Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Объектно-ориентированное программирование

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

“Анализ динамики показателей финансовой отчетности различных предприятий”

Выполнил: студент группы 953505

Басенко Кирилл Александрович

Проверил:

Летохо Александр Сергеевич

Минск 2021

**Содержание**

1. Введение 3

1.1. Постановка задачи 4

1.2. Инфологическое проектирование Базы данных 6

1.3. Анализ предметной области 6

1.4. Атрибуты. Идентифицирующие ключи 6

2. Логическое проектирование 7

2.1. Отношения 7

2.2. UML-диаграмма 8

2.3. Описание классов 10

2.3.1. Класс Main 10

2.3.2. Класс User 11

2.3.3. Класс Unit 11

2.3.4. Класс Indicator 11

2.3.5. Класс Enterprise 11

2.3.6. Класс Dynamic 11

2.3.7. Класс DynamicsRow 11

2.3.8. Класс DBHandler 12

2.3.9. Класс Configs 12

2.3.10. Класс SignUpController 12

2.3.11. Класс SignInController 13

2.3.12. Класс MainMenuController 13

2.3.13. Класс IndicatorsController 13

2.3.14. Класс UnitsController 13

2.3.15. Класс EnterprisesController 14

2.3.16. Класс DynamicsController 14

3. Демо 15

4. Заключение 21  
5. Список использованной литературы 22  
5. Приложение 23

**Введение**

Современные информационные системы ориентированы на конечного пользователя, не обладающего высокой квалификацией в области вычислительной техники. Поэтому клиентские приложения информационной системы должны обладать простым, удобным, легко осваиваемым интерфейсом.

На данный момент существует множество готовых инструментов, которые позволяют разрабатывать графический пользовательский интерфейс и работать с базами данных.

Данный проект реализован на языке Java в среде разработки Intellij Idea. Для базы данных использовался MySQL Database Service и MySQL Workbench. Создание графического пользовательского интерфейса осуществлялось в среде разработки Scene Builder с использованием пакета JavaFX.

Информационная система “Анализ динамики показателей финансовой отчетности различных предприятий” предназначена для удобного отображения финансовых показателей и их анализа.

Целью данного курсового проекта является создание удобного и понятного приложения, которое позволит пользователям с легкостью анализировать различные показатели.

**Постановка задачи**

Информационная система «Анализ динамики показателей финансовой отчетности различных предприятий» должна содержать следующий функционал:

* Регистрация новых пользователей
* Валидация введенных данных
* Проверку на существование пользователей с таким же username
* Логирование существующих пользователей
* Отображение сообщений при неудачном логировании или регистрации
* Просмотр курса валют
* Просмотр информации об предприятиях
* Просмотр показателей
* Просмотр показателей по предприятию и по дате

Все данные должны храниться в базе данных. Таблица пользователя должна содержать следующие поля:

* Уникальный id
* Уникальное имя пользователя username
* Пароль password

Таблица предприятий должна содержать следующие поля:

* Уникальный id
* Уникальное название предприятия name
* Контактный телефон phone
* Имя владельца предприятия name

Таблица показателей должна содержать следующие поля:

* Уникальный id
* Уникальное именование показателя name
* Важность показателя importance

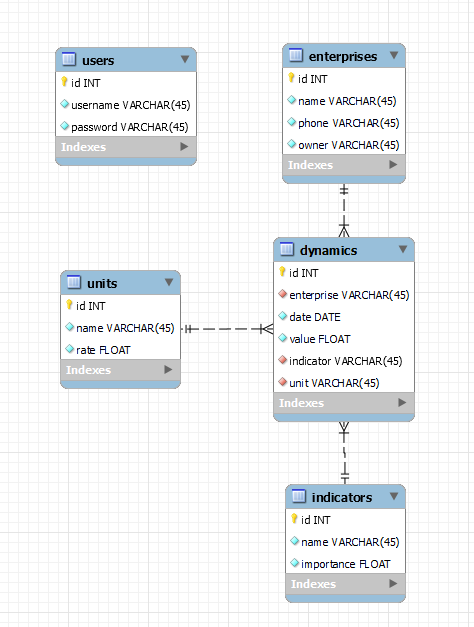
Таблица валют должна содержать следующие поля:

* Уникальный id
* Уникальное именование валюты name
* Курс валюты rate

Таблица динамик должна содержать следующие поля:

* Уникальный id
* Именование предприятия enterprise
* Дата отчета по показателю date
* Значение показателя value
* Именование показателя indicator
* Валюта значения показателя unit

Отношения между таблицами:



**Часть 1. Инфологическое проектирование Базы данных**

В данном курсовом проекте в качестве хранилища данных используется СУБД MySQL. Поскольку она является локальной, многопоточной, поддерживает несколько одновременных запросов, имеет гибкую систему привилегий и паролей, поддерживает записи фиксированной и переменной длины, имеет поддержку со стороны Java. Также имеет множество, простых в использовании, визуальных инструментов, что значительно ускоряет построение необходимой архитектуры БД.

**1.1 Анализ предметной области**

Предметная область информационной системы “Анализ динамики показателей финансовой отчетности различных предприятий” должна содержать информацию о пользователях, показателях, валютах, предприятиях, динамик. Для этого было выделены следующие объекты:

* User
* Indicator
* Unit
* Enterprise
* Dynamic

**1.2 Атрибуты. Идентифицирующие ключи**

Проанализировав объекты рассматриваемой предметной области, можно выделить следующие атрибуты и идентифицирующие ключи:

* Объект User содержит следующие атрибуты: “username”, “password”. Идентифицирующими ключами являются: “id” и “username”.
* Объект Indicator содержит следующие атрибуты: “name”, “importance”. Идентифицирующим ключом является “id”.
* Объект Unit содержит следующие атрибуты: “name”, “rate”. Идентифицирующим ключом является “id”.
* Объект Enterprise содержит следующие атрибуты: “name”, “phone”, “owner”. Идентифицирующим ключом является “id”.
* Объект Dynamic содержит следующие атрибуты: “enterprise”, “date”, “value”, “indicator”, “unit”.

**Часть 2. Логическое проектирование**

**2.1 Отношения**

На основе рассмотренных сущностей и атрибутов можно построить следующие отношения:

**Таблица 1. Отношение “User”**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | В программе |
| id | Числовой | id |
| username | Текстовый | username |
| password | Текстовый | password |

**Таблица 2. Отношение “Indicator”**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | В программе |
| id | Числовой | id |
| name | Текстовый | name |
| importance | Числовой | importance |

**Таблица 3. Отношение “Unit”**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | В программе |
| id | Числовой | id |
| name | Текстовый | name |
| rate | Числовой | rate |

**Таблица 4. Отношение “Enterprice”**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | В программе |
| id | Числовой | id |
| name | Текстовый | name |
| phone | Текстовый | phone |
| owner | Текстовый | owner |

**Таблица 5. Отношение “Dynamic”**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | В программе |
| id | Числовой | id |
| enterprise | Текстовый | enterprise |
| date | Числовой | date |
| value | Числовой | value |
| indicator | Текстовый | indicator |
| unit | Текстовый | unit |

**2.2 UML-диаграмма**

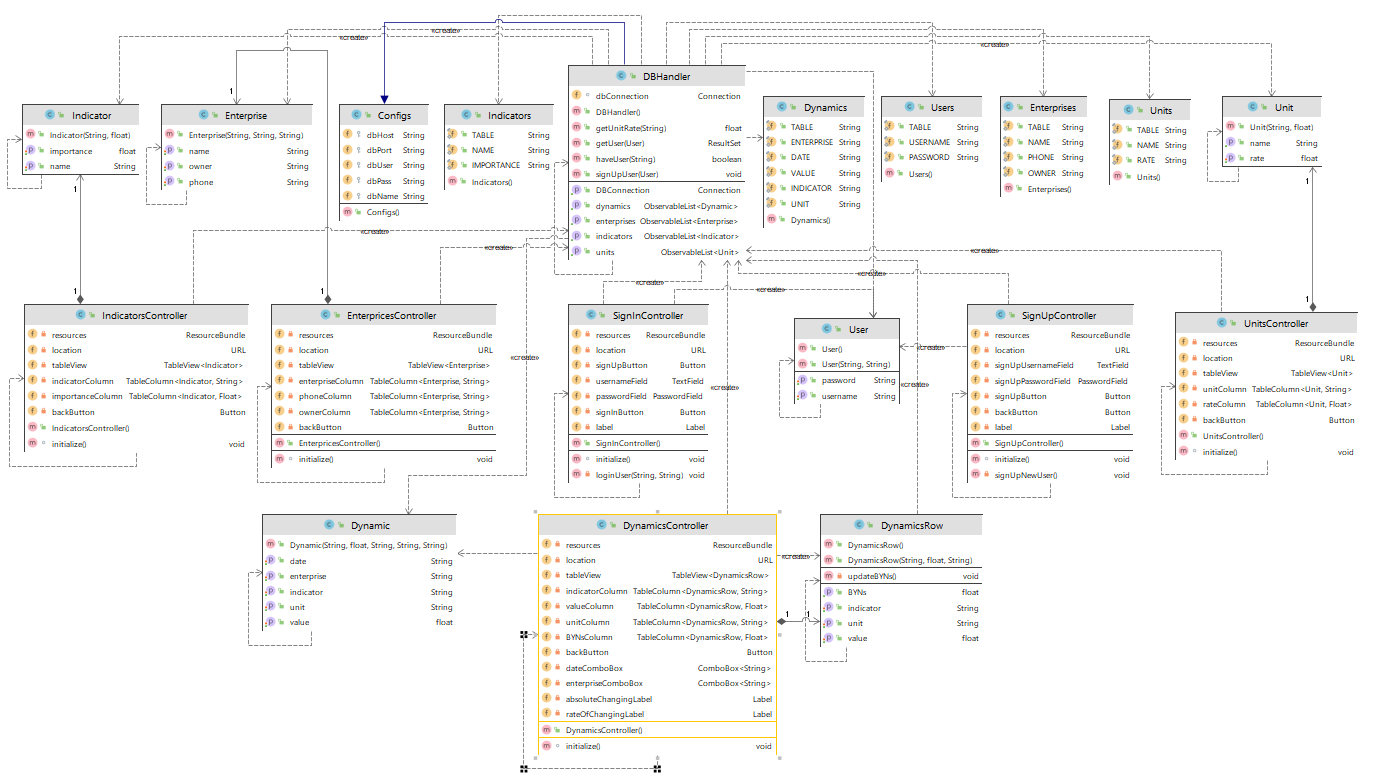
UML - это унифицированный язык моделирования. UML диаграммы могут применяться для различных задач:

* Проектирование. UML-диаграммы помогут при моделировании архитектуры больших проектов, в которой можно собрать как крупные, так и более мелкие детали и нарисовать каркас (схему) приложения. По нему впоследствии будет строиться код.
* Реверс-инжиниринг — создание UML-модели из существующего кода приложения, обратное построение. Может применяться, например, на проектах поддержки, где есть написанный код, но документация неполная или отсутствует.
* Из моделей можно извлекать текстовую информацию и генерировать относительно удобочитаемые тексты — документировать. Текст и графика будут дополнять друг друга.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы. На общем уровне графическая нотация UML содержит 4 основных типа элементов:

* фигуры;
* линии;
* значки;
* надписи.

UML-нотация является де-факто отраслевым стандартом в области разработки программного обеспечения, ИТ-инфраструктуры и бизнес-систем.



**2.3 Описание классов**

Подробно рассмотрим каждый класс изображенный на рис. 2. Всего программа состоить из следующих классов:

* Main
* User
* Unit
* Indicator
* Enterprise
* Dynamic
* DynamicsRow
* DBHandler
* Configs
* UnitsController
* SignUpController
* SignInController
* MainMenuController
* IndicatorsController
* EnterprisesController
* DynamicsController

**2.3.1 Класс Main**

Данный класс содержит следующий атрибут:

* stg – является основным окном программы. Используется для отображения сцены логирования, регистрации и каталога товаров.

Также в этом классе содержится три метода:

* public static void main(String[] args) – является основным методом программы.
* public void start(Stage primaryStage) – метод, который вызывается при запуске программы. Задача этого метода – создать основное окно и отобразить сцену логирования.

**2.3.2 Класс User**

Данный класс содержит следующий атрибуты:

* username – имя пользователя
* password – пароль пользователя

**2.3.3 Класс Unit**

Данный класс содержит следующий атрибуты:

* name – именование валюты.
* rate – курс валюты.

**2.3.4 Класс Indicator**

Данный класс содержит следующий атрибуты:

* name – именование показателя.
* importance – важность показателя.

**2.3.5 Класс Enterprise**

Данный класс содержит следующий атрибуты:

* name – именование предприятия.
* phone – контактный телефон.
* owner – имя владельца.

**2.3.6 Класс Dynamic**

Данный класс содержит следующий атрибуты:

* enterprise – именование предприятия
* date – дата отчета по показателю
* value – значение показателя
* indicator – именование показателя
* unit – валют

**2.3.7 Класс DynamicsRow**

Данный класс содержит следующий атрибуты:

* value – значение показателя
* indicator – именование показателя
* unit – валюта
* BYNs – значение показателя в белорусских рублях

**2.3.8 Класс DBHandler**

Данный класс содержит следующий атрибуты:

* connection – подключение к базе данных для обновления или взятия данных

Данный класс содержит следующие методы:

* public float getUnitRate(String unit) – служит для предоставления курса валюты по ее именованию
* public void signUpUser(User user) – служит для регистрации новых пользователей
* public boolean haveUser(String username) – служит для проверки уже зарегистрированных пользователей на указанное имя.
* public ObservableList<Unit> getUnits() – служит для представления таблицы в виде списка.
* public ObservableList<Enterprise> getEnterprises()– служит для представления таблицы в виде списка.
* public ObservableList<Indicator> getIndicators()– служит для представления таблицы в виде списка.
* public ObservableList<Dynamic> getDynamics()– служит для представления таблицы в виде списка.

**2.3.9 Класс Configs**

Данный класс содержит следующий атрибуты:

* dbHost – для установления соединения
* dbPort – для установления соединения
* dbUser – имя пользователя для установления соединения
* dbPass – пароль пользователя для установления соединения
* dbName – имя базы данных для подключения

**2.3.10 Класс SignUpController**

Данный класс содержит следующий атрибуты:

* signUpUsernameField – поле для ввода имени пользователя
* signUpPasswordField – поле для ввода пароля пользователя
* signUpButton – кнопка для регистрации пользователя
* label – метка для отображении сообщений
* backButton – кнопка для возврата в окно логгирования

Данный класс содержит следующий методы:

* void initialize() – метод устанавливаются “слушатели” для кнопок.

**2.3.11 Класс SignInController**

Данный класс содержит следующий атрибуты:

* signUpButton – кнопка для перехода в окно регистрации
* usernameField – поле для ввода имени пользователя
* passwordField– поле для ввода пароля пользователя
* signInButton – кнопка для входа в приложение
* label – метка для отображении сообщений

Данный класс содержит следующий методы:

* void initialize() – метод устанавливаются “слушатели” для кнопок.

**2.3.12 Класс MainMenuController**

Данный класс содержит следующий атрибуты:

* dynamicsButton – кнопка для перехода в окно динамик
* enterprisesButton – кнопка для перехода в окно предприятий
* indicatorsButton – кнопка для перехода в окно показателей
* unitsButton – кнопка для перехода в окно валют

Данный класс содержит следующий методы:

* void initialize() – метод устанавливаются “слушатели” для кнопок.

**2.3.13 Класс IndicatorsController**

Данный класс содержит следующий атрибуты:

* tableView – таблица для отображения показателей
* indicatorColumn – колона для отображение именования показателей
* importanceColumn – колона для отображения важности показателей
* backButton – кнопка для возврата в главное меню

Данный класс содержит следующий методы:

* void initialize() – метод, в котором происходит инициализация всех колонок таблицы, устанавливаются “слушатели” для кнопок.

**2.3.14 Класс UnitsController**

Данный класс содержит следующий атрибуты:

* tableView – таблица для отображения валют
* unitColumn – колона для отображения имени валюты
* rateColumn – колона для отображения курса валюты
* backButton – кнопка для возврата в главное меню

Данный класс содержит следующий методы:

* void initialize() – метод, в котором происходит инициализация всех колонок таблицы, устанавливаются “слушатели” для кнопок.

**2.3.15 Класс EnterprisesController**

Данный класс содержит следующий атрибуты:

* tableView – таблица для отображения информации об предприятиях
* enterpriseColumn – колона для отображения имени предприятия
* phoneColumn – колона для отображения контактного телефона
* ownerColumn – колона для отображения имени владельца предприятия
* backButton – кнопка для возврата в главное меню

Данный класс содержит следующий методы:

* void initialize() – метод, в котором происходит инициализация всех колонок таблицы, устанавливаются “слушатели” для кнопок.

**2.3.16 Класс DynamicsController**

Данный класс содержит следующий атрибуты:

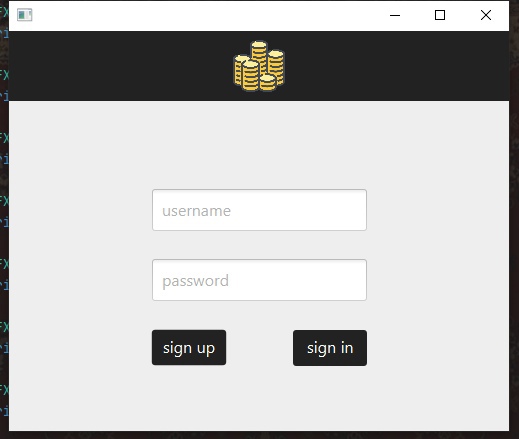
* tableView – таблица для отображения информации об динамиках
* indicatorColumn – колона для отображения именования индикатора
* valueColumn – колона для отображения значения индикатора
* unitColumn – колона для отображения валюты значения
* BYNsColumn – для отображения значения в белорусских рублях
* dateComboBox – выбор даты для отображения отчетов
* enterpriseComboBox – выбор имени предприятия для отображения отчетов
* backButton – кнопка для возврата в главное меню

Данный класс содержит следующий методы:

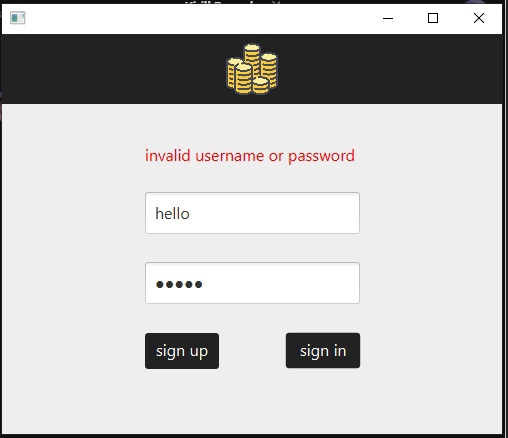
* void initialize() – метод, в котором происходит инициализация всех колонок таблицы, устанавливаются “слушатели” для кнопок.

**Часть 3. Демо программы**

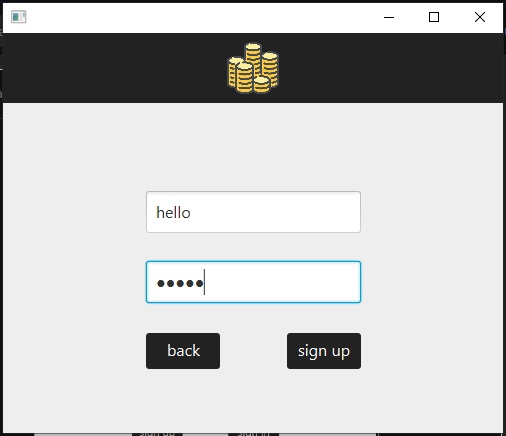
Окно логгирования

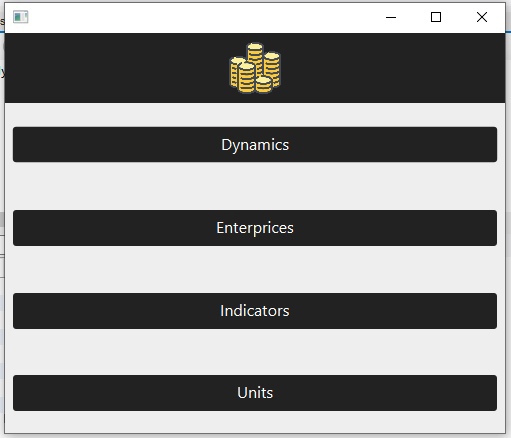


Несуществующий пользователь

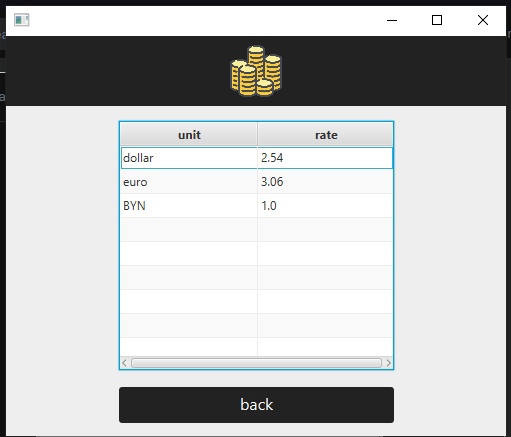


Регистрация нового пользователя

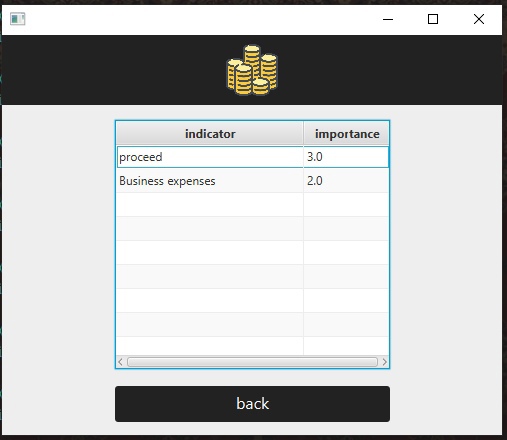


Главное меню

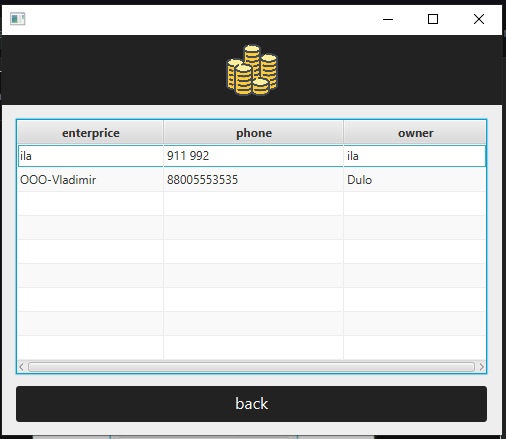
Валюты



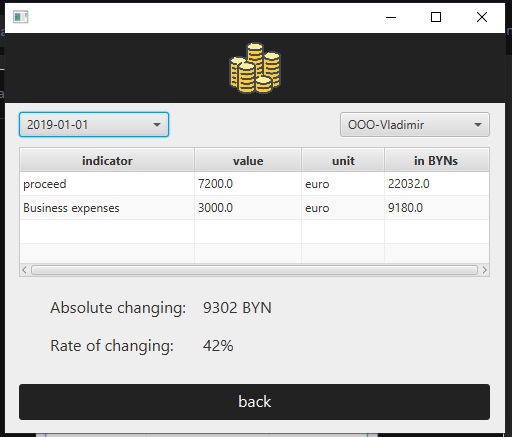
Показатели

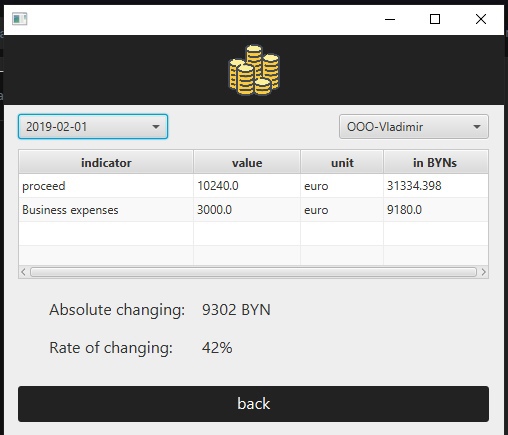


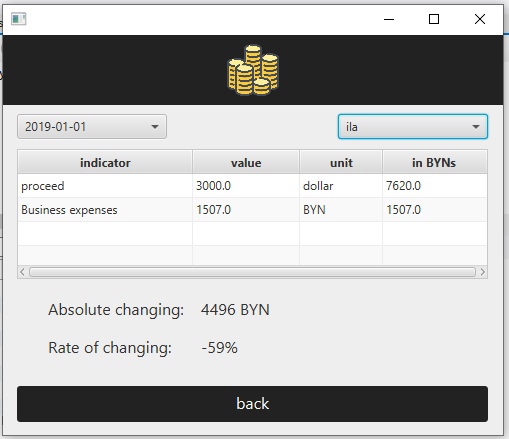
Предприятия

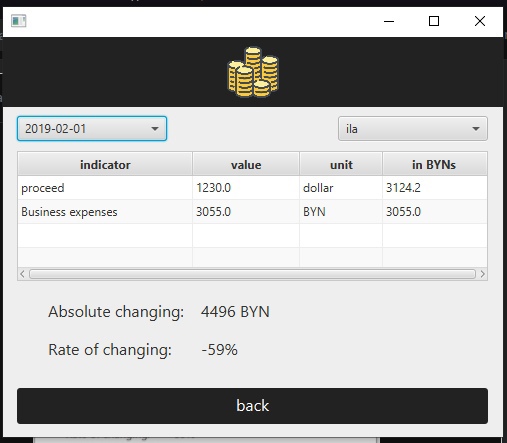


Динамики









**Заключение**

Реализована информационная система с удобным и информативным интерфейсом для хранения информации и реализации функции поиска информации в справочной системе. Благодаря понятной архитектуре и несложному управлению работать с базой данных может

и неподготовленный пользователь.

Выполненная курсовая работа на тему «Анализ динамики показателей финансовой отчетности различных предприятий» удобна в применении, понятна и доступна широкому кругу людей.

В программе реализовано:

* добавление данных в таблицы базы данных;
* удаление данных из базы данных;
* поиск данных в базе данных;

**Список использованных источников**

1. Java. Методы программирования – И. Н. Блинов, В. С. Романчик
2. openjfx.io – JavaFX
3. stackoverflow.com – Stack Overflow - Where Developers Learn, Share, & Build Careers
4. cyberforum.ru – Форум программистов и сисадминов Киберфорум
5. coderoad.ru – Вопросы – CodeRoad
6. codeproject.com – CodeProject - For those who code
7. Лекционные материалы

**Приложение**

Configs.java

package sample;  
  
public class Configs {  
 protected String dbHost = "localhost";  
 protected String dbPort = "3306";  
 protected String dbUser = "root";  
 protected String dbPass = "12345";  
 protected String dbName = "javak";  
}

DBHandler.java

package sample;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
  
import java.sql.\*;  
  
public class DBHandler extends Configs {  
 Connection dbConnection;  
  
 public Connection getDBConnection() throws SQLException, ClassNotFoundException {  
 String connectionString = "jdbc:mysql://" +  
 dbHost + ":" + dbPort + "/" + dbName;  
  
 Class.*forName*("com.mysql.jdbc.Driver");  
  
 dbConnection = DriverManager.*getConnection*(connectionString, dbUser, dbPass);  
 return dbConnection;  
 }  
  
 public float getUnitRate(String unit) {  
 ResultSet resultSet = null;  
 float res = 0;  
 String select = "SELECT \* FROM " + Units.*TABLE* + " WHERE " + Units.*NAME* + " = " + "\"" + unit + "\"";  
  
 try {  
 Statement statement = getDBConnection().createStatement();  
 resultSet = statement.executeQuery(select);  
  
 while (resultSet.next()) {  
 res = resultSet.getFloat("rate");  
 break;  
 }  
 } catch (SQLException | ClassNotFoundException | NullPointerException e) {  
 e.printStackTrace();  
 }  
  
 return res;  
 }  
  
 public void signUpUser(User user) throws SQLException, ClassNotFoundException {  
 String insert = "INSERT INTO " + Users.*TABLE* + "("  
 + Users.*USERNAME* + ", " + Users.*PASSWORD* + ")"  
 + "VALUES(?, ?)";  
  
 PreparedStatement prSt = getDBConnection().prepareStatement(insert);  
  
 prSt.setString(1, user.getUsername());  
 prSt.setString(2, user.getPassword());  
  
 try {  
 prSt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public boolean haveUser(String username) {  
 ResultSet resultSet = null;  
 String res = null;  
  
 String select = "SELECT COUNT(1) FROM " + Users.*TABLE* + " WHERE "  
 + Users.*USERNAME* + " = " + "\"" + username +"\"";  
  
 try {  
 Statement statement = getDBConnection().createStatement();  
 resultSet = statement.executeQuery(select);  
 resultSet.next();  
 res = resultSet.getString("username");  
 } catch (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 return res != null;  
 }  
  
 public ResultSet getUsers(User user) {  
 ResultSet resultSet = null;  
  
 String select = "SELECT \* FROM " + Users.*TABLE* + " WHERE "  
 + Users.*USERNAME* + "=? AND " + Users.*PASSWORD* + "=?";  
  
 try {  
 PreparedStatement prSt = getDBConnection().prepareStatement(select);  
 prSt.setString(1, user.getUsername());  
 prSt.setString(2, user.getPassword());  
  
 resultSet = prSt.executeQuery()  
; } catch (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 return resultSet;  
 }  
  
 public ObservableList<Unit> getUnits() {  
 ObservableList<Unit> res = FXCollections.*observableArrayList*();  
  
 String select = "SELECT \* FROM " + Units.*TABLE*;  
  
 try {  
 ResultSet result = null;  
 Statement statement = getDBConnection().createStatement();  
 result = statement.executeQuery(select);  
  
 while (result.next()) {  
 res.add(new Unit(result.getString("name"), result.getFloat("rate")));  
 }  
 } catch (SQLException | ClassNotFoundException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return res;  
 }  
  
 public ObservableList<Enterprise> getEnterprises() {  
 ObservableList<Enterprise> res = FXCollections.*observableArrayList*();  
  
 String select = "SELECT \* FROM " + Enterprises.*TABLE*;  
  
 try {  
 ResultSet result = null;  
 Statement statement = getDBConnection().createStatement();  
 result = statement.executeQuery(select);  
  
 while (result.next()) {  
 res.add(new Enterprise(  
 result.getString("name"),  
 result.getString("phone"),  
 result.getString("owner")));  
 }  
 } catch (SQLException | ClassNotFoundException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return res;  
 }  
  
 public ObservableList<Indicator> getIndicators() {  
 ObservableList<Indicator> res = FXCollections.*observableArrayList*();  
  
 String select = "SELECT \* FROM " + Indicators.*TABLE*;  
  
 try {  
 ResultSet result = null;  
 Statement statement = getDBConnection().createStatement();  
 result = statement.executeQuery(select);  
  
 while (result.next()) {  
 res.add(new Indicator(  
 result.getString("name"),  
 result.getFloat("importance")));  
 }  
 } catch (SQLException | ClassNotFoundException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return res;  
 }  
  
 public ObservableList<Dynamic> getDynamics() {  
 ObservableList<Dynamic> res = FXCollections.*observableArrayList*();  
  
 String select = "SELECT \* FROM " + Dynamics.*TABLE*;  
  
 try {  
 ResultSet result = null;  
 Statement statement = getDBConnection().createStatement();  
 result = statement.executeQuery(select);  
  
 while (result.next()) {  
 res.add(new Dynamic(  
 result.getString("date"),  
 result.getFloat("value"),  
 result.getString("indicator"),  
 result.getString("unit"),  
 result.getString("enterprise")));  
 }  
 } catch (SQLException | ClassNotFoundException throwables) {  
 throwables.printStackTrace();  
 }  
  
 return res;  
 }  
}

Dynamic.java

package sample;  
  
public class Dynamic {  
 private String date;  
 private float value;  
 private String indicator;  
 private String unit;  
 private String enterprise;  
  
 public Dynamic(String date, float value, String indicator,  
 String unit, String enterprise) {  
 this.date = date;  
 this.value = value;  
 this.indicator = indicator;  
 this.unit = unit;  
 this.enterprise = enterprise;  
 }  
  
 public String getEnterprise() {  
 return enterprise;  
 }  
  
 public void setEnterprise(String enterprise) {  
 this.enterprise = enterprise;  
 }  
  
 public String getDate() {  
 return date;  
 }  
  
 public void setDate(String date) {  
 this.date = date;  
 }  
  
 public float getValue() {  
 return value;  
 }  
  
 public void setValue(float value) {  
 this.value = value;  
 }  
  
 public String getIndicator() {  
 return indicator;  
 }  
  
 public void setIndicator(String indicator) {  
 this.indicator = indicator;  
 }  
  
 public String getUnit() {  
 return unit;  
 }  
  
 public void setUnit(String unit) {  
 this.unit = unit;  
 }  
}

Dynamics.java

package sample;  
  
public class Dynamics {  
 public static final String *TABLE* = "dynamics";  
 public static final String ENTERPRISE = "enterprise";  
 public static final String DATE = "date";  
 public static final String VALUE = "value";  
 public static final String INDICATOR = "indicator";  
 public static final String UNIT = "unit";  
}

DynamicsController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.text.DateFormat;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.\*;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.control.cell.TextFieldTableCell;  
import javafx.stage.Stage;  
import javafx.util.converter.FloatStringConverter;  
import sample.DBHandler;  
import sample.Dynamic;  
import sample.DynamicsRow;  
  
import static java.lang.Math.abs;  
import static java.lang.Math.round;  
  
public class DynamicsController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private TableView<DynamicsRow> tableView;  
  
 @FXML  
 private TableColumn<DynamicsRow, String> indicatorColumn;  
  
 @FXML  
 private TableColumn<DynamicsRow, Float> valueColumn;  
  
 @FXML  
 private TableColumn<DynamicsRow, String> unitColumn;  
  
 @FXML  
 private TableColumn<DynamicsRow, Float> BYNsColumn;  
  
 @FXML  
 private Button backButton;  
  
 @FXML  
 private ComboBox<String> dateComboBox;  
  
 @FXML  
 private ComboBox<String> enterpriseComboBox;  
  
 @FXML  
 private Label absoluteChangingLabel;  
  
 @FXML  
 private Label rateOfChangingLabel;  
  
 @FXML  
 void initialize() throws ParseException {  
 backButton.setOnAction(event -> {  
 backButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/MainMenu.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 ObservableList<Dynamic> dynamics = new DBHandler().getDynamics();  
  
 Set<String> enterprices = new HashSet<String>();  
 Set<String> dates = new HashSet<String>();  
  
 for (var i: dynamics) {  
 enterprices.add(i.getEnterprise());  
 dates.add(i.getDate());  
 }  
  
 for (var enterprise: enterprices) {  
 enterpriseComboBox.getItems().add(enterprise);  
 }  
 for (var date: dates) {  
 dateComboBox.getItems().add(date);  
 }  
  
 Map<String, Map<String, ObservableList<DynamicsRow>>> info = new HashMap<String, Map<String, ObservableList<DynamicsRow>>>();  
  
 for (var i: enterprices) {  
 info.put(i, new HashMap<String, ObservableList<DynamicsRow>>());  
  
 for (var j: dates) {  
 info.get(i).put(j, FXCollections.observableArrayList());  
 }  
 }  
  
 DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd", Locale.ENGLISH);  
 Date last = dateFormat.parse("1900-00-00");  
 Date preLast = last;  
  
 for (var i: dynamics) {  
 String indicator = i.getIndicator();  
 float value = i.getValue();  
 String unit = i.getUnit();  
 Date date = dateFormat.parse(i.getDate());  
  
 if (date.after(last)) {  
 last = date;  
 }  
  
 var obj = info.get(i.getEnterprise()).get(i.getDate());  
 obj.add(new DynamicsRow(indicator, value, unit));  
 }  
  
 for (var i: dynamics) {  
 Date date = dateFormat.parse(i.getDate());  
  
 if (last.after(date) && date.after(preLast)) {  
 preLast = date;  
 }  
 }  
  
 indicatorColumn.setCellFactory(TextFieldTableCell.forTableColumn());  
 valueColumn.setCellFactory(TextFieldTableCell.forTableColumn(new FloatStringConverter()));  
 unitColumn.setCellFactory(TextFieldTableCell.forTableColumn());  
 BYNsColumn.setCellFactory(TextFieldTableCell.forTableColumn(new FloatStringConverter()));  
  
 indicatorColumn.setCellValueFactory(new PropertyValueFactory<>("indicator"));  
 valueColumn.setCellValueFactory(new PropertyValueFactory<>("value"));  
 unitColumn.setCellValueFactory(new PropertyValueFactory<>("unit"));  
 BYNsColumn.setCellValueFactory(new PropertyValueFactory<>("BYNs"));  
  
 final String lastd = dateFormat.format(last);  
 final String preLastd = dateFormat.format(preLast);  
  
 enterpriseComboBox.setOnAction(e -> {  
 if (dateComboBox.getValue() != null && !dateComboBox.getValue().equals(dateComboBox.getPromptText())) {  
 tableView.setItems(info.get(enterpriseComboBox.getValue()).get(dateComboBox.getValue()));  
 }  
  
 float val1 = info.get(enterpriseComboBox.getValue()).get(lastd).get(0).getBYNs();  
 float val2 = info.get(enterpriseComboBox.getValue()).get(preLastd).get(0).getBYNs();  
  
 absoluteChangingLabel.setText(String.valueOf(round(abs(val1-val2))) + " BYN");  
 rateOfChangingLabel.setText(String.valueOf(round((val1-val2)\*100/val2)) + "%");  
 });  
  
  
 dateComboBox.setOnAction(e -> {  
 if (enterpriseComboBox.getValue() != null && !enterpriseComboBox.getValue().equals(enterpriseComboBox.getPromptText())) {  
 tableView.setItems(info.get(enterpriseComboBox.getValue()).get(dateComboBox.getValue()));  
 }  
 });  
 }  
}

DynamicsRow.java

package sample;  
  
import sample.DBHandler;  
  
public class DynamicsRow {  
 private String indicator;  
 private float value;  
 private String unit;  
 private float BYNs;  
  
 public DynamicsRow() {}  
  
 public DynamicsRow(String indicator, float value, String unit) {  
 this.indicator = indicator;  
 this.value = value;  
 this.unit = unit;  
 updateBYNs();  
 }  
  
 public float getBYNs() {  
 return BYNs;  
 }  
  
 public String getIndicator() {  
 return indicator;  
 }  
  
 private void updateBYNs() {  
 this.BYNs = this.value \* new DBHandler().getUnitRate(this.unit);  
 }  
  
 public void setIndicator(String indicator) {  
 this.indicator = indicator;  
 }  
  
 public float getValue() {  
 return value;  
 }  
  
 public void setValue(float value) {  
 this.value = value;  
 updateBYNs();  
 }  
  
 public String getUnit() {  
 return unit;  
 }  
  
 public void setUnit(String unit) {  
 this.unit = unit;  
 }  
}

EnterpricesController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.control.cell.TextFieldTableCell;  
import javafx.stage.Stage;  
import sample.DBHandler;  
import sample.Enterprise;  
  
public class EnterpricesController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private TableView<Enterprise> tableView;  
  
 @FXML  
 private TableColumn<Enterprise, String> enterpriseColumn;  
  
 @FXML  
 private TableColumn<Enterprise, String> phoneColumn;  
  
 @FXML  
 private TableColumn<Enterprise, String> ownerColumn;  
  
 @FXML  
 private Button backButton;  
  
 @FXML  
 void initialize() {  
 var dbHandler = new DBHandler();  
  
 enterpriseColumn.setCellFactory(TextFieldTableCell.*forTableColumn*());  
 phoneColumn.setCellFactory(TextFieldTableCell.*forTableColumn*());  
 ownerColumn.setCellFactory(TextFieldTableCell.*forTableColumn*());  
  
 enterpriseColumn.setCellValueFactory(new PropertyValueFactory<>("name"));  
 phoneColumn.setCellValueFactory(new PropertyValueFactory<>("phone"));  
 ownerColumn.setCellValueFactory(new PropertyValueFactory<>("owner"));  
  
 tableView.setItems(dbHandler.getEnterprises());  
  
 backButton.setOnAction(event -> {  
 backButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/MainMenu.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
 }  
}

Enterprise.java

package sample;  
  
public class Enterprise {  
 private String name;  
 private String phone;  
 private String owner;  
  
 public Enterprise(String name, String phone, String owner) {  
 this.name = name;  
 this.phone = phone;  
 this.owner = owner;  
 }  
  
 public void setName(String value) {  
 name = value;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setPhone(String value) {  
 phone = value;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setOwner(String value) {  
 owner = value;  
 }  
  
 public String getOwner() {  
 return owner;  
 }  
}

Enterprises.java

package sample;  
  
public class Enterprises {  
 public static final String TABLE = "enterprises";  
 public static final String NAME = "name";  
 public static final String PHONE = "phone";  
 public static final String OWNER = "owner";  
}

Indicator.java

package sample;  
  
public class Indicator {  
 private String name;  
 private float importance;  
  
 public Indicator(String name, float importance) {  
 this.name = name;  
 this.importance = importance;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String value) {  
 name = value;  
 }  
  
 public float getImportance() {  
 return importance;  
 }  
  
 public void setImportance(float value) {  
 importance = value;  
 }  
}

Indicators.java

package sample;  
  
public class Indicators {  
 public static final String *TABLE* = "indicators";  
 public static final String *NAME* = "name";  
 public static final String *IMPORTANCE* = "importance";  
  
}

IndicatorsController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.control.cell.TextFieldTableCell;  
import javafx.stage.Stage;  
import javafx.util.converter.FloatStringConverter;  
import sample.DBHandler;  
import sample.Indicator;  
  
public class IndicatorsController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private TableView<Indicator> tableView;  
  
 @FXML  
 private TableColumn<Indicator, String> indicatorColumn;  
  
 @FXML  
 private TableColumn<Indicator, Float> importanceColumn;  
  
 @FXML  
 private Button backButton;  
  
 @FXML  
 void initialize() {  
 var dbHandler = new DBHandler();  
  
 indicatorColumn.setCellFactory(TextFieldTableCell.*forTableColumn*());  
 importanceColumn.setCellFactory(TextFieldTableCell.*forTableColumn*(new FloatStringConverter()));  
  
 indicatorColumn.setCellValueFactory(new PropertyValueFactory<>("name"));  
 importanceColumn.setCellValueFactory(new PropertyValueFactory<>("importance"));  
  
 tableView.setItems(dbHandler.getIndicators());  
  
 backButton.setOnAction(event -> {  
 backButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/MainMenu.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
 }  
}

Main.java

package sample;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage primaryStage) throws Exception{  
 Parent root = FXMLLoader.*load*(getClass().getResource("Forms/SignIn.fxml"));  
  
 primaryStage.setTitle("");  
 primaryStage.setScene(new Scene(root, 500, 400));  
 primaryStage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

MainMenuController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.stage.Stage;  
  
public class MainMenuController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private Label label;  
  
 @FXML  
 private Button dynamicsButton;  
  
 @FXML  
 private Button enterpricesButton;  
  
 @FXML  
 private Button indicatorsButton;  
  
 @FXML  
 private Button unitsButton;  
  
 @FXML  
 void initialize() {  
 dynamicsButton.setOnAction(event -> {  
 dynamicsButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/Dynamics.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 indicatorsButton.setOnAction(event -> {  
 indicatorsButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/Indicators.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 enterpricesButton.setOnAction(event -> {  
 enterpricesButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/Enterprises.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 unitsButton.setOnAction(event -> {  
 unitsButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/Units.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
 }  
}

SignInController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ResourceBundle;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
  
public class SignInController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private Button signUpButton;  
  
 @FXML  
 private TextField usernameField;  
  
 @FXML  
 private PasswordField passwordField;  
  
 @FXML  
 private Button signInButton;  
  
 @FXML  
 private Label label;  
  
 @FXML  
 void initialize() {  
 signUpButton.setOnAction(event -> {  
 signUpButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/SignUp.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 signInButton.setOnAction(event -> {  
 String username = usernameField.getText().trim();  
 String password = passwordField.getText().trim();  
  
 if (!username.equals("") && !password.equals("")) {  
 loginUser(username, password);  
 }  
 else {  
 label.setText("fill both fields!");  
 }  
 });  
  
 }  
  
 private void loginUser(String username, String password) {  
 DBHandler dbHandler = new DBHandler();  
 User user = new User();  
 user.setUsername(username);  
 user.setPassword(password);  
 ResultSet res = dbHandler.getUsers(user);  
  
 int count = 0;  
  
 while (true) {  
 try {  
 if (!res.next()) break;  
 } catch ( NullPointerException ex) {  
 break;  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 count++;  
 }  
  
 if (count >= 1) {  
 label.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/MainMenu.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 }  
 else {  
 label.setText("invalid username or password");  
 }  
 }  
}

SignUpController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.sql.SQLException;  
import java.util.ResourceBundle;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
import sample.DBHandler;  
import sample.User;  
  
public class SignUpController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private TextField signUpUsernameField;  
  
 @FXML  
 private PasswordField signUpPasswordField;  
  
 @FXML  
 private Button signUpButton;  
  
 @FXML  
 private Button backButton;  
  
 @FXML  
 private Label label;  
  
 @FXML  
 void initialize() {  
 backButton.setOnAction(actionEvent -> {  
 backButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/SignIn.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
  
 signUpButton.setOnAction(event -> {  
 try {  
 if (!signUpUsernameField.getText().equals("") && !signUpPasswordField.getText().equals("")) {  
 String username = signUpUsernameField.getText();  
  
 if (new DBHandler().haveUser(username)) {  
 label.setText("username is already taken!");  
 return;  
 }  
 signUpNewUser();  
 } else {  
 label.setText("Fill both fields!");  
 }  
  
 signUpButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/SignIn.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
  
 } catch (SQLException | ClassNotFoundException throwables) {  
 throwables.printStackTrace();  
 }  
 });  
 }  
  
 private void signUpNewUser() throws SQLException, ClassNotFoundException {  
 DBHandler dbHandler = new DBHandler();  
 String username = signUpUsernameField.getText();  
 String password = signUpPasswordField.getText();  
  
 User user = new User(username, password);  
  
 dbHandler.signUpUser(user);  
 }  
}

Unit.java

package sample;  
  
public class Unit {  
 private final String name;  
 private float rate;  
  
 public Unit(String name, float rate) {  
 this.name = name;  
 this.rate = rate;  
 }  
  
 public float getRate() {  
 return rate;  
 }  
  
 public void setRate(float value) {  
 this.rate = value;  
 }  
  
 public String getName() {  
 return name;  
 }  
}

Units.java

package sample;  
  
public class Units {  
 public static final String *TABLE* = "units";  
 public static final String *NAME* = "name";  
 public static final String *RATE* = "rate";  
}

UnitsController.java

package sample;  
  
import java.io.IOException;  
import java.net.URL;  
import java.util.ResourceBundle;  
  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.control.cell.TextFieldTableCell;  
import javafx.stage.Stage;  
import javafx.util.converter.FloatStringConverter;  
import sample.DBHandler;  
import sample.Unit;  
  
public class UnitsController {  
  
 @FXML  
 private ResourceBundle resources;  
  
 @FXML  
 private URL location;  
  
 @FXML  
 private TableView<Unit> tableView;  
  
 @FXML  
 private TableColumn<Unit, String> unitColumn;  
  
 @FXML  
 private TableColumn<Unit, Float> rateColumn;  
  
 @FXML  
 private Button backButton;  
  
 @FXML  
 void initialize() {  
 var dbHandler = new DBHandler();  
  
 unitColumn.setCellFactory(TextFieldTableCell.*forTableColumn*());  
 rateColumn.setCellFactory(TextFieldTableCell.*forTableColumn*(new FloatStringConverter()));  
  
 unitColumn.setCellValueFactory(new PropertyValueFactory<>("name"));  
 rateColumn.setCellValueFactory(new PropertyValueFactory<>("rate"));  
  
 tableView.setItems(dbHandler.getUnits());  
  
 backButton.setOnAction(event -> {  
 backButton.getScene().getWindow().hide();  
  
 FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/Forms/MainMenu.fxml"));  
  
 try {  
 loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 Parent root = loader.getRoot();  
 Stage stage = new Stage();  
  
 stage.setScene(new Scene(root));  
 stage.show();  
 });  
 }  
}

User.java

package sample;  
  
public class User {  
 private String username;  
 private String password;  
  
 public User(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 public User() {}  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
}

Users.java

package sample;  
  
public class Users {  
 public static final String TABLE = "users";  
 public static final String USERNAME = "username";  
 public static final String PASSWORD = "password";  
}

Dynamics.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.ComboBox*?>  
<?*import javafx.scene.control.Label*?>  
<?*import javafx.scene.control.TableColumn*?>  
<?*import javafx.scene.control.TableView*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.DynamicsController">  
 <children>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <TableView fx:id="tableView" layoutX="14.0" layoutY="114.0" prefHeight="130.0" prefWidth="471.0">  
 <columns>  
 <TableColumn fx:id="indicatorColumn" prefWidth="175.0" text="indicator" />  
 <TableColumn fx:id="valueColumn" minWidth="0.0" prefWidth="107.0" text="value" />  
 <TableColumn fx:id="unitColumn" minWidth="0.0" prefWidth="83.0" text="unit" />  
 <TableColumn fx:id="BYNsColumn" minWidth="0.0" prefWidth="105.0" text="in BYNs " />  
 </columns>  
 </TableView>  
 <Button fx:id="backButton" layoutX="14.0" layoutY="351.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="471.0" style="-fx-background-color: #222222;" text="back" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <ComboBox fx:id="dateComboBox" layoutX="14.0" layoutY="79.0" prefWidth="150.0" promptText="Date" />  
 <ComboBox fx:id="enterpriseComboBox" layoutX="335.0" layoutY="79.0" prefWidth="150.0" promptText="Enterprise" />  
 <Label layoutX="45.0" layoutY="262.0" text="Absolute changing:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <Label layoutX="45.0" layoutY="300.0" text="Rate of changing:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <Label fx:id="absoluteChangingLabel" layoutX="198.0" layoutY="262.0">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <Label fx:id="rateOfChangingLabel" layoutX="198.0" layoutY="300.0">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

Enterprises.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.TableColumn*?>  
<?*import javafx.scene.control.TableView*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.EnterpricesController">  
 <children>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <TableView fx:id="tableView" layoutX="14.0" layoutY="84.0" prefHeight="255.0" prefWidth="471.0">  
 <columns>  
 <TableColumn fx:id="enterpriseColumn" prefWidth="147.0" text="enterprice" />  
 <TableColumn fx:id="phoneColumn" minWidth="0.0" prefWidth="180.0" text="phone" />  
 <TableColumn fx:id="ownerColumn" minWidth="0.0" prefWidth="143.0" text="owner" />  
 </columns>  
 </TableView>  
 <Button fx:id="backButton" layoutX="14.0" layoutY="351.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="471.0" style="-fx-background-color: #222222;" text="back" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 </children>  
</AnchorPane>

Indicators.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.TableColumn*?>  
<?*import javafx.scene.control.TableView*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.IndicatorsController">  
 <children>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <TableView fx:id="tableView" layoutX="113.0" layoutY="85.0" prefHeight="249.0" prefWidth="275.0">  
 <columns>  
 <TableColumn fx:id="indicatorColumn" prefWidth="188.0" text="indicator" />  
 <TableColumn fx:id="importanceColumn" minWidth="0.0" prefWidth="86.0" text="importance" />  
 </columns>  
 </TableView>  
 <Button fx:id="backButton" layoutX="113.0" layoutY="351.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="275.0" style="-fx-background-color: #222222;" text="back" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 </children>  
</AnchorPane>

MainMenu.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.Label*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.layout.ColumnConstraints*?>  
<?*import javafx.scene.layout.GridPane*?>  
<?*import javafx.scene.layout.RowConstraints*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.MainMenuController">  
 <children>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <Label fx:id="label" layoutX="143.0" layoutY="110.0" style="-fx-background-color: #EEEEEE;" textFill="RED">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <GridPane layoutX="8.0" layoutY="70.0" prefHeight="330.0" prefWidth="484.0">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <Button fx:id="dynamicsButton" mnemonicParsing="false" prefHeight="35.0" prefWidth="486.0" style="-fx-background-color: #222222;" text="Dynamics" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Button fx:id="enterpricesButton" mnemonicParsing="false" prefHeight="35.0" prefWidth="507.0" style="-fx-background-color: #222222;" text="Enterprices" textFill="WHITE" GridPane.rowIndex="1">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Button fx:id="indicatorsButton" mnemonicParsing="false" prefHeight="35.0" prefWidth="545.0" style="-fx-background-color: #222222;" text="Indicators" textFill="WHITE" GridPane.rowIndex="2">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Button fx:id="unitsButton" mnemonicParsing="false" prefHeight="35.0" prefWidth="558.0" style="-fx-background-color: #222222;" text="Units" textFill="WHITE" GridPane.rowIndex="3">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 </children>  
 </GridPane>  
 </children>  
</AnchorPane>

SignIn.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.Label*?>  
<?*import javafx.scene.control.PasswordField*?>  
<?*import javafx.scene.control.TextField*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.SignInController">  
 <children>  
 <Button fx:id="signUpButton" layoutX="143.0" layoutY="299.0" mnemonicParsing="false" style="-fx-background-color: #222222;" text="sign up" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <TextField fx:id="usernameField" layoutX="143.0" layoutY="158.0" prefHeight="42.0" prefWidth="215.0" promptText="username">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <PasswordField fx:id="passwordField" layoutX="143.0" layoutY="228.0" prefHeight="42.0" prefWidth="215.0" promptText="password">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </PasswordField>  
 <Button fx:id="signInButton" layoutX="284.0" layoutY="299.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="74.0" style="-fx-background-color: #222222;" text="sign in" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <Label fx:id="label" layoutX="143.0" layoutY="109.0" prefHeight="25.0" prefWidth="215.0" style="-fx-background-color: #EEEEEE;" textFill="RED">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

SignUp.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.Label*?>  
<?*import javafx.scene.control.PasswordField*?>  
<?*import javafx.scene.control.TextField*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.SignUpController">  
 <children>  
 <TextField fx:id="signUpUsernameField" layoutX="143.0" layoutY="158.0" prefHeight="42.0" prefWidth="215.0" promptText="username">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <PasswordField fx:id="signUpPasswordField" layoutX="143.0" layoutY="228.0" prefHeight="42.0" prefWidth="215.0" promptText="password">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </PasswordField>  
 <Button fx:id="signUpButton" layoutX="284.0" layoutY="300.0" mnemonicParsing="false" style="-fx-background-color: #222222;" text="sign up" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <Button fx:id="backButton" layoutX="143.0" layoutY="300.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="74.0" style="-fx-background-color: #222222;" text="back" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Label fx:id="label" layoutX="143.0" layoutY="110.0" style="-fx-background-color: #EEEEEE;" textFill="RED">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

Units.fxml

*<?*xml version="1.0" encoding="UTF-8"*?>  
  
<?*import javafx.scene.control.Button*?>  
<?*import javafx.scene.control.TableColumn*?>  
<?*import javafx.scene.control.TableView*?>  
<?*import javafx.scene.image.Image*?>  
<?*import javafx.scene.image.ImageView*?>  
<?*import javafx.scene.layout.AnchorPane*?>  
<?*import javafx.scene.text.Font*?>*<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="500.0" style="-fx-background-color: #EEEEEE;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.UnitsController">  
 <children>  
 <AnchorPane prefHeight="70.0" prefWidth="500.0" style="-fx-background-color: #222222;">  
 <children>  
 <ImageView fitHeight="58.0" fitWidth="69.0" layoutX="221.0" layoutY="6.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../assets/1.png" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 <TableView fx:id="tableView" layoutX="113.0" layoutY="85.0" prefHeight="249.0" prefWidth="275.0">  
 <columns>  
 <TableColumn fx:id="unitColumn" prefWidth="138.0" text="unit" />  
 <TableColumn fx:id="rateColumn" minWidth="0.0" prefWidth="136.0" text="rate" />  
 </columns>  
 </TableView>  
 <Button fx:id="backButton" layoutX="113.0" layoutY="351.0" mnemonicParsing="false" prefHeight="35.0" prefWidth="275.0" style="-fx-background-color: #222222;" text="back" textFill="WHITE">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 </children>  
</AnchorPane>