

**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO,
PURKYŇOVA, PŘÍSPĚVKOVÁ ORGANIZACE**



Elektronický obchodní deník

Rostislav Drápela

V4A

**PROFILOVÁ ČÁST MATURITNÍ
ZKOUŠKY MATURITNÍ PRÁCE**

BRNO 2019

Prohlášení

Prohlašuji, že jsem maturitní práci Elektronický obchodní deník vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že:

- Beru na vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu Střední průmyslové školy Brno, Purkyňova, příspěvkové organizace.
- Beru na vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, p. o., dostupná k prezenčnímu nahlednutí. Škola zajistí, že nebude pronikoho možné pořizovat kopie jakékoliv části práce.
- Beru na vědomí, že SPŠ Brno, Purkyňova, p. o. má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.
- Beru na vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

Rostislav Drápela

Marie Hubnerové 16

621 00 Brno-Řečkovice

V Brně dne 15. 04. 2019

.....

Vedoucí práce: Ing. Petr Pernes

Anotace

Tato maturitní práce se zabývá vytvořením elektronického obchodního deníku, který má za úkol vyhodnocovat obchodníkovy údaje, které do aplikace nahraje. Aplikace umožňuje zobrazení základních statistik obchodníkovy účtu (aktuální stav, průměrná výhra/prohra a další). Pro uložení dat jsem používal jazyk SQL a lokálně vytvořenou databázi.

Klíčová slova: *obchodník, burza, databáze*

Obsah

Prohlášení	2
Anotace	4
Obsah	5
Teoretický úvod	7
Obecné požadavky elektronického deníku	7
Cíl práce	8
Seznam použitých zkratk	9
Rozbor řešení	10
Problémy spojené s vytvářením aplikace	10
Nahrání statistického souboru	10
Vytvoření DB	10
Grafický design aplikace	10
Propojení DB s GUI	10
Použité technologie	11
Programovací jazyk Java	11
Databáze a MySQL	11
Návrh databáze	11
SQL příkazy použité v aplikaci	13
Případy užití	14
Typy uživatelů	14
Přihlášený uživatel	14
Nepřihlášený uživatel	14
Logika Systému	15

MainController	15
DBController	16
WindowMovement (interface)	17
.....	17
Testování aplikace	18
Závěr	19
Seznam Ilustrací	20
Citovaná literatura	20
GitHub	20
Přílohy	20
Úvod	22
Grafické rozhraní aplikace	22
Hlavní menu	22
Přidání obchodu do aplikace	23
Editování obchodu	25
Grafické zobrazení informací	27

Teoretický úvod

Na dnešním trhu existuje bezpočet elektronický deníků pro obchodníky na burze, které vynikají svými vlastnostmi, jako jsou například možnost připojení do aplikace pomocí internetu, či možnost vyhledávání obchodních příležitostí. Tyto aplikace jsou však často zpoplatněny vysokými částkami, a pro začínající obchodníky představují jejich vymoženosti spíše přítěž.

Proto jsem se rozhodl vytvořit vlastní elektronický deník, který by splňoval pouze základní požadavky. Výsledkem práce je proto jednoduchá aplikace, která zajišťuje základní informace a stavu účtu obchodníka.

Obecné požadavky elektronického deníku

Základními parametry, které by měl každý obchodník sledovat, jsou:

1. *Aktuální stav účtu*
2. *Procentuální úspěšnost obchodu*
3. *RRR(Risk Reward Ratio)*
4. *Průměrná výhra*

S těmito základními prvky je daný obchodník schopen určovat, zda jeho *obchodní strategie* je z dlouhodobého hlediska úspěšná či ne. Mezi další základní požadavky, patří také grafické zobrazení vývoje obchodníkovy účtu během daného časové období pro účely zkoumání výkonu strategie během různých období na finančním trhu.

Jako další je nutné zobrazit podrobnější informace o jednotlivém konkrétním obchodu. Mezi tyto informace patří:

1. *Výsledek obchodu*
2. *Velikost obchodu (lot size)*
3. *StopLoss (nejvyšší přijatelná ztráta)*
4. *Cenová hladina při vstupu*
5. *Cenová hladina při výstupu*

V rámci těchto informací jsem implementoval jenom ty nejdůležitější, které byly nezbytné pro výpočet statistik.

Cíl práce

Cílem práce je vytvořit aplikaci, která dokáže zpracovat a vyhodnotit základní parametry obchodníka v rámci jednoho či více obchodních účtů. Výsledná práci by měla usnadňovat zpracování informací, a následnou prezentaci těchto výsledků v jednoduché a přehledné grafické podobě.

Aplikace je napsána objektově orientovaným programováním v jazyce Java s prvky JavaFX a využití MySQL databáze pro uchování a zpracování uživatelského vstupu. Její grafický design využívá knihovny JFoenix.

Seznam použitých zkratek

DB - DataBase

Forex - Foreign Exchange

JDBC - Java Database Conectivity – soubor metod(API)

API - Application Programming Interface

ID - IDentification

Rozbor řešení

Elektronický obchodní deník jsem navrhl podle mých základních potřeb jakožto začínající obchodníka na *Forexovém* trhu. Forex je zkratka pro Foreign Exchange (anglicky směna cizích měn). Forex je ale též známý pod názvy Forex Trading, Currency Trading, Foreign Exchange Market nebo zkráceně FX. Je to mezinárodní obchodní systém pro směnu základních a vedlejších měnových párů, tedy devizový trh, jehož střední kurzy se považují za oficiální světové kurzy (Wikipedie, 2019)

Jak jsem již zmínil, soustředím se pouze na implementaci základních potřeb a vypouštím některé komerční funkce, jako například sledování otevřených obchodu a jiné, abych zajistil co nejjednodušší uživatelsky přívětivé prostředí.

Problémy spojené s vytvářením aplikace

Nahrání statistického souboru

Jako první bylo nutné zajistit vložení informací o obchodech, které daný obchodník provedl, aby aplikace měla data, ze kterých později vypočítala statistiky.

Vytvoření DB

V rámci vývoje aplikace byla nutnost vytvořit jednoduchou DB, která by splňovala jak normální formy DB, tak i účel pro který jsem ji navrhoval – jednoduché ukládání dat, tak aby se s nimi dalo později pracovat.

Grafický design aplikace

Při realizování této práce jsem myslel i na *GUI*, proto bylo potřeba navrhnout uživatelsky jednoduché a přívětivé prostředí, které je intuitivní, aby práce s aplikací byla co možná nejjednodušší.

Grafický design používá ikony z programu *Icons8*, kde jsou dostupné ikony z operačních systému *Android*, *IOS*, *Windows* a další .

Propojení DB s GUI

Pro funkčnost aplikace bylo nutné, abych propojil GUI s DB, abych mohl daná data zobrazovat v grafické formě. Tento problém byl poměrně obsáhlý, proto zabral nejvíce času na řešení.

Použité technologie

V následujících odstavcích představím technologie, které jsem při vytváření aplikace využil

Programovací jazyk Java

Celá aplikace běží na lokálním PC na technologii JavaFX a jejích grafických prvků. K vytvoření GUI jsem použil aplikaci *JavaFX Scene Builder 2.0*, která mi umožnila vytvářet grafické rozhraní pomocí grafických elementů.

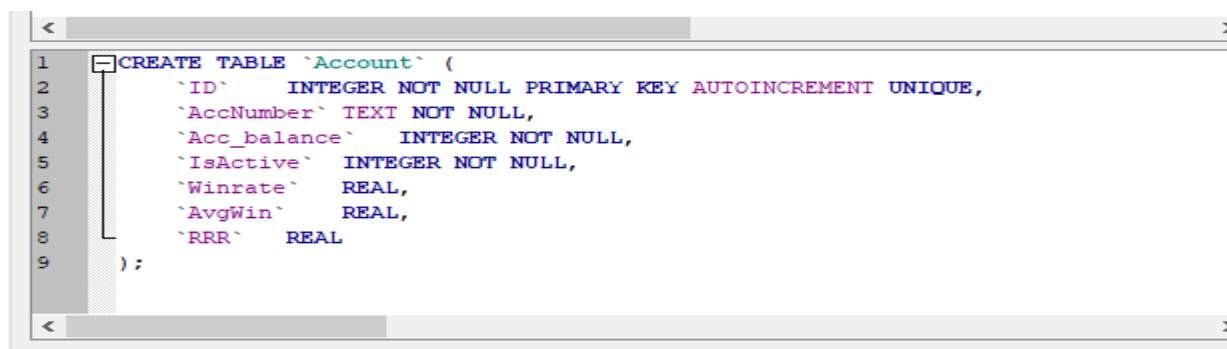
Databáze a MySQL

Pro ukládání dat jsem použil již zmíněnou DB, která umožňuje uchovávání a práci s daty i po vypnutí aplikace. Jako databázový systém jsem si vybral MySQL kvůli možnosti volného šíření. Komunikace funguje na bázi SQL příkazů přes API JDBC.

Návrh databáze

Při návrhu knihy, jsem se soustředil na základní prvky potřebné pro výpočet statistiky v rámci aplikace. V Případě účtu to byly například informace ohledně jeho aktuálního stavu, čísla (jména) účtu, *RRR* a další. V případě obchodu to byly informace ohledně cenové hladiny při vstupu a výstupu do obchodu, objem obchodu, datum vstupu a výstupu a další.

Vytvořená databáze se skládá ze 3 tabulek: *Account*, *Login*, a *Trade*. Návrh této databáze jsem vytvořil v programu *MySQLWorkbench*. Jakmile jsem vytvořil tento návrh, vytvořil jsem DB v programu *DB Browser for SQLite*. V obou tabulkách slouží jako primární klíč sloupec *ID*, pomocí kterého se spojují tabulky *Account* a *Trade*.



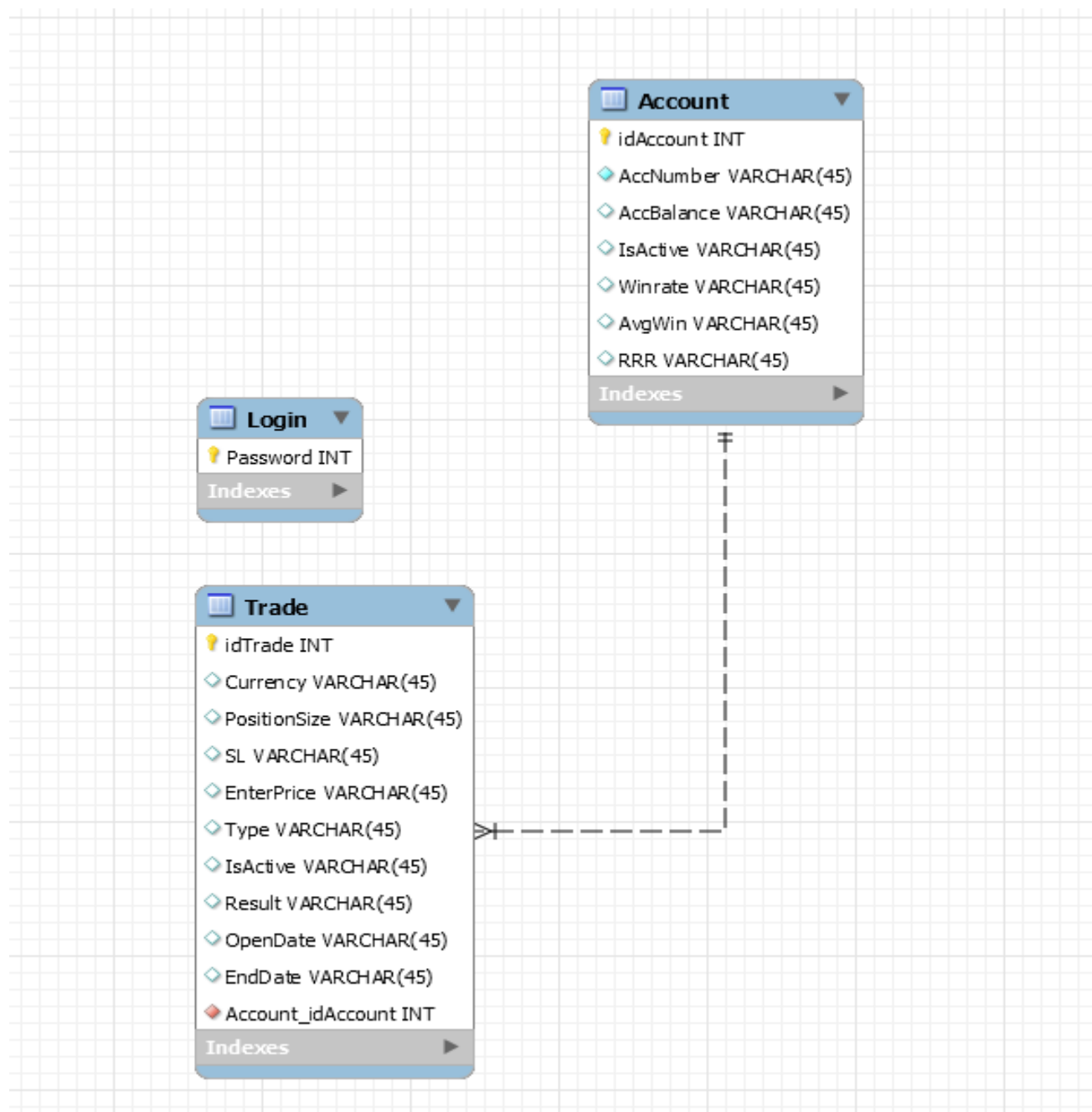
Obrázek 1 Vytvoření Tabulky Account

```

1 CREATE TABLE "Trade" (
2     "ID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
3     "Currency" TEXT NOT NULL,
4     "PositionSize" REAL NOT NULL,
5     "SL" REAL NOT NULL,
6     "EnterPrice" REAL NOT NULL,
7     "Type" TEXT NOT NULL,
8     "IsActive" INTEGER NOT NULL,
9     "AccID" INTEGER,
10    "Result" REAL,
11    "OpenDate" TEXT NOT NULL

```

Obrázek 2: Vytvoření tabulky Trade



Obrázek 3: Návrh DB

SQL příkazy použité v aplikaci

V rámci aplikace jsme použil několik SQL příkazů, pomocí kterých jsem byl schopen získat data z DB a zpracovat je v rámci aplikace.

Mezi nejvíce používanou rovníci patří:

```
=ry("Select Count(*) From Trade where AccID = " +id + " and OpenDate = current_date");

=ry("Select Count(*) From Trade where EndDate != 'null' and AccID = " + id + " and EndDate = current_date");

ry("SELECT SUM(result) FROM Trade WHERE AccID= " + id + " and EndDate =current_date");

ry("Select count (*) from Trade where AccID = " + id + " and Result > 0 and EndDate = current_date");

("Select Sum(Result) from Trade where AccID = " + id + " and Result < 0 " + " and EndDate = current_date");

("Select Sum(Result) from Trade where AccID = " + id + " and Result > 0 " + " and EndDate = current_date");
```

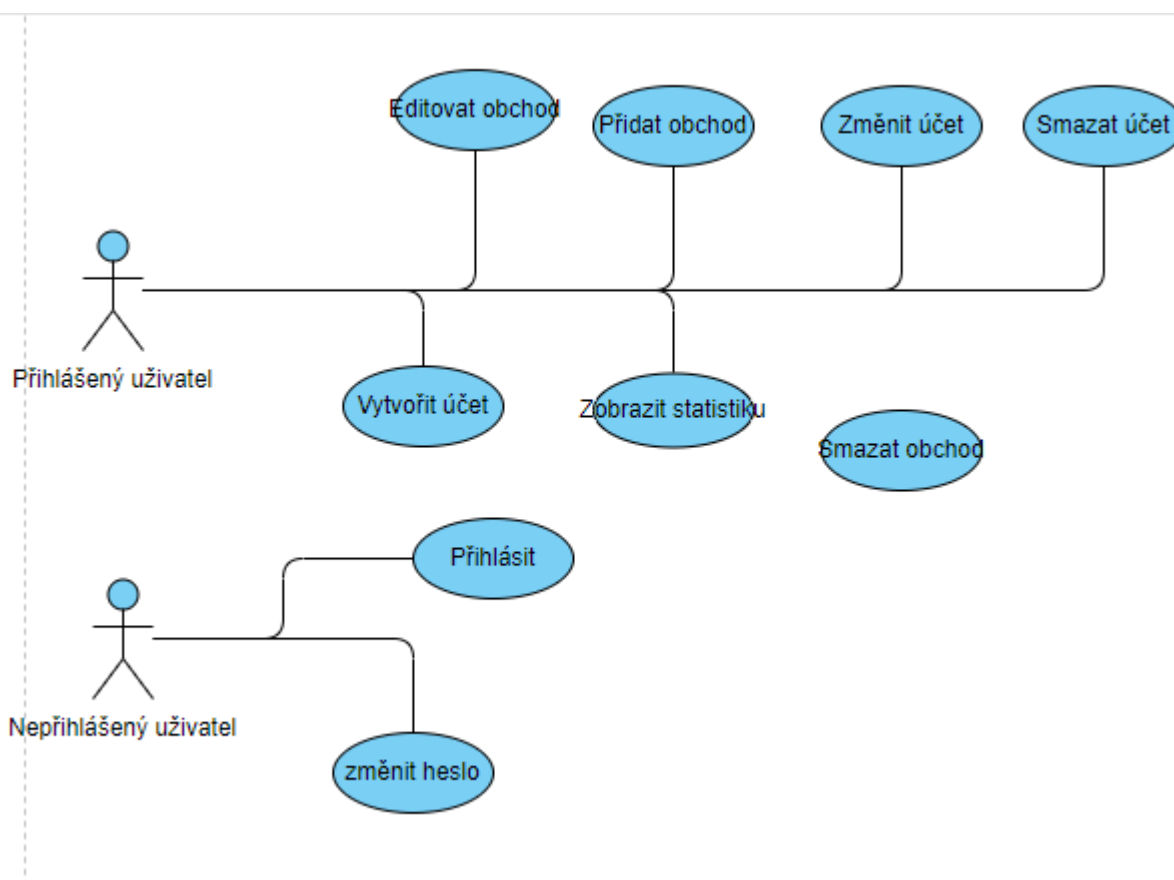
Tyto SQL příkazy získávají z DB data v následujícím pořadí:

1. Počet otevřených obchodů u daného účtu
2. Počet všech uzavřených obchodu u daného účtu
3. Součet výsledků všech uzavřených obchodů u daného účtu
4. Počet úspěšných obchodů
5. Součet výsledků všech neúspěšných obchodů
6. Součet výsledků všech úspěšných obchodů

Výše uvedené příkazy získávají data pro aktuální den. Pro týdenní a měsíční statistiku jsou rovnice mírně upraveny.

Případy užití

UML případ užití vyjadřuje, funkce, které můžu z pohledu uživatele použít. Přiložený diagram vyjadřuje možnosti přihlášeného/nepřihlášeného uživatele.



Obrázek 4: případ užití

Typy uživatelů

Přihlášený uživatel

Přihlášený uživatel má kompletní přístup k uživatelským datům. Má tedy možnost editovat již vložená data, přidávat nová obchodní data, popřípadě zobrazovat aktuální statistiky. Zároveň může vytvářet nové / mazat staré obchodní účty.

Nepřihlášený uživatel

Nepřihlášený uživatel má pouze dvě možnosti. První možnost je přihlásit se. Druhá možnost je změna uživatelského hesla. Aby heslo mohl změnit, musí znát staré přihlašovací heslo.

Logika Systému

MainController

V rámci programování logiky celého programu jsem použil třídu MainController, která obsahuje objekty DBController a všechny Scény a jejich controllerem.

Přesun dat mezi jednotlivými scénami zajišťuje právě třída MainController ve které se vždy rozhodne podle v metodě *changeScene* podle ID stisknutého tlačítka, jaká část kódu se provede. Následně se provedou instrukce spojené s DBControllerem, popřípadě controllerem dané scény, která se přepne při stisku tlačítka.

Jako příklad sem uvedl průběh při stisku tlačítka *login*. Jako první se zjistí, jestli zadané heslo je shodné s heslem v DB. Poté Metoda *changeScene* vyhodnotí, že se stisklo tlačítko *login* a odkáže se na objekt *dbController*, který provede metodu *doesAccExist()*.

Tato metoda zjišťuje, jestli už v DB existuje záznam o vytvoření uživatelského účtu. Pokud ano proběhne metoda *updateTab()*, které získá záznam o všech obchodech daného účtu, vypočítá statistiku, a tu následně pošle do *mainControlleru*. Jako poslední změnu scény na *mainScene*.

Pokud však v DB není nalezen záznam o vytvoření jakéhokoliv účtu, přepne se scéna na *createAccountScene*, kde uživatel vytváří nový účet.

```
public void changeScene(String id)
{
    switch(id)
    {
        // potvrzení hesla -> menu
        case "login":
            // nutnost zjistit jestli je vytvořený acc, pokud ne vytvořit, poté vyhledat aktivní acc, updatovat jeho statistiky a poslat ho do menuControlleru
            if (this.dbController.doesAccExist())
            {
                // this.dbController.updateAcc();
                this.menuController.updateTab(this.dbController.updateTab());
                this.menuController.setAccount(this.dbController.getActiveAccount());
                this.stage.setScene(menuScene);
                break;
            }
            // pokud už je acc vytvořený, vzít ho updatovat a poslat dál
        else
            this.stage.setScene(createAccountScene);
            //vytvořit nový acc
        break;
    }
}
```

Obrázek 5: MainController metoda ChangeScene

DBController

DBController je třída, která má za úkol zpracovat všechny instrukce týkající nebo související s DB. Její Konstruktor vytváří stabilní spojení mezi aplikací a DB, pomocí *API JDBC*.

Za zmínku stojí metoda *Login*, která porovnává zadané heslo uživatel s heslem, které je uloženo v DB.

```
public boolean login(String psw)
{
    ResultSet rs = null;
    try {
        rs = connection.createStatement().executeQuery("Select * from Login");
        if(psw.equals(rs.getString(1)))
        {
            rs.close();
            return true;
        }
        rs.close();
    } catch (SQLException ex) {
        Logger.getLogger(DBController.class.getName()).log(Level.SEVERE, null, ex);
    }

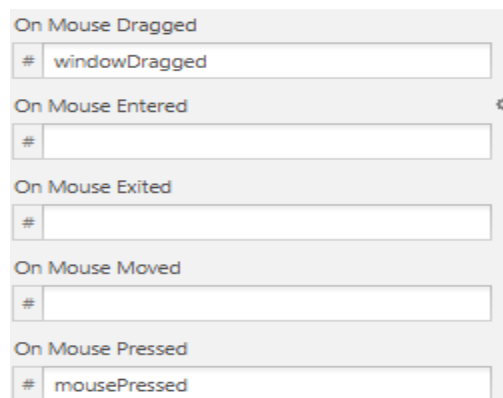
    return false;
}
```

Obrázek 6: DBController metoda Login

WindowMovement (interface)

Jelikož každý controller, který jsem prozatím vytvořil, sdílí určité množství stejných funkcí, vytvořil jsem interface, který tyto metody obsahuje, a který stačí pouze implementovat v jednotlivých controllerech, namísto toho abych v každé třídě vypisoval stejné řádky kódu.

Většina těchto metod se týká pohybu okna aplikace. Tento pohyb zajišťují metody *windowDragged* a *mousePressed*. Abych tyto metody propojil s vizuálním oknem aplikace, je nutné je v aplikaci *JavaFX Scene Builder* propojit s jednotlivými prvky. Toho dosáhneme přidáním jména metody, do jednotlivých akcí daného prvku.



Obrázek 7 propojení metody s GUI

Testování aplikace

V rámci testování aplikace jsem vytvořil tři cvičné DB, ve kterých byly uloženy rozdílné informace. Poté jsem aplikaci propojil s DB a porovnal výsledky, které se mi zobrazily s vypočítanými výsledky. Testování aplikace proběhlo úspěšně.

Závěr

V maturitní práci jsem splnil všechny body uvedené v zadání. Realizování maturitní práce bylo poněkud obtížnější, finální verze je 3. verzí této práce. Důvodem k přepracování byla nepřehlednost a nefunkčnost kódu v pozdější fázi vývoje aplikace.

V rámci realizace maturitní práce jsem si prohloubil znalosti týkající se OOP v Javě, propojením Javy s SQL DB, a celkově znalosti ohledně vývoje „větších“ projektů a jejich organizaci.

Mezi další důležité věci patří prohloubení znalostí v oblasti timemanagementu, neboť většina vývoje první verze aplikace proběhla ke konci roku 2018.

Výslednou aplikaci jsem naprogramoval podle svých osobních potřeb, neboť se obchodováním na burze zabývám, a poplatky za komerční produkty, které obsahují funkce pro mě nepotřebné, mi přišly až příliš vysoké.

Celá aplikace je dostupná ke stažení na <https://github.com/TeamRedRules/maturita>

Seznam Ilustrací

Obrázek 1 Vytvoření Tabulky Account	10
Obrázek 2: Vytvoření tabulky Trade	10
Obrázek 3: Návrh DB	11
Obrázek 4: případ užití	12
Obrázek 5: MainController metoda ChangeScene	13
Obrázek 6: DBController metoda Login	14
Obrázek 7 propojení metody s GUI	14

Citovaná literatura

(14. 04 2019). Načteno z Wikipedie: <https://cs.wikipedia.org/wiki/Forex>

GitHub

Celá práce a vývoj její třetí verze je zálohovaná zde:

<https://github.com/TeamRedRules/maturita>

Přílohy

Uživatelská příručka

**STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO,
PURKYŇOVA, PŘÍSPĚVKOVÁ ORGANIZACE**



Elektronický obchodní deník Uživatelská příručka

Rostislav Drápela

V4A

**PROFILOVÁ ČÁST MATURITNÍ
ZKOUŠKY MATURITNÍ PRÁCE**

BRNO 2019

Úvod

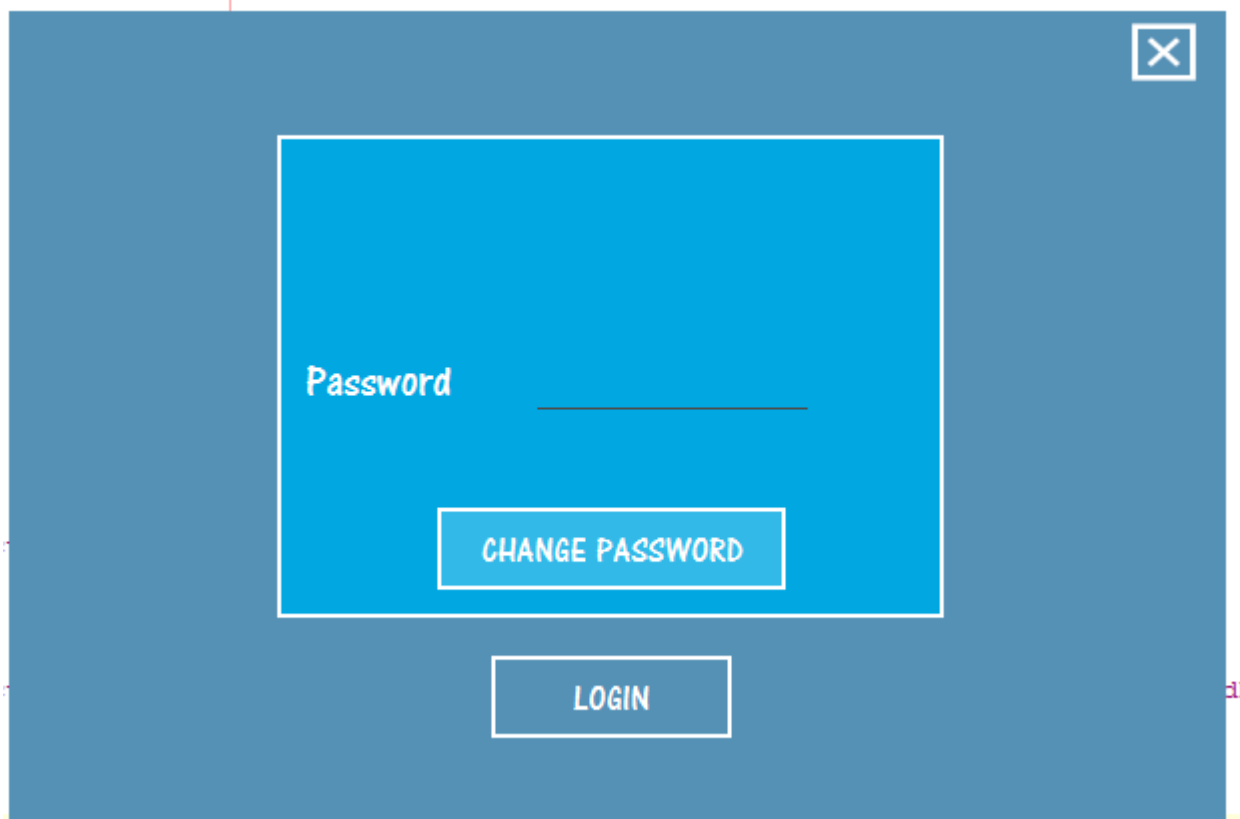
Elektronický obchodní deník je desktopová aplikace, určená pro zjednodušení zpracovávání statistiky obchodníka a jeho obchodní strategie.

Grafické rozhraní aplikace

Při startu aplikace se zobrazí přihlašovací okno aplikace. Zde má uživatel možnost zadat přístupové heslo do aplikace. V defaultním nastavení je heslo *Admin*. Zároveň se uživatel může rozhodnout heslo změnit.

Hlavní menu

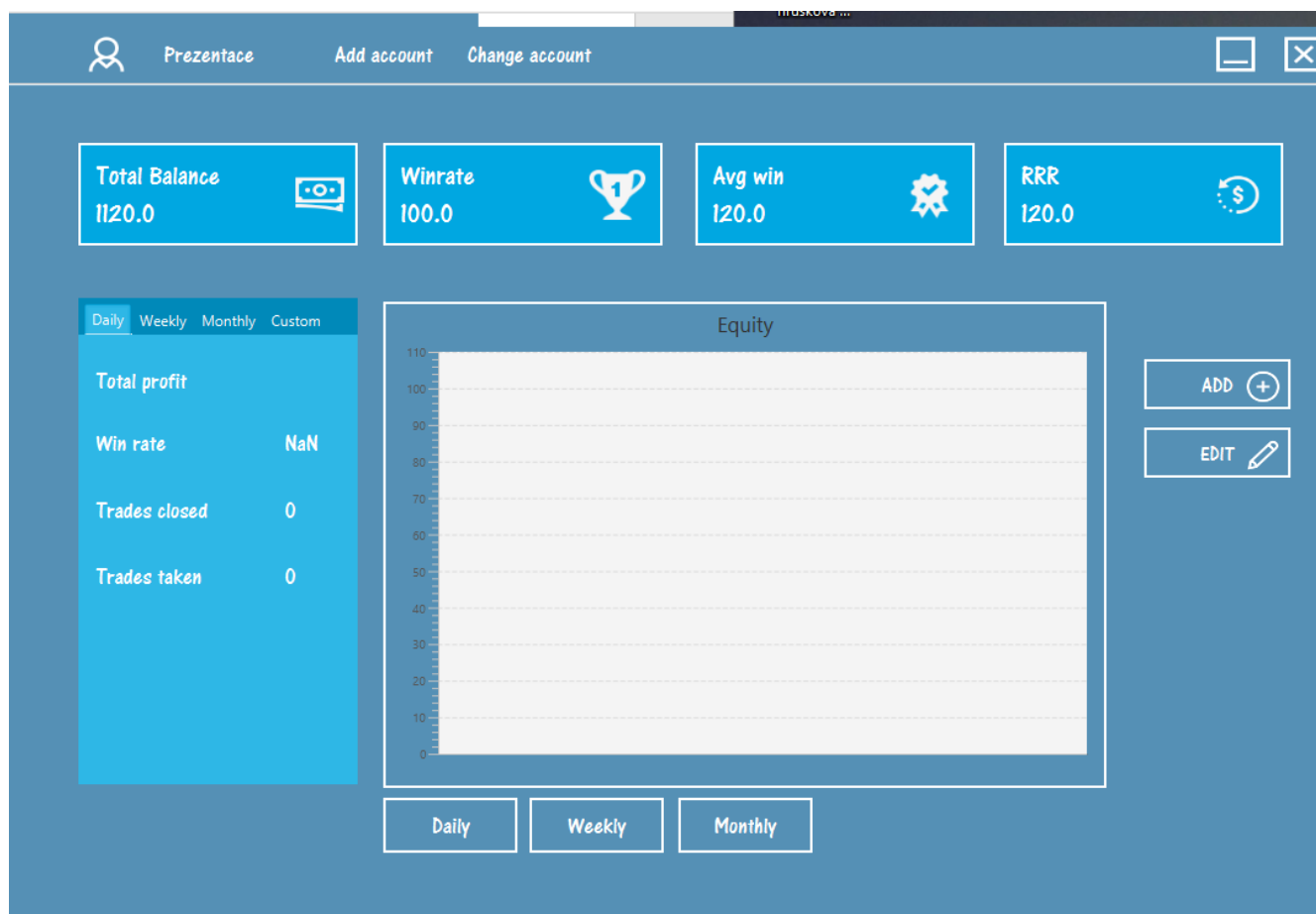
Po přihlášení se uživatel ocitne v hlavním menu aplikace, kde se mu zobrazí statistiky dané účtu. Uživatel má na výběr zobrazit si denní, týdenní, nebo měsíční statistiky. Zároveň si může vypsát statistiky za období, které si sám určí. Tyto statistiky se zobrazí po levé straně aplikace.



Obrázek 1 Přihlášení do aplikace

V horní liště aplikace se uživatel může rozhodnout přidat nový obchodní účet, popřípadě změnit obchodní účet.

Po pravé straně lze nalézt tlačítka *add trade* a *edit trade*. Tyto tlačítka slouží k přidání a editování přidanych obchodů.



Obrázek 2 Hlavní okno aplikace

Ve spodní části aplikaci si uživatel může zobrazit graf vývoje celkové bilance obchodu v denním, týdenním nebo měsíčním intervalu.

Přidání obchodu do aplikace

Pro přidání obchodu do aplikace stačí kliknout na tlačítko *Add*. Poté se uživatel dostane do okna, kde zadá parametry obchodu, který právě otevřel. Poté klikne na tlačítko *Add* a obchod se uloží do DB.

Add trade

Currency

Enter Price

Stop Loss

Position Size

Type

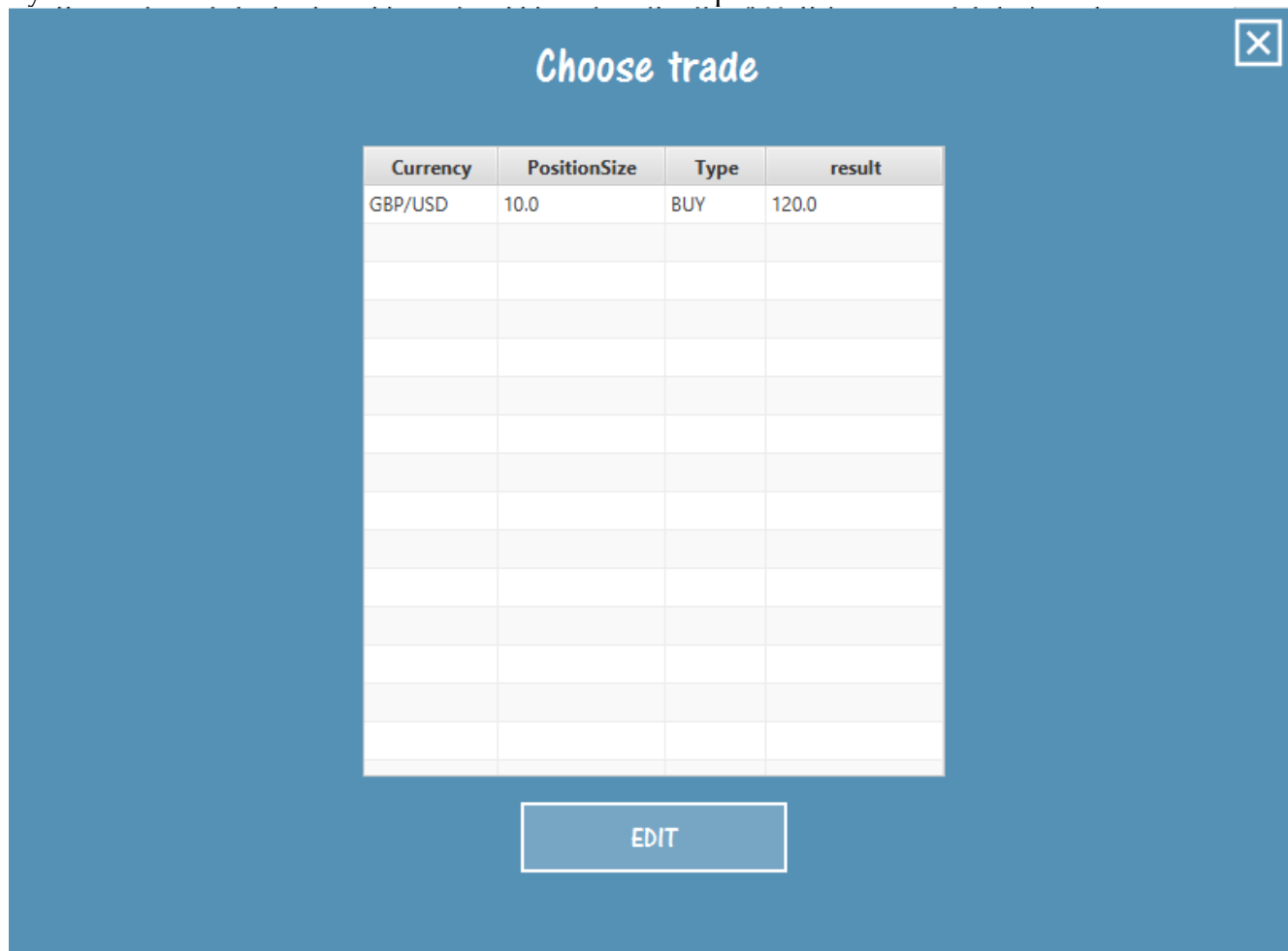
ADD

Obrázek 3 Vložení obchodu do aplikace

Editování obchodu

Poté co se vloží obchod do aplikace, je nutné po jeho uzavření editovat i obchod v aplikaci. Proto slouží tlačítko *Edit*. Po kliknutí se zobrazí seznam všech obchodů, které jsou k danému účtu přiřazeny. Uživatel si vybere obchod, který chce editovat a klikne na tlačítko.

Po stisku se mu zobrazí okno *editTrade*, kde může editovat vlastnosti obchodu včetně jeho výsledku. Po zadání všech informací stačí potvrdit a obchod se uloží do DB.



Currency	PositionSize	Type	result
GBP/USD	10.0	BUY	120.0

EDIT

Obrázek 4 Výběr obchodu k editaci

×

Edit trade

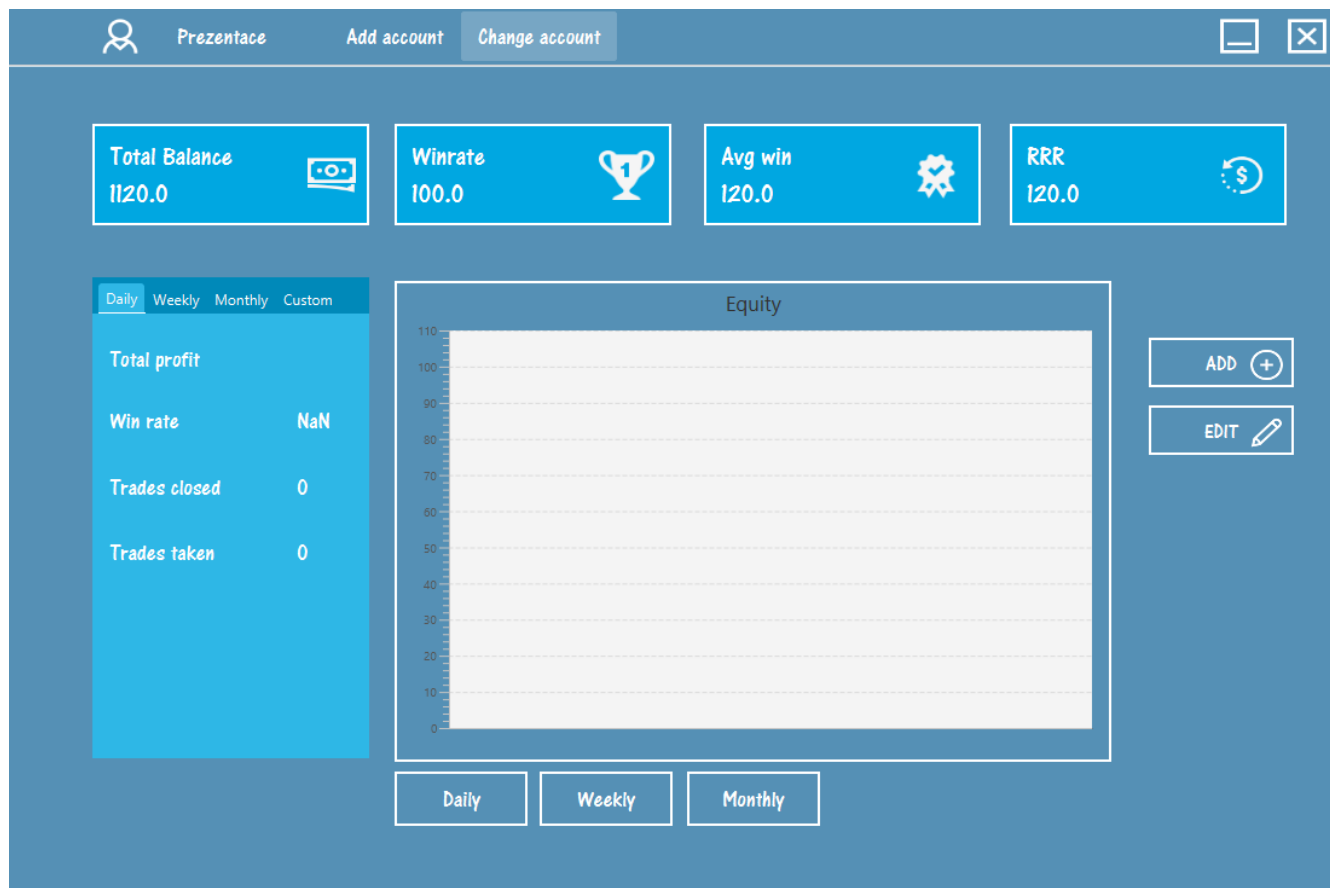
Currency	GBP/USD
Enter Price	10.0
Stop Loss	10.0
Position Size	10.0
Result	120.0
Type	BUY

Update

Obrázek 5 Editace obchodu

Grafické zobrazení informací

Pro zobrazení informací v podobě grafu stačí kliknout na jednu ze tří tlačítek umístěných ve spodní části aplikace pod grafem. Po kliknutí na tlačítka se graf aktualizuje podle dat uložených v DB.



Obrázek 6 Graf před zobrazením



Obrázek 7 Zobrazení dat v grafu