

Gymnázium, Praha 6, Arabská 14

Obor programování

Ročníková práce



2023

Timon Eiselt

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníková práce

Předmět : Programování

Téma : aplikace imitující internetové bankovníctví

Autor : Timon Eiselt

Třída : 2.E

Školní rok : 2022/2023

Vyučující : Mgr. Jan Lána

Třídní učitel : Mgr. Blanka Hniličková

Poděkování :

Rád bych zde na tomto místě poděkoval panu Mgr. Janu Lánovi za vedení tohoto projektu a RNDr. Mojmíru Adamcovi za cenné rady, nápady a pomoc, když jsem si při zpracování této práce nevěděl rady. Často jsem s ním konzultoval všechny problémy, které v mé práci nastaly a několikrát mi jeho pohled na věc pomohl a problém jsem byl schopen

vyřešit. Jsem jim velice vděčný za pomoc a podporu v průběhu mého projektu.

Čestné prohlášení:

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použítá literatura a další zdroje jsou v práci uvedené.

Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne

.....

Anotace :

Výsledkem tohoto ročníkového projektu je aplikace, která bude imitovat nejlépe a co nejpřesněji internetové bankovníctví. Aplikace by měla mít obdobné funkce, kterým se nám dostává i v klasickém internetovém bankovníctví. Po dobu zpracovávání tohoto projektu jsem využíval programovacího jazyku Java, CSS, softwarové platformy JavaFX a SQL pro jednoduchou práci s databází pro ukládání údajů.

Abstract (English) :

The result of this annual project is an application that will mimic internet banking as closely and accurately as possible. The application should have similar functions to those we get in the classic internet banking. While working on this project, I have used Java programming language, CSS, JavaFX software platform and SQL to easily work with the database to store data.

Zadání Ročníkového Projektu

Vytvoření aplikace internetového bankovníctví.

Upřesnění zadání :

- Aplikace nám umožní používat internetové bankovníctví bez toho, aby jsme museli používat reálné peníze. Hra může být také například pro děti 1.stupně základních škol, aby se trochu přiučili jak se zachází s penězi, které jsou na bankovním účtě.
- Po spuštění aplikace se objeví přihlašovací okno, které Vám nabídne možnost se přihlásit jako admin (správce všech účtů) nebo klient. Pro vytvoření klienta se daná osoba bude muset přihlásit jako admin (jeho přihlašovací údaje jsou předem dané a uložené v databázi). Osoba si pod adminem vytvoří profil klienta, bude automaticky vygenerována uživatelská adresa klienta, pod kterou se bude od teď do aplikace přihlašovat. Po přihlášení se zobrazí hlavní bankovní přehled. Klient má poté možnost provádět transakce mezi různými účty, převést si peníze z běžného na spořicí účet nebo naopak.

Platformy :

- Java, JavaFX, CSS, SQL

Obsah

1. Úvod

2. Struktura projektu v IntelliJ Ultimate

3. GUI - grafické uživatelské rozhraní

3.1. třída “OvladaniPrihlaseni.java”

3.2. třída “Model.java”

4. CSS

4.1. Ukázka CSS kódu ve třídě “Prihlaseni.css”

4.2. Rozdíl FXML souboru s a bez použití CSS stylingu

5. SQL databáze

5.1. Třída pracující s SQL databází : “OvladacDatabaze.java”

5.1.1. metoda “davajDataKlienta”

5.1.2. metoda "davajTransakce"

5.1.3. metoda "davajZustatekSporicihoUctu"

5.1.4. metoda "aktualizujZustatek"

5.1.5. metoda “novaTransakce ”

5.1.6. metoda “davajAdminData ”

5.1.7. metoda “vytvorKlienta ”

5.1.8. metoda “vytvorBeznyUcet ”

5.1.9. metoda “vytvorSporiciUcet ”

5.1.10. metoda “davajDataVsechKlientu”

5.1.11. metoda “ulozitUspory”

5.1.12. metoda “hledejKlienta”

5.1.13. metoda “davajIdPoslednihoKlienta”

5.1.14. metoda “davajDataSporicihoUctu”

5.1.15. metoda “davajDataSporiciUcet”

6. Závěr

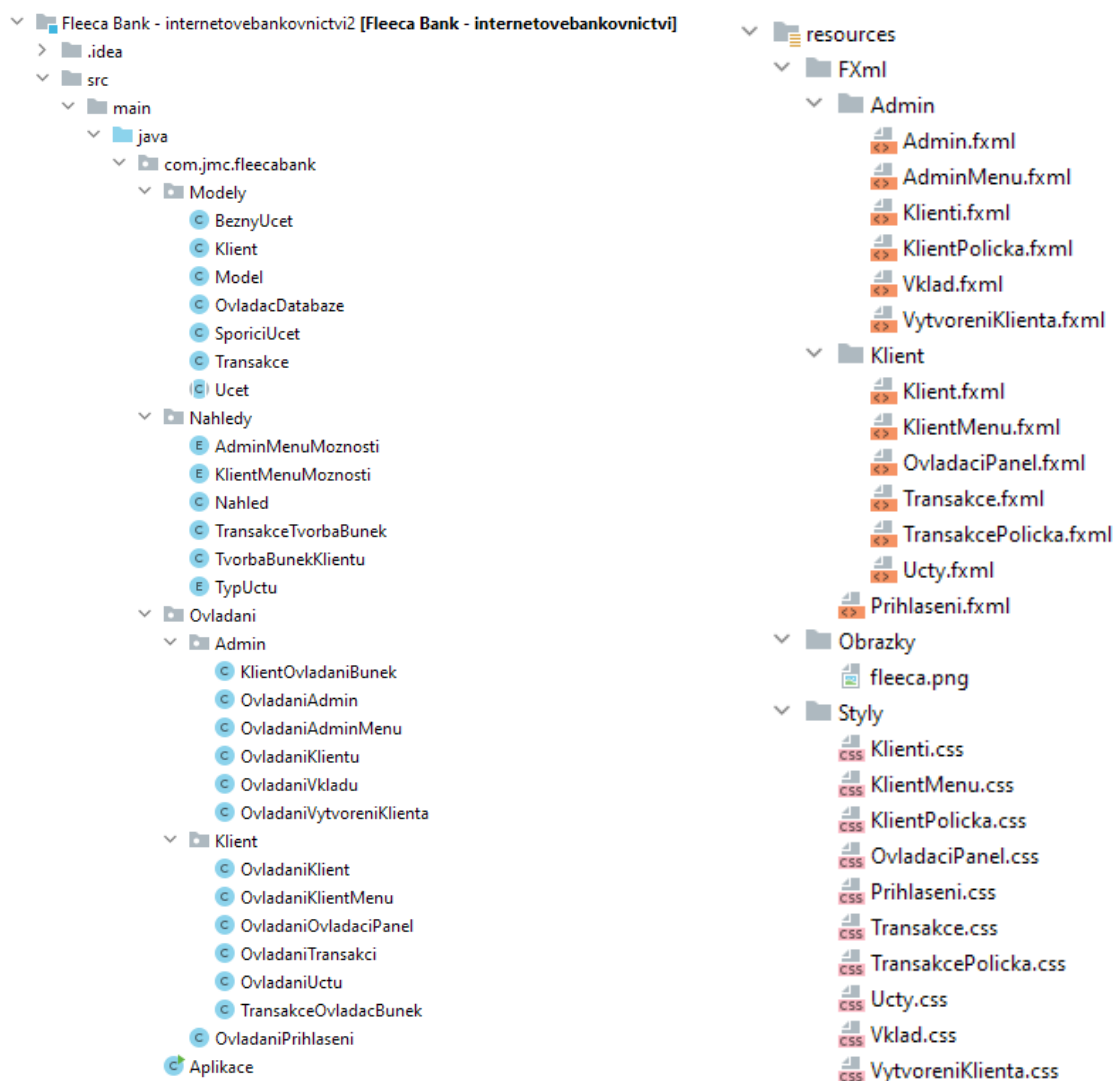
7. Zdroje dokumentace

8. Zdroje při vypracovávání projektu

1. Úvod

Předmětem tohoto ročníkového projektu bylo vypracovat aplikaci, která bude fungovat jako replika internetového bankovníctví. V aplikaci jsou dvě možnosti pod kým by se mohl daný uživatel přihlásit : Admin, Klient. Aplikace má několik funkcí jako jsou: vytvoření klienta pod adminem, vygenerování uživatelské adresy a její uložení do databáze, včetně všech zadaných údajů o klientovi. Klient má poté přístup do svého internetového bankovníctví, kde má předem určitý obnos peněz, jak na běžném, tak na spořicímu účtu. Peníze může rozesílat a přijímat od dalších klientů, kteří byli adminem vytvořeni a nacházejí se v databázi. Je zde možnost si zobrazit všechny poslední a provedené transakce a informace o nich. Všechny převody jsou automaticky zaznamenány do SQL databáze, takže zůstatek na účtě se po odeslání či příjmu peněz hned aktualizuje.

K vytvoření celého programu jsem využíval vývojového prostředí IntelliJ IDEA Ultimate, za použití programovacího jazyku Java, CSS, aplikace Gluon Scene Builder pro JavaFX a DB Browser pro zobrazování a upravování SQL databáze.

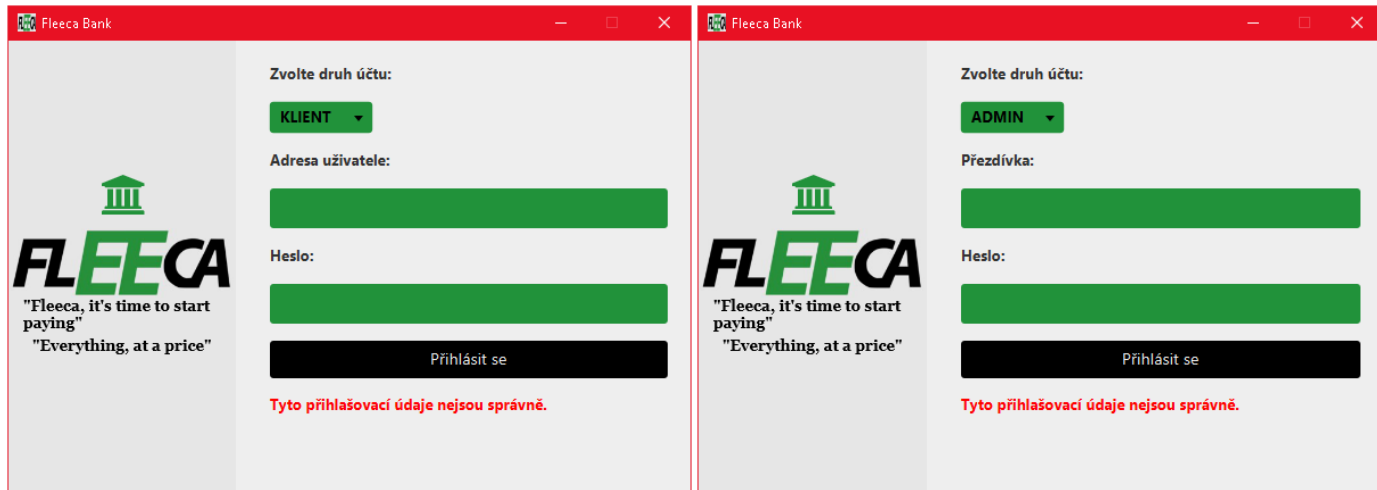


2. Struktura projektu v IntelliJ Ultimate

Obrázek č.1

Toto je struktura mého Java projektu. Projekt má dvě základní složky “java” a “resources”. Ve složce “java” se nachází další tři složky a jedna třída : “Modely”, “Nahledy”, “Ovladani” a třída “Aplikace”, která se stará o spuštění celého projektu. Ve složce “resources” se nachází složky “FXML”, “Obrazky” a “Styly” v každé z nich se nacházejí jim odpovídající třídy.

3. GUI - Grafické uživatelské rozhraní



přihlašovacího okna

Zde v této kapitole se pokusím popsat, alespoň část tříd a funkcí, které se v pozadí odehrávají během toho co je moje aplikace bankovníctví spuštěná.

Obrázek č.2

Obrázek č.3

3.1. třída “OvladaniPrihlaseni.java”

```
private void behemPrihlaseni() {
    Stage stejdz = (Stage) error_stitek.getScene().getWindow();
    if (Model.davajPriklad().davajNahled().davajPrihlaseniTypUctu() == TypUctu.KLIENT){
        //Vyhodnocení přihlašovacích údajů klienta
        Model.davajPriklad().vyhodnotKlientUdaje(adresaUzivatele_pole.getText(), heslo_pole.getText());
        if (Model.davajPriklad().davajIndikatorUspesnehoPrihlaseniKlienta()){
            Model.davajPriklad().davajNahled().ukazKlientOkno();
            // Zavření přihlašovací stage
            Model.davajPriklad().davajNahled().zavirajStejdz(stejdz);
        } else {
            adresaUzivatele_pole.setText("");
            heslo_pole.setText("");
            error_stitek.setText("Tyto přihlašovací údaje nejsou správně.");
        }
    } else {
        // Vyhodnocení přihlašovacích údajů admina
        Model.davajPriklad().vyhodnotAdminUdaje(adresaUzivatele_pole.getText(), heslo_pole.getText());
        if (Model.davajPriklad().davajIndikatorUspesnehoPrihlaseniAdmina()){
            Model.davajPriklad().davajNahled().ukazAdminOkno();
            // Zavření přihlašovací stage
            Model.davajPriklad().davajNahled().zavirajStejdz(stejdz);
        } else {
            adresaUzivatele_pole.setText("");
            heslo_pole.setText("");
            error_stitek.setText("Tyto přihlašovací údaje nejsou správně.");
        }
    }
}
```

```
}  
}
```

Tento kód obsahuje dvě metody, které se zavolají v určitých situacích při přihlášení uživatele do internetového bankovníctví.

Metoda "behemPrihlaseni()" se spouští při přihlašování uživatele a provádí následující akce:

- 1) Získání odkazu na aktuální stage (okno).
- 2) Vyhodnocení typu účtu (klient nebo admin).
- 3) Pokud je typ účtu klient, ověří se přihlašovací údaje klienta. Pokud jsou údaje správné, otevře se okno klienta a přihlašovací stage se zavře.
- 4) Pokud jsou přihlašovací údaje klienta nesprávné, pole pro přihlašovací údaje se vyprázdní a zobrazí se chybové hlášení.
- 5) Pokud je typ účtu admin, ověří se přihlašovací údaje admina. Pokud jsou údaje správné, otevře se okno admina a přihlašovací stage se zavře.
- 6) Pokud jsou přihlašovací údaje admina nesprávné, pole pro přihlašovací údaje se vyprázdní a zobrazí se chybové hlášení.

Metoda "nastavVolbuUctu()" se spouští při změně výběru typu účtu a provádí následující akce:

- 1) Nastaví typ účtu v datovém modelu aplikace na základě výběru uživatele.
- 2) Pokud byl vybrán typ účtu "admin", změní se popis pole pro adresu příjemce na "Přezdívka". V opačném případě se změní na "Adresa uživatele".

3.2. třída "Model.java"

```
public void vyhodnotKlientUdaje(String adresa, String heslo) {  
    BeznyUcet beznyUcet;  
    SporiciUcet sporiciUcet;  
    ResultSet vysledek = ovladacDatabase.davajDataKlienta(adresa, heslo);  
    try {  
        if (vysledek.isBeforeFirst()) {  
            this.klient.krestniA().set(vysledek.getString("Krestni"));  
            this.klient.prijmeniA().set(vysledek.getString("Prijmeni"));  
  
            this.klient.adresaPrijemceA().set(vysledek.getString("UzivatelaskaAdresa"));  
            String[] datumA = vysledek.getString("Datum").split("-");  
            LocalDate datum = LocalDate.of(Integer.parseInt(datumA[0]),  
Integer.parseInt(datumA[1]), Integer.parseInt(datumA[2]));  
            this.klient.datumA().set(datum);  
            beznyUcet = davajBeznyUcet(adresa);  
            sporiciUcet = davajSporiciUcet(adresa);  
            this.klient.beznyUcetA().set(beznyUcet);  
            this.klient.sporiciUcetA().set(sporiciUcet);  
            this.indikatorUspesnehoPrihlaseniKlienta = true;  
        }  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

```
}  
}
```

Tento kód se nachází v metodě `vyhodnotKlientUdaje` a slouží k získání dat klienta z databáze na základě jeho adresy a hesla. Metoda začíná definicí dvou proměnných pro uchování informací o běžném a spořicím účtu klienta. Poté se pomocí metody `davajDataKlienta` získají data klienta z databáze a uloží se do `ResultSet` objektu `vysledek`. Následně se zkontroluje, zda `vysledek` obsahuje alespoň jeden řádek dat, a pokud ano, uloží se informace o klientovi do příslušných proměnných, včetně informací o běžném a spořicím účtu. Nakonec se nastaví `indikatorUspesnehoPrihlaseniKlienta` na `true`, pokud bylo úspěšně nalezeno spojení mezi adresou a heslem klienta v databázi.

```
public void vyhodnotAdminUdaje(String uzivatelskeJmeno, String heslo) {  
    ResultSet nastaveniVysledku =  
    ovladacDatabaze.davajAdminData(uzivatelskeJmeno, heslo);  
    try {  
        if (nastaveniVysledku.isBeforeFirst()) {  
            this.indikatorUspesnehoPrihlaseniAdmina = true;  
        }  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

Tento kód definuje metodu "`vyhodnotAdminUdaje`", která přijímá jako vstupní parametry uživatelské jméno a heslo. Metoda získává výsledky dotazu na databázi, které obsahují informace o přihlášeném administrátorovi. Pokud jsou výsledky k dispozici, nastaví indikátor úspěšného přihlášení na hodnotu "`true`". Pokud nastane nějaká chyba při práci s výsledky dotazu, vypíše se chybová hláška na standardní výstup.

4. CSS

V aplikaci JavaFX je možné používat kaskádové styly, již zmiňované CSS, ve shrnutí slouží pro definování vzhledu grafických prvků. To znamená, že můžete definovat celou sadu stylů pro různé prvky a tyto styly se budou automaticky aplikovat na prvky, které mají přiřazenou odpovídající třídu stylu.

4.1. Ukázka CSS kódu ve třídě “Prihlaseni.css”

Zde příkládám ukázkou třídy Prihlaseni.css, která se stará o stylistiku přihlašovací strany, která je na další straně k vidění.

```
.prihlaseniKomplet {
    -fx-background-color: #EEEEEE;
}
.prihlaseniLogoKomplet {
    -fx-background-color: #e6e6e6;
    -fx-alignment: center;
}
.prihlaseniLogoKomplet Text {
    -fx-fill: #000000;
    -fx-font-size: 1.1em;
}
.prihlaseniLogoKomplet FontAwesomeIconView {
    -fx-fill: #21923a;
}
.vyberUctu {
    -fx-background-color: #21923a;
}
.vyberUctu .label {
    -fx-text-fill: #000000;
    -fx-font-size: 1.1em;
}
.vyberUctu .arrow {
    -fx-background-color: #000000;
}
#vyberUctuMenu {
    -fx-background-color: #000000;
}
#textVyberu {
    -fx-font-weight: 800;
}

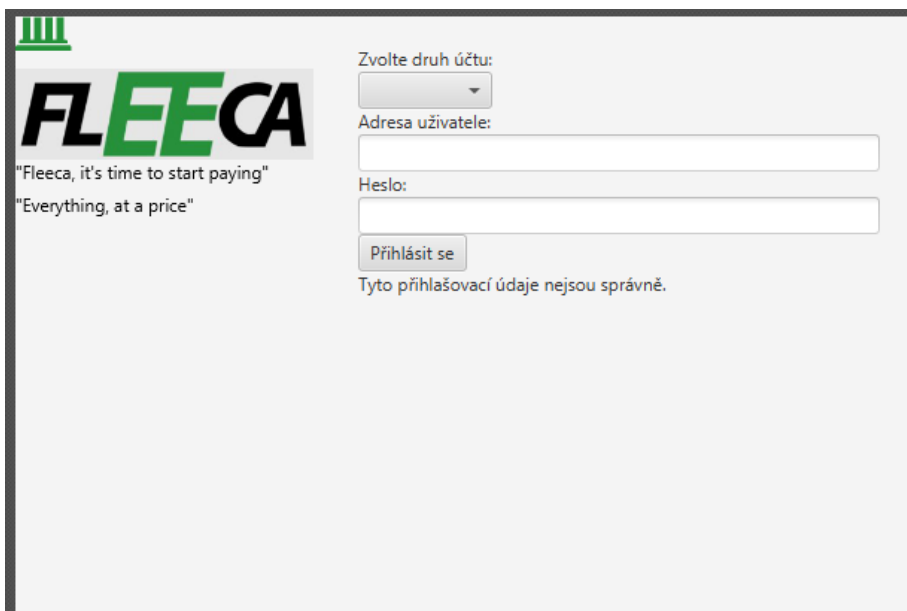
.prihlaseni {
    -fx-spacing: 15;
    -fx-alignment: top_left;
}
.prihlaseni Label {
    -fx-font-size: 1.1em;
    -fx-font-weight: bold;
}
.vstup {
    -fx-background-color: #21923a;
    -fx-pref-height: 35;
    -fx-font-size: 1.2em;
}
.prihlaseni Button {
    -fx-background-color: #000000;
    -fx-text-fill: #EEEEEE;
    -fx-font-size: 1.2em;
    -fx-pref-width: 350;
    -fx-pref-height: 30;
}
.prihlaseni Button:hover {
    -fx-cursor: hand;
}
.error {
    -fx-text-fill: #FF0000;
}
```

Obrázek 3 - třída Prihlaseni.css

Ukázka tohoto CSS kódu definuje hned několik stylistických vlastností pro grafické rozhraní v JavaFX. První pravidlo s třídou “prihlaseniKomplet” nastavuje barvu pozadí celého přihlašovacího okna na šedivou barvu. Třída “prihlaseniLogoKomplet” definuje pozadí, barvu textu a zarovnání pro horní část přihlašovací stránky, kde se nachází logo, název banky. Poté stejná stylová

třída určí barvu a velikost textu a barvu textu pro název aplikace a ikony. “.vyberUctu” definuje pozadí a barvu pro část stránky, label a šipku, kde si uživatel vybírá mezi typy účtů. Třída “.prihlaseni” definuje vzhled pro část stránky, kde se uživatel přihlašuje (její zarovnání do stran, výšku mezery mezi prvky). “.vstup” určuje pozadí výšku pole pro zadávání přihlašovacích údajů (uživatelská adresa a heslo). Poslední třída “.error” určuje jen barvu textu pro zobrazení nesprávných přihlašovacích údajů.

4.2. Rozdíl FXML souboru s a bez použití CSS stylingu

The image shows a web form for logging into Fleeca. On the left, there is a logo consisting of a green building icon above the word "FLEECA" in green and black, with the tagline "Fleeca, it's time to start paying" and "Everything, at a price" below it. To the right of the logo, the form has a label "Zvolte druh účtu:" followed by a small dropdown menu. Below that is a label "Adresa uživatele:" followed by a text input field. Then a label "Heslo:" followed by another text input field. A button labeled "Přihlásit se" is below the password field. At the bottom, there is a message "Tyto přihlašovací údaje nejsou správně." in a small font. The entire form is set against a plain white background with no styling.

Obrázek 4 -
Prihlaseni.fxml bez CSS

The image shows the same Fleeca login form as in the previous image, but with CSS styling applied. The logo and tagline are now styled with a green and black color scheme. The "Zvolte druh účtu:" dropdown menu is now a green button with a white arrow. The "Adresa uživatele:" and "Heslo:" text input fields are now green. The "Přihlásit se" button is now a black button with white text. The error message "Tyto přihlašovací údaje nejsou správně." is now in red text. The background of the form is a light gray color.

Obrázek 5 -
Prihlaseni.fxml s CSS

5. SQL Databáze

SQL (Structured Query Language) je jazyk určený pro práci s relačními databázemi. SQL umožňuje vytváření, modifikaci a správu relačních databází. Je to standardizovaný jazyk, který je podporován většinou databázových systémů. SQL umožňuje provádět mnoho operací s daty, jako například: vytváření tabulek pro ukládání dat, vkládání, průběžná aktualizace a mazání dat v tabulkách, vykonování dotazů pro získání dat z tabulek..

Admini

ID

INTEGER

UzivatelскеJmeno

TEXT

Heslo

TEXT

BezneUcty

ID

INTEGER

Vlastnik

TEXT

CisloUctu

TEXT

LimitTransakce

REAL

Zustatek

REAL

Klienti

ID

INTEGER

Krestni

TEXT

Prijmeni

TEXT

UzivatelскеAdresa

TEXT

Heslo

TEXT

Datum

TEXT

SporiciUcty

ID

INTEGER

Vlastnik

TEXT

CisloUctu

TEXT

LimitVyberu

REAL

Zustatek

REAL

Transakce

ID

INTEGER

Odesilatel

TEXT

Prijemce

TEXT

Castka

REAL

Datum

TEXT

Zprava

TEXT

sqlite_sequence

name

seq

ID	UzivatelскеJmeno	Heslo
Filtr	Filtr	Filtr
1	Admin	Admin

ID	Vlastnik	CisloUctu	LimitTransakce	Zustatek
Filtr	Filtr	Filtr	Filtr	Filtr
1	@bBobbins1	5741 10285	10.0	2500.0

ID	Krestni	Prijmeni	UzivatelскеAdresa	Heslo	Datum
Filtr	Filtr	Filtr	Filtr	Filtr	Filtr
1	Bob	Bobbins	@bBobbins1	macka	2023-04-23

ID	Vlastnik	CisloUctu	LimitVyberu	Zustatek
Filtr	Filtr	Filtr	Filtr	Filtr
1	@bBobbins1	5741 2212	2000.0	120000.0

ID	Odesilatel	Prijemce	Castka	Datum	Zprava
Filtr	Filtr	Filtr	Filtr	Filtr	Filtr
1	@bBobbins1	@bBobbins7	1000.0	2023-04-24	ahoj
2	@bBobbins1	@bBobbins7	1000.0	2023-04-24	cau
3	@bBobbins1	@bBobbins7	1500.0	2023-04-24	zdar
4	@bBobbins1	@bBobbins1	10000.0	2023-04-26	bob

name	seq
Filtr	Filtr
1 Transakce	4
2 SporiciUcty	1
3 Klienti	2
4 BezneUcty	1
5 Admini	1

Obrázek 6 - databáze SQL

Obrázek 6 - databáze SQL

Databáze je navržena jako SQL databáze, což znamená, že data jsou organizována do tabulek a používají se dotazy SQL pro manipulaci s daty. Každý záznam v databázi má svůj jednoznačný identifikátor, takže může být snadno vyhledán a aktualizován.

Tato databáze je klíčovou součástí mé aplikace, která slouží jako replika internetového bankovníctví. V databázi jsou ukládány veškeré informace : o adminovi/adminech, o klientech a jejich účtech, včetně převodů peněz mezi klienty a jako poslední také počet, kolik ID je uložených z každé tabulky.

Aplikace obsahuje dvě možnosti přihlášení: klient a admin. Admin má přístup ke všem informacím v databázi, může vytvářet nové klienty, účty pro klienty a vkládat klientům peníze na spořicí účet. Po vytvoření klienta jsou jeho osobní údaje(ID, jméno, příjmení, heslo, datum) včetně i automaticky vygenerované uživatelské adresy uloženy v databázi.

Každý klient má přístup do svého internetového bankovníctví, kde může vidět informace o svém účtu, jako je zůstatek, transakční historie, příjmy, výdaje a další informace. Klienti mohou provádět převody peněz mezi svými účty a účty jiných klientů, kteří jsou také zaregistrováni v databázi. Všechny převody jsou automaticky zaznamenány do databáze, takže zůstatek na účtu se okamžitě aktualizuje.

5.1. Třída pracující s SQL databází : OvladacDatabaze.java

Tato Java třída slouží k připojení k databázi a manipulaci s daty v ní uloženými. Základní funkcí třídy je konstruktor, který se připojí k databázi a inicializuje proměnnou "pripojeni". Všechny metody třídy pracují s touto proměnnou, která představuje otevřené připojení k databázi.

5.1.1. metoda "davajDataKlienta"

Vrátí ResultSet obsahující data klienta podle adresy příjemce a hesla. Metoda provede SELECT příkaz na tabulku "Klienti" s podmínkou, že hodnota sloupce "UzivatelaskaAdresa" musí být rovna zadané adrese a hodnota sloupce "Heslo" musí být rovna zadanému heslu. Výsledkem je ResultSet obsahující data o klientovi, který splňuje podmínku.

5.1.2. metoda "davajTransakce"

Vrátí ResultSet obsahující transakce spojené s danou adresou a počtem. Metoda provede SELECT příkaz na tabulku "Transakce" s podmínkou, že hodnota sloupce "Odesilatel" nebo "Prijemce" musí být rovna zadané adrese. Metoda také omezuje počet vrácených záznamů pomocí parametru "limit". Výsledkem je ResultSet obsahující transakce, které splňují podmínku.

5.1.3. metoda "davajZustatekSPoricihoUctu"

Vrátí zůstatek úspor klienta na základě jeho adresy. Metoda provede SELECT příkaz na tabulku "SporiciUcty" s podmínkou, že hodnota sloupce "Vlastnik" musí být rovna zadané adrese. Výsledkem je ResultSet obsahující informace o úsporách klienta, včetně aktuálního zůstatku.

5.1.4. metoda "aktualizujZustatek"

Aktualizuje zůstatek úspor klienta na základě jeho adresy a částky transakce. Metoda nejprve provede SELECT příkaz na tabulku "SporiciUcty" s podmínkou, že hodnota sloupce "Vlastnik" musí být rovna zadané adrese. Poté zkontroluje parametr "operace" a provede buď přičtení nebo odečtení částky transakce k aktuálnímu zůstatku. Výsledek je uložen zpět do databáze pomocí UPDATE příkazu.

5.1.5. metoda "novaTransakce"

Slouží k vytvoření nové transakce, tzn. vložení záznamu o převodu peněz mezi dvěma účty. Metoda přijímá následující argumenty: odesilatel (kdo posílá peníze), prijemce (komu posílá peníze), castka (kolik peněz se převádí) a zprava (volitelná zpráva k transakci). Metoda vytváří SQL příkaz pro vložení nového záznamu do tabulky "Transakce" a provede ho.

5.1.6. metoda "davajAdminData"

Slouží k získání údajů o administrátorovi na základě uživatelského jména a hesla. Metoda přijímá následující argumenty: uzivatskeJmeno (uživatelské jméno administrátora) a heslo (heslo administrátora). Metoda vytváří SQL příkaz pro výběr řádku z tabulky "Admini", kde se shoduje uživatelské jméno a heslo, a vrátí výsledky ve formě ResultSet.

5.1.7. metoda “vytvorKlienta”

Slouží k vytvoření nového klienta a vložení záznamu o něm do databáze. Metoda přijímá následující argumenty: krestni (křestní jméno klienta), prijmeni (příjmení klienta), uzivatelskaAdresa (uživatelská adresa, tj. e-mailová adresa nebo jméno v systému), heslo (heslo klienta) a datum (datum registrace klienta). Metoda vytváří SQL příkaz pro vložení nového řádku do tabulky "Klienti" a provede ho.

5.1.8. metoda “vytvortBeznyUcet”

Slouží k vytvoření nového běžného účtu a vložení záznamu o něm do databáze. Metoda přijímá následující argumenty: vlastnik (kdo je majitelem účtu), cislo (číslo účtu), limitTransakce (limit transakcí na účtu) a zustatek (počáteční zůstatek na účtu). Metoda vytváří SQL příkaz pro vložení nového řádku do tabulky "BezneUcty" a provede ho.

5.1.9. metoda “vytvorSporiciUcet”

Slouží k vytvoření nového spořicího účtu pro klienta banky. Při volání této metody je potřeba předat jako parametry jméno klienta, výchozí zůstatek a úrokovou sazbu. Metoda pak vytvoří nový objekt třídy SpořicíÚčet a inicializuje ho zadanými hodnotami.

5.1.10. metoda “davajDataVsechKlientu”

Vrací ResultSet obsahující všechny záznamy klientů v databázi. Metoda vytváří SQL příkaz pro výběr všech záznamů z tabulky "Klienti" a poté ho provede.

5.1.11. metoda “ulozitUspory”

Aktualizuje zůstatek na spořicím účtu daného uživatele. Metoda vytváří SQL příkaz pro aktualizaci zůstatku na spořicím účtu a poté ho provede.

5.1.12. metoda “hledejKlienta”

Vrací ResultSet obsahující informace o klientovi s daným uživatelským jménem. Metoda vytváří SQL příkaz pro výběr informací o klientovi s daným uživatelským jménem a poté ho provede.

5.1.13. metoda “davajIdPoslednihoKlienta”

Získává ID posledního klienta v databázi. Metoda používá SQL dotaz k vyhledání posledního řádku v tabulce "Klienti".

5.1.14. metoda “davajDataSporicihoUctu”

Získává data o spořicímu účtu pro daného klienta.

5.1.15. metoda “davajDataSporicihoUcet”

Vrací výsledky dotazu na databázi, který vyhledává všechny záznamy v tabulce "SporiciUcty", které mají vlastníka shodného s hodnotou "uzivatelskaAdresa". Všechny metody pracují s proměnnou "pripojeni", která představuje otevřené připojení k databázi.

6. Závěr

Závěrem bych chtěl shrnout, že v rámci tohoto ročníkového projektu jsem poměrně úspěšně vytvořil aplikaci, která imituje funkce internetového bankovníctví. Tato aplikace umožňuje uživatelům vytvořit si účet, přihlásit se, zobrazovat své údaje o bankovním účtu a provádět různé bankovní transakce s ostatními účty. Řekl bych, že projekt dopadl vcelku podle mých představ, až na několik skoro desítek komplikovaných chvil, které během jeho programování nastaly.

Rozhodně bych se nebál říci, že tento projekt byl pro mne velice přínosný, naučil mě mnoha novým věcem, které jsem dříve vůbec neuměl. Kupříkladu dokáži už upravovat vzhled FXML souborů zapomocí jazyku CSS a také pracovat s tabulkovými databázemi SQL, sice jenom na velmi základní úrovni, ale během zpracování projektu to stačilo.

7. Zdroje dokumentace

<https://www.dotnetportal.cz/clanek/50/Uvod-do-jazyka-SQL>

<https://www.interval.cz/clanky/databaze-a-jazyk-sql/>

https://www.tutorialspoint.com/javafx/javafx_css.htm

8. Zdroje při vypracovávání projektu

CSS:

<https://www.w3schools.com/css/>

<https://www.youtube.com/watch?v=1qVEuRhX27Q>

<https://www.section.io/engineering-education/add-an-external-css-file-to-a-javafx-application/>

https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm

<https://www.section.io/engineering-education/add-an-external-css-file-to-a-javafx-application/#ways-of-styling-a-javafx-application>

<https://www.youtube.com/watch?v=bvEbfqbfm4I>

http://www.java2s.com/Tutorials/Java/JavaFX/1300_JavaFX_CSS.htm

<https://jenkov.com/tutorials/javafx/css-styling.html>

<https://stackoverflow.com/questions/13946372/adding-css-file-to-stylesheets-in-javafx>

https://www.w3schools.com/howto/howto_css_sidebar_icons.asp

<https://www.youtube.com/watch?v=o-lAsVuskKI>

<https://www.youtube.com/watch?v=lL1HHWTBZm4>

<https://www.callicoder.com/javafx-css-tutorial/>

SQL:

<https://www.w3schools.com/mysql/default.asp>

<https://www.youtube.com/watch?v=V9nDH2iBJSM>

<https://www.youtube.com/watch?v=9vLOR4522eo>

<https://www.youtube.com/watch?v=1d85TID3V4U>

https://www.youtube.com/watch?v=_VxFv4HT17o

<https://www.youtube.com/watch?v=tY3srRqoGQ4>

<https://www.youtube.com/watch?v=52TIDl2UNLA>

<https://www.youtube.com/watch?v=9AREIOce4kY>

https://www.youtube.com/watch?v=I1X_giS3vKg

https://www.youtube.com/watch?v=i6hzXIRGF_M

https://www.youtube.com/watch?v=nv1_KpuzheM

https://www.youtube.com/watch?v=L8PSQG9_teM

<https://www.javatpoint.com/sql-tutorial>

<https://www.dotnetportal.cz/clanek/50/Uvod-do-jazyka-SQL>

<https://support.microsoft.com/cs-cz/office/jazyk-sql-v-accessu-z%C3%A1kladn%C3%AD-koncepty-slovn%C3%ADk-a-syntaxe-444d0303-cde1-424e-9a74-e8dc3e460671>

<http://books.fs.vsb.cz/SQLReference/Sadovski/SQL-PRVN.HTM>

JAVA FX:

<https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html>

<https://www.youtube.com/watch?v=ltX5AtW9v3Q>

<https://www.youtube.com/watch?v=fFq9-MLvIfE&t=562s>

<https://www.javatpoint.com/javafx-borderpane>

<https://www.javatpoint.com/javafx-hbox>

<https://www.javatpoint.com/javafx-label>

<https://www.javatpoint.com/javafx-button>

<https://www.javatpoint.com/javafx-stackpane>

<https://www.javatpoint.com/javafx-gridpane>

<https://www.javatpoint.com/javafx-vbox>

<https://www.javatpoint.com/javafx-checkbox>

<https://www.javatpoint.com/javafx-textfield>

OSTATNÍ:

https://www.w3schools.com/java/java_files_read.asp

<https://github.com/Saiprasaddabbe/Online-Banking-System>

<https://www.youtube.com/watch?v=bhB4rEYkYIA>

<https://www.youtube.com/watch?v=V9nDH2iBJSM>

<https://github.com/rissandimo/JavaFX-Bank-Application>

<https://www.youtube.com/watch?v=gDfi3BcrxMo>

<https://www.youtube.com/watch?v=AUOvEIsAj24&t=248s>

<https://www.youtube.com/watch?v=fFq9-MLvIfE&t=563s>

<https://www.chegg.com/homework-help/questions-and-answers/need-complete-bank-program-using-javafx-reads-file-instructions-classes-already-completed--q19060888>

<https://github.com/zahmed333/ColgateBanking>

<https://github.com/derickfelix/BankApplication>

<https://www.youtube.com/watch?v=4givX6cX3u4>

https://github.com/Harish7775/Online_Banking_System

<https://github.com/Sayed-5/scrawny-meat-2282>

<https://github.com/rajputyash006/Online-Banking-System>

<https://www.geeksforgeeks.org/mini-banking-application-in-java/>

<https://www.youtube.com/watch?v=CwfOV-TZEsc>

<https://github.com/alexpetrov/Mentor-Bank>

<https://github.com/ntt2k/OnlineBanking>