

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Университет управления «ТИСБИ»

Факультет «Информационных технологий»

Кафедра «Информационных технологий»

КУРСОВАЯ РАБОТА

по предмету «Управление базами данных»

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ОБЪЕКТНОЙ БАЗЫ ДАННЫХ
ИНФОРМАЦИОННОЙ СИСТЕМЫ
“Ресторан”**

Выполнил студент:

3 курса П-711 группы

Хайруллин Булат Маратович

Научный руководитель

Таренко Людмила Борисовна,

Кандидат педагогических наук, декан, доцент.

Казань 2019

Оглавление

Введение.....	3
1.Постановка задачи.....	5
1.1. Теоретические сведения по проектированию реляционных баз данных.....	5
1.2. Анализ и формализация предметной области: выявление основных сущностей и связей.....	8
1.3.Модель сущность-связь для базы данных задачи.....	12
1.4. Схема реляционной базы данных задачи.....	13
2. Разработка программного приложения.....	18
2.1. Описание структуры и основных методов (процедур и функций) приложения.....	18
2.2 Особенности проектирования пользовательского интерфейса.....	25
2.3. Результаты тестирования с примерами визуальных форм приложения.....	27
2.4. Руководство пользователю.....	32
Список литературы.....	44
Приложение 1.....	46
Приложение 2.....	53

1. Постановка задачи

Разработать базу данных «Ресторан», содержащую сведения о блюдах, заказах и официантах.

Написать и отладить приложение, реализующее просмотр базы данных, добавление и удаление, изменение записей, а также поиск по нужным запросам.

Пользователем данной программы является метрдотель (менеджер зала), координирующий работу обслуживания посетителей ресторана.

Цель курсовой работы – приобретение студентом практических навыков по формулированию требований к разрабатываемым базам данных, построению их моделей и разработке программных приложений.

При выполнении курсовой работы необходимо выполнить:

- Провести исследование предметной области (объекта исследования);
- На основе анализа предметной области сформулировать перечень задач;
- Построить инфологическую модель объекта учета;
- Построить логическую модель объекта;
- Спроектировать и построить в среде выбранной СУБД физическую компьютерную структуру данных;
- Разработать и отладить программное приложение для работы с БД;
- Заполнить таблицы контрольными данными и провести тестирование;
- Реализовать необходимые функции добавления, удаления, поиска и т.д. (умение работать с большим объемом информации, осуществлять обор информации по определенным признакам с использованием ИТ);
- оформить пояснительную записку.

Основные требования, предъявляемые к разрабатываемому приложению:

- Количество таблиц – не менее 4.
- Наличие связанных таблиц.
- Наличие вычисляемых столбцов.
- Размер таблиц – не менее 10 записей.

Для создания и редактирования базы данных был выбран MySQL.

Для доступа к базе данных используется JDBC (Подключение к Базе Данных Java), для которой была импортирована библиотека `mysql-connector-java-5.1.48.jar`.

1.1. Теоретические сведения по проектированию реляционных баз данных

Для того чтобы спроектировать базу данных необходимо перейти от неформального описания структуры к описанию с конкретными моделями. Для этого необходимо разделить проектирование на 4 этапа:

1. Провести системный анализ области с изучением структуры и выявлением объектов.
2. Формализованное описание в ER-модели. Структура базы данных, которая будет представлена в виде модели "сущность-связь".
3. Логическое проектирование, сравнительный анализ.
4. Физическое проектирование, т.е. методы доступа, хранение данных и т.д.

1.2. Анализ и формализация предметной области: выявление основных сущностей и связей

Ресторан – предприятие общественного питания с широким ассортиментом блюд сложного приготовления, включая заказные и фирменные.

В конечном счёте необходимо создать 4 таблицы:

- 1) Официант
- 2) Заказ
- 3) Заказ из меню
- 4) Блюдо

Таблицам соответствуют следующие атрибуты:

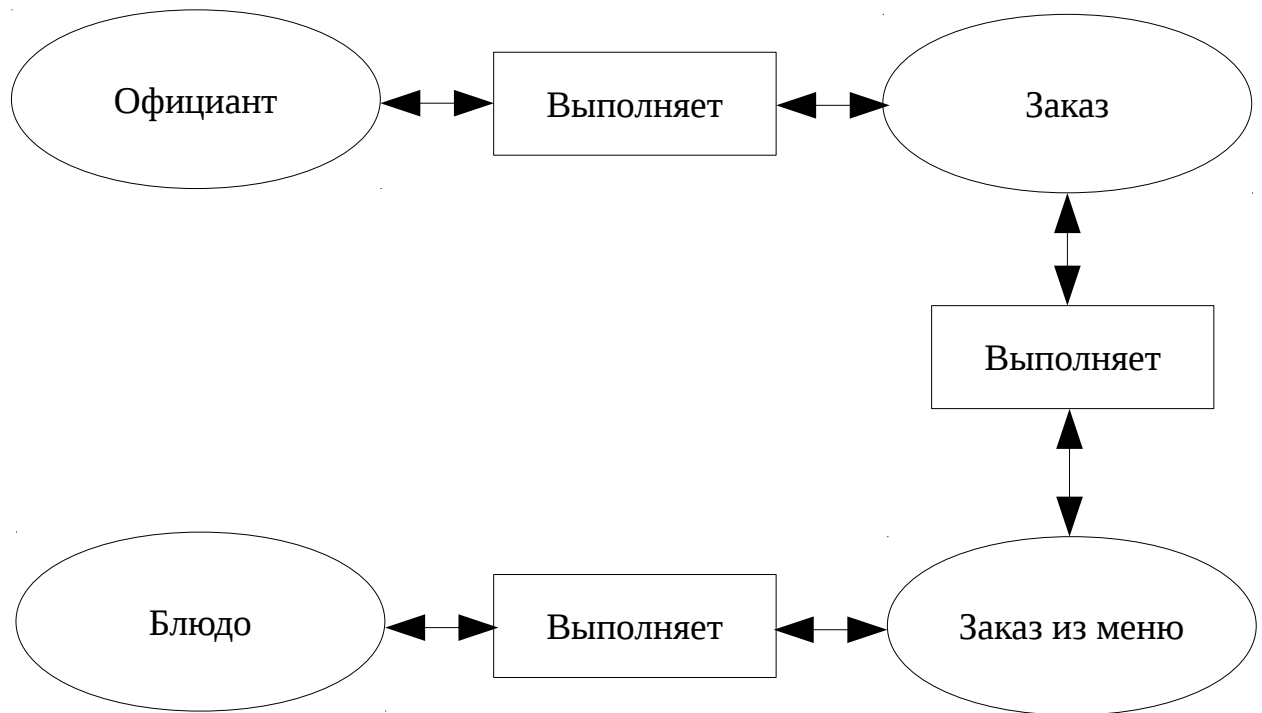
Официант: код официанта, имя, фамилия, отчество.

Заказ: код заказа, дата заказа, номер столика, код официанта, общая сумма.

Заказ из меню: код заказа из меню, код заказа, код блюда, количество, сумма.

Блюдо: код блюда, название, цена

1.3. Модель сущность-связь для базы данных задачи



1.4. Схема реляционной базы данных

Таблицы в MySQL Workbench.

Таблица «Официант»

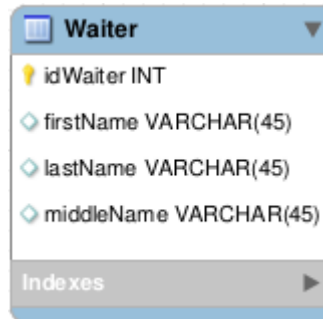


рис 1.4.1. Таблица «Waiter»

Таблица «Заказ»

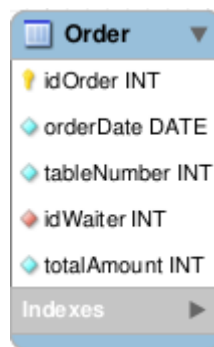


рис 1.4.2. Таблица «Order»

Таблица «Заказ из меню»

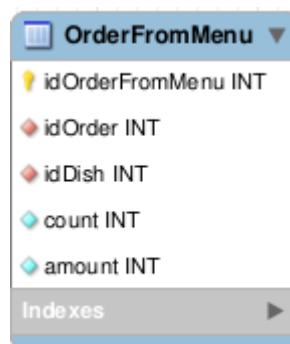


рис 1.4.3. Таблица «OrderFromMenu»

Таблица «Блюдо»



рис 1.4.4. Таблица «Dish»

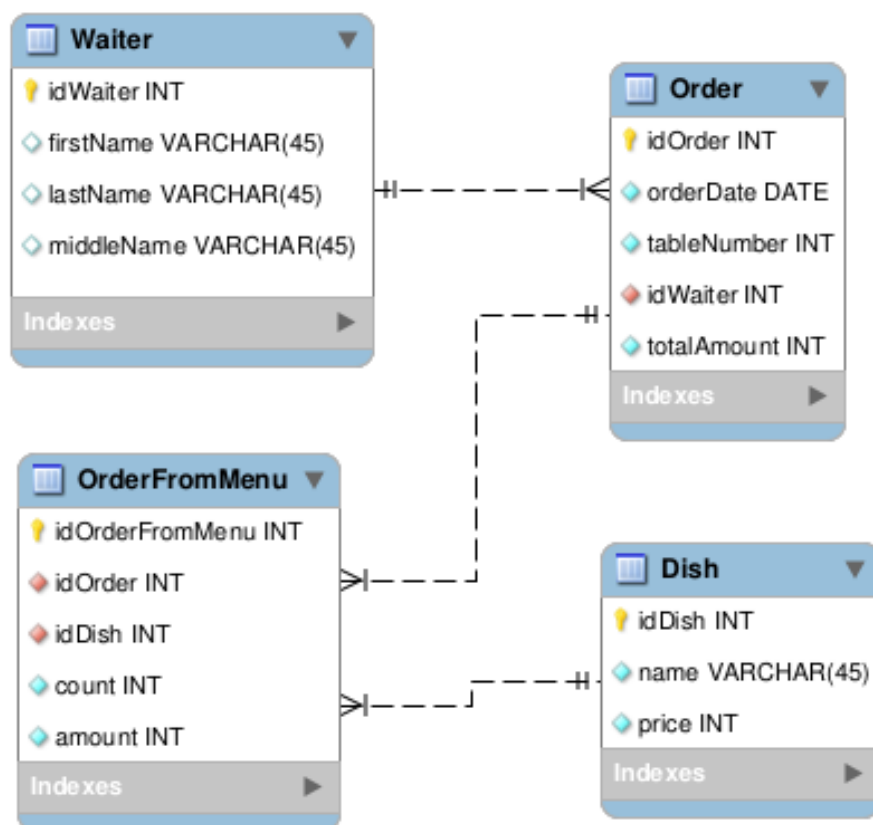


рис 1.4.5. Реляционная схема данных

2. Разработка программного приложения

2.1. Описание структуры и основных методов (процедур и функций) приложения; описание разработки интерфейса приложения

Для создания и редактирования базы данных был выбран MySQL. Для доступа к базе данных используется JDBC (Подключение к Базе Данных Java), для которой была импортирована библиотека `mysql-connector-java-5.1.48.jar`.

В самой программе для связи с базой данных используется классы из пакета `java.sql`. С помощью JDBC устанавливается соединение через класс `Connection` к MySQL к локальному серверу с базой данных “restaurant”. Для выполнения запросов есть класс `Statement`. Для получения самих результатов из запроса используется класс `ResultSet`.

Сам проект состоит из основных 2 классов: `DBProcessor` и `MainFrameGUI`.

Основной класс - `MainFrameGUI`, который нужен для запуска самой программы.

2.2. Особенности проектирования пользовательского интерфейса

Пользовательский интерфейс –это внешний вид продукта, способ общения между пользователем и программой. Интерфейс состоит из трех основополагающих частей таких как подача информации пользователю, взаимодействие и взаимосвязь с объектами. В данном приложении обмен информацией осуществляется через таблицы, которые генерируются пользователем и компьютером с помощью средств ввода и вывода, соответственно. Приложение имеет одно окно работы. Это окно содержит все необходимые для работы пользователя компоненты. В число этих компонентов входят таблицы, кнопки, вкладки и текстовые поля. Открытие приложения представит нам следующего вида окно:

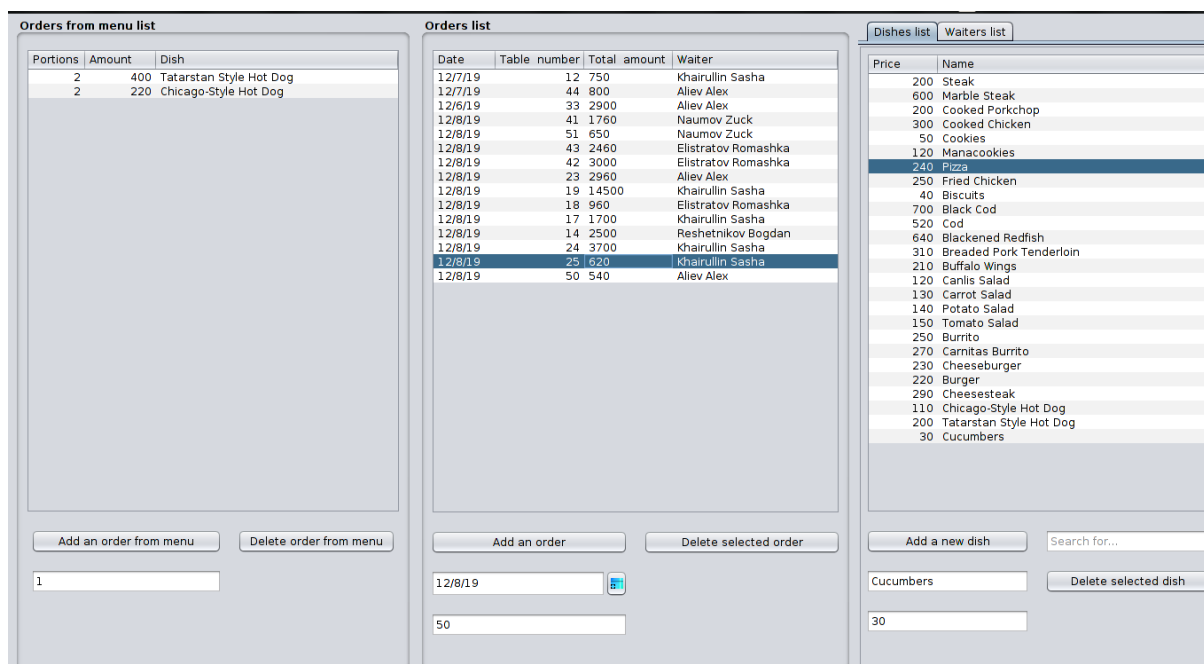


рис. 2.2.1. Вид приложения при запуске

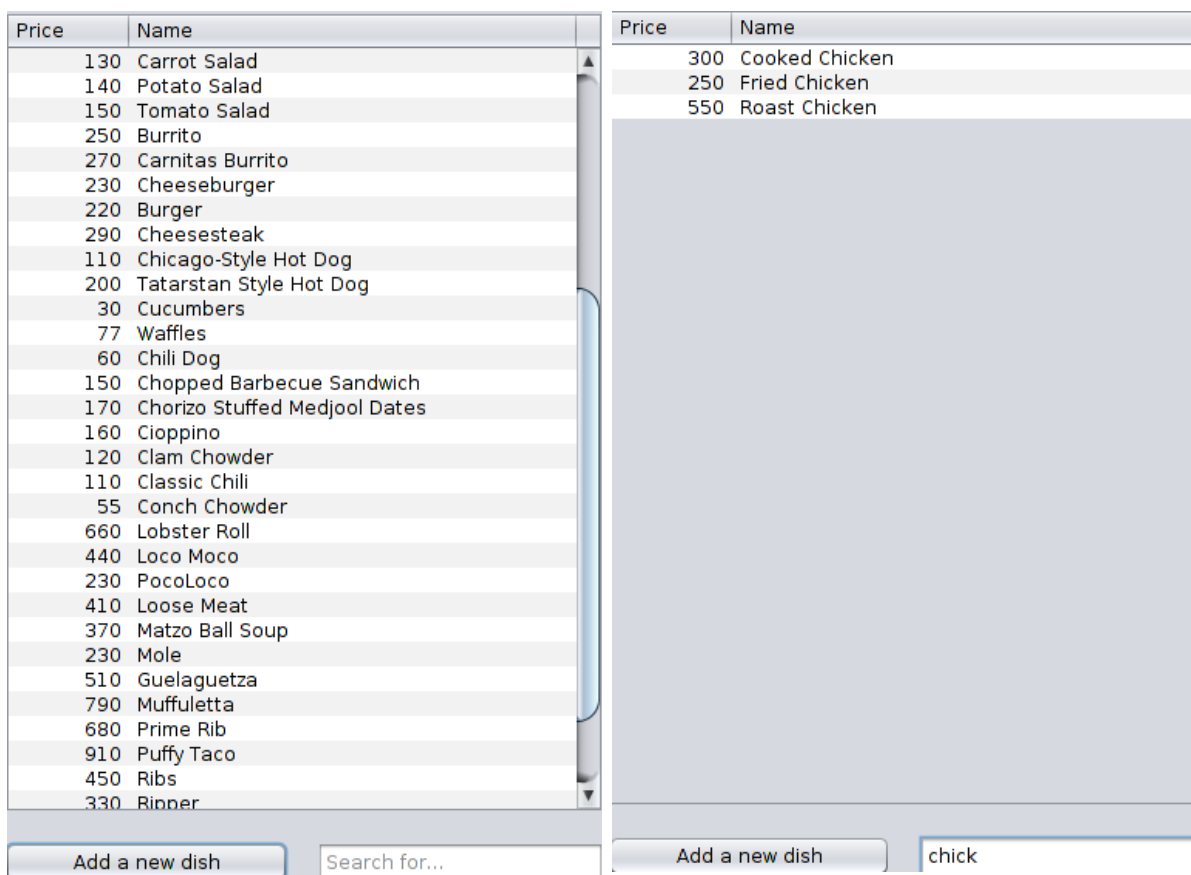
Приложение дает пользователю возможность выполнения некоторых действий (поиск, добавление, удаление, изменение), для которых пользователь определяет соответствующие данные и следствием выполнения которых является получение желаемых результатов.

Интерфейс позволяет выбрать необходимые операции из предложенных, предполагая реализацию множества сценариев работы, последовательность которых определяется самим пользователем

2.3. Результаты тестирования с примерами визуальных форм приложения

Результатом выполнения курсовой работы стало разработанное приложение баз данных, позволяющее оформлять заказы и управлять их содержимым, хранить и обрабатывать информацию о официантах и блюдах.

В программе осуществлен поиск по таблицам через ввод символов.



Price	Name
130	Carrot Salad
140	Potato Salad
150	Tomato Salad
250	Burrito
270	Carnitas Burrito
230	Cheeseburger
220	Burger
290	Cheesesteak
110	Chicago-Style Hot Dog
200	Tatarstan Style Hot Dog
30	Cucumbers
77	Waffles
60	Chili Dog
150	Chopped Barbecue Sandwich
170	Chorizo Stuffed Medjool Dates
160	Cioppino
120	Clam Chowder
110	Classic Chili
55	Conch Chowder
660	Lobster Roll
440	Loco Moco
230	PocoLoco
410	Loose Meat
370	Matzo Ball Soup
230	Mole
510	Guelaguetza
790	Muffuletta
680	Prime Rib
910	Puffy Taco
450	Ribs
330	Rinner

Price	Name
300	Cooked Chicken
250	Fried Chicken
550	Roast Chicken

At the bottom of each table is a button labeled "Add a new dish" and a search input field. In the right table, the search field contains the text "chick".

Рис 2.3.1. Демонстрация работы поиска по таблице Блюда.

First Name	Last Name	Middle Name
Sasha	Khairullin	Maratovich
Alex	Aliev	Alexandrovich
Bogdan	Reshetnikov	Bogdanovich
Zuck	Naumov	Zakharov
Romashka	Elistratov	Romanov

First Name	Last Name	Middle Name
Sasha	Khairullin	Maratovich
Alex	Aliev	Alexandrovich
Bogdan	Reshetnikov	Bogdanovich
Zuck	Naumov	Zakharov
Romashka	Elistratov	Romanov
John	Smith	

Рис. 2.3.2 Демонстрация добавления нового официанта в таблицу

Orders from menu list

Portions	Amount	Dish
1	150	Tomato Salad
1	140	Potato Salad
1	130	Carrot Salad
1	120	Canlis Salad

Orders from menu list

Portions	Amount	Dish
1	150	Tomato Salad
1	140	Potato Salad
1	130	Carrot Salad
1	120	Canlis Salad
1	250	Burrito

Рис. 2.3.3. Результат добавления нового блюда в состав заказа.

2.4. Руководство пользователя

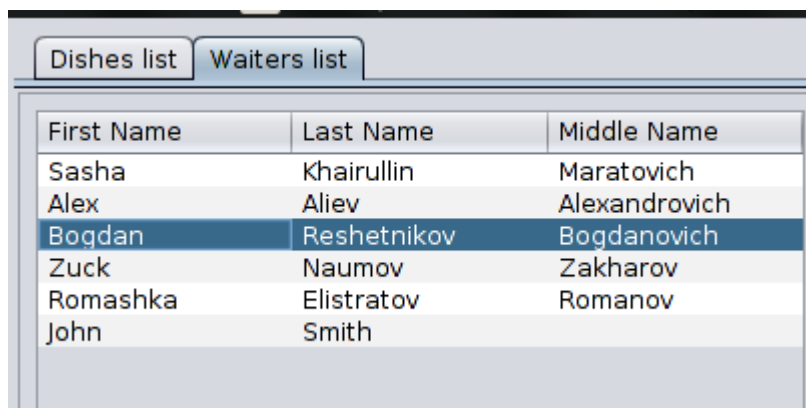
Приложение имеет множество функций. В их число входит добавление, изменение, удаление данных. Для работы с данными используются 4 таблицы:

1. **Orders from menu.** Эта таблица отображает данные о заказах из меню, которые содержатся в заказах. К примеру, если посетитель заказал 2 порции Cookies и 4 порции Manascookies, то таблица Orders from menu отобразит 2 строки: первая будет содержать «2 100 Cookies», а вторая - «4 480 Manascookies». Эти строки отображают количество, общую сумму (в первой строке: $2 * 50 = 100$, во второй строке: $4 * 120 = 480$) и название блюда.
2. **Orders.** Эта таблица отображает данные о заказах. К примеру, строка «12/8/19 12 580 Aliev Alex» означает, что дата заказа - декабрь, 8 число, 2019 год, номер столика — 12, общая сумма заказа — 580 (сумма 100 и 480 из предыдущего пункта), официант — Aliev Alex.
3. **Dishes list.** Эта таблица отображает данные о блюдах. К примеру, строка «200 Steak» означает, что цена блюда — 200, а название — Steak.
4. **Waiters list.** Эта таблица отображает данные о официантах. Таблица отображает имя (First Name), фамилию (Last Name), и, если имеется, то отчество (Middle Name).

Добавление нового официанта.

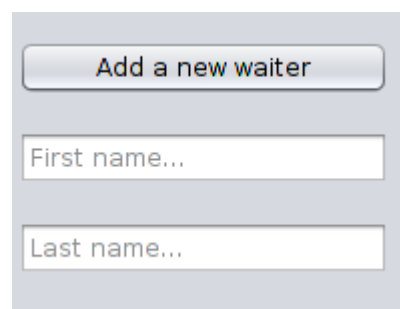
Для добавления нового официанта следуйте следующим инструкциям:

1. Откройте таблицу Waiters list.



First Name	Last Name	Middle Name
Sasha	Khairullin	Maratovich
Alex	Aliev	Alexandrovich
Bogdan	Reshetnikov	Bogdanovich
Zuck	Naumov	Zakharov
Romashka	Elistratov	Romanov
John	Smith	

2. Введите в поля First Name и Last Name имя и фамилию официанта соответственно и нажмите на кнопку Add a new waiter.

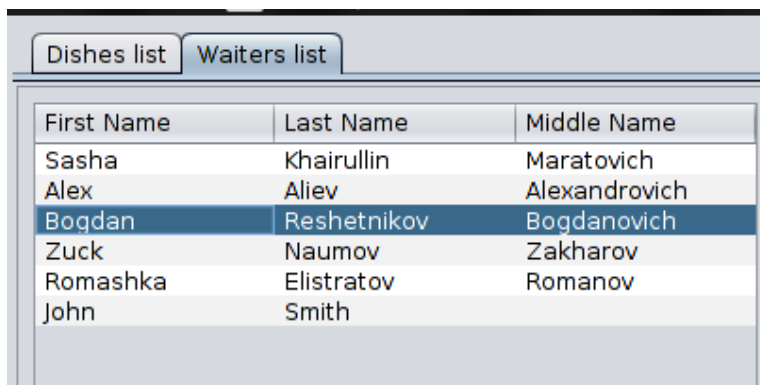


The form contains a button labeled "Add a new waiter" at the top. Below it are two text input fields: "First name..." and "Last name...".

Добавление нового заказа.


Для того чтобы добавить новый заказ следуйте следующим инструкциям:

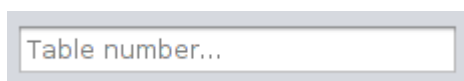
1. Для начала нужно выбрать официанта, который будет обслуживать посетителей, которые сделали этот заказ. Для этого откройте таблицу Waiter list и нажмите на имя этого официанта.



The screenshot shows a window with two tabs: "Dishes list" and "Waiters list". The "Waiters list" tab is active, displaying a table with three columns: "First Name", "Last Name", and "Middle Name". The row for "Bogdan Reshetnikov Bogdanovich" is highlighted.

First Name	Last Name	Middle Name
Sasha	Khairullin	Maratovich
Alex	Aliev	Alexandrovich
Bogdan	Reshetnikov	Bogdanovich
Zuck	Naumov	Zakharov
Romashka	Elistratov	Romanov
John	Smith	

2. Далее выберите дату совершения заказа. Для этого нажмите на иконку  и выберите день заказа, нажав два раза левой кнопкой мыши. Этот пункт можно пропустить, если выбранная по умолчанию дата совпадает с датой совершения заказа.
3. Далее нужно выбрать номер столика, с которого совершили заказ. Для этого введите номер столика в поле Table Number:



A text input field with the placeholder text "Table number..."

4. После того как выбраны официант, дата совершения заказа и номер столика, нажмите на кнопку Add an order. В результате в таблице Orders list должен появиться новый заказ с датой совершения заказа, номером столика, общей суммой заказа и именем официанта.

Наполнение заказа блюдами.

Для того чтобы наполнить заказ блюдами следуйте следующим инструкциям:

1. Выберите заказ, для которого Вы хотите добавить блюда. Для этого нажмите на этот заказ в таблице Orders list.

Orders list			
Date	Table number	Total amount	Waiter
12/7/19	12	750	Khairullin Sasha
12/7/19	44	800	Aliev Alex
12/6/19	33	2900	Aliev Alex
12/8/19	41	1760	Naumov Zuck
12/8/19	51	650	Naumov Zuck
12/8/19	43	2460	Elistratov Romashka
12/8/19	42	3000	Elistratov Romashka
12/8/19	23	2960	Aliev Alex
12/8/19	19	14500	Khairullin Sasha
12/8/19	18	960	Elistratov Romashka
12/8/19	17	1700	Khairullin Sasha
12/8/19	14	2500	Reshetnikov Bogdan
12/8/19	24	3700	Khairullin Sasha
12/8/19	25	620	Khairullin Sasha
12/8/19	50	790	Aliev Alex
12/8/19	23	0	Reshetnikov Bogdan

2. Далее выберите блюдо, которое вы хотите добавить в этот заказ. Для этого откройте таблицу Dishes list и нажмите на название этого блюда.
3. Далее введите количество порций блюда, которого вы хотите добавить в заказ. Для этого введите в поле Count количество порций.
4. Далее нажмите на кнопку Add an order from menu. В результате в таблице выше должно появиться название блюда, количество порций и сумма (количество порций * цена блюда), а общая сумма заказа должна обновиться.
5. Для того, чтобы добавить ещё блюда в заказ, повторите пункты 1-4.

Add an order from menu

Count...

Удаление блюда из заказа

Для того чтобы удалить блюдо из заказа следуйте следующим инструкциям:

1. Выберите заказ из таблицы Orders list
2. Выберите блюдо, которое хотите удалить из заказа в таблице Orders from menu list
3. Нажмите кнопку Delete order from menu
4. В результате блюдо удалится из заказа, а общая сумма заказа уменьшится(изменится)

Удаление заказа

Для того чтобы удалить заказ следуйте следующим инструкциям:

1. Выберите заказ из таблицы Orders list
2. Нажмите кнопку Delete selected order
3. В результате заказ удалится

Удаление блюда

Для того чтобы удалить блюдо следуйте следующим инструкциям:

1. Откройте таблицу Dishes list
2. Выберите блюдо для удаления из таблицы Dishes list
3. Нажмите кнопку Delete selected dish.

Удаление официанта

Для того чтобы удалить официанта следуйте следующим инструкциям:

1. Откройте таблицу Waiters list
2. Выберите официанта для удаления
3. Нажмите кнопку Delete selected waiter

Изменение данных о блюдах

Для того чтобы изменить данные о блюде (цена и/или название) следуйте следующим инструкциям:

1. Откройте таблицу Dishes list

2. Нажмите на блюдо, которое хотите изменить, два раза в таблице Dishes list
3. Введите новое значение для блюда
4. Нажмите клавишу Enter на клавиатуре

Изменение данных о официантах

Для того чтобы изменить данные о официантах(имя, фамилия или отчество) следуйте следующим инструкциям:

1. Откройте таблицу Waiters list
2. Нажмите два раза на данных официанта, которые хотите изменить
3. Введите новое значение
4. Нажмите Enter

Поиск блюд или официантов

Для поиска блюда или официанта следуйте следующим инструкциям:

1. Откройте соответствующую таблицу. Это может быть Dishes list или Waiters list.
2. Поиск официанта: Введите в поле Search for имя, фамилию или отчество того официанта, которого Вы хотите найти
Поиск блюда: Введите в поле Search for цену или название блюда, которое Вы хотите найти
3. Таблица обновится и выведет вам соответствующий результат

Список литературы

1. «Университетская библиотека онлайн»
2. Mihalcea V. High-Performance Java Persistence, 2016 г, — 486 с. — ISBN 978-9730228236
3. www.iprbookshop.ru – Электронно-библиотечная система
4. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2018 г, — 352 с. — ISBN 978-5-4461-0772-8
5. Введение в СУБД MySQL / . — 2-е изд. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 228 с. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/73650.html>
6. Крис, Файлы SQL / Файлы Крис ; перевод А. В. Хаванов. — 2-е изд. — Саратов : Профобразование, 2019. — 452 с. — ISBN 978-5-4488-0103-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/87984.html>
7. Режим доступа: <https://stackoverflow.com>
8. Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму, — 2018 с. — ISBN 978-5-4461-0696-7
9. Шилдт Г. Java 8: полное руководство. – М.: Вильямс, 2015 г, — 1376с. — ISBN 978-5-8459-1918-2
10. Эккель Б. Философия Java. – СПб.: Питер, 2015 г
11. Молинаро Э. SQL. Сборник рецептов - Символ-Плюс, 2009 г, — 672 с. — ISBN 978-5-93286-125-7
12. Система ответов и вопросов о программировании Stack Overflow. –

Приложение

```
package com.akaleaf.course;
import java.awt.Color;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.table.DefaultTableModel;
public class MainFrameGUI extends javax.swing.JFrame {
    public MainFrameGUI() {
        setExtendedState(javax.swing.JFrame.MAXIMIZED_BOTH);
        setTitle("Restaurant");
        initComponents();
        dishTable.getColumnModel().getColumn(1).setMinWidth(78);
        dishTable.getColumnModel().getColumn(1).setMaxWidth(78);
        orderFromMenuTable.getColumnModel().getColumn(1).setMinWidth(64);
        orderFromMenuTable.getColumnModel().getColumn(1).setMaxWidth(64);
        orderFromMenuTable.getColumnModel().getColumn(2).setMinWidth(80);
        orderFromMenuTable.getColumnModel().getColumn(2).setMaxWidth(80);
        orderTable.getColumnModel().getColumn(1).setMinWidth(70);
        orderTable.getColumnModel().getColumn(1).setMaxWidth(70);
        orderTable.getColumnModel().getColumn(2).setMinWidth(100);
        orderTable.getColumnModel().getColumn(2).setMaxWidth(100);
        orderTable.getColumnModel().getColumn(3).setMinWidth(100);
        orderTable.getColumnModel().getColumn(3).setMaxWidth(100);
        updateDishTable("");
        updateWaiterTable("");
        updateOrderTable();
        updateOrderFromMenuTable();
    }
    // Метод отображающий данные таблицы блюд
    private void updateDishTable(String parameter) {
        try {
            DBProcessor db = new DBProcessor();
            Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
            String query;
            if (parameter.equals("")) {
                query = "select * from restaurant.Dish";
            } else {
                query = "select * from restaurant.Dish where name like '%" + parameter + "%' OR price like '%" + parameter + "%'";
            }
            PreparedStatement prepStatement = connection.prepareStatement(query);
            ResultSet resultSet = prepStatement.executeQuery();
            DefaultTableModel dishTableModel = (DefaultTableModel) dishTable.getModel();
            dishTableModel.setRowCount(0);
            int i = 0;
            while (resultSet.next()) {
                dishTableModel.addRow(new Object[]{resultSet.getInt("idDish")});
                dishTableModel.setValueAt(resultSet.getInt("price"), i, 1);
            }
        }
    }
}
```

```

dishTableModel.setValueAt(resultSet.getString("name"), i, 2);
        i++;
    }
    preparedStatement.close();
    connection.close();
} catch (SQLException ex) {
    ex.printStackTrace();
}
}

// Метод отображающий данные таблицы официантов
private void updateWaiterTable(String parameter) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String query;
        if (parameter.equals("")) {
            query = "select * from restaurant.Waiter";
        } else {
            query = "select * from restaurant.Waiter where firstName like '%" + parameter + "%' OR lastName like '%" + parameter + "%' OR middleName like '%" + parameter + "%'";
        }
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();
        DefaultTableModel waiterTableModel = (DefaultTableModel) waiterTable.getModel();
        waiterTableModel.setRowCount(0);
        int i = 0;
        while (resultSet.next())
            waiterTableModel.addRow(new Object[] {resultSet.getString("idWaiter"),
            waiterTableModel.setValueAt(resultSet.getString("firstName"), i, 1);
            waiterTableModel.setValueAt(resultSet.getString("lastName"), i, 2);
            waiterTableModel.setValueAt(resultSet.getString("middleName"), i, 3);
            i++;
        }
        preparedStatement.close();
        connection.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

// Метод отображающий данные таблицы заказов
private void updateOrderTable() {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String query = "select * from restaurant.Order";
        PreparedStatement preparedStatement = connection.prepareStatement(query);
        ResultSet resultSet = preparedStatement.executeQuery();
        DefaultTableModel orderTableModel = (DefaultTableModel) orderTable.getModel();
        orderTableModel.setRowCount(0);
        int i = 0;

```

```

while (resultSet.next()) {
    orderTableModel.addRow(new Object[]{resultSet.getInt("idOrder")});
    orderTableModel.setValueAt(resultSet.getString("orderDate"), i, 1);
    orderTableModel.setValueAt(resultSet.getInt("tableNumber"), i, 2);
    orderTableModel.setValueAt(resultSet.getInt("totalAmount"), i, 3);
    String queryWaiter = "select * from restaurant.Waiter where idWaiter like \"\" + resultSet.getInt("idWaiter") + \"\"";
    PreparedStatement queryWaiterStatement = connection.prepareStatement(queryWaiter);
    ResultSet queryWaiterResultSet = queryWaiterStatement.executeQuery();
    queryWaiterResultSet.next();
    orderTableModel.setValueAt(queryWaiterResultSet.getString("lastName") + " " + queryWaiterResultSet.getString("firstName"), i,
4);
    i++;
}
prepStatement.close();
connection.close();
} catch (SQLException ex) {
    ex.printStackTrace();
}
}
// Метод отображающий данные таблицы заказов из меню
private void updateOrderFromMenuTable() {
    if (orderTable.getSelectedRow() == -1) {
        return;
    }
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        int idOrder = (Integer) orderTable.getValueAt(orderTable.getSelectedRow(), 0);
        String query = "select * from restaurant.OrderFromMenu where idOrder like \"\" + idOrder + \"\"";
        PreparedStatement prepStatement = connection.prepareStatement(query);
        ResultSet resultSet = prepStatement.executeQuery();
        DefaultTableModel orderFromMenuTableModel = (DefaultTableModel) orderFromMenuTable.getModel();
        orderFromMenuTableModel.setRowCount(0);
        int totalAmount = 0;
        int i = 0;
        while (resultSet.next()) {
            orderFromMenuTableModel.addRow(new Object[]{resultSet.getInt("idOrder")});
            orderFromMenuTableModel.setValueAt(resultSet.getInt("count"), i, 1);
            orderFromMenuTableModel.setValueAt(resultSet.getInt("amount"), i, 2);
            String queryDish = "select * from restaurant.Dish where idDish like \"\" + resultSet.getInt("idDish") + \"\"";
            PreparedStatement queryDishStatement = connection.prepareStatement(queryDish);
            ResultSet queryDishResultSet = queryDishStatement.executeQuery();
            queryDishResultSet.next();
            orderFromMenuTableModel.setValueAt(queryDishResultSet.getString(2), i, 3);
            queryDishStatement.close();
            i++;
        }
        prepStatement.close();
        connection.close();
    } catch (SQLException ex) {

```

```

        ex.printStackTrace();
    }
}

// Метод, добавляющий новый заказ в базу данных и в таблицу
private void addOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String INSERT = "insert into restaurant.Order (orderDate, tableNumber, idWaiter, totalAmount) values (?, ?, ?, ?)";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setString(1, dateChooser.getText());
        prepInsert.setInt(2, Integer.parseInt(tableNumberTextField.getText()));
        prepInsert.setInt(3, Integer.parseInt((String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 0)));
        prepInsert.setInt(4, 0);
        prepInsert.execute();
        connection.close();
        updateOrderTable();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

// Метод, удаляющий заказ из базы данных и из таблицы
private void deleteOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String INSERT = "delete from restaurant.Order where idOrder=?";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setInt(1, (Integer) orderTable.getValueAt(orderTable.getSelectedRow(), 0));
        prepInsert.execute();
        connection.close();
        updateOrderTable();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

// Метод, добавляющий новый заказ из меню в базу данных и в таблицу
private void addOrderFromMenuButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        int dishTableSelectedRow = dishTable.getSelectedRow();
        int dishTableIDOrder = (Integer) dishTable.getValueAt(dishTableSelectedRow, 0);
        int orderTableIDOrder = (Integer) orderTable.getValueAt(orderTable.getSelectedRow(), 0);
        String INSERT = "insert into restaurant.OrderFromMenu (idOrder, idDish, count, amount) values (?, ?, ?, ?)";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setInt(1, orderTableIDOrder);
        prepInsert.setInt(2, dishTableIDOrder);
        prepInsert.setInt(3, Integer.parseInt(countTextField.getText()));
        String dishQuery = "select * from restaurant.Dish where idDish = " + dishTableIDOrder;
    }
}

```



```

        PreparedStatement dishQueryStatement = connection.prepareStatement(dishQuery);
        ResultSet dishQueryResultSet = dishQueryStatement.executeQuery();
        dishQueryResultSet.next();
        prepInsert.setInt(4, (Integer.parseInt(countTextField.getText()) * dishQueryResultSet.getInt("price")));
        prepInsert.execute();
        String UPDATE = "update restaurant.Order set totalAmount = ? where idOrder = " + orderTableIDOrder;
        PreparedStatement prepUpdate = connection.prepareStatement(UPDATE);
        String orderFromMenuQuery = "select * from restaurant.OrderFromMenu where idOrder = " + orderTableIDOrder;
        PreparedStatement orderFromMenuQueryStatement = connection.prepareStatement(orderFromMenuQuery);
        ResultSet orderFromMenuQueryResultSet = orderFromMenuQueryStatement.executeQuery();
        int totalAmount = 0;
        while (orderFromMenuQueryResultSet.next()) {
totalAmount += orderFromMenuQueryResultSet.getInt("amount");
        }
        prepUpdate.setInt(1, totalAmount);
        prepUpdate.execute();
        connection.close();
        updateOrderFromMenuTable();
        updateOrderTable();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void countTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(countTextField, "Count...");
}

private void tableNumberTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(tableNumberTextField, "Table number...");
}

private void countTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(countTextField, "Count...");
}

private void tableNumberTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(tableNumberTextField, "Table number...");
}

// Удаление официанта из базы данных и из таблицы
private void deleteWaiterActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String INSERT = "delete from restaurant.Waiter where idWaiter = ?";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setInt(1, Integer.parseInt((String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 0)));
        prepInsert.execute();
        connection.close();
        updateWaiterTable("");
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}
}

```

```

private void lastNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(lastNameTextField, "Last name...");
}

private void lastNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(lastNameTextField, "Last name...");
}

private void firstNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(firstNameTextField, "First name...");
}

private void firstNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(firstNameTextField, "First name...");
}

// Метод, добавляющий нового официанта в базу данных и в таблицу
private void addNewWaiterButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String INSERT = "insert into restaurant.Waiter (firstName, lastName, middleName) values (?, ?, ?)";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setString(1, firstNameTextField.getText());
        prepInsert.setString(2, lastNameTextField.getText());
        prepInsert.setString(3, "");
        prepInsert.execute();
        updateWaiterTable("");
        connection.close();
        updateDishTable("");
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void searchForTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(searchForTextField, "Search for...");
}

private void searchForTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(searchForTextField, "Search for...");
}

// Метод удаления блюда из базы данных и из таблицы
private void deleteDishActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String INSERT = "delete from restaurant.Dish where idDish=?";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setInt(1, (Integer) dishTable.getValueAt(dishTable.getSelectedRow(), 0));
        prepInsert.execute();
        connection.close();
        updateDishTable("");
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

```

```

private void priceOfDishTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(priceOfDishTextField, "Price of dish...");
}

private void priceOfDishTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(priceOfDishTextField, "Price of dish...");
}

private void nameOfDishTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(nameOfDishTextField, "Name of dish...");
}

private void nameOfDishTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(nameOfDishTextField, "Name of dish...");
}

// Метод добавления блюда в базу данных и в таблицу
private void addDishButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        String query = "select * from restaurant.Dish";
        String INSERT = "insert into restaurant.Dish (name, price) values (?, ?)";
        PreparedStatement prepInsert = connection.prepareStatement(INSERT);
        prepInsert.setString(1, nameOfDishTextField.getText());
        prepInsert.setInt(2, Integer.parseInt(priceOfDishTextField.getText()));
        prepInsert.execute();
        connection.close();
        updateDishTable("");
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void searchForTextFieldKeyReleased(java.awt.event.KeyEvent evt) {
    updateDishTable(searchForTextField.getText());
}

private void orderTableMouseClicked(java.awt.event.MouseEvent evt) {
    updateOrderFromMenuTable();
}

private void waiterTableKeyReleased(java.awt.event.KeyEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        int idWaiter = Integer.parseInt((String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 0));
        if (waiterTable.getSelectedColumn() == 1) {
            String updateFirstName = "update restaurant.Waiter set firstName = ? where idWaiter = " + idWaiter;
            PreparedStatement prepUpdateFirstName = connection.prepareStatement(updateFirstName);
            prepUpdateFirstName.setString(1, (String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 1));
            prepUpdateFirstName.execute();
        }
        if (waiterTable.getSelectedColumn() == 2) {
            String updateLastName = "update restaurant.Waiter set lastName = ? where idWaiter = " + idWaiter;
            PreparedStatement prepUpdateLastName = connection.prepareStatement(updateLastName);
            prepUpdateLastName.setString(1, (String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 2));
        }
    }
}

```

```

        prepUpdateLastName.execute();
    }
    if (waiterTable.getSelectedColumn() == 3) {
        String updateMiddleName = "update restaurant.Waiter set middleName = ? where idWaiter = " + idWaiter;
        PreparedStatement prepUpdateMiddleName = connection.prepareStatement(updateMiddleName);
        prepUpdateMiddleName.setString(1, (String) waiterTable.getValueAt(waiterTable.getSelectedRow(), 3));
        prepUpdateMiddleName.execute();
    }
    connection.close();
} catch (SQLException ex) {
    ex.printStackTrace();
}
}

private void dishTableKeyReleased(java.awt.event.KeyEvent evt) {
    try {
        DBProcessor db = new DBProcessor();
        Connection connection = db.getConnection(URL, USERNAME, PASSWORD);
        int idDish = (Integer) dishTable.getValueAt(dishTable.getSelectedRow(), 0);
        if (dishTable.getSelectedColumn() == 1) {
            String updatePrice = "update restaurant.Dish set price = ? where idDish = " + idDish;
            PreparedStatement prepUpdatePrice = connection.prepareStatement(updatePrice);
            prepUpdatePrice.setInt(1, (Integer) dishTable.getValueAt(dishTable.getSelectedRow(), 1));
            prepUpdatePrice.execute();
        }
        if (dishTable.getSelectedColumn() == 2) {
            String updateName = "update restaurant.Dish set name = ? where idDish = " + idDish;
            PreparedStatement prepUpdateName = connection.prepareStatement(updateName);
            prepUpdateName.setString(1, (String) dishTable.getValueAt(dishTable.getSelectedRow(), 2));
            prepUpdateName.execute();
        }
        connection.close();
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
}

private void searchForWaiterTextFieldKeyReleased(java.awt.event.KeyEvent evt) {

```

```

    updateWaiterTable(searchForWaiterTextField.getText());
}

private void searchForWaiterTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    textFieldFocusGained(searchForWaiterTextField, "Search for...");
}

private void searchForWaiterTextFieldFocusLost(java.awt.event.FocusEvent evt) {
    textFieldFocusLost(searchForWaiterTextField, "Search for...");
}

```

```

private void countTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorInteger(evt, countTextField);
}
private void tableNumberTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorInteger(evt, tableNumberTextField);
}
private void priceOfDishTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorInteger(evt, priceOfDishTextField);
}
private void nameOfDishTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorString(evt, nameOfDishTextField);
}
private void firstNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorString(evt, firstNameTextField);
}
private void lastNameTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorString(evt, lastNameTextField);
}
private void searchForWaiterTextFieldKeyPressed(java.awt.event.KeyEvent evt) {
    censorString(evt, searchForWaiterTextField);
}
private void censorInteger(java.awt.event.KeyEvent evt, javax.swing.JTextField textField) {
    char character = evt.getKeyChar();
    if (Character.isDigit(character) || Character.isISOControl(character)) {
        textField.setEditable(true);
    } else {
        textField.setEditable(false);
    }
}
private void censorString(java.awt.event.KeyEvent evt, javax.swing.JTextField textField) {
    char character = evt.getKeyChar();
    if (Character.isLetter(character) || Character.isWhitespace(character) || Character.isISOControl(character)) {
        textField.setEditable(true);
    } else {
        textField.setEditable(false);
    }
}
private void textFieldFocusLost(javax.swing.JTextField textField, String text) {
    if (textField.getText().equals("")) {
        textField.setText(text);
        textField.setForeground(Color.GRAY);
    }
}
private void textFieldFocusGained(javax.swing.JTextField textField, String text) {
    if (textField.getText().equals(text)) {
        textField.setText("");
        textField.setForeground(Color.BLACK);
    }
}
/**

```

```

* @param args the command line arguments
*/
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
    * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(MainFrameGUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(MainFrameGUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(MainFrameGUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(MainFrameGUI.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new MainFrameGUI().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton addDishButton;
private javax.swing.JButton addNewWaiterButton;
private javax.swing.JButton addOrderButton;
private javax.swing.JButton addOrderFromMenuButton;
private javax.swing.JTextField countTextField;
private datechooser.beans.DateChooserCombo dateChooser;
private javax.swing.JButton deleteDish;
private javax.swing.JButton deleteOrderButton;
private javax.swing.JButton deleteWaiter;
private javax.swing.JTable dishTable;
private javax.swing.JScrollPane dishesListScrollPane;
private javax.swing.JTextField firstNameTextField;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel5;
private javax.swing.JPanel jPanel7;
private javax.swing.JScrollPane jScrollPane1;

```

```

private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane5;
private javax.swing.JScrollPane jScrollPane6;
private javax.swing.JTextField lastNameTextField;
private javax.swing.JTextField nameOfDishTextField;
private javax.swing.JTable orderFromMenuTable;
private javax.swing.JTable orderTable;
private javax.swing.JTextField priceOfDishTextField;
private javax.swing.JTextField searchForTextField;
private javax.swing.JTextField searchForWaiterTextField;
private javax.swing.JTabbedPane tabbedPane;
private javax.swing.JTextField tableNumberTextField;
private javax.swing.JScrollPane waiterListScrollPane;
private javax.swing.JTable waiterTable;
// End of variables declaration
// MySQL Variables declaration
private static final String USERNAME = "root";
private static final String PASSWORD = "steinHER35.-";
private static final String URL = "jdbc:mysql://localhost:3306/mysql?useSSL=false";
// End of MySQL variables declaration
}

```

DBProcessor

```

package com.akaleaf.course;
import com.mysql.fabric.jdbc.FabricMySQLDriver;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBProcessor {
    private Connection connection;
    public DBProcessor() throws SQLException {
        DriverManager.registerDriver(new FabricMySQLDriver());
    }
    public Connection getConnection(String url, String username, String password) throws SQLException {
        if (connection != null)
            return connection;
        connection = DriverManager.getConnection(url, username, password);
        return connection;
    }
}

```