

Esemény vezérelt programozás beadandó



Banki alkalmazás

Hujber Ádám

Kovács Kornél

Kulman Ferdinánd

Tartalom

A program és a projekt rövid bemutatása	2
Program specifikáció	2
Jegyzőkönyv a banki alkalmazás specifikálásáról	3
Rendszerterv	5
Java dióhéjban	5
JavaFx bemutatása	5
Alkalmazott technológiák	6
A program megoldási terve	6
Az SQL dióhéjban	7
Adatbázishoz alkalmazott technológia	7
Adatbázis felépítése	7
Bejelentkezés	8
Kezdőképernyő	8
Új ügyfél	9
Ügyfél adatok	9
Számlatörténet	10
Átutalás	10
Befizetés/kifizetés	11
Számla nyitás/zárás	11
Tesztelési terv	12
Java osztályok UML diagramjai	14

A program és a projekt rövid bemutatása

A projekt:

A Gintonic Bank alkalmazás az Eötvös Lóránd egyetem Esemény vezérelt programozás tárgyának a beadandó csapatmunkájaként készült el. Célunk egy olyan könnyen kezelhető, átlátható, felhasználóbarát grafikus asztali alkalmazás létrehozása volt, amely megkönnyíti a banki alkalmazott munkáját a rendkívül letisztult, intuitív grafikus felhasználói felületének köszönhetően. Az alkalmazás jövőbiztos lesz, garantáljuk a rendszeres rendszerfrissítéseket és egyéb biztonsági fejlesztéseket is.

A projekten dolgozó csapat heti rendszerességgel, tudatosan, az elvárásoknak megfelelően építette fel a projektet. A projekthez tartozó GitHub repository, a Gantt diagram és a minden más fájl rendszeresen kezelve volt.

Szükséges támogatás:

- Pénzügyi szakemberek
- Anyagi támogatás
- Jogászati tanácsadás
- Megfelelő hardver környezet kialakítása

Program specifikáció

Megvalósíthatósági tanulmány

A megrendelőnél már megvannak a telepítéshez szükséges feltételek. Vállalták a szerverparkjuk fejlesztését amennyiben ez szükséges. A kliens számítógépek Windows 10-et futtatnak, és teljesítményük is bőven megfelel a tervezett program futtatására. A fejlesztési időt 2 hónapban maximalizáltuk. Ennek költségét a megrendelő fizeti. Az átadást követően a szoftver támogatását 1 évig vállaljuk, ezalatt az esetleges kisebb hibák javítását elvégezzük, létező funkciók továbbfejlesztését vállaljuk, de nem adunk hozzá újabb funkciókat, vagy bővítjük a már meglévőket. Ez utóbbiak igény esetén külön díj megfizetése ellenében további egyeztetés kérdése.

Követelmény feltárása, elemzés

A banknak nincs nagyobb rendszere, ami alá be kellene tagozódnunk, így ezzel nem kell foglalkozni. A cég egy letisztult, könnyen kezelhető, egyértelműen átlátható rendszert kért.

Funkcionális követelmények:

A programnak támogatnia kell legalább 5 felhasználó egyidejű munkáját. Főbb funkcióit tekintve tudnia kell ügyfelek adatait tárolni és számlaadatokat nyilvántartani. Ezekkel kapcsolatos lekérdezéseket lebonyolítani GUI felületen. Alapszintű biztonsági és megbízhatósági elveknek megfelelni.

Nem funkcionális követelmény

Komoly biztonsági előírások teljesítése, több felhasználó esetén is megfelelően gyors legyen a lekérdezés. Mivel naponta újraindítsa, nem kell 24-órás sessionokra felkészülni.

Termék követelmények

Ne használja túlzottan az erőforrásokat, más program is tudjon futni mellette. Fontos a megbízhatóság, az adatok nem veszhetnek el, és a tranzakciók se „érhetnek össze”. Hordozhatóság szempontjából a cégnél csak windows eszközök vannak és nem is terveznek váltani szóval más

operációs rendszert nem kell figyelembe venni. Végül pedig fontos, hogy a program használata könnyen tanulható, elsajátítható legyen. Mivel a programot csak egy szűk kör fogja használni, így az akadálymentességre sem kell kiemelkedően nagy figyelmet fordítani. Fontos működési feltétel, hogy a használathoz elegendő legyen alapszintű számítógépes ismeret.

A program fejlesztése Java nyelven fog történni, a grafikus felület megvalósításához szükséges külső könyvtárak bevonásával.

Külső követelmények

A személyes adatokkal kapcsolatos szabályozásoknak megfelelően kell eljárni az adatok tárolása során.

Jegyzőkönyv a banki alkalmazás specifikálásáról

(- fejlesztő; + ügyfél)

-Milyen szoftvert képzeltek el nagyvonalakban?

+Egy grafikus rendszert szeretnénk fejleszteni, ami a mai modern elvárásoknak megfelelő kezelőfelülettel bír, de nem kell túl extravagánsnak lennie, mivel az ügyintézők fogják használni. Fontos azonban, hogy intuitív legyen, minden egyértelműen látható legyen rajt, és gyorsan elérhetőek legyenek a különböző menüpontok.

-Milyen funkciókkal kell, hogy rendelkezzen?

+Bankunknak a folyószámlák kezelésére kellene ez a rendszer, emiatt fontos, hogy az ezzel kapcsolatos ügyeket lehessen vele intézni. Ide tartozik a számlanyitás, számlák közti átutalás, valamint a pénzkivétel és betétel is. Ha belefér a fejlesztési keretbe még az jó lenne, ha a számlatörténetet le lehetne kérdezni.

-Ehhez nyilván adatbázisban kell gondolkodni. Pontosan milyen adatokat tárolnának el az ügyfelekről, valamint a számlákról?

+Az ügyfelekről mindenképp tudnunk kell a személyes adatokat. Ide tartoznak a személyi szám, anyja neve, születési helye és ideje, a neve természetesen és az elérhetőségei. A számlákról csak azt tartanánk nyilván, hogy mi a számlaszáma, ki a tulajdonosa, mennyi az egyenlege és hogy van-e rajta hitelkeret. Amennyiben igen, akkor fedezet nélkül is kezdeményezhet utalást.

-Egyszerre többen is szeretnék használni az alkalmazást?

+Jelenleg 3 ügyintézőnk van. Ők egyszerre dolgoznak és egyszerre is használnák az alkalmazást. További bővítést bár egyelőre nem tervezünk, szeretnénk, ha legalább 5-en tudnának egyszerre dolgozni a rendszerben, hátha később mégiscsak szükség lenne rá, ne kelljen új szoftvert fejleszteni.

-Rendben érthető, hogy inkább a létszámlimittel inkább feljebb szeretnék lőni. Ha viszont többen használják szükség lesz dedikált adatbázis szerverre, amire az alkalmazás csatlakozik. Ennek beszerzése és beállítása további költségként merülhet fel, azonban a többfelhasználós működéshez

elengedhetetlen. Továbbá ezzel külön céggel kell szerződniük, mert mi csak a szoftverért felelünk, a hardveres kiépítéshez nincs megfelelő szakemberünk.

+Van már az irodában egy kisebb külön szerverszoba. Abban el tudjuk helyezni a megfelelő eszközöket és elő is tudjuk készíteni a program számára a megfelelő környezetet, amennyiben pontosan tudjuk, hogy mire van szükség.

-Konkrétan majd a specifikáció során döntünk róla, hogy milyen adatbáziskezelő rendszert lesz célszerű használni, de amint biztos lesz, értesítjük önöket, hogy el tudják indítani a beszerzési folyamatot.

+Rendben a hálózat kiépítése nekünk is idő lesz, de igyekszünk párhuzamosan a fejlesztési projekttel haladni.

-A program elkészültét követően, illetve a fejlesztés folyamán milyen tesztelési lehetőségeink lesznek? Esetleg tudjuk kérni az alkalmazottak segítségét a program tökéletesítése céljából?

+Ha szükséges tudunk teszt adatokat küldeni, illetve természetesen kollégáink is rendelkezésre állnak, amennyiben ez valóban a szoftver minőségét javítja.

-Természetesen. Nagyon is sokat tud segíteni a rendszer felhasználóinak a véleménye, észrevételei, mert végső soron nekik kell majd használni a programot. Ők tudják, hogy minek hol kézenfekvő lennie. Ezen kívül szakmai tapasztalatukkal is hozzá tudnak járulni a szoftver kifogástalan működéséhez.

+Rendben van még más tisztázni való?

-A kész termék átvételét követően még a felhasználói oktatásról eshet szó. Tudunk készíteni egy felhasználói útmutatót a programhoz, ami minden fontos funkció leírását részletesen tartalmazza. Szerintem ez elég lesz a program használatának elsajátításához, nem lesz szükség külön oktatásra.

+Amennyiben tényleg olyan egyszerű lesz a kezelés, mint ígérték valószínűleg elég lesz.

-Rendben köszönöm a segítségét. Visszamegyünk az irodába és egy héten belül elkészítjük a pontos specifikációt, ami alapján meg tudjuk majd írni a szerződést, és a hivatalos megrendelést.

Rendszerterv

A program SQL adatbázist használ az adatok tárolására. Tudja kezelni a különböző felhasználókat és a hozzájuk kapcsolódó jogosultságokat (admin, basic). Külön táblában tárolja a számlák adatait. Számlák közötti átutalásokat tud indítani. és ezeket a műveleteket naplózza. Ezen kívül készpénzes befizetést és kifizetést is lehetővé tesz.

A program indulásakor egy bejelentkezési képernyő jelenik meg, amibe az ügyintéző beírja a felhasználónevét és jelszavát. Ezt követően tudja majd használni a rendszert. A megfelelő jogosultságokkal.

Ez után megjelenik a kezdőlap, ahol a program különböző funkciói között válogathat. Az első panel a számla keresés lesz. Itt egy személy adatait megadva megkereshetjük a számláit, vagy egy számla azonosítóból lekérdezhetjük a tulajdonos adatait. A második panelen lehetőség lesz készpénzes befizetés vagy kifizetés kezdeményezésére. A harmadik blokkban két számla közti átutalásra lesz lehetőség. Ehhez ellenőrizni kell, hogy a számlán van-e elég fedezet, vagy van-e rajta hitelkeret. Amennyiben minden megfelelő megtörténik a tranzakció és mentésre kerül. Az utolsó blokkban lehetőség lesz listázni egy adott számlához tartozó tranzakciókat.

Java dióhéjban

A Java egy magas szintű objektum orientált, osztályalapú programozási nyelv, amely lehetővé teszi a programozók számára, hogy egyszer kódoljanak és azt bárhol futtassák. A Java alkalmazások bájtkódra fordulnak és a Java Virtuális gépen bárhol futtathatók (JVM).

A nyelvet James Gosling fejlesztette ki, de a Java jelenleg az Oracle tulajdonában van és a legfrissebb JDK verzió a fejlesztéshez a 19-es.

A Java nyelv a szintaxisát főleg a C és a C++ nyelvektől örökölte, viszont sokkal egyszerűbb objektummodellel rendelkezik, mint a C++. A JavaScript szintaxisa és neve hasonló ugyan a Java-hoz, de a két nyelv nem áll olyan szoros rokonságban, mint azt ezekből a hasonlóságokból gondolhatnánk.

Bár a nyelv neve kezdetben Oak (tölgyfa) volt, (James Gosling, a nyelv atyja nevezte így az írodája előtt növő tölgyfáról), később kiderült, hogy ilyen elnevezésű nyelv már létezik, ezért végül Java néven vált ismertté. A Java szó a Oracle védjegye. Ennélfogva engedélye nélkül nem használható mások által kifejlesztett termékek megjelölésére; még például Java-szerű... stb. összetételben sem, mert ez a védjegyjogosult jogaiba ütközik.

JavaFx bemutatása

A JavaFX egy szoftverplatform asztali alkalmazások , valamint gazdag webalkalmazások létrehozására és szállítására , amelyek számos eszközön futhatnak. A JavaFX támogatja az asztali számítógépeket és webböngészőket Microsoft Windows , Linux és macOS rendszeren , valamint iOS és Android rendszert futtató mobil eszközöket .

A JavaFX-et a Java SE szabványos grafikus felhasználói felületének könyvtáraként szánták a Swing helyére, de kikerült az új Standard Edition-ből, míg a Swing és az AWT továbbra is szerepel, feltehetően azért, mert a JavaFX piaci részesedését „a „mobile first” és a „web” térnyerése csökkentette. első jelentkezések.” A JDK 11 2018-as kiadásával az Oracle az OpenJFX projekt keretében a JavaFX-et az OpenJDK részévé tette, hogy növelje fejlesztési ütemét. A JavaFX Oracle támogatása a Java JDK 8-hoz is elérhető 2025 márciusáig.

A nyílt forráskódú JavaFXPorts működik iOS (iPhone és iPad) és Android rendszeren, valamint beágyazottan (Raspberry Pi); és a kapcsolódó kereskedelmi szoftver, amelyet "Gluon" néven hoztak létre, ugyanazokat a mobil platformokat támogatja további funkciókkal és asztali számítógéppel. Ez lehetővé teszi, hogy egyetlen forráskódbázis alkalmazásokat hozzon létre asztali, iOS és Android eszközökhöz.

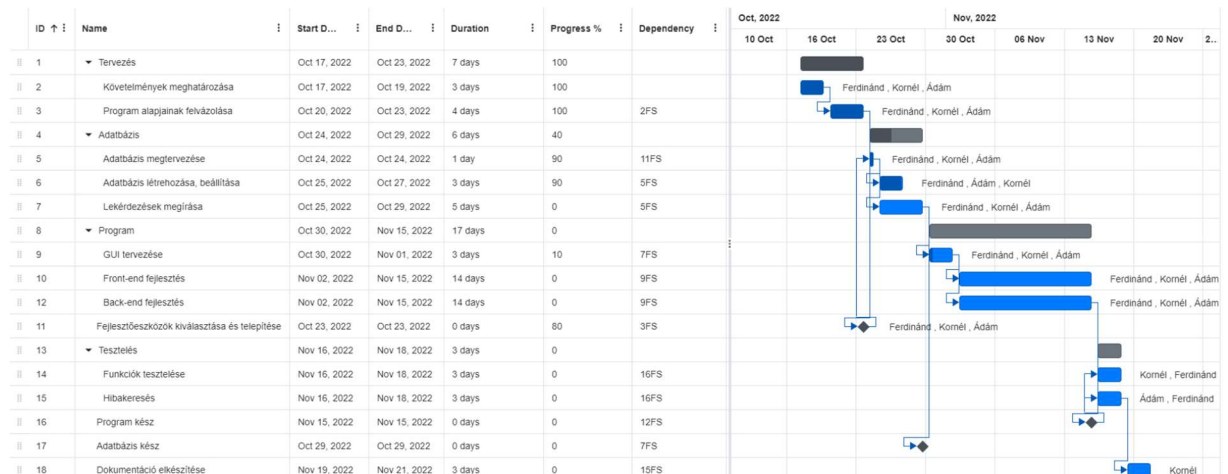
Alkalmazott technológiák

A fejlesztéshez az Eclipse IDE-t használtuk, a JavaFx grafikus felületek megalkotásához pedig a Scene Builder nevű alkalmazást.

A Gantt diagramm az Online-Gantt oldalon készült, az UML diagramok draw.io-ban, illetve a plantUml oldalon. A verziókövetéshez pedig a GitHub-ot választottuk.

A program megoldási terve

A program elkészítésekor nagyon fontos a megfontoltság és az előre tervezés. Ezért csapatunk a következő Gantt diagramm szerint haladt a megvalósítással. A határidők betartása és a mérföldkövek rendkívül fontosak.



Az SQL dióhéjban

A Structured Query Language (SQL) egy szabványos programozási nyelv, amelyet relációs adatbázisok kezelésére használnak, és különféle műveleteket hajtanak végre a bennük lévő adatokon. Az eredetileg az 1970-es években létrehozott SQL-t nemcsak adatbázis-adminisztrátorok, hanem adatintegrációs szkripteket író fejlesztők és adatelemzők is rendszeresen használják, akik elemző lekérdezéseket szeretnének beállítani és futtatni.

Az SQL-lekérdezések és egyéb műveletek utasításként írt parancsok formájában valósulnak meg, és olyan programokba vannak összesítve, amelyek lehetővé teszik a felhasználók számára, hogy adatokat adjanak hozzá, módosítsanak vagy lekérjenek adatbázistáblázatokról.

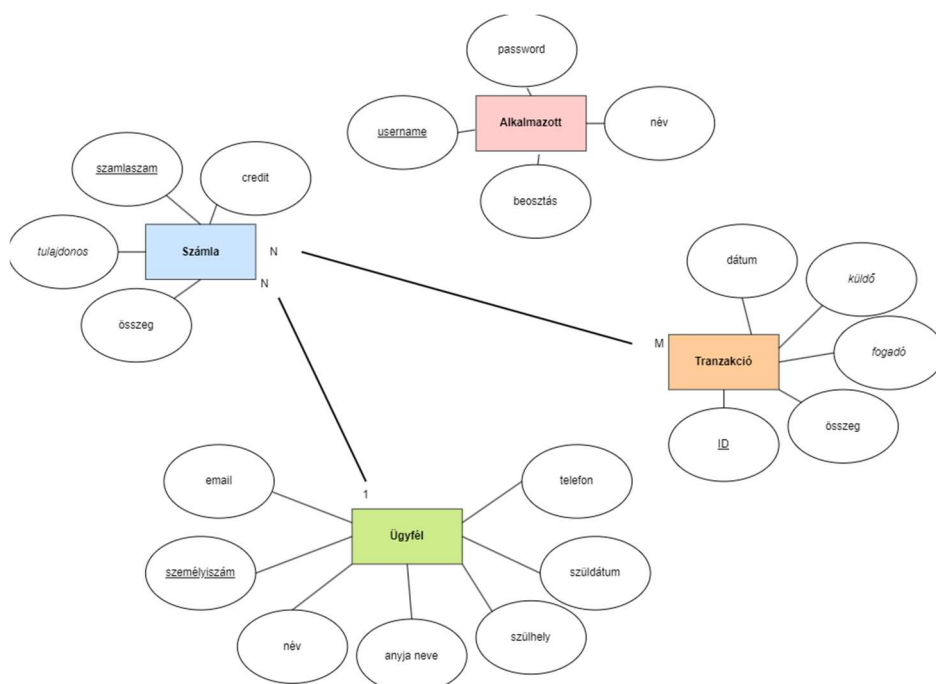
Adatbázishoz alkalmazott technológia

A Microsoft SQL Server egy relációs adatbázis-kezelő rendszer, amelyet a Microsoft fejlesztett ki. Adatbázis-szerverként ez egy szoftvertermék, amelynek elsődleges funkciója az adatok tárolása és visszakeresése más szoftveralkalmazások által kért módon – amelyek futhatnak ugyanazon a számítógépen vagy egy másik számítógépen a hálózaton (beleértve az internetet is). A Microsoft a Microsoft SQL Server legalább egy tucat különböző kiadását forgalmazza, amelyek különböző közönségeknek és munkaterhelésnek felelnek meg, a kis egygépes alkalmazásoktól a nagy, internetre néző alkalmazásokig sok egyidejű felhasználóval.

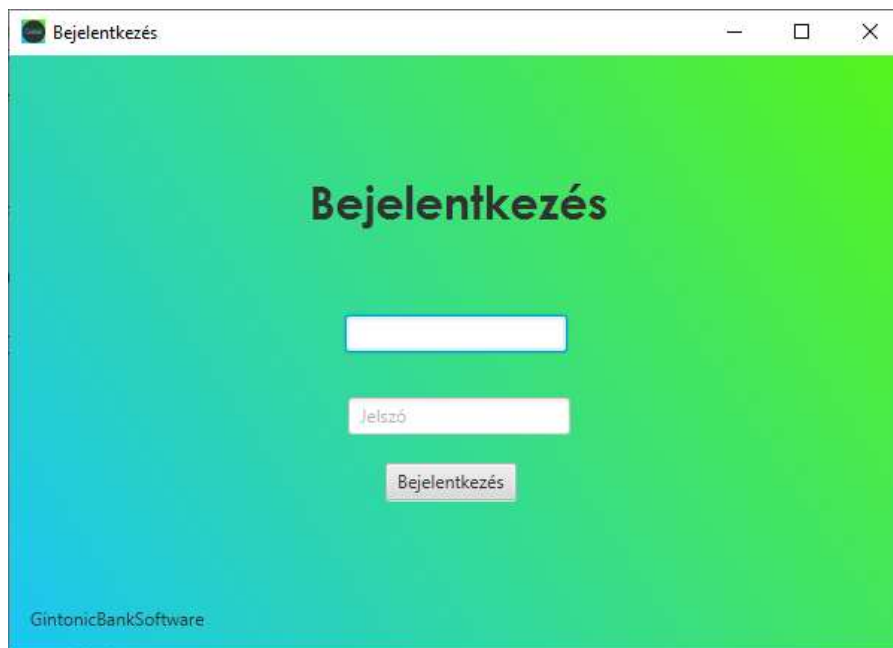
Az adatbázist ennek a segítségével hoztuk létre és kezeltük. Az SQL kód alapján generáltuk az adatbázist, amit a Java alkalmazás futásidőben képes kezelni. Az adatbázisunk természetesen támogatja a tranzakciókezelést és az egyidőben való felhasználást több kliensről.

Adatbázis felépítése

Az adatbázisunkban 4 táblánk van. Az adatbázishoz tartozó EK diagram itt alul látható.

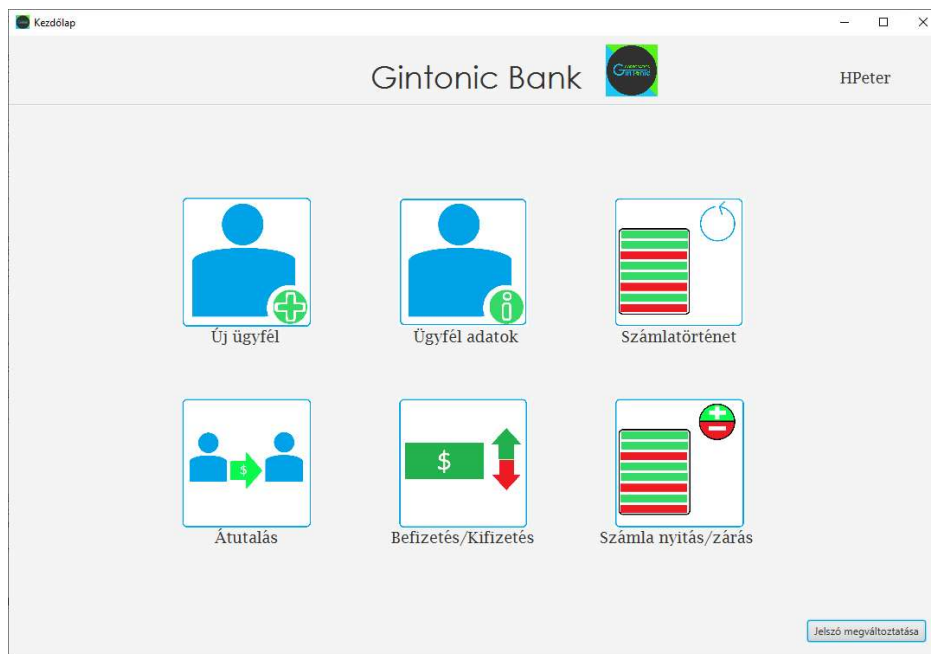


Bejelentkezés



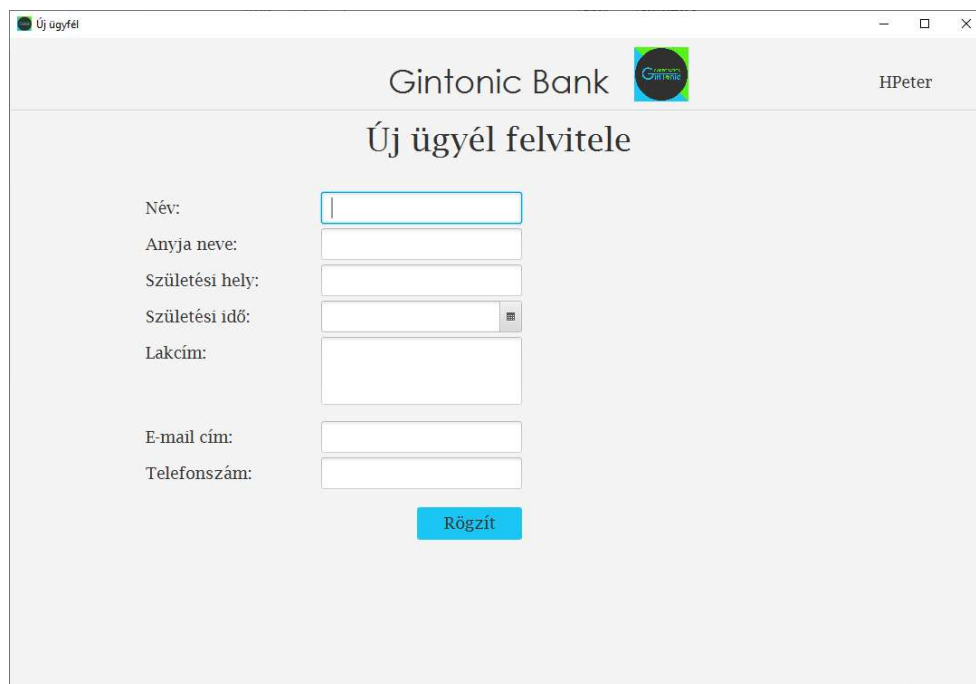
Kezdőképernyő

Sikeres bejelentkezést követően a kezdőképernyő fogad minket. Innen van lehetőségünk elérni a további hat menüpontot.



Új ügyfél

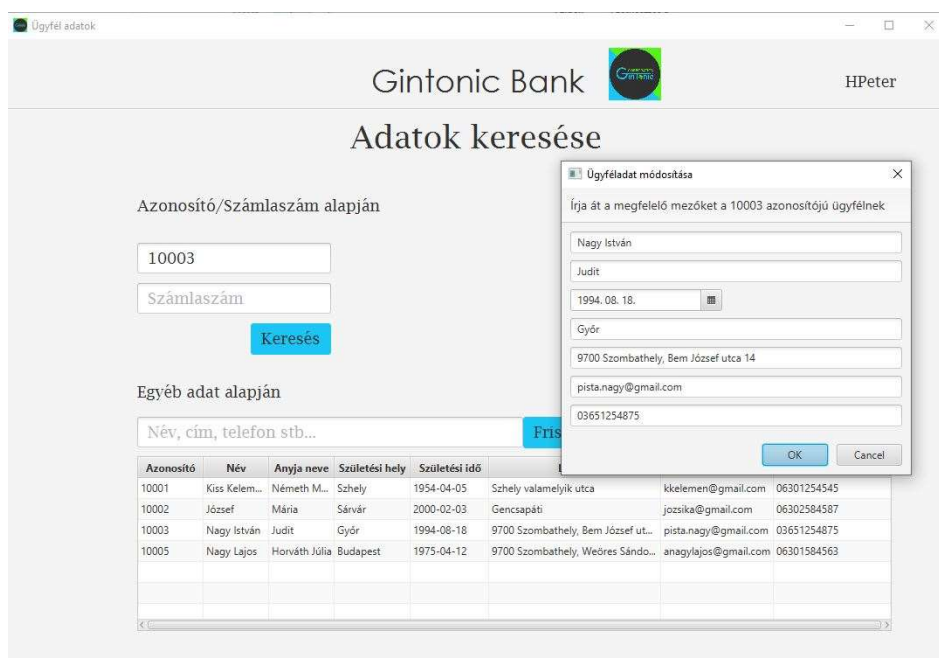
Ebben a menüpontban van lehetősége az alkalmazottnak új ügyfelet felvennie az adatbázisba.



The screenshot shows a web application window titled 'Új ügyfél'. The header includes the Gintonic Bank logo and the username 'HPeter'. The main heading is 'Új ügyfél felvitele'. Below this, there are several input fields for customer data: 'Név:', 'Anyja neve:', 'Születési hely:', 'Születési idő:', 'Lakcím:', 'E-mail cím:', and 'Telefonszám:'. A blue 'Rögzít' (Save) button is at the bottom.

Ügyfél adatok

Ebben a menüpontban az ügyfelekkel kapcsolatban futtathatunk lekérdezéseket. Azonosító és számlaszám alapján. Ha megtaláltuk, akit szerettünk volna ott lehetőségünk nyílik az adatok módosítására is.



The screenshot shows a web application window titled 'Ügyfél adatok'. The header includes the Gintonic Bank logo and the username 'HPeter'. The main heading is 'Adatok keresése'. Below this, there are two search methods: 'Azonosító/Számlaszám alapján' (by ID/Account Number) and 'Egyéb adat alapján' (by other data). The first method has input fields for '10003' and 'Számlaszám', with a blue 'Keresés' (Search) button. The second method has a text input field 'Név, cím, telefon stb...' and a blue 'Fris' (Refresh) button. Below the search options is a table with columns: 'Azonosító', 'Név', 'Anyja neve', 'Születési hely', 'Születési idő', 'Szé...', 'Gy...', 'E-mail', and 'Telef.'. The table contains five rows of data. A modal window titled 'Ügyféladat módosítása' (Modify Customer Data) is open, showing the details of the selected customer (Nagy István) and a 'Rögzít' (Save) button.


Azonosító	Név	Anyja neve	Születési hely	Születési idő	Szé...	Gy...	E-mail	Telef.
10001	Kiss Kelem...	Németh M...	Szé...	1954-04-05	Szé...		kkelemen@gmail.com	06301254545
10002	József	Mária	Sárvár	2000-02-03	Gencsapáti		jozsika@gmail.com	06302584587
10003	Nagy István	Judit	Győr	1994-08-18	9700 Szombathely, Bem József ut...		pista.nagy@gmail.com	03651254875
10005	Nagy Lajos	Hornváth Júlia	Budapest	1975-04-12	9700 Szombathely, Weöres Sándó...		anagylajos@gmail.com	06301584563

Számlatörténet

Ebben a menüpontban a számláról és a számlára történt átutalásokat tudjuk nyomon követni.

Számlatörténet

Gintonic Bank



HPeter

Számlatörténet

Keresendő számla

Lekérdezés

Dátum-tól

Küldő	Fogadó	Összeg	Dátum
1373	1281	1000	2022-11-08 17:35:39.0
1373	1281	1000	2022-11-08 17:40:21.0
1281	1373	1000	2022-11-08 17:41:13.0
1281	1373	1000	2022-11-08 17:48:23.0

Átutalás

Ebben a menüpontban az alkalmazottak egy ügyfél számlájáról tudnak pénzt átutalni egy másik számlára.

Átutalás

Gintonic Bank



HPeter

Átutalás

Küldő számlaszám

Fogadó számlaszám

Fogadó neve

Átutalni kívánt összeg

Átutalás

Befizetés/kifizetés

Ebben a menüpontban nyílik lehetőség egyenlegfeltöltésre vagy készpénzfelvételre.

Befizetés/Kifizetés

Gintonic Bank

HPeter

Készpénz befizetés/felvétel

Készpénz befizetés

Összeg

Jóváhagy

Készpénz felvétel

Számlaszám

Összeg

Jóváhagy

Számla nyitás/zárás

Ebben a menüpontban van lehetősége az alkalmazottaknak számlákat nyitni és zárni.

Számla nyitás/zárás

Gintonic Bank

HPeter

Számla ügyintézés

Számlanyitás

Ügyfél azonosító

OK

Számla lezárása

Számlaszám

OK

Számlák keresése ügyfél alapján

10005

Keresés

Számlaszám	Egyenleg
1120	12000
1189	0
1281	4000
1350	1000

Tesztelési terv

A teszteléseket a program a Tester.java fileban végzi. A programon keresztüli adatbáziselérés a tesztelés célja, hogy minden adatbázis művelet hibátlanul működjön. (INSERT, UPDATE, DELETE)

Tesztszám	Teszt típusa	Leírás
1	INSERT	új ügyfél hozzáadása
2	DELETE	legnagyobb azonosítójú ügyfél eltávolítása
3	INSERT	új admin felvétele, test névvel
4	DELETE	test admin eltávolítása
5	INSERT	új alkalmazott felvétele
6	DELETE	legnagyobb azonosítójú alkalmazott eltávolítása
7	INSERT	új fiók nyitása
8	DELETE	legnagyobb azonosítójú fiók bezárása
9	INSERT	tranzakció
10	DELETE	tranzakció visszavonása
11	UPDATE	név módosítása
12	UPDATE	név módosítása
13	UPDATE	születési hely módosítása
14	UPDATE	születési hely módosítása
15	UPDATE	születésnap módosítása
16	UPDATE	születésnap módosítása
17	UPDATE	anyja nevének módosítása
18	UPDATE	anyja nevének módosítása
19	UPDATE	lakcím frissítése
20	UPDATE	lakcím frissítése
21	UPDATE	másik email megadása

22	UPDATE	másik email megadása
23	UPDATE	telefonszám frissítése
24	UPDATE	telefonszám frissítése
25	UPDATE	másik email megadása
26	UPDATE	telefonszám frissítése
27	INSERT	új alkalmazott felvétele
28	UPDATE	alkalmazott felhasználónevének a frissítése
29	INSERT	számlanyitás
30	UPDATE	készpénzfelvétel/átutalás
31	INSERT	tranzakció rögzítése
32	INSERT	tranzakció rögzítése

Ezekután a program már csak le ellenőrzi, hogy mindenhol a feltételezett adatok szerepelnek az adatbázisban.

Java osztályok UML diagramjai

NewAccountController	
-	accAzon : javafx.scene.control.TextField
-	accID : javafc.scene.TableColumn<Data,Integer>
-	accountsTable : javafx.scene.control.TableView<Data>
-	balance : javafc.scene.TableColumn<Data,Integer>
~	con : DatabaseConnection
-	custAzon : javafx.scene.control.TextField
-	customerToAcc : javafx.scene.control.TextField
-	usernameLabel : javafx.scene.control.Label
+	createAccount() : void
+	deactivateAccount : void
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	seatchForAccounts() : void
«constructor»	NewAccountController()

NewClientController	
-	addressText : javafx.scene.control.TextArea
-	birthPlaceText : javafx.scene.control.TextField
-	birthTimeText : javafx.scene.control.DatePicker
-	conn : DatabaseConnection
-	emailText : java fx.scene.control.TextField
-	motherText : javafx.scene.control.TextField
-	nameText : javafx.scene.control.TextField
-	phonetext : javafx.scene.control.TextField
-	usernameLabel : javafx.scene.control.Label
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	insertIntoDatabase(javafx.event.ActionEvent : event : void)
«constructor»	NewClientController

QueryController	
-	bdate_Coll : javafx.scene.control.TableColumn<Customer,String>
-	bpalce_Coll : javafx.scene.control.TableColumn<Customer,String>
~	con : DatabaseConnection
-	customerTable : javafx.scene.control.TableView<Customer,String>
-	email_Coll : javafx.scene.control.TableColumn<Customer,String>
-	haddress_Coll : javafx.scene.control.TableColumn<Customer,String>
-	ID_Coll : javafx.scene.control.TableColumn<Customer,Integer>
-	mother_Coll : javafx.scene.control.TableColumn<Customer,String>
-	name_Coll : javafx.scene.control.TableColumn<Customer,String>
~	obserList : javafx.collections.ObservableList<Customer>
-	phone_Coll : javafx.scene.control.TableColumn<Customer,String>
-	searchAccount : javafx.scene.control.TextField
-	searchID : javafx.scene.control.TextField
-	searchKW : javafx.scene.control.TextField
-	usernameLabel : javafx.scene.control.Label
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	refreshList() : void
+	searchToModify() : void
«constructor»	QueryController()

SceneSwitcher	
-	<u>isadmin : boolean</u>
-	pwdModify : javafx.scene.control.Button
-	root : javafx.scene.Parent
-	scene : javafx.scene.Scene
-	stage : javafx.scene.Stage
-	<u>user : String</u>
-	userAdd : javafx.scene.control.Button
-	usernameLabel : javafx.scene.control.Label
+	addNewUser() : void
+	changePassword() : void
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	setUser(String : u, boolean : a) : void
+	switchToHandleCash() : void
+	switchToMenu(javafx.event.Event : event) : void
+	switchToNewAccount() : void
+	switchToNewClient() : void
+	switchToQuery() : void
+	switchToTransaction() : void
+	switchToTransactionLog() : void
«constructor»	SceneSwitcher()

TransactionController	
~	con : DatabaseConnection
-	receiverAcc : javafx.scene.control.TextField
-	receiverName : javafx.scene.control.TextField
-	senderAcc : javafx.scene.control.TextField
-	transferAmount : javafx.scene.control.TextField
-	usernameLabel : javafx.scene.control.Label
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	transferMoney() : void
«constructor»	TransactionController()

TransactionLogController	
-	accID :javafx.scene.control.TextField
-	amount : javafx.scene.control.TableColumn<DataOfTransacion, Integer>
~	con : DatabaseConnection
-	dateColl : javafx.scene.control.TableColumn<DataOfTransacion, String>
-	fromDate : javafx.scene.control.DatePicker
-	logTable : javafx.scene.control.TableView<DataOfTransaction>
-	receiver : javafx.scene.control.TableColumn<DataOfTransaction,Integer>
-	sender : javafx.scene.control.TableColumn<DataOfTransaction,Integer>
-	usernameLabel : javafx.scene.control.Label
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	searchTransacionLog() : void
«constructor»	TransactionLogController()

User	
-	isAdmin : boolean
-	password : String
-	realname : String
-	username : String
«constructor»	User()

Customer	
-	bdate : String
-	bplace : string
-	email : String
-	haddress : String
-	id : int
-	mother : String
-	name : String
-	phone : String
+	getBdate() : String
+	getBplace() : String
+	getEmail() : String
+	getHadress() : String
+	getId() : int
+	getMother() : String
+	getName() : String
+	getPhone() : String
+	setBdate(String : bdate) : void
+	setBplace(String : bplace) : void
+	setEmail(String : email) : void
+	setHaddress(String : haddress) : void
+	setId(int : id) : void
+	setMother(String : mother) : void
+	setName(String : name) : void
+	setPhone(String : phone) : void
«constructor»	Customer(int : id, String : name, String : bdate, String : bplace, String : mother, String : haddress, String : email, String phone)

Data	
-	D_accID : int
-	D_balance : int
+	getD_accID() : int
+	getD_balance
+	setD_accID(int : d_accID) : void
+	setD_balance(int : d_balance) : void
«constructor»	Data(int : a, int : b)

DatabaseConnection	
-	con : Connection
-	stmt : Statement
-	url : String
+	doQuery(string : SQL) : ResultSet
+	getConnection() : Connection
+	insertIntoDatabase(String : SQL) : void
+	processTransaction(ArrayList<String> : SQL) : void
+	updateDatabase(String : SQL) : void
«constructor»	DatabaseConnection()

DataOfTransaction	
~	amount : int
~	date : String
~	receiver : int
~	sender : int
+	getAmount() : int
+	getDate() : String
+	getReceiver() : int
+	getSender() : int
+	setAmount(int : amount) : void
+	setDate(String : date) : void
+	setReceiver(int : receiver) : void
+	setSender(int : sender) : void
«constructor»	DataOfTransaction(int : sender,
	int : receiver, int : amount,
	String : date)

LoginController	
-	loginAnchorPane : javafx.scene.control.AnchorPane
-	loginButton : javafx.scene.control.Button
-	passwd : javafx.scene.control.PasswordField
-	sw : SceneSwitcher
-	username : javafx.scene.control.TextField
+	enterButton(javafx.scene.input.KeyEvent : e) : void
+	initialize(URL :arg0, ResourceBundle : arg1) : void
+	loginAttempt(javafx.event.Event : e) : void
+	passwordEncryption(String : pass) : String
«constructor»	LoginController()

Main	
+	main(String[] : args) : void
+	start(javafx.stage.Stage : stage)
«constructor»	Main()

Tartalom

A program és a projekt rövid bemutatása	2
Program specifikáció	2
Jegyzőkönyv a banki alkalmazás specifikálásáról	3
Rendszerterv	5
Java dióhéjban	5
JavaFx bemutatása	5
Alkalmazott technológiák	6
A program megoldási terve	6
Az SQL dióhéjban	7
Adatbázishoz alkalmazott technológia	7
Adatbázis felépítése	7
Bejelentkezés	8
Kezdőképernyő	8
Új ügyfél	9
Ügyfél adatok	9
Számlatörténet	10
Átutalás	10
Befizetés/kifizetés	11
Számla nyitás/zárás	11
Tesztelési terv	12
Java osztályok UML diagramjai	14