Bankovní systém – Simulace bankomatu

Úvod projektu a vývoj dovedností

Tento projekt představoval významný krok v rozvoji mých dovedností v oblasti **Java Swing**. Během jeho implementace, inspirován tutoriály na YouTube, se mi jasně ukázala kritická důležitost **kvalitní databázové struktury**. Na cestě k dokončení jsem narazil na mnoho překážek, které však vedly k osvojení nových a cenných poznatků, a to i z materiálů, které byly sice starší, ale stále relevantní. Nejdůležitějším přínosem celého procesu bylo získání silné **motivace k dalšímu studiu Javy a programování obecně**.

Stejně tak bylo podstatné praktické pochopení **interakce SQL dotazů s databází a správného nastavení databázového připojení**. Zkušenosti s **JDBC SQLite** byly neocenitelné. Snaha o imitaci specifického bankovního systému nejen pomohla řešit konkrétní technické problémy, ale také vybudovala praktický základ, který bude velmi užitečný v mých budoucích programovacích snahách. I když si uvědomuji, že můj projekt má určité nedostatky, vzhledem k tomu, že je to můj první takto komplexní vývoj, představuje silný přínos pro mé univerzitní portfolio. Jasně ukazuje reálné aplikace Javy, databází a návrhu uživatelského rozhraní.

Cíle projektu

Hlavní cíle projektu byly:

- **Simulovat základní funkce bankomatu** v samostatné Java aplikaci.
- **Implementovat JDBC (Java Database Connectivity)** s využitím SQLite.
- **Vytvořit grafické uživatelské rozhraní** pomocí Java Swing.
- **Osvojit návrhové vzory**, konkrétně DAO (Data Access Object).

Systém imituje bankovní ATM v zjednodušené, ale funkční podobě, zaměřené na běžné operace, a zahrnuje také přihlašovací systém.

Popis systému a funkcionalita

Aplikace se spouští **přihlašovací obrazovkou**, kde se uživatel může přihlásit, nebo zaregistrovat. Veškeré informace jsou perzistentně uloženy v databázi SQLite.

Přihlášení a registrace:

- **Registrace nového uživatele**: Uživatelé zadávají své jméno, PIN a počáteční vklad. Během tohoto procesu je prováděna **důkladná validace vstupů** pro zajištění

správnosti a úplnosti dat (např. kontrola formátu e-mailu, délky PINu, či zda je počáteční vklad kladné číslo).

- **Přihlášení**: Ověření údajů probíhá pomocí **prepared statements**, což zvyšuje bezpečnost a předchází zranitelnostem typu SQL injection. V případě neúspěšného přihlášení systém poskytuje obecnou chybovou zprávu ("Nesprávné číslo karty nebo PIN"), aby neprozradil konkrétní důvod selhání a neposkytl útočníkovi nápovědu.

Funkce bankomatu:

Po úspěšném přihlášení má uživatel přístup k následujícím funkcím prostřednictvím hlavního menu bankomatu (ATM.java):

- **Zobrazit zůstatek**: Slouží k rychlému zobrazení aktuálního zůstatku na účtu.
- **Výpis z účtu**: Poskytuje detailní historii předchozích transakcí, včetně data a času, typu transakce (např. 'Vklad', 'Výběr') a částky. Implementace této funkce vyžadovala efektivní načítání a formátování dat pro zobrazení v uživatelském rozhraní.
- **Vklad**: Umožňuje vložení finančních prostředků na účet. Zadaná částka je okamžitě přičtena k zůstatku v databázi a transakce je zaznamenána.
- **Výběr**: Umožňuje výběr hotovosti. Systém provádí **kritickou kontrolu zůstatku**, aby se zabránilo výběru vyšší částky, než je na účtu k dispozici. V případě nedostatku prostředků je uživatel upozorněn.
- **Rychlý výběr**: Nabízí přednastavené částky pro rychlé transakce (např. 500 Kč, 1000 Kč). I zde probíhá kontrola zůstatku.
- **Demo peníze**: Funkce určená primárně pro testovací účely. Umožňuje rychlé přidání libovolných finančních prostředků na účet pro usnadnění testování transakcí bez nutnosti skutečného vkladu.
- **Změna PINu**: Umožňuje uživateli bezpečně změnit svůj stávající PIN za nový, s validací nového PINu.
- **Návrat**: Tlačítko pro návrat na přihlašovací obrazovku nebo pro ukončení aktuální uživatelské relace.

Všechny tyto operace jsou provázány s databází pomocí návrhového vzoru **DAO (Data Access Object)**, který zajišťuje čisté oddělení logiky přístupu k datům od logiky uživatelského rozhraní.

Technická struktura

Systémové požadavky:

- **Java Development Kit (JDK):** Oracle OpenJDK 23 nebo novější.
- **Databáze:** JcDB SQLite (integrovaná knihovna pro práci s SQLite databází).
- **Platformy:** Kompatibilní s Windows, macOS, Linux.
- **Doporučený hardware:** Minimálně 4 GB RAM, 100 MB volného místa na disku.

Architektura programu:

Program je navržen s využitím principů objektově orientovaného programování, s jasně definovanými třídami a jejich zodpovědnostmi:

- Main: Spouštěcí bod aplikace.
- Login: Třída pro grafické rozhraní přihlašovací obrazovky.
- SignUp: Třída pro grafické rozhraní registračního formuláře.
- ATM (dříve ATMUI): Hlavní třída pro GUI bankomatu, která slouží jako centrum pro navigaci mezi jednotlivými transakčními obrazovkami.
- Deposit, Withdrawl, Balance, fastCash, BankStatements, PINChange, DemoMoney: Jednotlivé třídy pro grafické rozhraní specifických operací bankomatu. Tyto třídy komunikují s instancí ATM pro návrat a předávání dat.
- User: Datový model reprezentující uživatelské informace. Obsahuje pouze atributy a metody pro jejich získání/nastavení, bez obchodní logiky.
- UserDAO: Vrstva pro přístup k datům. Zodpovídá za všechny operace s databází související s uživateli a transakcemi (např. ověřování, aktualizace, získávání dat).
- DatabaseConnection: Spravuje singleton instanci připojení k databázi. Zajišťuje, že v aplikaci je vždy jen jedno aktivní databázové připojení a správně se s ním nakládá (otevírání, zavírání).

Databázová struktura:

- **Tabulka users**: Slouží k ukládání uživatelských informací. Zahrnuje sloupce jako id (primární klíč, auto-increment), firstName, lastName, nationality, region, city, phoneNumber, email, gender, maritalStatus, pin (unikátní index), homeAddress, cardNumber (unikátní index) a balance (reálné číslo, s výchozí hodnotou 0.00).
- **Tabulka bank (nebo transactions)**: Uchovává záznamy o všech transakcích. Obsahuje transaction_id (primární klíč, auto-increment), pin (cizí klíč odkazující na tabulku users), date (TEXT pro uložení data a času transakce), type (TEXT pro typ transakce, např. 'Deposit', 'Withdrawal') a amount (REAL pro částku transakce).

V celém systému jsou důsledně použity **prepared statements** pro bezpečné a efektivní provádění SQL dotazů.

Testování a ověřování

Pro zajištění funkčnosti, spolehlivosti a bezpečnosti aplikace byly provedeny následující testy:

- **Ověření přihlášení**: Testy zahrnovaly pokusy o přihlášení s platnými i neplatnými údaji (nesprávné číslo karty, chybný PIN). Rovněž byly provedeny pokusy o simulaci **SQL injection** vložením speciálních znaků do vstupních polí, aby se potvrdila robustnost použití prepared statements.
- **Testy transakcí**: Po provedení vkladu nebo výběru byl vždy ověřen aktuální zůstatek uživatele a záznam v historii transakcí, aby se potvrdila správná aktualizace dat v databázi.
- **Hranicové případy**: Systém byl testován s mezními hodnotami a neočekávanými scénáři, jako je pokus o výběr částky vyšší, než je aktuální zůstatek, nebo zadání nulových či záporných částek pro vklad/výběr.
- **Ošetření vstupů**: Bylo ověřeno, že numerická vstupní pole (např. pro částku vkladu/výběru) správně odmítají nečíselné znaky, čímž se předchází NumberFormatException a zajišťuje se integrita dat.

Testování probíhalo jak na úrovni jednotlivých funkcí (funkční testy), tak v rámci celého uživatelského průchodu, od prvního přihlášení až po odhlášení, pro ověření plynulé integrace všech komponent.

Uživatelská příručka

Spuštění programu:

1. Ujistěte se, že máte nainstalovanou **Javu Development Kit (JDK)**, verzi Oracle OpenJDK 23 nebo novější.
2. Spusťte aplikaci přes hlavní třídu **Main**.

Ovládání aplikace:

- **Registrace**: Vyplňte příslušné položky v registračním formuláři a potvrďte.
- **Přihlášení**: Zadejte své číslo karty a PIN do přihlašovacího formuláře.

- **Menu bankomatu**: Vyberte požadovanou funkci kliknutím na příslušné tlačítko. Informace a výsledky operací jsou zobrazovány v kontextových oknech (JOptionPane).

Uživatelská data, včetně zůstatku a historie transakcí, jsou perzistentně uchovávána v databázi mezi jednotlivými spuštěními aplikace.

Zkušenosti a výzvy

Přínosy:

- **Osvojení návrhových vzorů (DAO)**: Implementace DAO výrazně zlepšila modularitu a udržitelnost kódu, oddělila databázovou logiku od prezentace.
- **Bezpečné provádění databázových operací**: Klíčové bylo pochopení a důsledné používání prepared statements, což významně zvýšilo bezpečnost aplikace proti injekčním útokům.
- **Vytvoření GUI s Java Swing**: Získání praktických zkušeností s návrhem, implementací a interakcí uživatelského rozhraní, včetně dynamické manipulace s komponentami.
- **Validace vstupů a ladění kódu**: Zlepšení dovedností v oblasti zajištění datové integrity prostřednictvím vstupní validace a systematického odstraňování chyb v průběhu vývoje.

Výzvy:

- **Správa připojení k databázi**: Tato oblast představovala značné obtíže. Bylo nutné důkladně pochopit životní cyklus Connection objektů (kdy je otevírat a kdy zavírat) a zásady pro prevenci úniků zdrojů a přetížení systému. Konkrétní problémy zahrnovaly SQLException při pokusech o použití již uzavřeného připojení nebo naopak situace, kdy příliš mnoho otevřených připojení vedlo k nestabilitě. Řešením bylo systematické využívání try-with-resources bloků pro automatické uzavírání zdrojů a implementace singleton vzoru pro DatabaseConnection, který zajišťuje pouze jedno aktivní databázové připojení.
- **Návrh intuitivního GUI a správa oken**: Složitost spočívala v efektivním rozvržení komponent a zajištění plynulého uživatelského toku. Konkrétní problémem bylo **správné řízení zobrazení a skrytí jednotlivých JFrame instancí**. Na počátku se často stávalo, že se po akci otevřelo nové okno bez zavření předchozího, což vedlo k mnoha souběžně otevřeným oknům a zmatku. Řešením bylo systematické používání setVisible(false) pro skrytí aktuálního okna a dispose() pro jeho uvolnění z paměti, spolu s předáváním reference na hlavní ATM okno do podřízených tříd pro zajištění správného návratu.

- **Zajištění přesnosti a konzistence transakcí**: Klíčovým úkolem bylo zajistit, aby finanční operace (např. výběr) byly **atomické**. To znamená, že buď se celá sekvence operací (kontrola zůstatku, odečtení peněz, záznam transakce) dokončí úspěšně, nebo se neprovede vůbec nic. Cílem bylo předejít nekonzistentním stavům databáze (např. peníze odečteny, ale transakce nezaznamenána). To vyžadovalo pochopení a implementaci **databázových transakcí v JDBC**, včetně použití conn.setAutoCommit(false), conn.commit() pro potvrzení a conn.rollback() pro vrácení změn v případě chyby. Pochopení těchto konceptů bylo na začátku složité, ale zásadní pro spolehlivost systému.

Závěr

Projekt **„Bankovní systém – Simulace bankomatu“** mi dal novou motivaci učit se programovat a pomohl mi lépe si zorganizovat práci. Díky němu jsem začal více objevovat, co všechno se dá v Javě vytvořit, a zjistil jsem, že mě tento jazyk opravdu baví. Práce na projektu mi ukázala, jak důležité je přemýšlet nad strukturou kódu, pracovat pečlivě a hledat chyby. Tato zkušenost mě posunula dál a vzbudila ve mně chuť pokračovat v programování i v dalších projektech.

Zdroje

1. Dokumentace Oracle Java – https://docs.oracle.com/en/java/
2. SQLite – https://www.sqlite.org/
3. Java Swing tutorial – https://docs.oracle.com/javase/tutorial/uiswing/
4. YouTube návody (různí autoři)
5. Knihovna JcDB SQLite – [dokumentace]
6. Problém s SQLite - https://www.youtube.com/watch?v=1DeFr3h4O8E