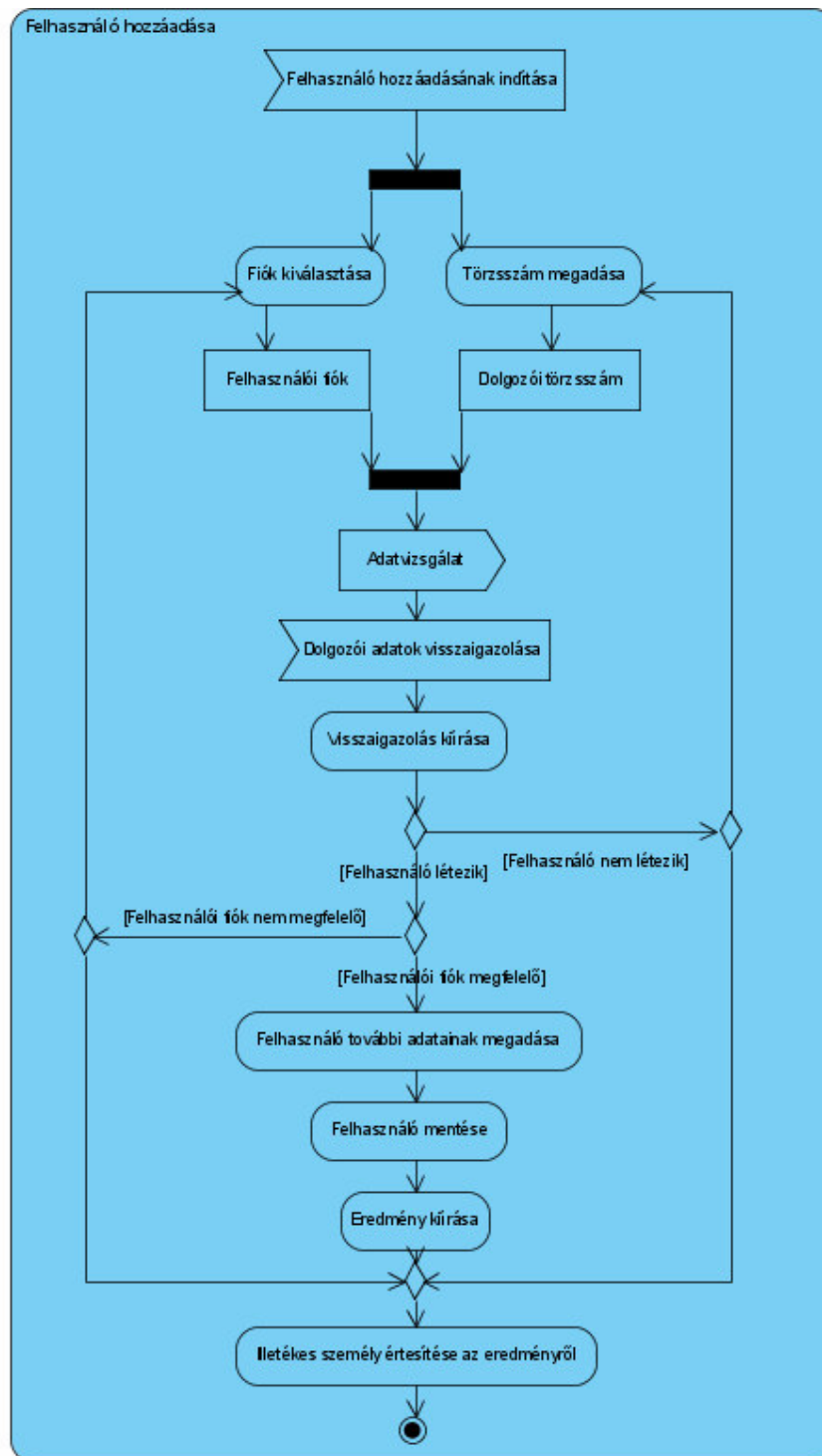
Egy bonyolultabb aktivitás diagramon keresztül bemutatom az aktivitás diagram lényegét. A rendszer többi aktivitás diagramja azok mérete miatt a dolgozat mellékleteiben található meg.

A folyamatban a lekerekített sarkú téglalapok jelentik a tevékenységeket, a nyilak pedig a folyamat irányát. A folyamat fentről lefelé halad.

A diagram tetején a konkáv ötszöggel jelölt csomópontból indul a folyamat. Ez a jelölés azt jelenti, hogy a folyamat jelre vár, hogy megindulhasson. Jelen esetben a felhasználó hozzáadás funkció elindítása a jel, amely nem lesz más, mint az alkalmazásban a megfelelő menüpontra kattintás. A használati esetekből pedig tudjuk, hogy ezt csak az Adminisztrátor végezheti el.

7. UML diagram

Aktivitás diagram: felhasználó hozzáadása



A folyamatban a lekerekített sarkú téglalapok jelentik a tevékenységeket, a nyilak pedig a folyamat irányát. A folyamat fentről lefelé halad.

A diagram tetején a konkáv ötszöggel jelölt csomópontból indul a folyamat. Ez a jelölés azt jelenti, hogy a folyamat jelre vár, hogy megindulhasson. Jelen esetben a felhasználó hozzáadás funkció elindítása a jel, amely nem lesz más, mint az alkalmazásban a megfelelő menüpontra kattintás. A használati esetekből pedig tudjuk, hogy ezt csak az Adminisztrátor végezheti el.

Ezután egy feltétel nélküli elágazás következik (ezt jelzi a fekete téglalap), mely után a folyamat két szálon futhat. A fiók kiválasztása és a törzsszám megadása után következő téglalapok kimenő adatokat jelentenek, amelyet a következő tevékenység kap meg. Aztán a két szál összekapcsolódik, és az adatvizsgálat következik, amelyet külső adatbázis segítségével végez el a rendszer. A nyílszerű téglalap jelzi, hogy a tevékenység külső szereplő bevonásával történik, majd vár annak válaszára. Ez tulajdonképpen egy adatbázis lekérdezésnek és az eredmény visszaadásának felel meg a gyakorlatban.

A visszaigazolás kiírása után következő rombusz feltételes elágazást jelent. Az utána következő nyilakon szerepel a feltétel. Amennyiben valamelyik megadott adat nem megfelelő, akkor szükséges az elágazás. Ekkor vagy visszatérünk az adatok bevitelére vagy egyből a folyamat végéhez léphetünk.

A folyamat végét a fekete pont szimbolizálja.

2. 4. 2. OSZTÁLYOK KIALAKÍTÁSA

2. 4. 2. 1. AZ OSZTÁLYDIAGRAMOKRÓL

Az osztályok az objektumorientált rendszer szívéét képezik, ezért a legnépszerűbb UML diagramok az osztálydiagramok. Az osztályok leírják azokat a különböző objektumokat, amelyekkel a rendszer működése során rendelkezhet. Az osztálydiagramok bemutatják ezeket az osztályokat és kapcsolataikat.

Az osztálydiagramok a kódoláshoz legközelebb álló modellező eszközök. Az osztályok attribútumainak és módszereinek meghatározása a forráskód előállításának egyik első lépése. Az UML CASE (Computer Aided Software

Engineering) eszközök könnyen tudnak generálni osztály template-eket az osztálydiagramok alapján, az implementáció nyelvével függően.

Az osztályok közötti legfőbb kapcsolat, az öröklődés, amely a legfőbb erőssége az osztályoknak. Az öröklődés a minőséget és produktivitást fokozza az újrafelhasználhatóságon keresztül.

Az osztályok, csak az attribútumaikkal, a rendszer egyedeit képezik, amelyek könnyen reprezentálhatnak adatbázisbeli táblákat. Tehát az osztályok az adatbázis megtervezését is elősegítik.

2. 4. 2. 2. A RENDSZER OSZTÁLYAI

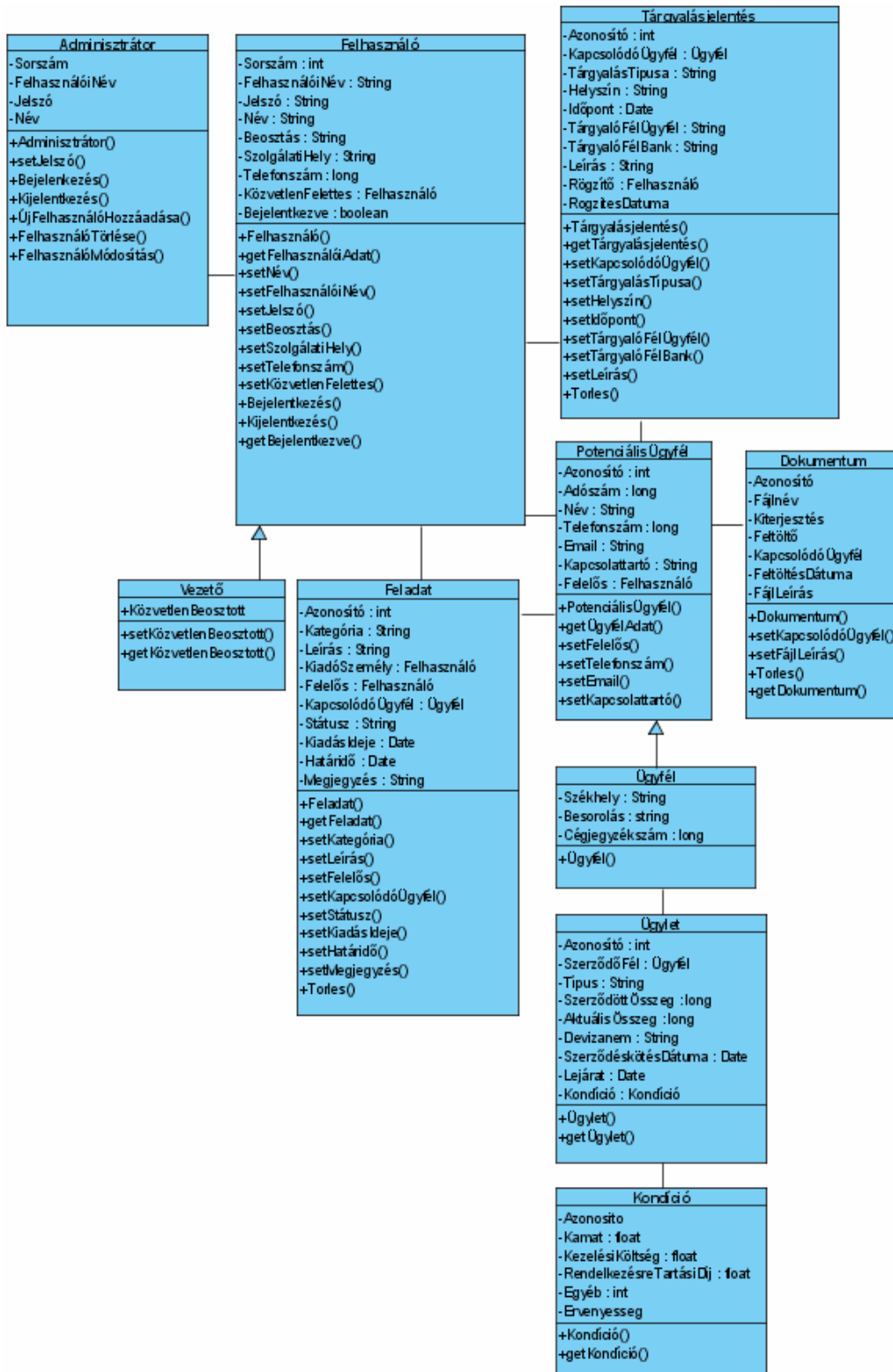
A rendszer osztályainak meghatározásához először vettem a rendszer külső szereplőit. A szereplők közül csak az emberi felhasználókból képeztem osztályokat, hiszen ők fogják használni a rendszert. Ezek a szereplők: az Adminisztrátor, a Felhasználó és a Vezető. További osztályok létrehozásához vettem a rendszerkövetelmények által előírt funkciókkal kapcsolatos főneveket, és azokból képeztem osztályokat: Tárgyalásjelentés, Dokumentum, Feladat, Ügyfél, PotenciálisÜgyfél, Ügylet. Az eredményt az osztály diagramokat ábrázoló 8. UML diagram mutatja.

Az UML az osztályok egy téglalapban ábrázolja. Legfelül az osztály neve található, alatta közvetlenül az attribútumai, a legalsó részben pedig a metódusok. A metódusok közül az első az osztály nevével megegyező nevű konstruktor. Az attribútumok és metódusok neve előtt szereplő jel a láthatóságra utal. A „+” jelenti a public és „-” a private láthatóságot. Az osztályok tartalmát nem részletezem, az attribútumok és módszerek a használati eset dokumentációk és az aktivitás diagramok alapján kerültek meghatározásra.

Az osztályok közötti kapcsolatokat jelentik az osztályok között megrajzolt vonalak és nyilak. Az Adminisztrátor osztály kapcsolatban áll a Felhasználó osztállyal, hiszen az előbbi példányai tudnak létrehozni, módosítani, törölni Felhasználó példányokat. Az öröklődési kapcsolat a Felhasználó és a Vezető osztály között itt is fennáll, ahogy a szereplők között, a Vezető őse a Felhasználó. Az öröklődést a nyíl jelenti, amely az ősoosztály felé mutat. Ugyanígy öröklődési viszony áll fenn a PotenciálisÜgyfél és az

8. UML diagram

Osztálydiagram



Ügyfél között, hiszen az Ügyfél a PotenciálisÜgyfél egy kiterjesztett formája, azt az ügyfelet reprezentálja, aki a bankkal szerződéses viszonyban áll, azaz van Ügylete. Az Ügylet osztály dokumentálásakor találtam egy újabb osztályt, a Kondíciót. Minden banki ügylet rendelkezik kondíciókkal, amely alatt az adott ügylet hiteldíjait (kamat, kezelési költség, rendelkezésre tartási jutalék, stb.) kell érteni.

2. 4. 3. AZ ADATBÁZIS MEGTERVEZÉSE

Az elkészített osztálydiagram jó kiindulási pont az adatbázis megtervezéséhez. Ez természetesen nem az osztályok automatikus táblákká alakítását jelenti. Ugyanis a forráskódban használt osztályok, objektumok tárolása adatbázisban egyszerűbben is megvalósítható, mintha egy az egyben letárolnánk az attribútumokat. Igyekeztem ezért minél kevesebb táblát kialakítani, hogy a hasonló szerkezetű osztályok (ez főleg az öröklődési viszonyban álló osztályokra igaz) ne kerüljenek külön táblába.

A következőkben bemutatom az elkészített táblákat.

29. táblázat

Felhasznalo tábla

Mező	Típus
Sorszam	bigint(20)
FelhasznaloiNev	varchar(15)
Jelszo	varchar(10)
Nev	varchar(50)
Beosztas	varchar(30)
SzolgalmatiHely	text
Telefonszam	int(10)
KozvetlenFelettes	varchar(50)
AdminE	tinyint(1)
PwLejarat	date
LoginProba	int(11)
RegIdo	date

A **Felhasznalo** táblát úgy alakítottam ki, hogy a rendszer összes felhasználója (Felhasználó, Vezető, Adminisztrátor) ebbe kerüljön tárolásra. A három különböző felhasználó típust egyetlen mező különböztet meg, az *AdminE*. Ha értéke 0, „sima” felhasználóról van szó, ha 1, akkor Adminisztrátor, és ha 2, akkor pedig Vezető. Látható, hogy az összes, Felhasználó osztályhoz tervezett attribútum itt mezőként szerepel, kivéve egyet: ez a *Bejelentkezve* attribútum. Azt, hogy az adott felhasználó

épp be van-e jelentkezve, nem szükséges és nem is érdemes adatbázisban tárolni, mert megoldja a PHP session-ökkel és sütikkel. A Vezető osztályból nem került be a *KözvetlenBeosztott* attribútum, mert ennek tárolása szintén felesleges; bármikor lekérdezhető az adatbázisból a *KozvetlenFelettes* mező alapján. Végül bekerült még néhány mező a táblába, melyeknek a biztonság szempontjából van szerepe. A *Sorszam* mező az adott egyed egyedi azonosítója, a tábla elsődleges kulcsa.

30. táblázat

Ugyfel tábla

Mező	Típus
Azonosito	bigint(20)
Adoszam	int(11)
Nev	text
Cegjegyzekszam	int(11)
Szekhely	text
Kapcsolattarto	varchar(50)
Telefonszam	int(11)
Email	varchar(50)
Besorolas	varchar(3)
Felelos	int(11)

Az **Ugyfel** tábla az Ügyfél és a PotenciálisÜgyfél adatait tartalmazza. A kettőt az fogja megkülönböztetni egymástól, hogy a PotenciálisÜgyfél-nek nincs ügylete (sem élő, sem lezárt), amely egyszerűen lekérdezhető az *Ugylet* tábla segítségével. A *Felelos* mező a *Felhasznalo* táblával való kapcsolatra utalra. A tábla a gyakori keresések miatt az *Adoszamra* és a *Felelosre* indexált. Elsődleges kulcs az *Azonosito*.

31. táblázat

Ugylet tábla

Mező	Típus
Azonosito	bigint(20)
SzerzFel	bigint(20)
Tipus	varchar(50)
SzerzOsszeg	double
AktOsszeg	double
Devizanem	varchar(3)
SzerzDatum	date
Lejarat	date
Kamatbazis	int(11)
Kamatfelar	float
KezKoltseg	float
RendTart	float

Mező	Típus
EgyebDij	float
Fedezet	text

Az **Ügylet** tábla tárolja az Ügylet és a Kondíció példányok adatait. A Kondíciónak nem hoztam létre külön táblát, mert bonyolultabb lenne a kezelése, ezáltal lassabb is, továbbá a kondícióknak önmagukban nincs értelmük, csak az ügylettel együtt. Viszont a kondícióknál ki kellett alakítanom egy *Kamatbasis* mezőt, amely egy másik táblára (*Kamatbasis*) hivatkozik. Erre a Kondíció osztályban megtervezett Kamat attribútum jellege miatt volt szükség. Ugyanis a kamat mindig egy kamatbázisból és egy kamatfelárból áll. A kamatbázis mindig az alapkamat, amelyhez képest az ügyfél kockázatától függően a kamatfelárat megállapítják. A kamatbázisok között van különbség a kamatozási periódusokat és a devizanemet illetően is (pl. 3 havi BUBOR, 1 havi CHF LIBOR, stb.). Ezért a kamatbázisokat egy külön táblában tárolom, amely nem más, mint egy lista az összes lehetséges kamatbázisról, és az éppen aktuális mértékekről (ld. 32. táblázat).

32. táblázat

Kamatbasis tábla

Mező	Típus
Azonosító	bigint(20)
Megnevezés	varchar(50)
Érték	float

A **Feladat**, **Tárgyalásjelentés** és **Dokumentum** táblában tárolt adatok teljes mértékben megegyeznek az adott osztály tervezett attribútumaival. A *Felelos*, *Kiado*, *Rogzito*, *Feltolto* mezők a *Felhasznalo* tábla elsődleges kulcsaira, a *KapcsUgyf* mező pedig az *Ugyfel* tábla elsődleges kulcsaira hivatkozik.

33. táblázat

Feladat tábla

Mező	Típus
Azonosító	bigint(20)
Kategoria	varchar(50)
Leiras	text
Kiado	bigint(20)
Felelos	bigint(20)
KapcsUgyf	bigint(20)
Statusz	varchar(30)
KiadasIdeje	datetime

Mező	Típus
Hatarido	datetime
Megjegyzes	text
Teljldeje	datetime

34. táblázat

Targyalasjelentes tábla

Mező	Típus
Azonosito	bigint(20)
KapcsUgyf	bigint(20)
Tipus	varchar(50)
Helyszin	varchar(50)
Idopont	date
TFUgyf	varchar(100)
TFBank	varchar(100)
Leiras	text
Rogzito	bigint(20)
RogzDatum	date

35. táblázat

Dokumentum tábla

Mező	Típus
Azonosito	bigint(20)
Fajlnev	varchar(50)
Feltolto	bigint(20)
KapcsUgyf	bigint(20)
FeltDatum	date
Leiras	text

2. 4. 4. AZ ALKALMAZÁS KÜLSŐ MEGJELENÉSE

Az alkalmazás grafikus felülete a funkcionalitáshoz képest másodlagos szempont. Természetesen egy vállalati használatra szánt webes alkalmazásnál a grafikus design kevésbé olyan fontos, mint egy internetre tervezett oldalnál. Azonban érdemes erre egy kis időt szánni, mert mégsem teljesen mindegy – főleg a felhasználónak -, hogy milyen az alkalmazás megjelenése.

Ezért az implementáció előtt megterveztem a grafikus felületet is. Az elsődleges célom az volt, hogy minél átláthatóbb, olvashatóbb legyen. Kialakítottam egy vékony fejléceket, amely tartalmazza a legördülő menüt is. Ezzel helyet takarítottam meg az alkalmazás többi részének. Próbáltam olyan színskombinációkat használni, amelyeket

nem fárasztják a szemet: ezek közül a legjobbak a világos háttéren sötét betű variációk.

2. 4. 5. A HARDVER

A tervezés magában fogja szoftver és hardver elemek megtervezését is. Ami a hardvert illeti, szükség van egy olyan számítógépre, amely ellátja a webszerver, az adatbázisszerver és az alkalmazásszerver szerepét. Ahogy korábban említésre került, a PHP, MySQL, Apache hármas nagyon sokféle platformon elfut, és alacsony gépigénnyel rendelkezik.

A rendszernek intraneten kell elérhetőnek lennie, ezért szükség van egy belső hálózatra, ahová a felhasználók számítógépei csatlakoznak, és elérhetik a webszerver által üzemeltetett rendszert. A PHP szerveroldali jellege miatt a felhasználók számítógépein nem szükséges semmilyen speciális szoftvert telepíteni, elegendő egy böngésző.

2. 5. VERIFIKÁCIÓ ÉS VALIDÁCIÓ

Miután végeztünk a tervezéssel, dokumentálással érdemes átvizsgálnunk azok minőségét. Ebben segít a verifikáció és validáció technikája (Verification and Validation, V&V). A V&V egy minőségellenőrzési technika, amely segít a hibák és inkonzisztenciák feltárásában. A V&V olyan tevékenységeket és ellenőrzéseket hajt végre, amely biztosítja, hogy a létrehozott modell helyes. A verifikáció a szoftver és a modell szintaktikai helyességét ellenőrzi. A validáció a szemantikáért felelős, azaz a rendszer felhasználói felé van jelentőséggel, mert azt ellenőrzi, hogy a rendszer mennyire teljesíti az elvárt felhasználói követelményeket. (Unhelkar, 2005)

A V&V ellenőrzéseknek három ismert szintjét szokták megvizsgálni: szintaktika, szemantika és esztétika.

Első körben a modell verifikációját és validációját érdemes elvégezni. Mivel a tervezés UML-ben történt, amely mint vizuális nyelv, rendelkezik saját szintaktikával, ezért a szintaktikai ellenőrzés annak ellenőrzése, hogy az UML eszközöket megfelelően használtuk-e. Ebben segítenek a modellező szoftverek is.

A szemantikai ellenőrzés azt jelenti, hogy az előállított diagramok és elemeik valóban azt jelentik-e, amit ábrázolni szerettünk volna. Ennek azért van jelentősége, mert egy szoftverfejlesztő cégnél mások végzik a tervezést és az aztán következő feladatokat, és egy diagramoknak ugyanazt a jelentést kell hordoznia mindenki számára. Olyanokra kell odafigyelnünk itt, hogy pl. ahol Felhasználót írtunk, ott valóban annak kell-e szerepelnie, és nem Adminisztrátornak vagy Vezetőnek.

Végül az esztétikai vizsgálat abban segít, hogy előállított diagramok, dokumentációk világosak, átláthatóak, megnyerőek legyenek.

A szempontok alapján megvizsgáltam az általam készített modellt, ahol hibát találtam, javítottam.

2. 6. IMPLEMENTÁCIÓ

2. 6. 1. AZ ALKALMAZÁS FELÉPÍTÉSE

Leegyszerűsítve, az alkalmazás lényegében adatbeviteli űrlapokból és megjelenítő oldalakból áll. Az átláthatóság érdekében az alkalmazás több fájlból tevődik össze.

Az alkalmazás fő része a **main.php**-ben található. Az adatok feldolgozása, a funkciók végrehajtása itt történik. A *main.php* vezérli az outputot is, és az összes többi fájl futás közben ide kerül importálásra (include paranccsal). A fájl első része a változó átvétellel kapcsolatos parancsokat tartalmazza, aztán *switch-case* utasítással történik a megfelelő funkció kiválasztása. Hogy éppen melyik funkcióra van szükség, azt űrlapok vagy belső linkek küldik, az alkalmazás menüjének vagy *hidden* (rejtett) input mezők segítségével. Ezután kiválasztásra és végrehajtásra kerülnek az adott *case*-en belüli utasítások, majd az eredmény továbbítódik a böngészőnek.

Az alkalmazás **includes** mappájában találhatóak a futás közben importálásra kerülő fájlok. Ezek egy része űrlap, az adatbevitelt szolgálja, másik része az eredmény megjelenítéséért felelős. Az űrlapok kódja szintén PHP, nem tisztán HTML, mert több űrlapot többször is felhasznál az alkalmazás, amikor nagyon hasonló tevékenységeket kell végezni. Ilyenek pl. a kereső funkciók: ha feladat kiadáshoz és tárgyalásjelentéshez keresünk ügyfelet, a rendszer ugyanazt az űrlapot fogja használni. Flag-ek irányítják a megfelelő működést. Az include könyvtár a következő

fájlokat tartalmazza (a név utal a funkcióra): *FajlFeltolt.php*, *FeladatKeres.php*, *FeladatKeresOutput.php*, *FeladatKiadas.php*, *FelhasznaloHozzaadas.php*, *FelhasznaloiAdatModositas.php*, *FelhasznaloiJelszoModositas.php*, *FelhasznaloKeres.php*, *FelhasznaloKeresOutput.php*, *FelhasznaloModositas.php*, *login.php*, *menu.php*, *TJHozzaad.php*, *UFKeres.php*, *UFKeresOutput.php*, *UgyfelAdatModositas.php*, *UgyfelHozzaadas.php*, *UgyfelKeres.php*, *UgyfelKeresOutput.php*.

Az alkalmazás **classes** könyvtára tartalmazza az osztályokat. Az osztályok tekintetében az implementációnál jelentős egyszerűsítést hajtottam végre a tervezettekhez képest. Az osztálydiagramon felvázolt osztályok közül csak a *Felhasznalo* osztályt használtam fel. Az összes többit ugyanis tárolja az adatbázis, az alkalmazás futása közben kerülnek beolvasásra és majdnem közvetlenül kiírásra. Ezért nem érdemes osztályokat kialakítani és példányosítani őket, mert csak a kódot bonyolítja, és a futást lassítja: ha pl. minden egyes ügyfélre létrehozok egy példányt, amikor beolvasok egy sok ügyfélből álló listát. A *Felhasznalo* viszont érdemes példányosítani. Egy *Felhasznalo* példány mindig egy aktuálisan bejelentkezett felhasználót fog jelenteni (az adatok előzetesen természetesen az adatbázisból kerülnek beolvasásra).

Továbbá létrehoztam egy *Adatbazis* és egy *Output* osztályt, amelyek nem szerepeltek a tervekben. Az *Adatbazis* egy példánya mindig az adatbázissal való kapcsolatot jelenti. Módszereinek nagy részét a PHP MySQL lekérdezéseinek általánosított, rövidített változatai alkotják. Az *Output* osztály pedig az általánosítható megjelenítési funkciókat menedzseli. Tehát a *class* mappa tartalma a következő három fájl: *Adatbazis.php*, *Felhasznalo.php*, *Output.php*.

A fentiekén kívüli fájloknak a megjelenítésben van szerepe:

menu_admin.js, menu_vezeto.js,	Az alkalmazás menüjének forráskódja
menu_felhaszn.js és	(JavaScript). Felhasználói típus alapján
awmData-menu könyvtár	különböző a megjelenített menü.
stilus.css	Az alkalmazás megjelenítési stílusait
	tartalmazza (Cascade Style Sheets).
img könyvtár	A headerben látható képfájl tartalmazza.

2. 6. 2. BIZTONSÁG

Az alkalmazás felhasználói névvel és jelszóval védett. Felhasználót, ahogy láttuk, az adminisztrátor tud hozzáadni. Ekkor a kezdeti jelszó kerül beállításra: Pass1234. A felhasználónak első bejelentkezésekor ezt rögtön meg kell változtatnia. A jelszó követelményei: minimum 8 karakter, legyen benne nagybetű és szám is. Erre szükség van, mert a felhasználók hajlamosak túl egyszerű és könnyen megfejhető jelszavakat használni. A rendszer három sikertelen belépési kísérlet (rossz jelszó) után letiltja az adott felhasználót. Feloldás csak az adminisztrátor segítségével lehetséges. A jelszavak cseréjét havonta el kell végezni, ezt a rendszer ki is kényszeríti. A jelszavak kódolva kerülnek tárolásra az adatbázisban. A titkosításhoz az autentikációnál gyakran használt base64 kódolást alkalmaztam. Szükség esetén az összes, adatbázisba kerülő adat kódolása megoldható.

A rendszeren belül nagy szerepe van a jogosultságvizsgálatnak, ahogy ez már látszott az UML diagramokon is. A jogosultságvizsgálat első lépcsője megtörténik a bejelentkezéskor. A rendszer felhasználói típusonként más menüt jelenít meg. Ezenkívül, a kódban is vannak elhelyezve vizsgálatok, flag-ek, amelyek segítenek megakadályozni, hogy ne lehessen megkerülni a rendszert, és elérni olyan funkciókat, amelyre a felhasználó nem jogosult.

2. 7. TESZTELÉS ÉS TELEPÍTÉS

A forráskód tesztelése az implementáció során folyamatos volt. Ahogy elkészült egy újabb funkció, rögtön tesztelés követte. Ez sokkal egyszerűbb volt így, mintha a végén teszteltem volna az egészet. Természetesen az implementáció befejezése után, egy teljes teszt következett, ahol már csak apróbb (pl.: megjelenítésbeli) hibák fordultak elő.

Mivel a rendszer nem konkrét megrendelésre, konkrét felhasználóknak készült, ezért az alkalmazás széles körű, több felhasználó általi tesztelése nem valósult meg.

A teszteléshez használt környezet: Ubuntu linux operációs rendszer, Apache 2.0, PHP 5.1.6, MySQL 5.0.24a. Az alkalmazás kliensoldala Mozilla Firefox 2 és MS Internet Explorer 6 böngészőkkel került tesztelésre.

A PHP forráskódja nem igényel fordítást, ezért az alkalmazás telepítésekor elég a szükséges fájlokat felmásolni egy PHP futtatásra alkalmas webszerverre. Ezután az adatbázist kell létrehozni az adatbázis szerveren, az sql fájl segítségével, amely tartalmazza az adatbázis szerkezetét.

3

A RENDSZER BEMUTATÁSA

A következőkben a rendszer működését mutatom be, funkciók szerint, képernyőképekkel illusztrálva, a teljesség igénye nélkül.

7. ábra

A bejelentkező képernyő.

The screenshot shows a web application interface for 'int Bank'. The header bar has the 'int Bank' logo on the left and the 'UKTR' logo with the text 'Ügyfélkapcsolat Támogató Rendszer' on the right. The background of the header is a collage of US dollar bills. The main content area is a light gray rectangle. In the center of this area is a blue login form titled 'Bejelentkezés'. The form has two input fields: 'Felhasználói név:' and 'Jelszó:'. Below these fields is a button labeled 'Bejelentkezés'. The footer of the page is a dark blue bar with the text '©2007 Szentesi Viktor' on the right side.

Az **adminisztrátor** által elérhető funkciókat mutatja a következő kép:

8. ábra

Az admin menü.



Az *Adataim* menüpontban a felhasználó saját adatai jeleníthetők meg, onnan érhető el a felhasználó adatainak módosítása és a jelszó megváltoztatása.

A következő menüpontban a meglévő felhasználók karbantartásával kapcsolatos funkciók érhetők el. Először egy felhasználó keresés történik, majd a keresési feltételeknek megfelelő felhasználókra érhetőek el a módosítás, törlés, aktiválás, letiltás funkciók. Felhasználó keresése név, felhasználói név, beosztás, szolgálati hely és közvetlen felettes alapján történhet, legördülő (*select*) menüből kell kiválasztani a szűrőfeltételt. A letiltott felhasználók külön menüpontból listázhatóak.

9. ábra

Felhasználó keresése.



A *Felhasználó hozzáadás*-sal kerülhet új felhasználó a rendszerbe. Itt meg kell adni a felhasználó adatait. Legördülő listából választható ki a rendszerben lévő vezetők közül a közvetlen felettes, továbbá a felhasználó fiók típusa (vezetői, nem vezetői, adminisztrátor). Az adatok később módosíthatóak.

10. ábra

Felhasználó hozzáadása.

Új felhasználó hozzáadása

Név:

Felhasználói név:

Beosztás:

Szolgálati hely:

Telefon:

Közvetlen felettes:

Felhasználói fiók típusa:

Hozzáad

Ahogy látható, az adminisztrátorok szerepe kizárólag a felhasználói adminisztrációra korlátozódik. A rendszer többi felhasználója azonban ezeket a funkciókat nem vagy csak korlátozottan érheti el. Felhasználói fiók típusának megváltoztatására csak az admin jogosult. Közvetlen felettes beállításra pedig jogosult még a vezető, de minden vezető csak a saját magát állíthatja be vezetőként beosztottjainál.

Nem vezetői hozzáféréssel bejelentkezve a *Felhasználó* menüpontban már csak az *Adataim* és a *Felhasználó keresése* funkciók érhetőek el. Az *Ügyfél* menüben az *Ügyfeleim-re* kattintva listázhatóak ki az adott felhasználóhoz tartozó ügyfelek. Onnan tovább lehet lépni az ügyfél részletes adatainak megtekintésére, módosítására, továbbá az adott ügyfél ügyleteinek listájára. Az ügyfél ügyleteinek listája a legfontosabb adatokat tartalmazza az ügyletről (típus, összeg, devizanem, lejárat). A Részletek gombra kattintva jelennek meg az egyéb adatok az ügyletről. A lista alján az ügyletek aktuális összege kerül összegzésre, forintba átváltva.

11. ábra

Ügyfél menü.



12. ábra

Ügyfél ügyletei.

Ügylet azonosító	Ügyfél	Ügylet	Aktuális összege	Devizanem	Lejárat	
3	Teszt Kommunikációs Rt.	bankgarancia	3 000 000,00	EUR	2008.04.10	Részletek
4	Teszt Kommunikációs Rt.	folyószámlahitel	89 310,00	USD	2007.10.11	Részletek
1	Teszt Kommunikációs Rt.	hitel	2 356 436,00	HUF	2007.04.30	Részletek
2	Teszt Kommunikációs Rt.	hitel	353 254,00	CHF	2012.04.29	Részletek
Összes kockázatvállalás forintban:			841 460 802,24			

Az *Ügyfél keresés* menüpontból érhetőek el a nem a felhasználó nevéen lévő ügyfelek adatai. Itt módosításra már nincs jogosultság, csak a megtekintés engedélyezett. Az *Új potenciális ügyfél*-re kattintva vihető fel új ügyfél a rendszerbe, amely még nem szerződött ügyfele a banknak. Az alkalmazás felvitelkor adószám alapján ellenőrzi, hogy van-e már olyan ügyfél a rendszerben.

A *Feladat* menüpontban a felhasználó elintézendő, kiadott feladatai és összes feladata listázható. A listából lehet megtekinteni a részleteket, lehet teljesíteni a feladatot. Feladat módosításra és törlésre csak a feladat teljesítése előtt van lehetőség, mert utólag már nincs értelme.

Ebből a menüpontból érhető el a *Feladat kiadás* is. Feladat kiadást nem vezetői jogosultsággal csak saját maga részére hajthat végre a felhasználó. Feladat kiadás előtt ki kell választani (keresés), melyik ügyfélhez kapcsolódik a feladat.

13. ábra

Ügyfél keresés feladat kiadás előtt.

A *Tárgyalásjelentés* és *Dokumentum* menüben mindegyik funkció kereséssel indul. Itt is szükség van a kapcsolódó ügyfél megadására. A rendszerben bármely felhasználó által rögzített tárgyalásjelentés és feltöltött dokumentum megtekinthető. Erre azért van szükség, mert több felhasználó is kapcsolatba kerülhet egy céggel (helyettesítés, munka-átcsoportosítás). Tárgyalásjelentés, dokumentumot módosítani, törölni csak az eredeti rögzítő, feltöltő tud.

A vezetők a fentiekén túl további funkciókat érhetnek el. A vezető saját magához rendelhet felhasználókat (*Felhasználó vezetőhöz rendelése*), kilistázhatja az irányítása alá tartozó felhasználókat. A felhasználók adatainál megtekinthető egy összegzés az adott felhasználó ügyfeleit, ügyleteit, feladatait illetően (15.ábra).

14 ábra

Vezetői funkciók.

Felhasználó	Ügyfél	Feladat
Adataim		
Felhasználó keresés		
Felhasználó vezetőhöz rendelése		
Közvetlen beosztottaim		

Ügyfél	Feladat	Tárgyalásjelen
Ügyfeleim		
Ügyfél keresés		
Új potenciális ügyfél		
Adott felhasználó ügyfelei		
Ügyfél felhasználóhoz rendelése		

15. ábra

Felhasználó statisztika.

Szentesi Viktor	
Ügyfelek száma:	6 db
Ügyletek száma:	4 db
Kezelt portfólió (szerződött):	858 570 994,84,- Ft
Kezelt portfólió (aktuális):	841 460 802,24,- Ft
Átlagos ügylet összeg:	214 642 748,71,- Ft
Legkisebb szerződött ügylet:	32 424,00,- Ft
Legnagyobb szerződött ügylet	768 450 000,00,- Ft
Összes feladat:	4 db
Ebből teljesített:	1 db
Ebből elintézendő:	3 db
Határidőben teljesített:	1 db
Határidőn túl teljesített:	0 db

A vezető megtekintheti bármely felhasználó ügyfeleit, és ügyfeleket rendelhet felhasználókhoz.

Továbbá a vezető más felhasználó részére is adhat ki feladatot, nem csak saját maga részére, ahogy a felhasználó.

ÖSSZEFOGLALÁS

Dolgozatom célja egy webes alkalmazás fejlesztése és a fejlesztés menetének bemutatás volt. A fejlesztés előtt már rendelkeztem a szükséges webfejlesztési (PHP, MySQL, HTML) ismeretekkel, azonban az UML eszközeivel részletesebben meg kellett ismerkednem, hogy a tervezést megfelelően hajtsam végre. Nem volt szükség az összes UML eszköz használatára, a rendszer viszonylag kis mérete miatt.

Az elkészült rendszer teljesíti a rendszerkövetelményeket, csak néhány ponton kényszerültem módosításra az implementációban. A rendszer, ahogy korábban említésre került, a bank bevonása nélkül került fejlesztésre. Ezért nem értem hozzá az alkalmazás szempontjából külső rendszereknek számító adatbázisokhoz, azok szerkezetéhez. Ezek hiányában a külső adatbázisból való lekérdezéseket is a rendszer által használt adatbázisból való lekérdezések helyettesítik. A rendszerkövetelményekben meghatározott funkciók közül csak a jövedelmezőség számolást nem tudtam megvalósítani, mert ahhoz a banki rendszerekben lévő, állomány és kamatadatokat tartalmazó, valós idejű idősorok lettek volna szükségesek, amelyek nem álltak rendelkezésemre.

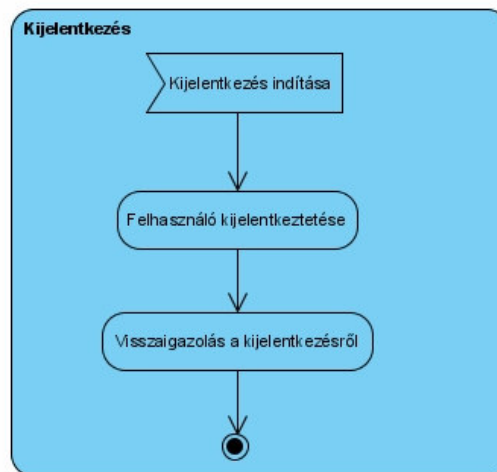
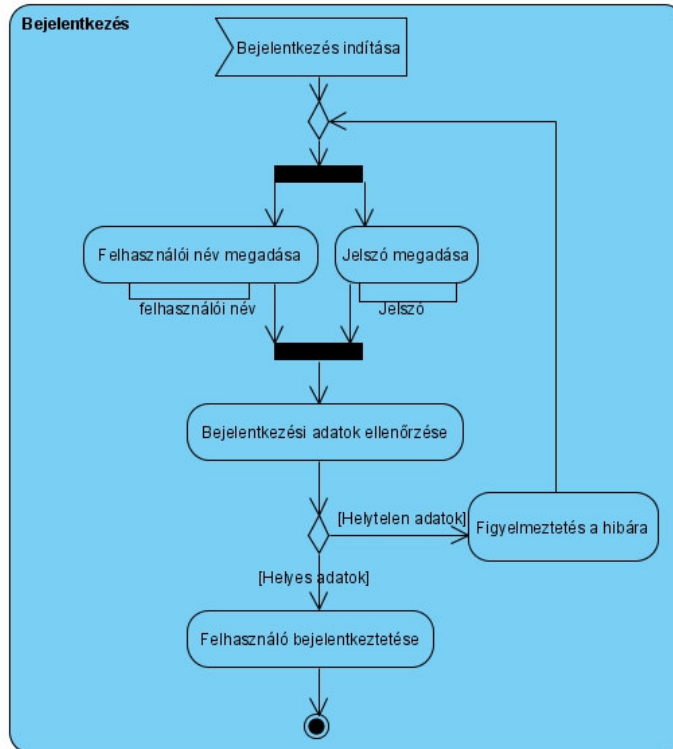
A kész alkalmazás előnye, hogy nemcsak bankban használható, hanem bármely olyan területen, ahol a munkavégzés része az ügyfelekkel történő kapcsolattartás. A program használata előtt természetesen szükséges az adott munkakörnyezetbe való beillesztés, más rendszerekkel való kapcsolatok kialakítása.

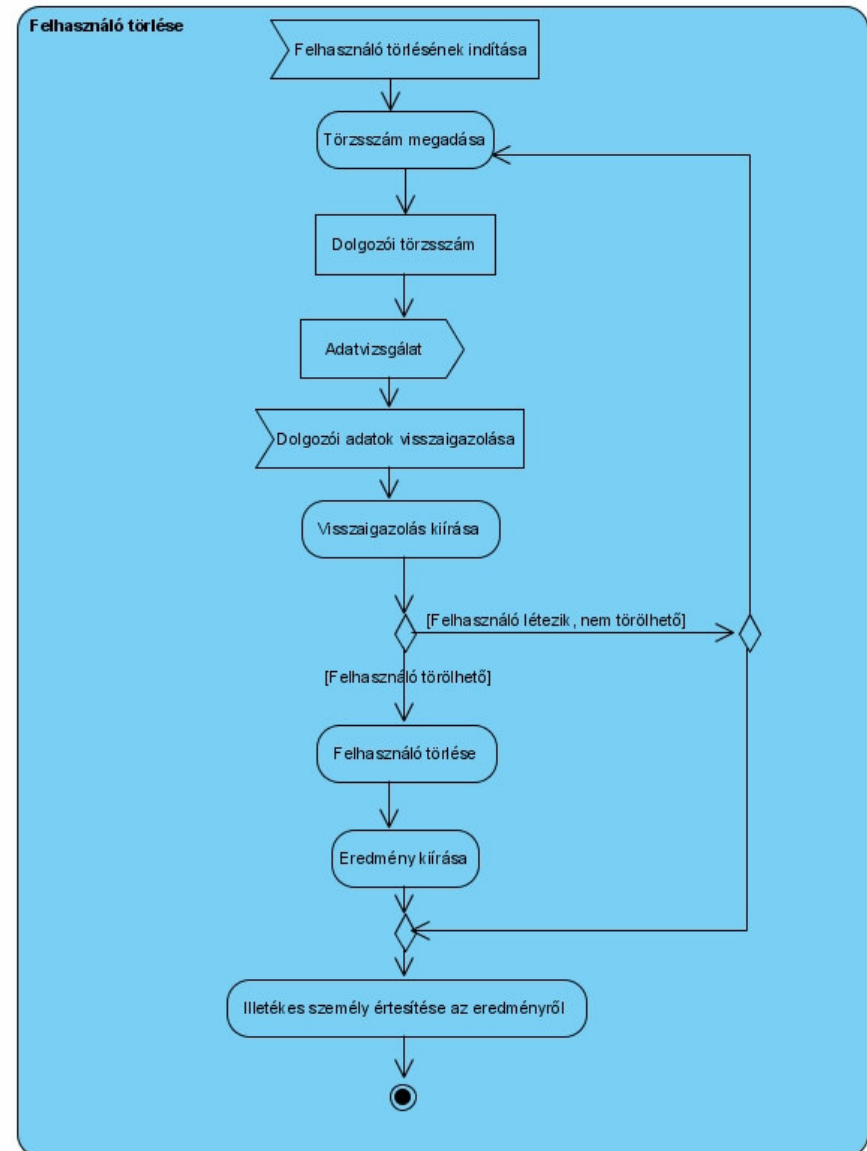
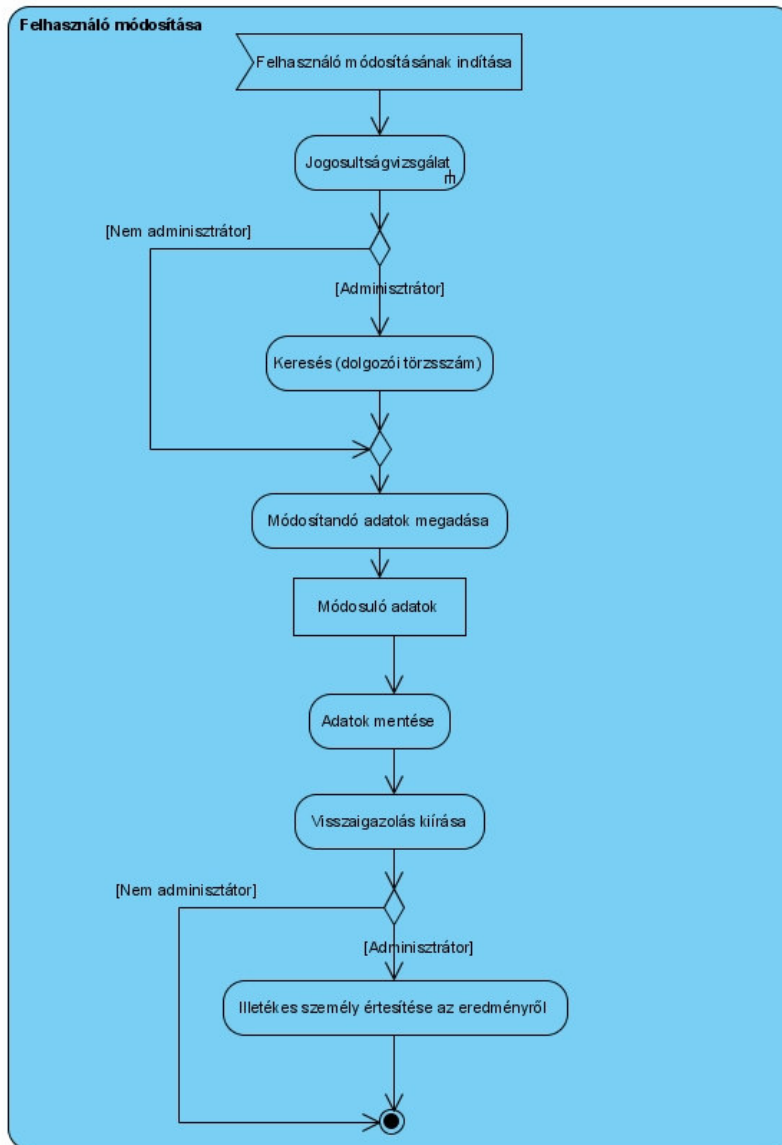
A dolgozat megírása során szerzett UML és rendszerfejlesztési ismeretek jó kiindulási alapot jelentenek majd számomra egy következő, komplexebb alkalmazás fejlesztése során.

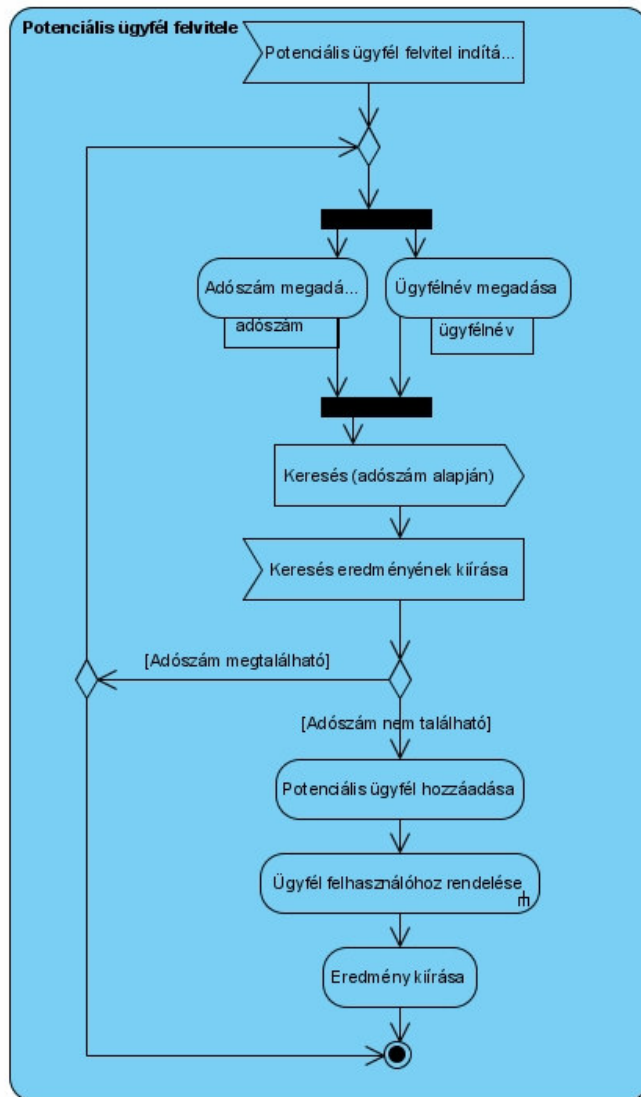
MELLÉKLETEK

1. melléklet

A felhasználói adminisztráció aktivitás diagramjai.

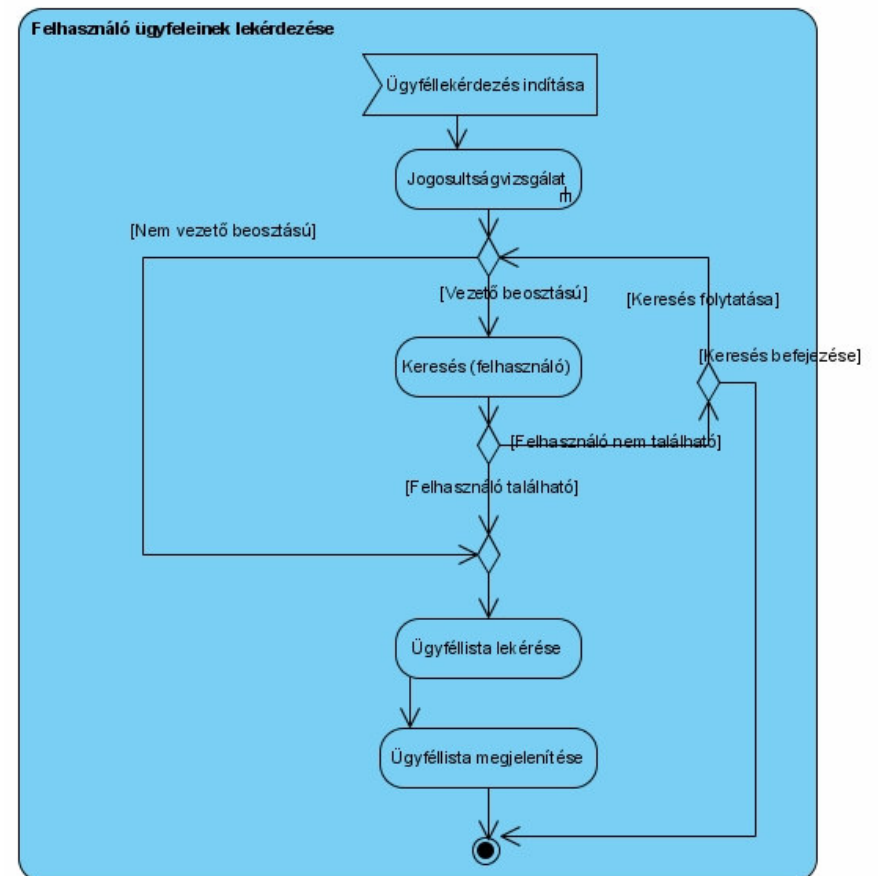


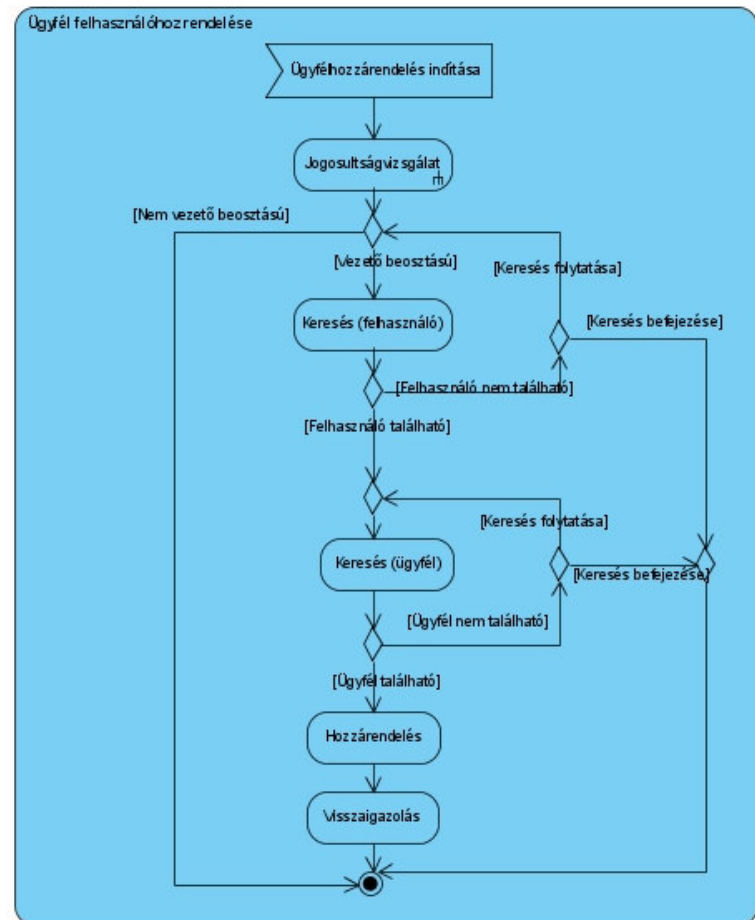
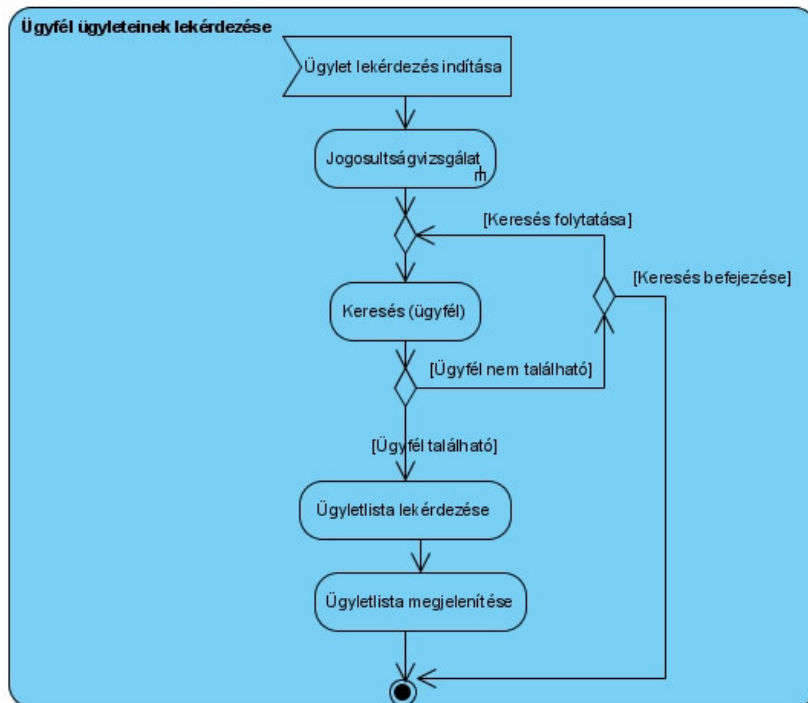




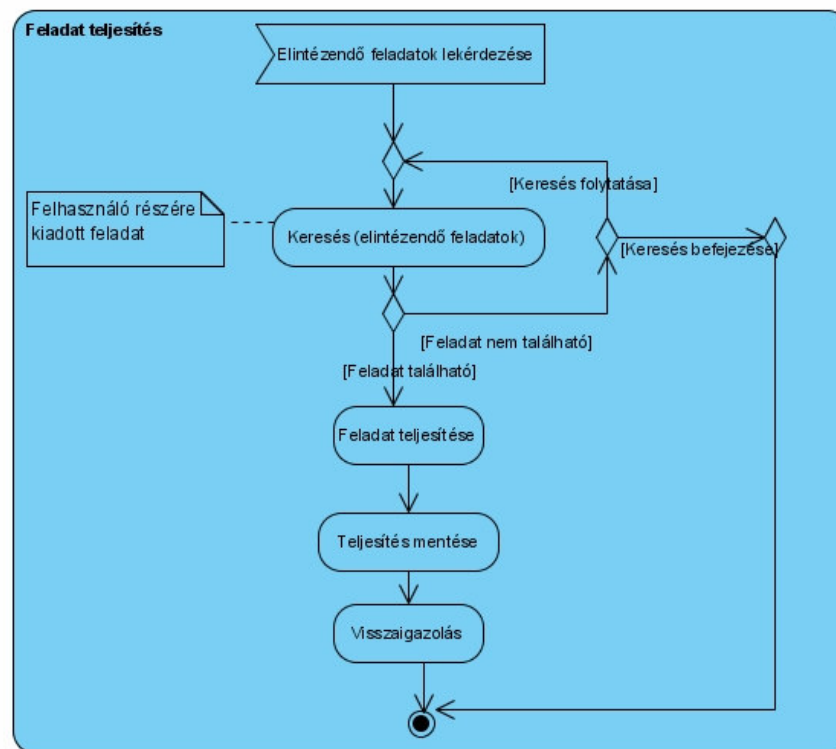
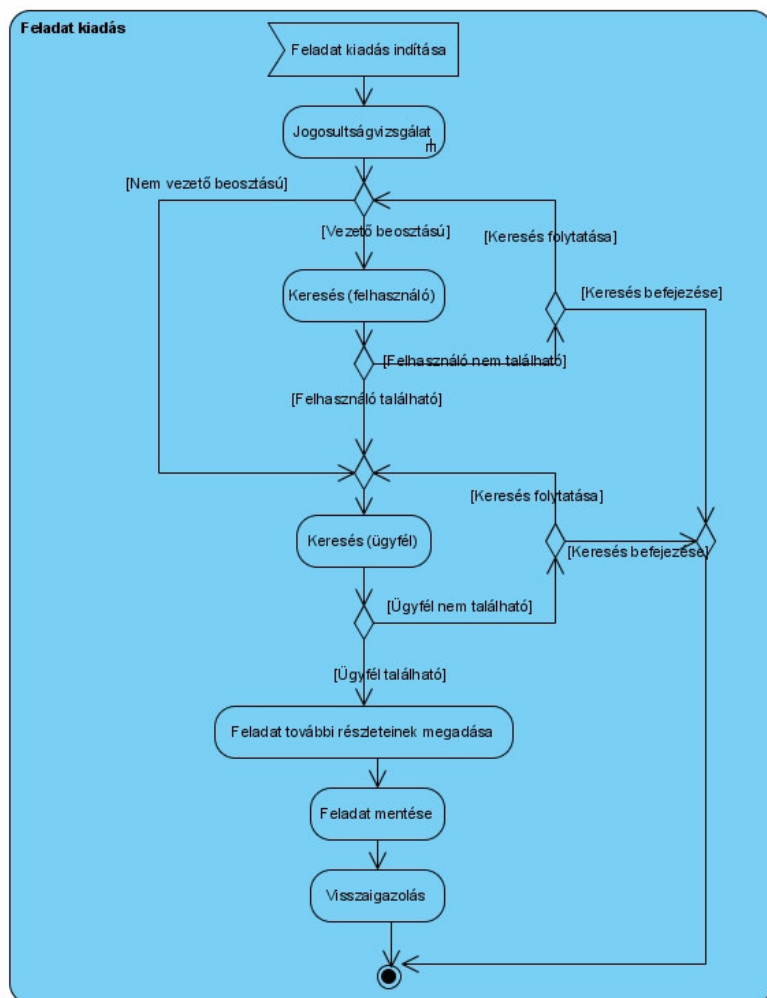
2. melléklet

Az ügyfél adminisztráció aktivitás diagramjai.



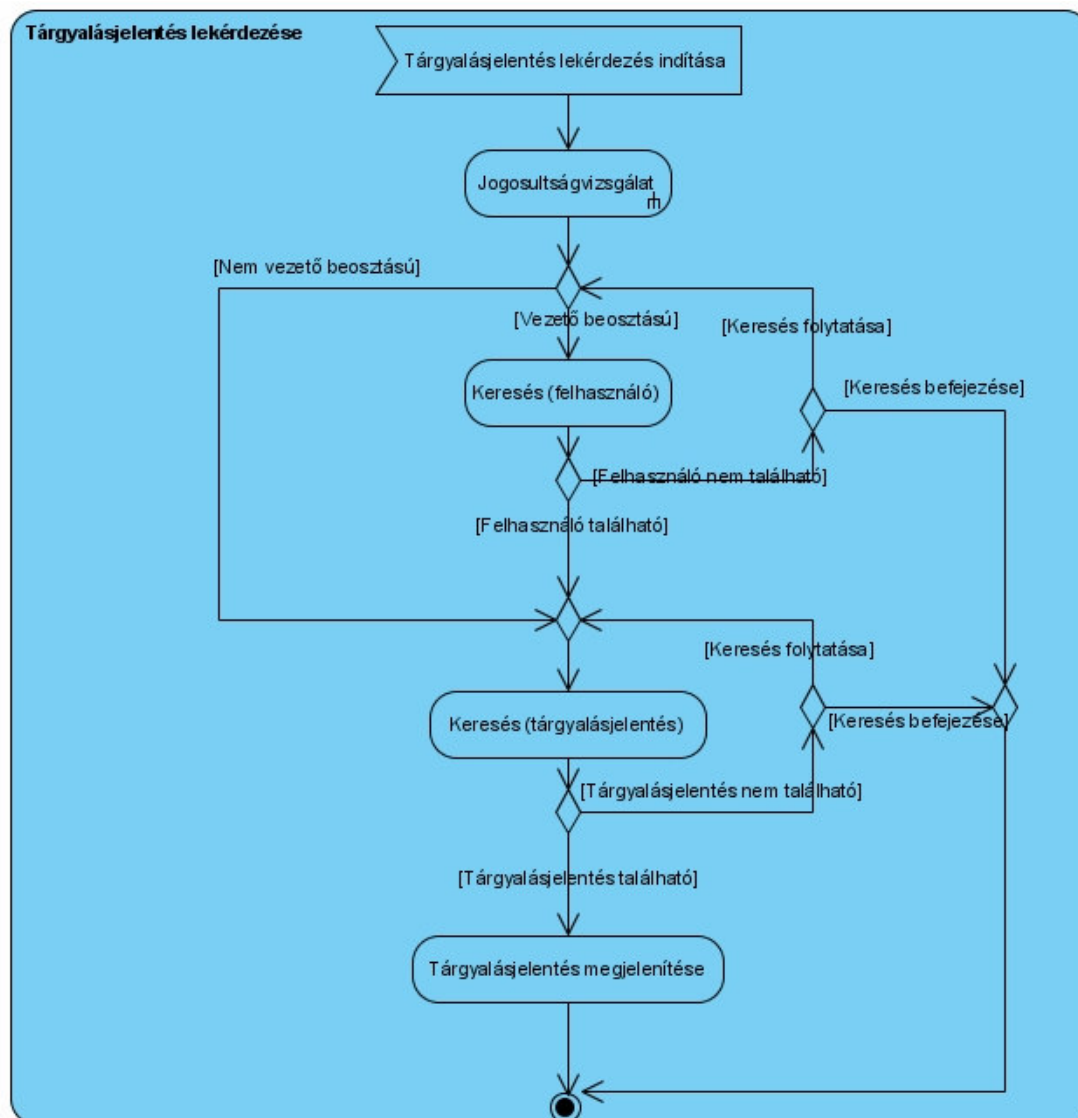
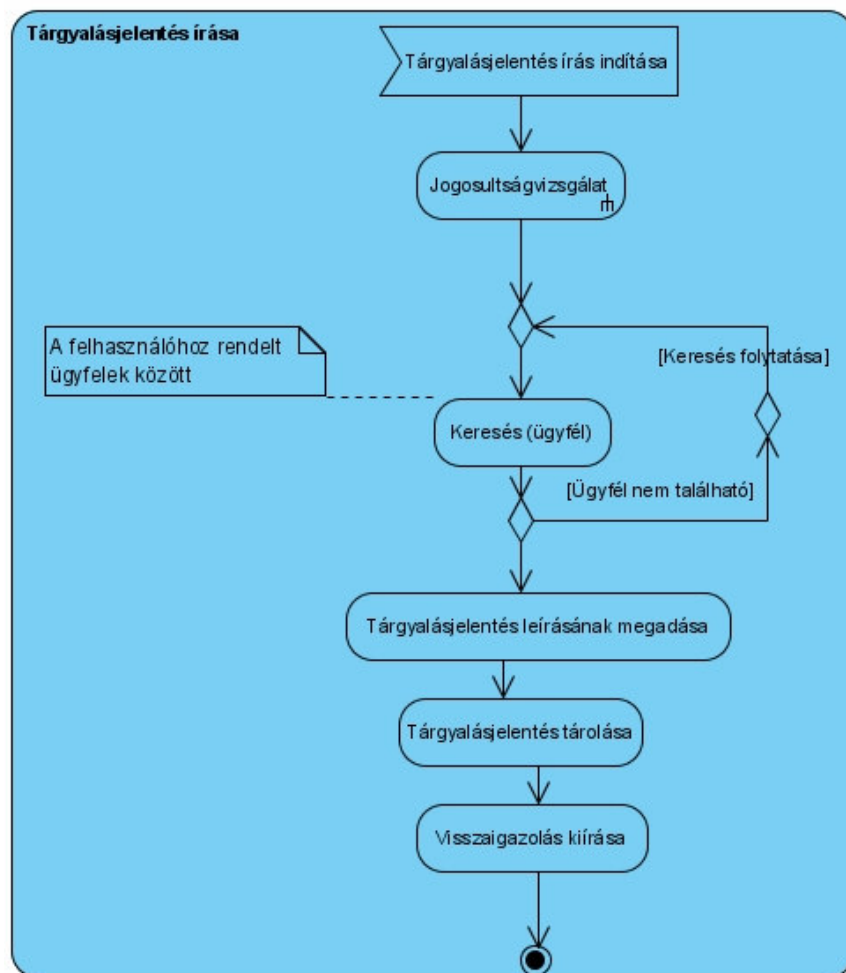


3. melléklet
Feladatok aktivitás diagramjai



4. melléklet

Tárgyalásjelentések aktivitás diagramjai



IRODALOMJEGYZÉK

- Converse, T. - Park, J. – Morgan, C. (2004): PHP5 and MySQL Bible.
Wiley Publishing, Inc, USA.
- Glass, M. – Le Scouarnec, Y. – Naramore, E. – Mailer, G. – Stolz, G. - Gerner, J.
(2004): Beginning PHP, Apache, MySQL Web Development.
Wiley Publishing, Inc, USA.
- Vaswani, V. (2005): How to Do Everything with PHP & MySQL.
McGraw-Hill/Osborne, USA.
- Bakken, S. S. – Schmid, E. (2002): PHP kézikönyv.
PHP Documentation Group.
- Bacon, J. (2007): Practical PHP and MySQL: Building Eight Dynamic Web
Applications. Prentice Hall, USA.
- MySQL AB (2005): MySQL® Administrator's Guide. Sams Publishing, USA.
- Kofler, M. (2004): The Definitive Guide to MySQL, Second Edition. Apress, USA.
- Pilone, D. – Pitman, N. (2005): UML 2.0 in a Nutshell. O'Reilly, USA.
- Hamilton, K. – Miles, R. (2006): Learning UML 2.0. O'Reilly, USA.
- O'Docherty, M. (2005): Object-Oriented Analysis and Design: Understanding System
Development with UML 2.0. John Wiley & Sons Ltd., England.
- Garland, J. – Antony, R. (2003): Large-Scale Software Architecture: A Practical
Guide using UML. John Wiley & Sons Ltd., England.
- Unhelkar, B. (2005): Verification and Validation for Quality of UML 2.0 Models. John
Wiley & Sons Ltd., USA.