**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc533373004)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 6](#_Toc533373005)

[1.1 Анализ предметной области 6](#_Toc533373006)

[1.2 Постановка задачи 7](#_Toc533373007)

[2 РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ БАЗЫ ДАННЫХ 9](#_Toc533373008)

[2.1 Построение диаграммы «сущность-связь» в нотации П.Чена 9](#_Toc533373009)

[2.2 Построение модели основанной на ключах (Key Based model, KB) 13](#_Toc533373010)

[2.3 Построение полной атрибутивной модели в нотации IDEF1X 13](#_Toc533373011)

[3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 17](#_Toc533373012)

[3.1 Обоснование выбора СУБД 17](#_Toc533373013)

[3.2 Разработка физической модели базы данных 17](#_Toc533373014)

[3.3 Разработка запросов к базе данных 18](#_Toc533373015)

[3.4 Разграничение прав доступа 19](#_Toc533373016)

[3.5 Исследование информационных параметров базы данных 19](#_Toc533373017)

[4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 22](#_Toc533373018)

[4.1 Обоснование выбора языка программирования 22](#_Toc533373019)

[4.2 Разработка интерфейса пользователя 22](#_Toc533373020)

[ЗАКЛЮЧЕНИЕ 32](#_Toc533373021)

[БИБЛИОГРАФИЧЕСКИЙСПИСОК 33](#_Toc533373022)

[ПРИЛОЖЕНИЕА 34](#_Toc533373023)

[ПРИЛОЖЕНИЕ Б 89](#_Toc533373024)

# **ВВЕДЕНИЕ**

**Актуальность темы.** На сегодняшний день, очень затруднительно представить любую информационную систему, которая не использует базы данных. Их развитие было очень тесно связано с развитием автоматизированной обработки данных и увеличением количества различных категорий пользователей, что послужило к построению более мощных СУБД, имеющие средства защиты информации, высокую отказоустойчивость и производительность. С развитием сферы услуг и ростом желающих получить различные услуги в области красоты появляется задача автоматизированной обработки данных о данных услугах. База данных облегчает работу с огромной информацией, необходимой для организации записи клиентов на предоставляемые услуги.

**Цель и задачи работы.** Целью данного курсового проекта является создание базы данных, используемой для упрощения обработки данных в салоне красоты. Для реализации данной работы, необходимо провести следующие этапы:

* анализ предметной области (определяются пользовательские системы, функциональные требования системы, выделяются основные сущности разрабатываемой системы и отношения между ними, на основе выделенных сущностей, определяются их атрибуты и строится ER-диаграмма);
* разработка логической схемы базы данных (построение и нормализация реляционной модели);
* реализация базы данных (обоснование выбора СУБД, разработка физической схемы базы данных, реализация и тестирование базы данных, запросов к ней, разграничение прав доступа);
* исследование информационных параметров базы данных (подсчитывание значений некоторых параметров БД и приложения в целом);
* разработка клиентского приложения (обоснование выбора языка программирования, разработка клиентского приложения, описание интерфейса пользователя, тестирование системы).

**Предмет и объект исследования.** Объектом исследования настоящей работы является база данных выбранной предметной области.

**Структура работы.** Данная работа состоит из пояснительной записки, включающей в себя введение, четыре раздела, выводы, список использованных источников, приложения и программного модуля на электронном носителе.

В первом разделе описана предметная область и постановка задачи.

* Второй раздел посвящен разработке логической схемы базы данных,

а именно построению диаграммы «сущность-связь» в нотации П.Чена, модели основанной на ключах (Key Based model, KB) и полной атрибутивной модели в нотации IDEF1X.

* В третьем разделе рассмотрена физическая модель базы данных, представлено обоснование выбора СУБД, разграничение прав доступа, а также исследованы информационные параметры базы данных.
* В четвертом разделе описана разработка клиентского приложения, выбранный язык программирования, представлен интерфейс пользователя.

1. **АНАЛИТИЧЕСКАЯ ЧАСТЬ**

# **1.1 Анализ предметной области**

Салон красоты – заведение, занимающееся косметическим обслуживанием мужчин и женщин, оказывая услуги в области лица и тела людей. Для организации работы салона красоты необходимо создать программное обеспечение по хранению и обработки информации о клиентах, сотрудниках и об оказываемых услугах. В обязанности пользователям будет входить следующие действия:

* создание и составление таблиц с данными о клиентах, сотрудниках и услугах;
* ведение списков сотрудников;
* учет информации о клиентах;
* редактирование информации об оказываемых услугах;
* учет посещаемости клиентов.

Учёт посещаемости клиентов имеет очень большое значение для эффективной проверки качества работы сотрудников в салоне красоты. Статистика даёт возможность оценивать предпочтения клиентов. Она также может быть использована с целью развития салона красоты.

Исходя из того, что пользователями приложения будут администратор и иные пользователи, должна быть предусмотрена авторизация и разграничение доступа к закрытой информации (информация о пользователях). Определим следующие функциональные требования для приложения:

* + авторизация;
  + хранение актуальной информации о сотрудниках, клиентах и услугах;
  + возможность добавления, удаления и редактирования данных о сотрудниках, клиентах и услугах;
  + возможность изменять списки, исключая и добавляя, в них сотрудников, клиентов и услуги;
  + возможность учёта посещаемости клиентов;
  + интуитивно понятный интерфейс;
  + предоставление общей информации о группе и студентах.

Согласно вышеперечисленным функциональным требованиям, выделим следующие сущности, которые будут описывать нашу систему: клиент, услуга, сотрудник, сотрудник-услуга, посещение.

Каждой из перечисленной сущности соответствует свой набор атрибутов:

* клиент: идентификатор, фио, номер телефона, посещения;
* сотрудник: идентификатор, фио, номер телефона, процентная ставка, дата принятия на работу;
* услуга: идентификатор, наименование услуги, стоимость, длительность
* посещение: идентификатор клиента, идентификатор сотрудника, идентификатор услуги, дата, время, статус;
* услуга-сотрудник: идентификатор сотрудника, идентификатор услуги.

Таким образом, в ходе анализа предметной области были выявлены основные сущности и отношения между ними.

# **1.2 Постановка задачи**

Для реализации базы данных салоны красоты, а также клиентского приложения, для работы с разработанной БД, были поставлены следующие задачи:

* разработать логическую модель базы данных (построить диаграмму «сущность-связь» в нотации П.Чена, построить модели основанной на ключах, а также построить полную атрибутивную модель IDEF1X. Нормализовать полученную логическую модель до третьей нормальной формы);
* разработать физическую модель базы данных (выбрать аппаратную и программную платформу для реализации базы данных, а также её реализовать, протестировать полученную базу данных, разграничить права доступа, и рассчитать информационные параметры базы данных);
* разработать пользовательский интерфейс (выбрать и обосновать выбор языка программирования, разработать интерфейс пользователя, алгоритм работы каждого модуля, и протестировать разработанное приложение).

Поэтому основной задачей курсового проекта является сбор и обработка данных, построение на их основании логической модели будущей базы данных, разработка базы данных, а также разработка полноценного клиентского приложения для обеспечения взаимодействия различных групп пользователей с разработанной базой данных.

**Выводы раздела 1**

Для того, чтобы хранить данные в виде реляционной БД, их были систематизировать и выявлены основные сущности и логические отношения между ними.

Можно сделать вывод, что основной задачей курсового проекта является сбор данных, их анализ и построение на их основании логической модели будущей базы данных, разработка базы данных, обеспечивающей хранение всех необходимых данных и их систематизацию, а также разработка полноценного многопользовательского клиентского приложения для обеспечения взаимодействия различных групп пользователей с разработанной базой данных.

1. **РАЗРАБОТКА ЛОГИЧЕСКОЙ СХЕМЫ БАЗЫ ДАННЫХ**

Для проектирования базы данных была выбрана реляционная модель. Такие модели позволяют достигнуть гораздо более высокого уровня абстракции, чем в сетевой или иерархической моделях. В основе реляционной модели лежит математическая теория отношений, имеющая строгие правила проектирования, включающие в себя необходимость проведения процедуры нормализации. Представление самих данных в модели не зависит от способа их физической реализации. Единственной используемой конструкцией является «таблица», что обеспечивает простоту и доступность для понимания пользователями различного уровня подготовки. Модели обеспечивают полную независимость данных, что подразумевает минимальные изменения в интерфейсе при внесении поправок в структуру модели. Стоит также отметить, что реляционные модели существенно сокращают избыточность и дублирование исходных данных, обеспечивают сохранение их целостности.

# **2.1 Построение диаграммы «сущность-связь» в нотации П.Чена**

Необходимо выполнить построение сложной сетевой структуры и дальнейшее ее преобразование к простой сетевой структуре, а после – к древовидной.  Основными сущностями в сложной сетевой структуре разрабатываемой базы данных являются клиент, сотрудник салона красоты и услуга, которая должна быть оказана. Все три сущности связаны отношениями «многие ко многим»: клиент может получать услугу у нескольких сотрудников и получать множество услуг, сотрудник – оказывать услугу нескольким клиентам, услугу возможно оказать многим клиентам различными сотрудниками. Сложная сетевая структура базы данных представлена на рисунке 2.1.

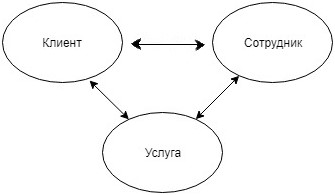


Рисунок 2.1 – Сложная сетевая структура

Для построения простой сетевой структуры (рисунок 2.2.) необходимо устранить связи «многие ко многим». Это возможно посредством введения в структуру новых сущностей – клиент-услуга, клиент-сотрудник и услуга-сотрудник.

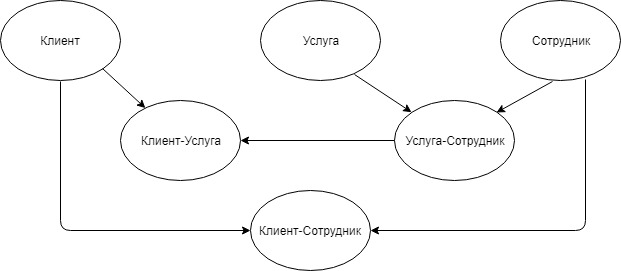


Рисунок 2.2 – Простая сетевая структура

Следующий этап проектирования – построение древовидной структуры. Древовидная стриктура представлена на рисунке 2.3.

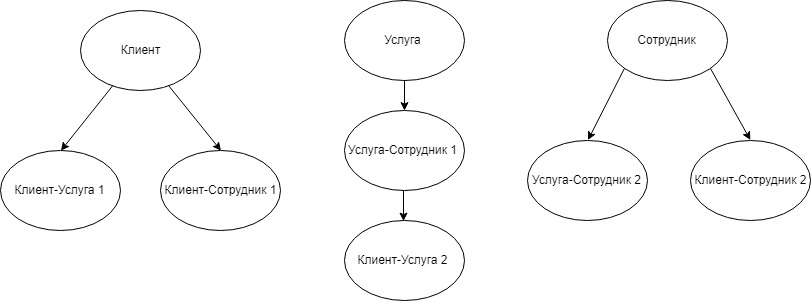


Рисунок 2.2 – Древовидная структура

После того, как была проведен анализ предметной области, и выделены основные сущности, участвующие в работе салона красоты, стало возможным построение диаграммы «сущность-связь» в нотации П.Чена (рисунок 2.4).

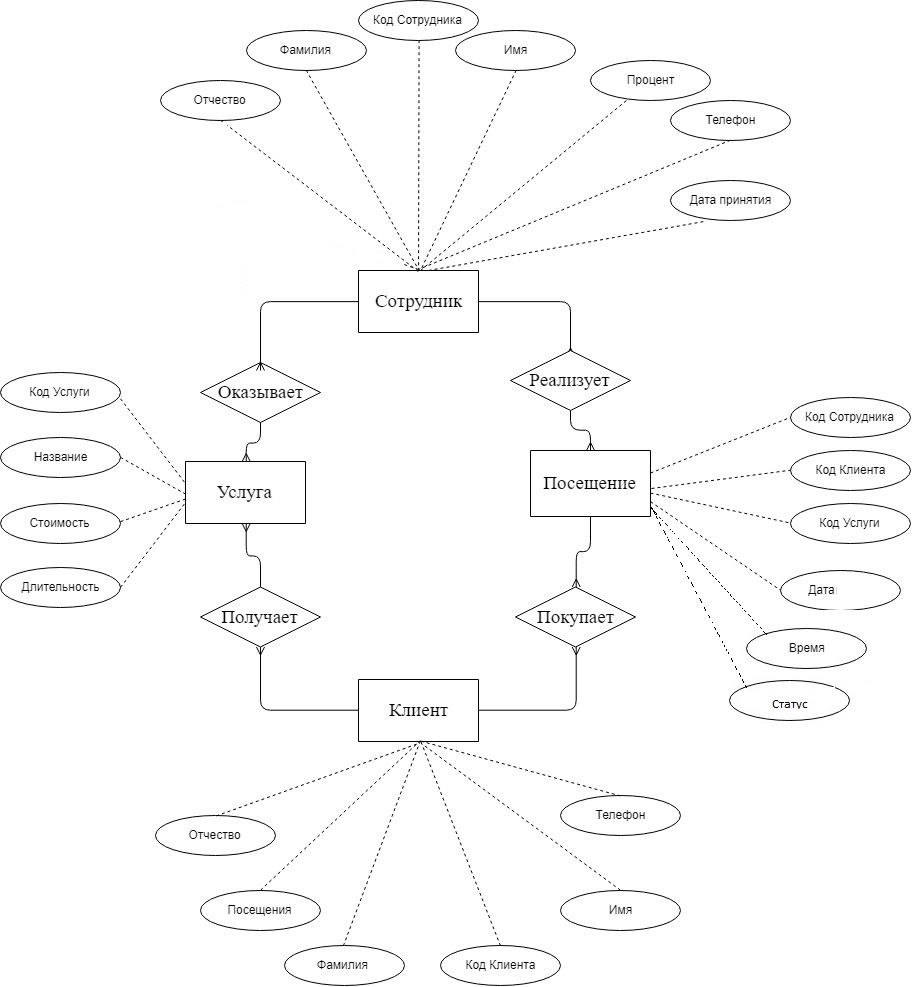


Рисунок 2.4 – Диаграмма «сущность-связь» в нотации П.Чена

В итоге, была построена диаграмма «сущность-связь» в нотации П.Чена, которая отражает логическую модель базы данных, на ней показаны основные сущности и их атрибуты, а также характер их взаимодействия.

# **2.2 Построение модели основанной на ключах (Key Based model, KB)**

Модель, основанная на ключах описывает основные структуры данных, которые охватывают предметная область. Все сущности и первичные ключи включены вместе с примерами атрибутов. Основной целью модели, основанной на ключах, является широкий обзор структур данных и ключей, нужных для поддержки определенной области. Модель показывает ту же область что и область ERD, но, вместе с тем, отображает больше деталей.

На рисунке 2.5 представлена модель, основанная на ключах, отображающая выделенные выше сущности, их свойства и связи.

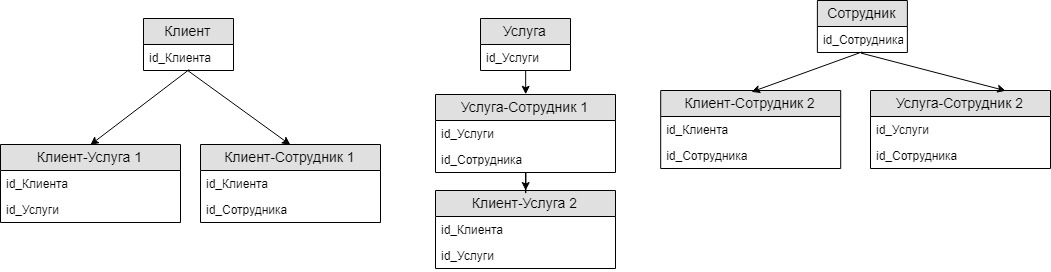


Рисунок 2.5 – Модель, основанная на ключах

# **2.3 Построение полной атрибутивной модели в нотации IDEF1X**

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором каждый факт появляется лишь в одном месте, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных. За основу возьмем выделенные ранее сущности с соответствующими им атрибутами. Также добавим сущность Услуга-Сотрудник для того, чтобы за сотрудником можно было привязать услугу. Модель представлена на рисунке 2.6.

****

Рисунок 2.6 – Полная атрибутивная модель в нотации IDEF1X

Рассмотрим все выделенные нами сущности, и проверим их на атомарность. В каждой из сущности соблюдается атомарность, следовательно, делаем вывод о том, что наша база данных находится в первой нормальной форме.

Так как наша база данных, находится в первой нормальной форме (адрес сотрудника, выявленные отклонения в контексте данной базы будем считать атомарными полями), рассмотрим её на удовлетворение второй нормальной форме (Отношение *R* находится во второй нормальной форме в том и только  
в том случае, когда находится в 1НФ, и каждый не ключевой атрибут полностью  
зависит от первичного ключа). Каждая сущность содержит уникальный, искусственно введенный, первичный ключ, не обладающий смысловой нагрузкой, от которого полностью зависят все не ключевые атрибуты, это удовлетворяет условиям второй нормальной форме, следовательно, делаем вывод о том, что база данных находится во второй нормальной форме.

Так как наша база данных находится во второй нормальной форме, рассмотрим на её удовлетворение третьей нормальной форме (Отношение *R* находится в третьей нормальной форме в том и только в  
том случае, если находится в 2НФ и каждый не ключевой атрибут не транзитивно  
зависит от первичного ключа). Во всех созданных сущностях транзитивные зависимости отсутствуют, что говорит о том, что наша база данных находится в третьей нормальной форме.

В ходе нормализации была построена полная атрибутивная модель в нотации IDEF1X, которая представлена на рисунке 2.7.

****

Рисунок 2.7 – Полная атрибутивная модель в нотации IDEF1X

Таким образом, в ходе нормализации модели базы данных, была достигнута третья нормальная форма, были убраны транзитивные зависимости неключевых атрибутов.

**Выводы раздела 2**

Таким образом, в процессе разработки логической схемы базы данных были построены сложная сетевая , простая сетевая и древовидная структуры – в результате была получена диаграмма «сущность-связь» в нотации П.Чена, которая отражает логическую модель базы данных. Следующим этапом была разработана модель, основанная на ключах, которая показывает ту же область что и область ERD, но, вместе с тем, отображает больше деталей.

В ходе нормализации модели базы данных, была достигнута третья нормальная форма, были убраны транзитивные зависимости неключевых атрибутов, была достигнута целостность базы данных.

1. **РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

# **3.1 Обоснование выбора СУБД**

Выбор **системы управления** базами данных (СУБД) является одним из важных этапов при разработке приложений баз данных и представляет собой сложную задачу. Выбранная СУБД должна удовлетворять как текущим, так и будущим потребностям системы. В настоящее время существует достаточно много различных СУБД, но в настоящей курсовой работе, была использована *MySQL*. так как она имеет ряд преимуществ: поддерживается неограниченное количество пользователей, одновременно работающих с базой данных; количество строк в таблицах может достигать 50 млн. У *MySQL* простая и эффективная система безопасности;

К тому же, в отличии от других СУБД, *MySQL* эта платформа отличается высокой скоростью обработки данных.

*MySQL* обеспечивает высокий уровень кроссплатформенности данных и кода, созданных с помощью *MySQL*. Благодаря чему возможно перенести БД в любую другую современную СУБД, также поддерживающую язык структурированных запросов.

# **3.2 Разработка физической модели базы данных**

Так как у нас уже имеется логическая модель базы данных в третьей нормальной форме, а также выбрана СУБД, которая её будет реализовать, то на основании разработанной логической модели БД, составим диаграмму, представляющая физическую модель разрабатываемой (рисунок 3.1). На диаграмме отображены реальные имена таблиц и столбцов, а также соответствующие этим столбцам типы данных.

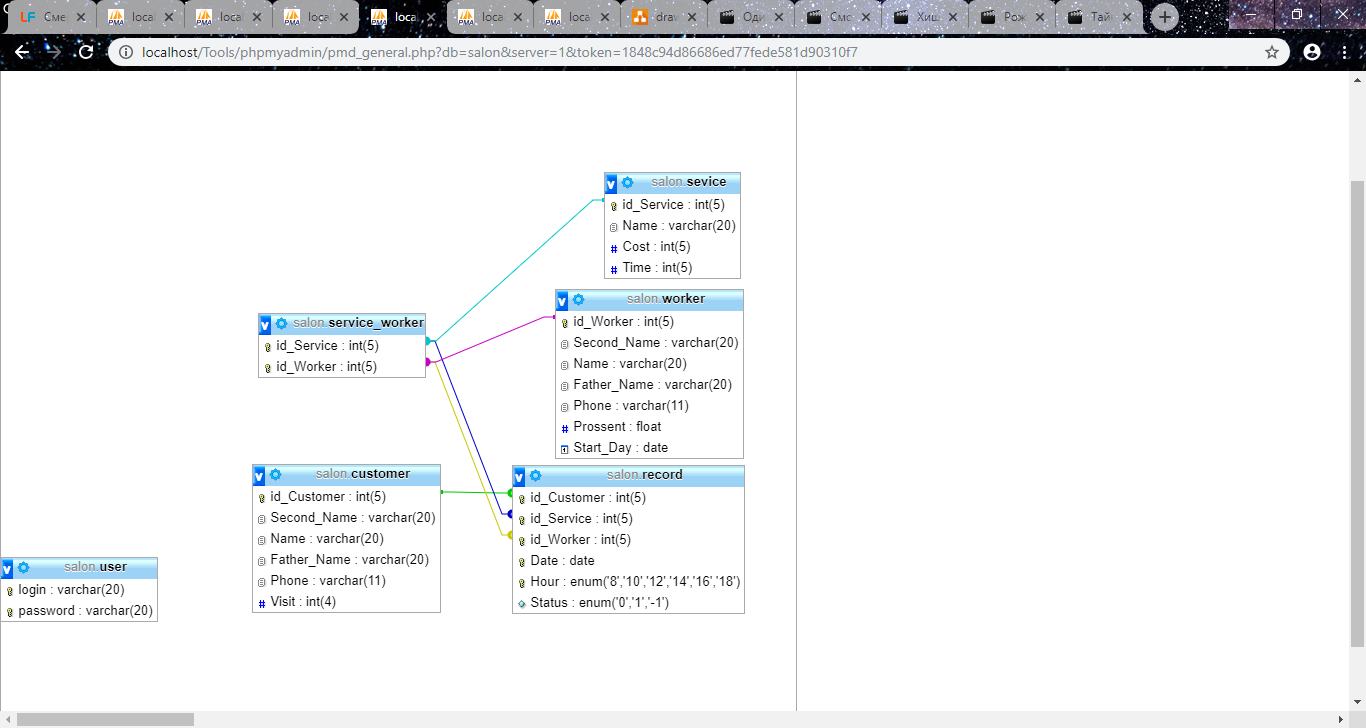


Рисунок 3.1 – Физическая модель базы данных

Таким образом, была построена физическая модель базы данных салона красоты, отражающая реальные параметры, а именно: тип атрибута, размер атрибута.

# **3.3 Разработка запросов к базе данных**

При анализе предметной области, были разработаны функциональные требования к базе данных, на основании которых, была разработана логическая и физическая модель базы данных.

Исходя из функциональных требований, запросы к базе данных, должны реализовывать:

* + выборку клиентов с определенными условиями сужения выборки;
  + выборку сотрудников с определенными условиями сужения выборки;
  + выборку клиентов с определенными условиями сужения выборки;
  + подсчет суммарного количества посещений.

Запросы к базе данных представлены в приложении Б.

Все приведенные тестирования показывают, что данная модель имеет высокую целостность данных, а значит она легко изменяемая, при сохранении целостности.

# **3.4 Разграничение прав доступа**

В разработанной базе данных существует два уровня доступа пользователей: администратор и иные пользователи. В зависимости от выбранной роли могут выполняться различные действия.

Обычный пользователь имеет возможность добавлять, изменять, удалять данные в таблицах с информацией о клиентах, сотрудника и услугах; осуществлять запись на услугу; изменять статус посещения.

Администратор имеет те же возможности, а также возможность добавлять, изменять и удалять новых пользователей.

Таким образом, база данных имеет разграничение доступа к ней, что существенно улучшает удобство использования, а также снижает количество времени на разработку, так как интерфейс программы единый для пользователей разных групп.

# **3.5 Исследование информационных параметров базы данных**

Длина логической записи j-ого файла определяется как сумма длин полей, её составляющих и указателей, если они организуются разработчиком

|  |  |  |
| --- | --- | --- |
|  | [байт], | (3.1) |

где *Mj* – число групп полей в записях;

*lij* длина группы [байт].

Вычислим:

Lклиент = 20 + 3 \* (20 + 1) + (11 + 1) + 20 = 125 байт.

Lсотрудник = 20 + 3 \* (20 + 1) + (11 + 1) + 4 + 3 = 112 байта.

Lуслуга = 20 + (20 + 1) + 20 + 20 = 81 байта.

Lуслуга-сотрудник = 20 + 20 = 40 байт.

Lпосещение = 20 + 20 + 20 + 3 + 1 +1= 65 байт.

Объем памяти, необходимый для размещения информационного фонда без учёта системных данных и указателей составит

|  |  |  |
| --- | --- | --- |
|  | [байт], | (3.2) |

где *N* – число типов записей в информационном фонде;

– количество записей *j*-го файла.

Предположим, что база содержит:

500 клиентов, 20 сотрудников, 10 услуг, 40 связей услуга-сотрудник, 2000 посещений.

.

Приращение информационного фонда

|  |  |  |
| --- | --- | --- |
|  | [байт-1], | (3.3) |

где *N* – число добавленных типов записей

– интенсивность добавления записей в файл *j*-го типа.

Время резервного копирования определяется интенсивностью отказов, сопровождающихся потерей данных

(дней)

Таким образом, были рассчитаны информационные параметры модели базы данных салона красоты. Так же было рассчитано время резервного копирования, которое показывает, насколько часто придется делать бекап базы данных, и составило 154 дней, что является средним и допустимым параметром.

**Вывода раздела 3**

В итоге, была выбрана СУБД MySQL за ее удобство и ряд преимуществ перед другими СУБД. Была построена физическая модель базы данных салона красоты, отражающая реальные параметры, а именно: тип атрибута, размер атрибута.

Приведенные тестирования показывают, что база данных имеет высокую целостность данных, а значит она легко изменяемая, при сохранении целостности.

Также база данных имеет разграничение доступа к ней, что существенно улучшает удобство использования, а также снижает количество времени на разработку.

1. **РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ**

# **4.1 Обоснование выбора языка программирования**

При написании приложения был выбран язык программирования *Java*. Этот выбор обусловлен тем, что данный язык позволяет применять объектно-ориентированный подход, кроссплатформенен, и также, что немало важно – сам интерпретатор *Java* является бесплатным.

Приложение написано с помощью среды разработки *Eclipse Java Photon* с использованием платформы *Java SWING*. Эта платформы выбрана за высокий уровень безопасности и удобства работы с ним, так как имеет множество реализованных функций, позволяющих упростить создание приложений. Для подключения и реализации работы с базой данных использовался *JDBC.*

Таким образом, был выбран язык для реализации клиентского приложения был выбран язык *Java*, для подключения базы данных использовался *JDBC.*

# **4.2 Разработка интерфейса пользователя**

Как упоминалось выше, для создания приложения использовался платформы *Java SWING*, позволяющий быстро создать удобный интерфейс пользователя. Были спроектированные следующие страницы: страница авторизации, страница с данными о клиентах, страница с данными о сотрудниках, страница с данными об услугах, запись клиента и изменение статуса услуги (оказана/ не оказана). Также во всех списках сущностей, доступ к редактированию которых имеется у пользователя, появляется соответствующие кнопки редактирования и удаления.

Для каждого поля ввода данных была добавлена валидация. Для каждой сущности реализованы страницы для редактирования и просмотра общей информации. Далее будут рассмотрены все перечисленные страницы на рисунках.

На рисунке 4.1 представлен скриншот приложения демонстрирующий страницу авторизации.

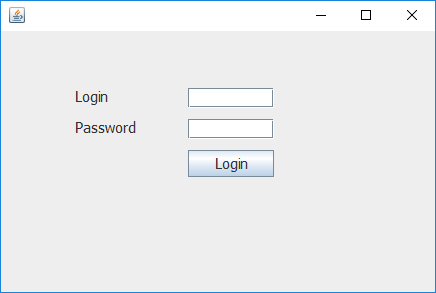


Рисунок 4.1 – Страница авторизации

В зависимости от того, какой пользователей авторизировался – администратор или другой пользователей – отрывается соответствующее окно главного меню. Главное меню администратора и главное меню пользователя представлены соответственно на рисунках 4.2 и 4.3.

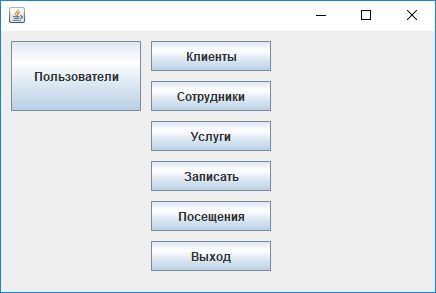


Рисунок 4.2 – Главное меню администратора

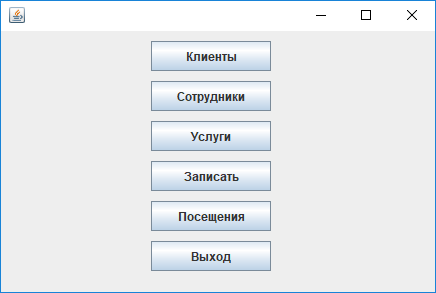


Рисунок 4.3 – Главное меню пользователя

На странице главного меню пользователя представлены кнопки «Клиенты», «Сотрудники», «Услуги», «Записать», «Посещения», «Выход». По нажатию на кнопку «Клиенты» - открывается окно, представленное на рисунке 4.4.

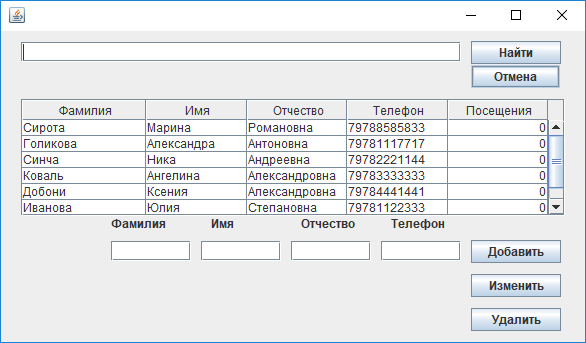


Рисунок 4.4 – Окно «Клиенты»

На данном окне расположена таблица с информацией о клиентах, кнопки «Добавить», «Изменить», «Удалить», по нажатию на которые происходят соответствующие названиям кнопок действия. Данные для добавления и изменения необходимо вводить в соответствующие поля под таблицей. По нажатию на строку в таблице поля ввода значений заполняются данными выбранной строки. Кнопка «Найти» осуществляет поиск по введенному в соответствующее поле слово и выводит результаты поиска в таблицу ; кнопка «Отмена» отменяет поиск и возвращает данные обо всех клиентах.

Аналогично устроены окна, открывающиеся по нажатию на кнопки «Сотрудники» и «Услуги» (рисунки 4.5 и 4.6).

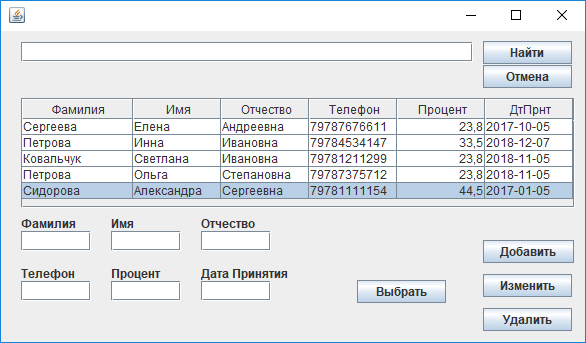


Рисунок 4.5 – Окно «Сотрудники»

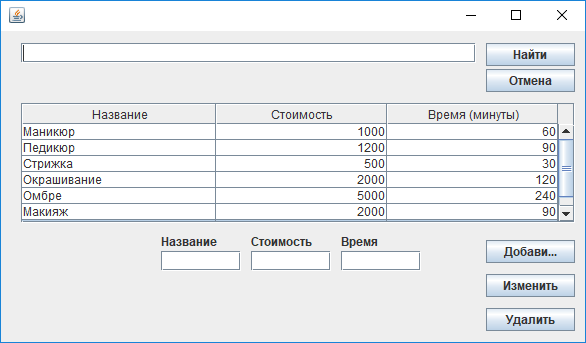


Рисунок 4.6 – Окно «Услуги»

В окне «Сотрудники» имеется кнопка «Выбрать» для выбора услуги, которую осуществляет сотрудник, по нажатию на которую открывается окно с услугами (рисунок 4.7).

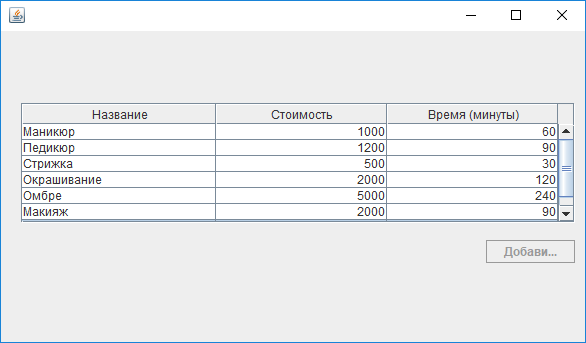


Рисунок 4.7 – Окно «Сотрудники»

Кнопка «Записать» открывает окно с формой для записи клиента (рисунок 4.8). По нажатию на кнопку «Записать» осуществляется запись и окно закрывается.

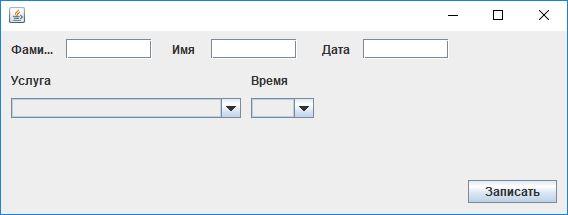


Рисунок 4.8 – Окно «Записать»

Кнопка «Посещения» открывает окно таблицей записей на услуги (рисунок 4.9). Кнопки «Оказано» и «Не оказано» не активны до тех пор, пока пользователей не выберет строку в таблице, нажав на нее ЛКМ. По нажатию на кнопку «Оказано» статус услуги меняется на «1», и становится активной кнопка «Печать чека» (см. далее) (рисунок 4.10); по нажатию на кнопку «Не оказано» статус услуги меняется на «-1». По нажатию на кнопку «Печать чека» данные об оказанной услуге сохраняются в файл, который в дальнейшем можно напечатать.

Также можно осуществить поиск по нажатию на кнопку «Найти».

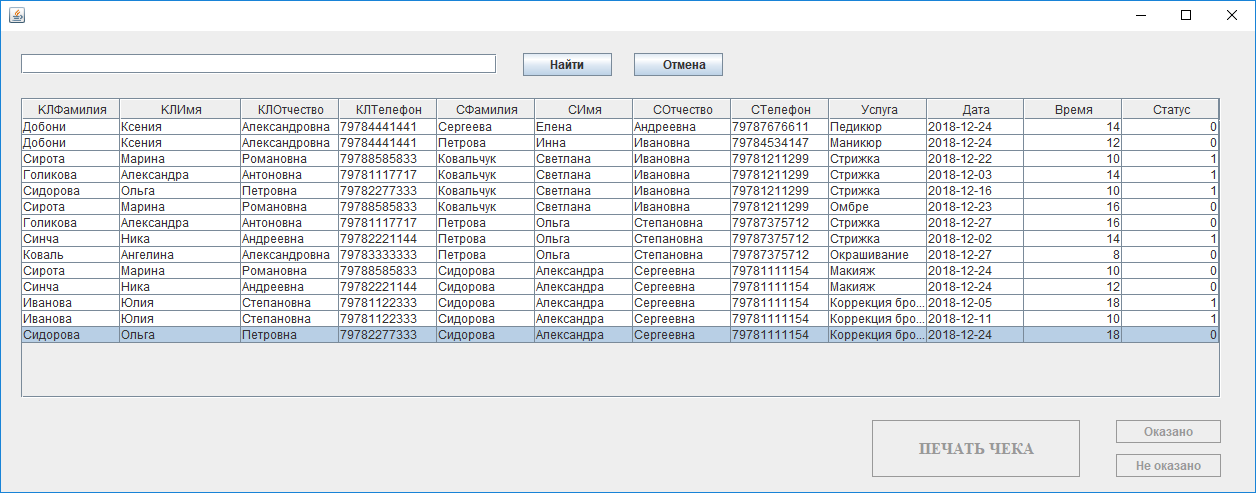


Рисунок 4.9 – Окно «Посещения» с не активными кнопками

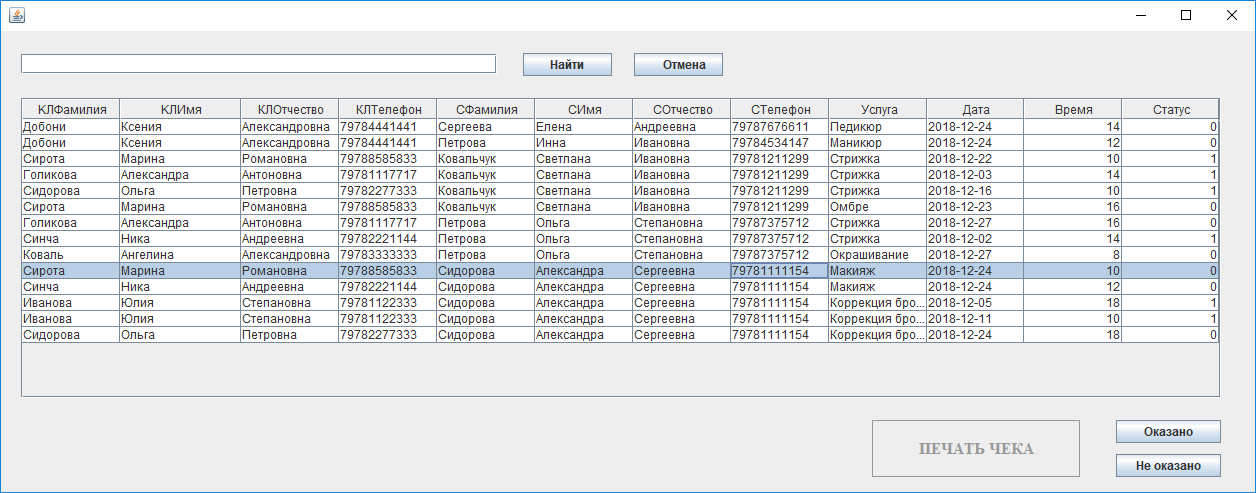


Рисунок 4.10 – Окно «Посещения» с активными кнопками

Кнопка «Выход» возвращает на страницу авторизации.

Страница главного меню администратора устроена аналогично странице главного меню пользователя, за исключение наличия кнопки «Пользователи».

По нажатию на кнопку «Пользователи» - открывается окно, представленное на рисунке 4.11.

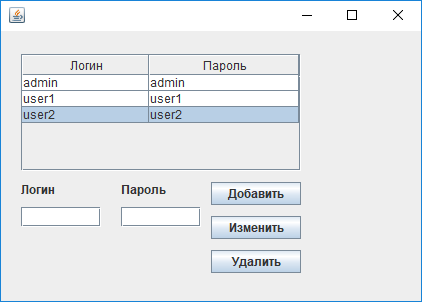


Рисунок 4.11 – Окно «Пользователи»

Приложение предусматривает некорректный ввод данных при добавлении, изменении данных таблицы и записи на услугу. На рисунках 4.12-4.13 представлены сообщения об ошибках при вводе фамилии и телефона соответственно.

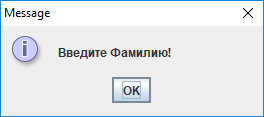


Рисунок 4.12 – Окно ошибки ввода фамилии

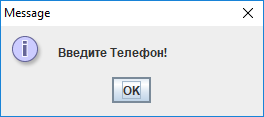


Рисунок 4.13 – Окно ошибки ввода номера телефона

При успешном выполнении действия появляется окно, представленное на рисунке 4.14.

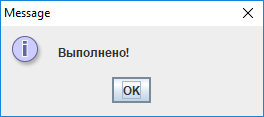


Рисунок 4.14 – Окно успешного выполнения команды

Произведем вывод клиентов с именем «Ангелина». Результаты представлены на рисунке 4.15 (выборка приложения) и рисунке 4.16 (выборка написанного вручную запроса в СУБД).

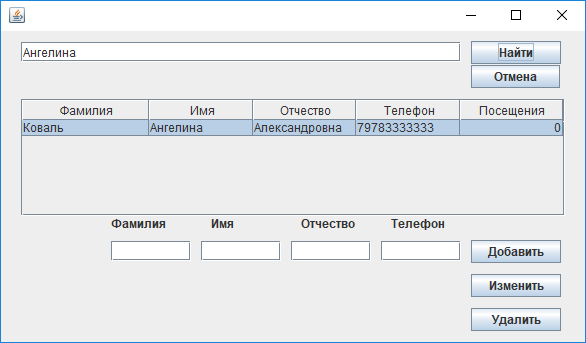


Рисунок 4.15 – Выборка приложением

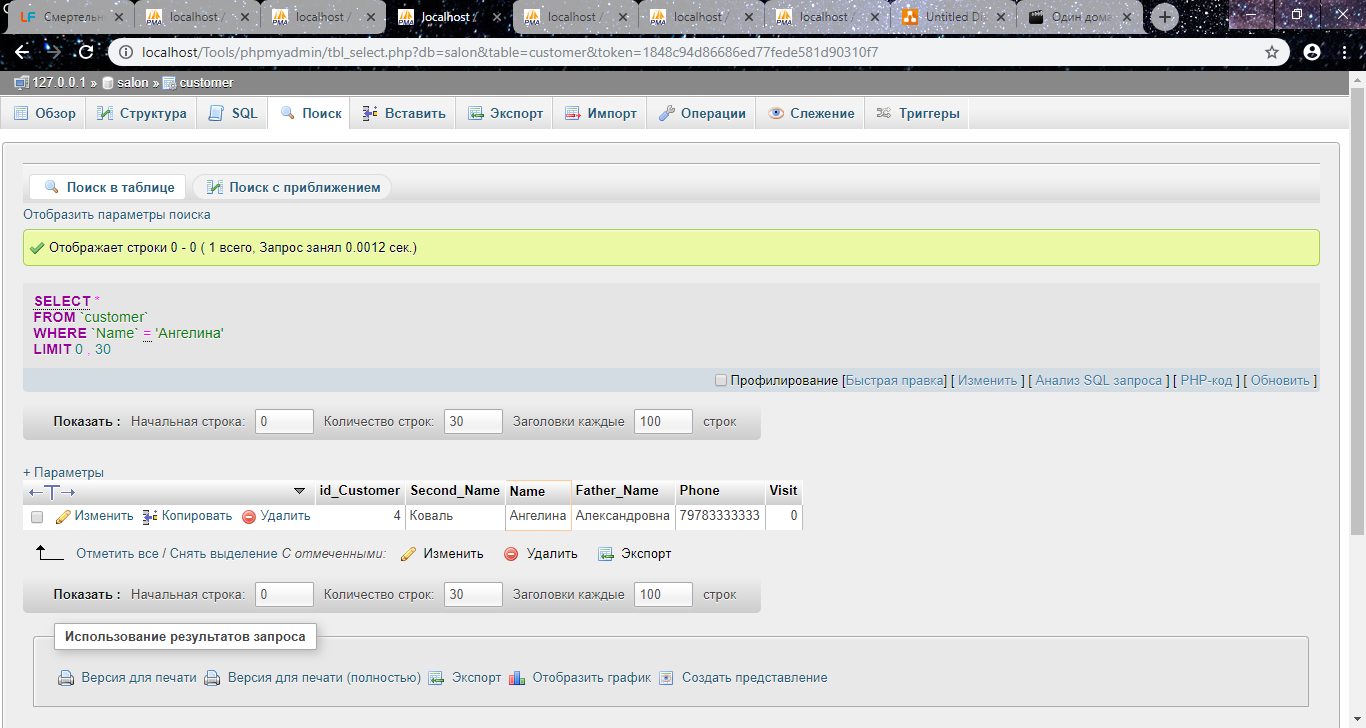


Рисунок 4.16 – Выборка запросом в СУБД

Исходя из рисунков 4.15 и 4.16 делаем вывод, о том, что приложение работает корректно.

В итоге, программа получила дружественный интерфейс пользователя, множество различных форм и меню, удобное в навигации. Также все формы имеют защиту и отклик на действия пользователя. Все формы и меню выдержано в едином стиле, что добавляет программе привлекательности.

**Выводы раздела 4**

В итоге, был выбран язык для реализации клиентского приложения был выбран язык *Java*, для подключения базы данных использовался *JDBC.*

Также можно сделать вывод, что программа получила дружественный интерфейс пользователя, множество различных форм и меню, удобное в навигации.

В ходе тщательного тестирования программы было выяснено, что программа соответствует стандартам и имеет весь необходимый функционал для выполнения поставленной задачи, а также предотвращения ошибок неправильного ввода пользователя. Были предусмотрены ошибки разного рода, в том числе неправильный ввод данных пользователем, предусмотрены вспомогательные окна для информирования пользователя о каких-либо изменениях в программе.

# **ЗАКЛЮЧЕНИЕ**

В курсовом проекте была спроектирована реляционная база данных на тему «База данных салона красоты». Написано приложение на языке *Java*, реализующее работу с базой данных. Также был разработан простой и изящный интерфейс для работы с базой данных. Кроме того, была обеспечена защита базы данных, путём разграничение прав доступа по строкам на прикладном уровне.

Для спроектированной логической модели базы данных была проведена нормализация до третьей нормальной формы, которая позволила уменьшить избыточность данных, а также сохранить целостность данных. Реализована физическая модель базы данных.

Были закреплены навыки проектирования логической модели базы данных, проведения их нормализации, построения физической модели, разработки программного обеспечения, использующие базы данных как способ хранения данных и был изучен *JDBC* для работы с базами данных.

В результате выполнения курсового проекта был получен программный продукт, реализующий работу с данными об услугах салоны красоты.

# **БИБЛИОГРАФИЧЕСКИЙСПИСОК**

1. Ульман Д. Основы систем баз данных/Д.Ульман.– М.: Финансы и статистика, 1983. – 334 с.
2. Barker R. CASE-Method. Entity-Relationship Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.
3. IDEF1 Information Modeling Method. Описание. Режим доступа – http://www.idef.com/IDEF1.htm.
4. Дейт К. Введение в системы баз данных. - К.: Диалектика, 1998.
5. Мейер Д. Теория реляционных баз данных. - М.: Мир, 1987. - 608с.

# **ПРИЛОЖЕНИЕА**

**Текст программы**

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Севастопольский государственный университет

Кафедра Информационных систем

Программа «Салон красоты»

Текст программы

РАЗРАБОТАЛА

Студент гр. ИС-б/32-O

Сирота М.Р.

2018

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Утверждено 23.12.2018

Среда программирования Eclipse IDE for Java Developers Version: Photon Release (4.8.0)

Дата последней коррекции: 15.12.2018

Версия 1.0

/\*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

customer.java

**package** ud;

**public** **class** customer **implements** Comparable<customer> {

**protected** **int** id;

**protected** String secondName;

**protected** String name;

**protected** String fatherName;

**protected** String phone;

**protected** **int** visit;

**public** customer(**int** id\_, String secondName\_, String name\_,String fatherName\_, String phone\_, **int** visit\_) {

id=id\_;

secondName=secondName\_;

name=name\_;

fatherName=fatherName\_;

phone=phone\_;

visit=visit\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getSecondName() {

**return** secondName;

}

**public** String getName() {

**return** name;

}

**public** String getFatherName() {

**return** fatherName;

}

**public** String getPhone() {

**return** phone;

}

**public** **int** getVisit() {

**return** visit;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setSecondName(String secondName) {

**this**.secondName = secondName;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setFatherName(String fatherName) {

**this**.fatherName = fatherName;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** **void** setVisit(**int** visit) {

**this**.visit = visit;

}

@Override

**public** **int** compareTo(customer c) {

**return** **this**.secondName.compareTo(c.secondName);

}

}

CustomerTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class CustomerTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<customer> CustomerT\_obj;

public CustomerTableModel(LinkedList<customer> CustomerT\_obj) {

this.CustomerT\_obj = CustomerT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 6;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Фамилия";

case 2:

return "Имя";

case 3:

return "Отчество";

case 4:

return "Телефон";

case 5:

return "Посещения";

}

return "";

}

public int getRowCount() {

return CustomerT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

customer p = CustomerT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getSecondName();

case 2:

return p.getName();

case 3:

return p.getFatherName();

case 4:

return p.getPhone();

case 5:

return p.getVisit();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void addRow(customer nr) { // добавление строки в таблицу

CustomerT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = CustomerT\_obj.indexOf(nr);

tablePane.table.setRowSelectionInterval(index, index);

}

public void updateRow(int index, customer nr) {// обновление строки в таблице

CustomerT\_obj.set(index, nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем измененную строку - иначе после обновления таблицы теряет выделение

tablePane.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) { // удаление строки из таблицы

Iterator<customer> isph = CustomerT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();// вызываем для обновления таблицы

} else

// JOPCT\_objionPane.showMessageDialog(myJframe.this,"Запись "+Model+" ненайдена!")

;

}

public void deleteAll() { // удаление строки из таблицы

int size = CustomerT\_obj.size();

CustomerT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

insert.java

package ud;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JOptionPane;

public class insert {

Statement st;

public insert() {

}

public int insertRow(String sql) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

try {

st = connection.createStatement();

int i = st.executeUpdate(sql, Statement.RETURN\_GENERATED\_KEYS);

if (i > 0) {

JOptionPane.showMessageDialog(null, "Выполнено!");

ResultSet rs = st.getGeneratedKeys();

if (rs.next()) {

return rs.getInt(1);

}

} else {

JOptionPane.showMessageDialog(null, "Невозможно выпонить команду!");

}

return i;

} catch (SQLException e) {

e.printStackTrace();

return -1;

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

return -1;

}

}

}

Java\_login.java

package ud;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;

import javax.swing.JPanel;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JButton;

import java.awt.Font;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.awt.event.ActionEvent;

import java.awt.BorderLayout;

import javax.swing.JPasswordField;

import javax.swing.JRootPane;

public class Java\_login extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private JTextField LoginTextField;

private JLabel lblLogin;

private JLabel lblPassword;

private JPasswordField passwordTextField;

public static boolean isAdmin(String x)

{

Pattern p = Pattern.compile("^admin$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Java\_login frame = new Java\_login();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public Java\_login() {

setBounds(100, 100, 450, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(new BorderLayout());

JPanel panel = new JPanel();

panel.setBounds(0, 0, 434, 261);

getContentPane().add(panel);

panel.setLayout(null);

JPanel panelMenu = new JPanel();

panelMenu.setBounds(0, 0, 434, 261);

panelMenu.setLayout(null);

JButton btnCustomerButton = new JButton("\u041A\u043B\u0438\u0435\u043D\u0442\u044B");

btnCustomerButton.setBounds(150, 10, 120, 30);

btnCustomerButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePane frame = new tablePane();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnCustomerButton);

JButton btnWorkerButton = new JButton("\u0421\u043E\u0442\u0440\u0443\u0434\u043D\u0438\u043A\u0438");

btnWorkerButton.setBounds(150, 50, 120, 30);

btnWorkerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForWorker frame = new tablePanelForWorker ();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnWorkerButton);

JButton btnServiceButton = new JButton("\u0423\u0441\u043B\u0443\u0433\u0438");

btnServiceButton.setBounds(150, 90, 120, 30);

btnServiceButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForService frame = new tablePanelForService(2);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnServiceButton);

JButton btnRecondButton = new JButton("Записать");

btnRecondButton.setBounds(150, 130, 120, 30);

btnRecondButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

visitAdd frame = new visitAdd();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnRecondButton);

JButton btnShowVisit = new JButton("Посещения");

btnShowVisit.setBounds(150, 170, 120, 30);

btnShowVisit.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

panelRecord frame = new panelRecord();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnShowVisit);

JButton btnExitButton = new JButton("Выход");

btnExitButton.setBounds(150, 210, 120, 30);

btnExitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

getContentPane().remove(panelMenu);

setContentPane(panel);

LoginTextField.setText(null);

passwordTextField.setText(null);

panel.setLayout(null);

update(getGraphics());

/\*dispose();

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Java\_login frame = new Java\_login();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});\*/

}

});

panelMenu.add(btnExitButton);

JButton btnUsers = new JButton("Пользователи");

btnUsers.setBounds(10, 10, 130, 70);

btnUsers.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForUser frame = new tablePanelForUser();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

});

panelMenu.add(btnUsers);

btnUsers.setVisible(false);

LoginTextField = new JTextField();

LoginTextField.setBounds(187, 57, 86, 20);

panel.add(LoginTextField);

//LoginTextField.setColumns(10);

JButton btnSignUp = new JButton("Login");

btnSignUp.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

// JOptionPane.showMessageDialog(null, "Connection to Store DB succesfull!");

String uName = LoginTextField.getText();

if(isAdmin(uName)){

btnUsers.setVisible(true);

}

@SuppressWarnings("deprecation")

String query = "SELECT \* FROM `user` WHERE `login`='"+LoginTextField.getText()+"' and `password`='"+passwordTextField.getText()+"'";

PreparedStatement statement = connection.prepareStatement(query);

//statement.setString(1, uName);

//statement.setString(2, pass);

ResultSet set = statement.executeQuery();

if (set.next()) {

getContentPane().remove(panel);

// panel.removeAll();

// panel.add(panelMenu);

// getContentPane().add(panelMenu);

setContentPane(panelMenu);

panelMenu.setLayout(new BorderLayout());

update(getGraphics());

} else {

JOptionPane.showMessageDialog(null, "Login uncorrect!");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

passwordTextField = new JPasswordField();

passwordTextField.setBounds(187, 88, 86, 20);

panel.add(passwordTextField);

passwordTextField.setColumns(10);

btnSignUp.setBounds(187, 119, 86, 27);

panel.add(btnSignUp);

btnSignUp.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblPassword = new JLabel("Password");

lblPassword.setBounds(74, 87, 61, 19);

panel.add(lblPassword);

lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblLogin = new JLabel("Login");

lblLogin.setBounds(74, 56, 33, 19);

panel.add(lblLogin);

lblLogin.setFont(new Font("Tahoma", Font.PLAIN, 15));

JRootPane rootPane = SwingUtilities.getRootPane(btnSignUp);

rootPane.setDefaultButton(btnSignUp);

}

}

record.java

**package** ud;

**public** **class** record {

**protected** String secondNameCustomer;

**protected** String nameCustomer;

**protected** String fatherNameCustomer;

**protected** String phoneCustomer;

**protected** String secondNameWorker;

**protected** String nameWorker;

**protected** String fatherNameWorker;

**protected** String phoneWorker;

**protected** String nameService;

**protected** String dateService;

**protected** **int** hourService;

**protected** **int** statusService;

**public** record (String secondNameCustomer\_, String nameCustomer\_,String fatherNameCustomer\_, String phoneCustomer\_,

String secondNameWorker\_, String nameWorker\_, String fatherNameWorker\_, String phoneWorker\_,

String nameService\_, String dateService\_, **int** hourService\_, **int** statusService\_) {

secondNameCustomer=secondNameCustomer\_;

nameCustomer=nameCustomer\_;

fatherNameCustomer=fatherNameCustomer\_;

phoneCustomer=phoneCustomer\_;

secondNameWorker = secondNameWorker\_;

nameWorker = nameWorker\_;

fatherNameWorker = fatherNameWorker\_;

phoneWorker = phoneWorker\_;

nameService=nameService\_;

dateService=dateService\_;

hourService=hourService\_;

statusService=statusService\_;

}

**public** String getSecondNameCustomer() {

**return** secondNameCustomer;

}

**public** String getNameCustomer() {

**return** nameCustomer;

}

**public** String getFatherNameCustomer() {

**return** fatherNameCustomer;

}

**public** String getPhoneCustomer() {

**return** phoneCustomer;

}

**public** **void** setSecondNameCustomer(String secondNameCustomer) {

**this**.secondNameCustomer = secondNameCustomer;

}

**public** **void** setNameCustomer(String nameCustomer) {

**this**.nameCustomer = nameCustomer;

}

**public** **void** setFatherNameCustomer(String fatherNameCustomer) {

**this**.fatherNameCustomer = fatherNameCustomer;

}

**public** **void** setPhoneCustomer(String phoneCustomer) {

**this**.phoneCustomer = phoneCustomer;

}

**public** String getSecondNameWorker() {

**return** secondNameWorker;

}

**public** String getNameWorker() {

**return** nameWorker;

}

**public** String getFatherNameWorker() {

**return** fatherNameWorker;

}

**public** String getPhoneWorker() {

**return** phoneWorker;

}

**public** **void** setSecondNameWorker(String secondNameWorker) {

**this**.secondNameWorker = secondNameWorker;

}

**public** **void** setName(String nameWorker) {

**this**.nameWorker = nameWorker;

}

**public** **void** setFatherNameWorker(String fatherNameWorker) {

**this**.fatherNameWorker = fatherNameWorker;

}

**public** **void** setPhoneWorker(String phoneWorker) {

**this**.phoneWorker = phoneWorker;

}

**public** String getNameService() {

**return** nameService;

}

**public** **int** getHourService() {

**return** hourService;

}

**public** String getDateService() {

**return** dateService;

}

**public** **int** getStatusService() {

**return** statusService;

}

**public** **void** setNameService(String nameService) {

**this**.nameService = nameService;

}

**public** **void** setHourService(**int** hourService) {

**this**.hourService = hourService;

}

**public** **void** setDateService(String dateService) {

**this**.dateService = dateService;

}

**public** **void** setStatusService(**int** statusService) {

**this**.statusService = statusService;

}

}

RecordTableModel.java

package ud;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class RecordTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<record> RecordT\_obj;

public RecordTableModel(LinkedList<record> RecordT\_obj) {// конструктор модели

this.RecordT\_obj = RecordT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 12;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "KЛФамилия";

case 1:

return "KЛИмя";

case 2:

return "КЛОтчество";

case 3:

return "КЛТелефон";

case 4:

return "СФамилия";

case 5:

return "СИмя";

case 6:

return "СОтчество";

case 7:

return "СТелефон";

case 8:

return "Услуга";

case 9:

return "Дата";

case 10:

return "Время";

case 11:

return "Статус";

}

return "";

}

public int getRowCount() {

return RecordT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

record p = RecordT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getSecondNameCustomer();

case 1:

return p.getNameCustomer();

case 2:

return p.getFatherNameCustomer();

case 3:

return p.getPhoneCustomer();

case 4:

return p.getSecondNameWorker();

case 5:

return p.getNameWorker();

case 6:

return p.getFatherNameWorker();

case 7:

return p.getPhoneWorker();

case 8:

return p.getNameService();

case 9:

return p.getDateService();

case 10:

return p.getHourService();

case 11:

return p.getStatusService();

}

return "";

}

public void addRow(record nr) { // добавление строки в таблицу

RecordT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = RecordT\_obj.indexOf(nr);

panelRecord.table.setRowSelectionInterval(index, index);

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, record nr) {

RecordT\_obj.set(index, nr);

fireTableDataChanged();

panelRecord.table.setRowSelectionInterval(index, index);

}

public void deleteAll() { // удаление строки из таблицы

int size = RecordT\_obj.size();

RecordT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

service.java

**package** ud;

**import** java.util.Comparator;

**public** **class** service **implements** Comparator<service> {

**protected** **int** id;

**protected** String name;

**protected** **int** cost;

**protected** **int** time;

**public** service(**int** id\_, String name\_, **int** cost\_, **int** time\_) {

id=id\_;

name=name\_;

cost=cost\_;

time=time\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getName() {

**return** name;

}

**public** **int** getCost() {

**return** cost;

}

**public** **int** getTime() {

**return** time;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setCost(**int** cost) {

**this**.cost = cost;

}

**public** **void** setTime(**int** time) {

**this**.time = time;

}

@Override

**public** **int** compare(service c1, service c2) {

**return** c1.getID() - c2.getID();

}

}

ServiceTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class ServiceTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<service> ServiceT\_obj;

public ServiceTableModel(LinkedList<service> ServiceT\_obj) {

this.ServiceT\_obj = ServiceT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 4;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Название";

case 2:

return "Cтоимость";

case 3:

return "Время (минуты)";

}

return "";

}

public void addRow(service nr) { // добавление строки в таблицу

ServiceT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = ServiceT\_obj.indexOf(nr);

tablePanelForService.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return ServiceT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

service p = ServiceT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getName();

case 2:

return p.getCost();

case 3:

return p.getTime();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, service nr) {

ServiceT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForService.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) {

Iterator<service> isph = ServiceT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = ServiceT\_obj.size();

ServiceT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

tablePane,java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePane extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldName;

private JTextField textFieldSecondName;

private JTextField textFieldFatherName;

private JTextField textFieldPhone;

private JTextField textFieldForSearch;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePane frame = new tablePane();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public static boolean isNumeric(String x)

{

Pattern p = Pattern.compile("^\\d{11}$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public tablePane() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 574, 311);

getContentPane().add(panel);

final LinkedList<customer> tbl = new LinkedList<customer>();

final CustomerTableModel CustomerT\_obj = new CustomerTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(470, 209, 90, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p) && sn.length() != 11) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1) {

String query = "INSERT INTO `salon`.`customer` (`id\_Customer`, `Second\_Name`, `Name`, `Father\_Name`, `Phone`, `Visit`)"

+ " VALUES (NULL, '" + sn + "', '" + n + "', '" + fn + "', '" + p + "', '0');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.addRow(new customer(newId, sn, n, fn, p, 0));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Неверные данные" + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 68, 544, 117);

panel.add(scrollPane);

table = new JTable(CustomerT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblName = new JLabel("\u0418\u043C\u044F");

lblName.setBounds(210, 185, 70, 14);

panel.add(lblName);

JLabel lblSecondName = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondName.setBounds(110, 185, 70, 14);

panel.add(lblSecondName);

JLabel lblFatherName = new JLabel("\u041E\u0442\u0447\u0435\u0441\u0442\u0432\u043E");

lblFatherName.setBounds(300, 185, 70, 14);

panel.add(lblFatherName);

JLabel lblPhone = new JLabel("\u0422\u0435\u043B\u0435\u0444\u043E\u043D");

lblPhone.setBounds(390, 185, 70, 14);

panel.add(lblPhone);

textFieldName = new JTextField();

textFieldName.setBounds(200, 210, 80, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

textFieldSecondName = new JTextField();

textFieldSecondName.setBounds(110, 210, 80, 20);

textFieldSecondName.setColumns(10);

panel.add(textFieldSecondName);

textFieldFatherName = new JTextField();

textFieldFatherName.setBounds(290, 210, 80, 20);

textFieldFatherName.setColumns(10);

panel.add(textFieldFatherName);

textFieldPhone = new JTextField();

textFieldPhone.setBounds(380, 210, 80, 20);

textFieldPhone.setColumns(10);

panel.add(textFieldPhone);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(470, 243, 90, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p) && sn.length() != 11) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagN == 1 && flagP == 1) {

String query = "UPDATE `salon`.`customer` SET `Second\_Name` = '" + sn + "', `Name` = '" + n

+ "', `Father\_Name` = '" + fn + "', `Phone` = '" + p + "'"

+ " WHERE `customer`.`id\_Customer` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.updateRow(table.getSelectedRow(), new customer(i, sn, n, fn, p, 0));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(470, 277, 90, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `customer` WHERE `customer`.`id\_Customer` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

CustomerT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.setBounds(470, 10, 90, 23);

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search= textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `customer` "

+ "WHERE `Second\_Name` LIKE '" + search + "' OR `Name` LIKE '" + search + "' OR `"

+ "Father\_Name`LIKE '" + search + "' OR `Phone`LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

//if (!res.next()) {

//JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

CustomerT\_obj.deleteAll();

while (res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

panel.add(btnSearch);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 11, 440, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

CustomerT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `customer`");

ResultSet res = statement.executeQuery();

while(res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2),

res.getString(3),res.getString(4),res.getString(5),res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnCancel.setBounds(470, 34, 89, 23);

panel.add(btnCancel);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldSecondName.setText(table.getValueAt(row, 1).toString());

textFieldName.setText(table.getValueAt(row, 2).toString());

textFieldFatherName.setText(table.getValueAt(row, 3).toString());

textFieldPhone.setText(table.getValueAt(row, 4).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `customer`");

ResultSet res = statement.executeQuery();

while(res.next()) {

CustomerT\_obj.addRow(new customer(res.getInt(1), res.getString(2),

res.getString(3),res.getString(4),res.getString(5),res.getInt(6)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForService.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForService extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldTime;

private JTextField textFieldCost;

private JTextField textFieldName;

private JTextField textFieldForSearch;

public String idServiceForWorker;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isNumeric(String x)

{

Pattern p = Pattern.compile("^\\d+(?:\\.\\d+)?$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public String getIDForWorker(String x)

{

return idServiceForWorker;

}

/\*\*

\* Create the frame.

\*/

public tablePanelForService(int choose) {

if (choose==1)

loadTwo();

else load();

}

public void load() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<service> tbl = new LinkedList<service>();

final ServiceTableModel ServiceT\_obj = new ServiceTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(485, 209, 89, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

String n=textFieldName.getText();

String c=textFieldCost.getText();

String t=textFieldTime.getText();

int flagN = 1;

int flagC = 1;

int flagT = 1;

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Название!");

}

if (!isNumeric(c)) {

flagC = 0;

JOptionPane.showMessageDialog(null, "Введите Стоимость!");

}

if (!isNumeric(t)) {

flagT = 0;

JOptionPane.showMessageDialog(null, "Введите Длительность!");

}

if (flagN == 1 && flagC == 1 && flagT == 1) {

String query="INSERT INTO `salon`.`sevice` (`id\_Service`, `Name`, `Cost`, `Time`)"

+" VALUES (NULL, '"+n+"', '"+c+"', '"+t+"');";

insert it=new insert();

int newId = it.insertRow(query);

if(newId > 0 ) {

ServiceT\_obj.addRow(new service(newId,n,Integer.parseInt(textFieldCost.getText()),

Integer.parseInt(textFieldTime.getText())));

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 72, 554, 120);

panel.add(scrollPane);

table = new JTable(ServiceT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblTime = new JLabel("\u0412\u0440\u0435\u043C\u044F");

lblTime.setBounds(340, 203, 50, 14);

panel.add(lblTime);

JLabel lblCost = new JLabel("\u0421\u0442\u043E\u0438\u043C\u043E\u0441\u0442\u044C");

lblCost.setBounds(250, 203, 70, 14);

panel.add(lblCost);

textFieldTime = new JTextField();

textFieldTime.setBounds(340, 220, 80, 20);

panel.add(textFieldTime);

textFieldTime.setColumns(10);

textFieldCost = new JTextField();

textFieldCost.setBounds(250, 220, 80, 20);

textFieldCost.setColumns(10);

panel.add(textFieldCost);

JLabel lblName = new JLabel("\u041D\u0430\u0437\u0432\u0430\u043D\u0438\u0435");

lblName.setBounds(160, 203, 70, 14);

panel.add(lblName);

textFieldName = new JTextField();

textFieldName.setBounds(160, 220, 80, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldName.setText(table.getValueAt(row, 1).toString());

textFieldCost.setText(table.getValueAt(row, 2).toString());

textFieldTime.setText(table.getValueAt(row, 3).toString());

idServiceForWorker=table.getValueAt(row, 0).toString();

// сделать кнопку обновления недоступной(пока нет изменений)

//btnUpd.setEnabled(false);

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(485, 243, 89, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String n=textFieldName.getText();

String c=textFieldCost.getText();

String t=textFieldTime.getText();

int flagN = 1;

int flagC = 1;

int flagT = 1;

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Название!");

}

if (!isNumeric(c)) {

flagC = 0;

JOptionPane.showMessageDialog(null, "Введите Стоимость!");

}

if (!isNumeric(t)) {

flagT = 0;

JOptionPane.showMessageDialog(null, "Введите Длительность!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagN == 1 && flagC == 1 && flagT == 1) {

String query = "UPDATE `salon`.`sevice` SET `Name` = '" + n + "', `Cost` = '" + c

+ "', `Time` = '" + t + "'"

+ " WHERE `sevice`.`id\_Service` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

ServiceT\_obj.updateRow(table.getSelectedRow(), new service(i, n,Integer.parseInt(textFieldCost.getText()),

Integer.parseInt(textFieldTime.getText())));

textFieldName.setText(null);

textFieldCost.setText(null);

textFieldTime.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(485, 277, 89, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `sevice` WHERE `sevice`.`id\_Service` = " + table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

ServiceT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search= textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `sevice` "

+ "WHERE `Name` LIKE '" + search + "' OR `Cost` LIKE '" + search + "' OR `"

+ "Time`LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

//if (!res.next()) {

//JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

ServiceT\_obj.deleteAll();

while (res.next()) {

ServiceT\_obj.addRow(new service(res.getInt(1), res.getString(2),res.getInt(3),res.getInt(4)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnSearch.setBounds(485, 12, 89, 23);

panel.add(btnSearch);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ServiceT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet res = statement.executeQuery();

while(res.next()) {

ServiceT\_obj.addRow(new service(res.getInt(1), res.getString(2),res.getInt(3),res.getInt(4)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

btnCancel.setBounds(485, 38, 89, 23);

panel.add(btnCancel);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 12, 455, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

ServiceT\_obj.addRow(new service(rest.getInt("id\_Service"),rest.getString("Name"), rest.getInt("Cost"),

rest.getInt("Time")));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

public String idConn;

public void setIDConn(String idConn) {

this.idConn = idConn;

}

public void loadTwo() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<service> tbl = new LinkedList<service>();

final ServiceTableModel ServiceT\_obj = new ServiceTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(485, 209, 89, 23);

btnAdd.setEnabled(false);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

int row = table.getSelectedRow();

String iy = table.getValueAt(row, 0).toString();

String query="INSERT INTO `salon`.`service\_worker` (`id\_Service`, `id\_Worker`) "

+" VALUES ('"+iy+"', '"+idConn+"');";

insert it=new insert();

it.insertRow(query);

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

dispose();

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 72, 554, 120);

panel.add(scrollPane);

table = new JTable(ServiceT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

btnAdd.setEnabled(true);

int row = table.getSelectedRow();

table.getValueAt(row, 0).toString();

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `sevice`");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

ServiceT\_obj.addRow(new service(rest.getInt("id\_Service"),rest.getString("Name"), rest.getInt("Cost"),

rest.getInt("Time")));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForUser.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForUser extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldPassword;

private JTextField textFieldLogin;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForUser frame = new tablePanelForUser();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[A-Za-z0-9]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public tablePanelForUser() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE );

setBounds(100, 100, 436, 309);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 420, 270);

getContentPane().add(panel);

final LinkedList<user> tbl = new LinkedList<user>();

final userTableModel UserT\_obj = new userTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(210, 151, 90, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldLogin.getText();

String n = textFieldPassword.getText();

int flagSn = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите логин!");

}

if (flagSn == 1) {

String query = "INSERT INTO `salon`.`user` (`login`, `password`)"

+ " VALUES ('" + sn + "', '" + n + "');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.addRow(new user(sn, n));

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Неверные данные" + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 23, 280, 117);

panel.add(scrollPane);

table = new JTable(UserT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblPassword = new JLabel("\u041F\u0430\u0440\u043E\u043B\u044C");

lblPassword.setBounds(120, 151, 70, 14);

panel.add(lblPassword);

JLabel lblLogin = new JLabel("\u041B\u043E\u0433\u0438\u043D");

lblLogin.setBounds(20, 151, 70, 14);

panel.add(lblLogin);

textFieldPassword = new JTextField();

textFieldPassword.setBounds(120, 176, 80, 20);

panel.add(textFieldPassword);

textFieldPassword.setColumns(10);

textFieldLogin = new JTextField();

textFieldLogin.setBounds(20, 176, 80, 20);

textFieldLogin.setColumns(10);

panel.add(textFieldLogin);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(210, 185, 90, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldLogin.getText();

String n = textFieldPassword.getText();

int flagSn = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

int row = table.getSelectedRow();

if (flagSn == 1) {

String query = "UPDATE `salon`.`user` SET `login` = '" + sn + "', `password` = '" + n

+ "'"

+ " WHERE `user`.`login` = '" + table.getValueAt(row, 0).toString() + "';";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.updateRow(table.getSelectedRow(), new user(sn, n));

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(210, 219, 90, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

String i = table.getValueAt(row, 0).toString();

String query = "DELETE FROM `salon`. `user` WHERE `user`.`login` = '" + table.getValueAt(row, 0).toString() + "';";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

UserT\_obj.deleteRow(i);

}

textFieldLogin.setText(null);

textFieldPassword.setText(null);

}

});

panel.add(btnDelete);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldLogin.setText(table.getValueAt(row, 0).toString());

textFieldPassword.setText(table.getValueAt(row, 1).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" +

"FROM `user`");

ResultSet res = statement.executeQuery();

while(res.next()) {

UserT\_obj.addRow(new user(res.getString(1), res.getString(2)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

}

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.LinkedList;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

public class tablePanelForWorker extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

static JTable table;

private JTextField textFieldName;

private JTextField textFieldSecondName;

private JTextField textFieldFatherName;

private JTextField textFieldPhone;

private JTextField textFieldProssent;

private JTextField textFieldDate;

private JTextField textFieldForSearch;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

tablePanelForWorker frame = new tablePanelForWorker();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isNumeric(String x) {

Pattern p = Pattern.compile("^\\d{11}$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isWorld(String x) {

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isDate(String x) {

Pattern p = Pattern.compile("^20\\d{2}-\\d{1,2}-\\d{1,2}$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Create the frame.

\*/

public tablePanelForWorker() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 600, 350);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

getContentPane().setLayout(null);

contentPane.setLayout(null);

JPanel panel = new JPanel();

panel.setBounds(0, 0, 584, 311);

getContentPane().add(panel);

final LinkedList<worker> tbl = new LinkedList<worker>();

final WorkerTableModel WorkerT\_obj = new WorkerTableModel(tbl);

JButton btnAdd = new JButton("\u0414\u043E\u0431\u0430\u0432\u0438\u0442\u044C");

btnAdd.setBounds(482, 209, 91, 23);

btnAdd.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

String pr = textFieldProssent.getText();

String d = textFieldDate.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

int flagPr = 1;

int flagD = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p)) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1 && flagPr == 1 && flagD == 1) {

String query = "INSERT INTO `salon`.`worker` (`id\_Worker`, `Second\_Name`, `Name`, `Father\_Name`, `Phone`, `Prossent`, `Start\_Day`) "

+ " VALUES (NULL, '" + sn + "', '" + n + "', '" + fn + "', '" + p + "', '" + pr + "', '"

+ d + "');";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.addRow(

new worker(newId, sn, n, fn, p, Float.parseFloat(textFieldProssent.getText()), d));

}

int row = table.getSelectedRow();

String idWorkerForInsertToTableWorkerService = table.getValueAt(row, 0).toString();

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

frame.setIDConn(idWorkerForInsertToTableWorkerService);

} catch (Exception e1) {

e1.printStackTrace();

}

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.setLayout(null);

panel.add(btnAdd);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(20, 67, 554, 110);

panel.add(scrollPane);

table = new JTable(WorkerT\_obj);

table.getColumnModel().getColumn(1).setPreferredWidth(98);

scrollPane.setViewportView(table);

JLabel lblName = new JLabel("\u0418\u043C\u044F");

lblName.setBounds(110, 185, 36, 14);

panel.add(lblName);

JLabel lblSecondName = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondName.setBounds(20, 185, 70, 14);

panel.add(lblSecondName);

JLabel lblFatherName = new JLabel("\u041E\u0442\u0447\u0435\u0441\u0442\u0432\u043E");

lblFatherName.setBounds(200, 185, 70, 14);

panel.add(lblFatherName);

JLabel lblPhone = new JLabel("\u0422\u0435\u043B\u0435\u0444\u043E\u043D");

lblPhone.setBounds(20, 235, 70, 14);

panel.add(lblPhone);

textFieldName = new JTextField();

textFieldName.setBounds(110, 200, 70, 20);

panel.add(textFieldName);

textFieldName.setColumns(10);

textFieldSecondName = new JTextField();

textFieldSecondName.setBounds(20, 200, 70, 20);

textFieldSecondName.setColumns(10);

panel.add(textFieldSecondName);

textFieldFatherName = new JTextField();

textFieldFatherName.setBounds(200, 200, 70, 20);

textFieldFatherName.setColumns(10);

panel.add(textFieldFatherName);

textFieldPhone = new JTextField();

textFieldPhone.setBounds(20, 250, 70, 20);

textFieldPhone.setColumns(10);

panel.add(textFieldPhone);

JLabel lblProssent = new JLabel("\u041F\u0440\u043E\u0446\u0435\u043D\u0442");

lblProssent.setBounds(110, 235, 70, 14);

panel.add(lblProssent);

textFieldProssent = new JTextField();

textFieldProssent.setBounds(110, 250, 70, 20);

panel.add(textFieldProssent);

textFieldProssent.setColumns(10);

JLabel label = new JLabel("\u0414\u0430\u0442\u0430 \u041F\u0440\u0438\u043D\u044F\u0442\u0438\u044F");

label.setBounds(200, 235, 91, 14);

panel.add(label);

textFieldDate = new JTextField();

textFieldDate.setBounds(200, 250, 70, 20);

panel.add(textFieldDate);

textFieldDate.setColumns(10);

JButton btnUpdate = new JButton("\u0418\u0437\u043C\u0435\u043D\u0438\u0442\u044C");

btnUpdate.setBounds(482, 243, 89, 23);

btnUpdate.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (table.getSelectedRow() >= 0) {

String sn = textFieldSecondName.getText();

String n = textFieldName.getText();

String fn = textFieldFatherName.getText();

String p = textFieldPhone.getText();

String pr = textFieldProssent.getText();

String d = textFieldDate.getText();

int flagSn = 1;

int flagN = 1;

int flagFn = 1;

int flagP = 1;

int flagPr = 1;

int flagD = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (!isWorld(fn)) {

flagFn = 0;

JOptionPane.showMessageDialog(null, "Введите Отчество!");

}

if (!isNumeric(p)) {

flagP = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

if (flagSn == 1 && flagN == 1 && flagFn == 1 && flagP == 1 && flagPr == 1 && flagD == 1) {

String query = "UPDATE `salon`.`worker` SET `Second\_Name` = '" + sn + "', `Name` = '" + n

+ "', `Father\_Name` = '" + fn + "', `Phone` = '" + p + "'," + "`Prossent` = '" + pr

+ "',`Start\_Day` = '" + d + "'" + " WHERE `worker`.`id\_Worker` = "

+ table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.updateRow(table.getSelectedRow(),

new worker(i, sn, n, fn, p, Float.parseFloat(textFieldProssent.getText()), d));

textFieldSecondName.setText(null);

textFieldName.setText(null);

textFieldFatherName.setText(null);

textFieldPhone.setText(null);

textFieldProssent.setText(null);

textFieldDate.setText(null);

}

}

}

}

});

panel.add(btnUpdate);

JButton btnDelete = new JButton("\u0423\u0434\u0430\u043B\u0438\u0442\u044C");

btnDelete.setBounds(482, 277, 89, 23);

btnDelete.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

int i = (int) table.getValueAt(row, 0);

String query = "DELETE FROM `salon`. `worker` WHERE `worker`.`id\_Worker` = "

+ table.getValueAt(row, 0).toString() + ";";

insert it = new insert();

int newId = it.insertRow(query);

if (newId > 0) {

WorkerT\_obj.deleteRow(i);

}

}

});

panel.add(btnDelete);

textFieldForSearch = new JTextField();

textFieldForSearch.setBounds(20, 11, 452, 20);

panel.add(textFieldForSearch);

textFieldForSearch.setColumns(10);

JButton btnSearch = new JButton("\u041D\u0430\u0439\u0442\u0438");

btnSearch.setBounds(482, 10, 89, 23);

btnSearch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String search = textFieldForSearch.getText();

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection

.prepareStatement("SELECT \* FROM `worker` " + "WHERE `Second\_Name` LIKE '" + search

+ "' OR `Name` LIKE '" + search + "' OR `" + "Father\_Name` LIKE '" + search

+ "' OR `Phone` LIKE '" + search + "'" + "OR `Prossent` LIKE '" + search + "'");

ResultSet res = statement.executeQuery();

// if (!res.next()) {

// JOptionPane.showMessageDialog(null, "Ничего не найдено...");

// }

WorkerT\_obj.deleteAll();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.add(btnSearch);

JButton btnCancel = new JButton("\u041E\u0442\u043C\u0435\u043D\u0430");

btnCancel.setBounds(482, 34, 89, 23);

btnCancel.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

WorkerT\_obj.deleteAll();

textFieldForSearch.setText(null);

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" + "FROM `worker`");

ResultSet res = statement.executeQuery();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3),

res.getString(4), res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

}

});

panel.add(btnCancel);

JButton btnSelect = new JButton("\u0412\u044B\u0431\u0440\u0430\u0442\u044C");

btnSelect.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

int row = table.getSelectedRow();

String idWorkerForInsertToTableWorkerService = table.getValueAt(row, 0).toString();

try {

tablePanelForService frame = new tablePanelForService(1);

frame.setVisible(true);

frame.setIDConn(idWorkerForInsertToTableWorkerService);

} catch (Exception e1) {

e1.printStackTrace();

}

}

});

btnSelect.setBounds(356, 249, 89, 23);

panel.add(btnSelect);

table.addMouseListener(new MouseListener() {

public void mouseClicked(MouseEvent e) {

int row = table.getSelectedRow();

textFieldSecondName.setText(table.getValueAt(row, 1).toString());

textFieldName.setText(table.getValueAt(row, 2).toString());

textFieldFatherName.setText(table.getValueAt(row, 3).toString());

textFieldPhone.setText(table.getValueAt(row, 4).toString());

textFieldProssent.setText(table.getValueAt(row, 5).toString());

textFieldDate.setText(table.getValueAt(row, 6).toString());

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

});

try (Connection connection = DriverManager.getConnection(

"jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root", "")) {

PreparedStatement statement = connection.prepareStatement("SELECT \* \r\n" + "FROM `worker`");

ResultSet res = statement.executeQuery();

while (res.next()) {

WorkerT\_obj.addRow(new worker(res.getInt(1), res.getString(2), res.getString(3), res.getString(4),

res.getString(5), res.getFloat(6), res.getString(7)));

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..." + ex);

System.out.println(ex);

}

table.getColumnModel().getColumn(0).setMinWidth(0);

table.getColumnModel().getColumn(0).setMaxWidth(0);

table.getColumnModel().getColumn(0).setWidth(0);

}

}

tablePanelForWorker.java

**package** ud;

**public** **class** user {

**protected** String login;

**protected** String password;

**public** user (String login\_, String password\_) {

login=login\_;

password=password\_;

}

**public** String getLogin() {

**return** login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

user.java

**package** ud;

**public** **class** user {

**protected** String login;

**protected** String password;

**public** user (String login\_, String password\_) {

login=login\_;

password=password\_;

}

**public** String getLogin() {

**return** login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

userTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class userTableModel extends AbstractTableModel{

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<user> UserT\_obj;

public userTableModel(LinkedList<user> UserT\_obj) {

this.UserT\_obj = UserT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 2;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "Логин";

case 1:

return "Пароль";

}

return "";

}

public void addRow(user nr) { // добавление строки в таблицу

UserT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = UserT\_obj.indexOf(nr);

tablePanelForUser.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return UserT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

user p = UserT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getLogin();

case 1:

return p.getPassword();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, user nr) {

UserT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForUser.table.setRowSelectionInterval(index, index);

}

public void deleteRow(String login) {

Iterator<user> isph = UserT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getLogin().equals(login)) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = UserT\_obj.size();

UserT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

visitAdd.java

package ud;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JComboBox;

import javax.swing.JButton;

import javax.swing.JTextPane;

public class visitAdd extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JTextField textFieldSecondNameCustomer;

private JTextField textFieldNameCustomer;

private JTextField textFieldDay;

public String idCustomerSelected;

public String idServiceSelected;

public String idWorkerSelected;

public String daySelected;

public String serviceIdSelected;

public String workerIdSelected;

public String timeSelected ;

public String serviceNameWorkerNameSelected;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

visitAdd frame = new visitAdd();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public static boolean isWorld(String x)

{

Pattern p = Pattern.compile("^[а-яА-ЯёЁ]+$");

Matcher m = p.matcher(x);

return m.matches();

}

public static boolean isDate(String x)

{

Pattern p = Pattern.compile("^20\\d{2}-\\d{2}-\\d{2}$");

Matcher m = p.matcher(x);

return m.matches();

}

/\*\*

\* Create the frame.

\*/

public void checkFio() {

if(textFieldNameCustomer.getText()!=null && textFieldSecondNameCustomer.getText()!=null)

{

String sn=textFieldSecondNameCustomer.getText();

String n=textFieldNameCustomer.getText();

int flagSn = 1;

int flagN = 1;

if (!isWorld(sn)) {

flagSn = 0;

JOptionPane.showMessageDialog(null, "Введите Фамилию!");

}

if (!isWorld(n)) {

flagN = 0;

JOptionPane.showMessageDialog(null, "Введите Имя!");

}

if (flagSn == 1 && flagN == 1) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `id\_Customer` \r\n" +

"FROM `customer`"

+ "where `Second\_Name`='"+sn+"'AND `Name`='"+n+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

idCustomerSelected=rest.getString(1);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

}

}

public int checkDateTime() {

String d=textFieldDay.getText();

String flagForCheckExistenceTimeAndDate=null;

int flag=0;

int flagD=1;if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

if (flagD == 1) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT \* FROM `record` "

+ "WHERE `id\_Service` = '"+idServiceSelected+"' AND `id\_Worker` = '"+idWorkerSelected+"' AND `Date` = '"+textFieldDay.getText()+"' AND `Hour` = '"+timeSelected+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

flagForCheckExistenceTimeAndDate=rest.getString(1);

flag=1;

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex+flagForCheckExistenceTimeAndDate);

System.out.println(ex);

}

}

return flag;

}

public visitAdd() {

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 582, 222);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblDay = new JLabel("\u0414\u0430\u0442\u0430");

lblDay.setBounds(321, 11, 31, 14);

contentPane.add(lblDay);

JLabel lblTime = new JLabel("\u0412\u0440\u0435\u043C\u044F");

lblTime.setBounds(250, 42, 46, 14);

contentPane.add(lblTime);

JLabel lblSecondNameCustomer = new JLabel("\u0424\u0430\u043C\u0438\u043B\u0438\u044F");

lblSecondNameCustomer.setBounds(10, 11, 46, 14);

contentPane.add(lblSecondNameCustomer);

JLabel lblNameCustomer = new JLabel("\u0418\u043C\u044F");

lblNameCustomer.setBounds(171, 11, 31, 14);

contentPane.add(lblNameCustomer);

textFieldSecondNameCustomer = new JTextField();

textFieldSecondNameCustomer.setBounds(65, 8, 86, 20);

contentPane.add(textFieldSecondNameCustomer);

textFieldSecondNameCustomer.setColumns(10);

textFieldNameCustomer = new JTextField();

textFieldNameCustomer.setBounds(210, 8, 86, 20);

contentPane.add(textFieldNameCustomer);

textFieldNameCustomer.setColumns(10);

textFieldDay = new JTextField();

textFieldDay.setBounds(362, 8, 86, 20);

contentPane.add(textFieldDay);

textFieldDay.setColumns(10);

JComboBox<String> comboBoxServiceWorker = new JComboBox<String>();

comboBoxServiceWorker.setBounds(10, 67, 230, 20);

contentPane.add(comboBoxServiceWorker);

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `sevice`.`Name`, `worker`.`Second\_Name`, `worker`.`Name` FROM `sevice` \r\n" +

"inner join `service\_worker` \r\n" +

"on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service`) \r\n" +

"inner join `worker` \r\n" +

"on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)\r\n" +

"\r\n" +

"");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

String concat= rest.getString(1) +" "+rest.getString(2)+" "+ rest.getString(3);

comboBoxServiceWorker.addItem(concat);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

comboBoxServiceWorker.setSelectedItem(null);

JLabel lblServiceWorker = new JLabel("\u0423\u0441\u043B\u0443\u0433\u0430");

lblServiceWorker.setBounds(10, 42, 46, 14);

contentPane.add(lblServiceWorker);

JComboBox<String> comboBoxTime = new JComboBox<String>();

comboBoxTime.setBounds(250, 67, 63, 20);

contentPane.add(comboBoxTime);

comboBoxTime.addItem("8");

comboBoxTime.addItem("10");

comboBoxTime.addItem("12");

comboBoxTime.addItem("14");

comboBoxTime.addItem("16");

comboBoxTime.addItem("18");

comboBoxTime.setSelectedItem(null);

comboBoxTime.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

@SuppressWarnings("unchecked")

JComboBox<String> comboBoxTime = (JComboBox<String>)e.getSource();

timeSelected = (String)comboBoxTime.getSelectedItem();

JOptionPane.showMessageDialog(null, timeSelected);

}

});

comboBoxServiceWorker.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

@SuppressWarnings("unchecked")

JComboBox<String> comboBoxService = (JComboBox<String>)e.getSource();

serviceNameWorkerNameSelected = (String)comboBoxService.getSelectedItem();

JOptionPane.showMessageDialog(null, serviceNameWorkerNameSelected);

String str []=serviceNameWorkerNameSelected.split(" ");

String s=str[0];

String sn=str[1];

String n=str[2];

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

PreparedStatement statement = connection.prepareStatement("SELECT `service\_worker`.`id\_Service`, `service\_worker`.`id\_Worker` FROM `service\_worker` \r\n" +

"inner join `sevice` \r\n" +

"on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service` and `sevice`.`Name`='"+s+"' ) \r\n" +

"inner join `worker` \r\n" +

"on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker` and `worker`.`Name`='"+n+"' and `worker`.`Second\_Name`='"+sn+"' )\r\n" +

"");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

idServiceSelected=rest.getString(1);

idWorkerSelected=rest.getString(2);

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

}

});

JButton btnWriteRecord = new JButton("\u0417\u0430\u043F\u0438\u0441\u0430\u0442\u044C");

btnWriteRecord.setBounds(467, 149, 89, 23);

btnWriteRecord.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try (Connection connection = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/salon?serverTimezone=Europe/Moscow&useSSL=false", "root",""))

{

String d=textFieldDay.getText();

int flagD=1;

int flagC=1;

int flagS=1;

int flagW=1;

String flagForCheckExistenceRecord=null;

if (!isDate(d)) {

flagD = 0;

JOptionPane.showMessageDialog(null, "Введите Телефон!");

}

checkFio();

if(idCustomerSelected==null)

{

flagC=0;

JOptionPane.showMessageDialog(null, "Такого клиента не существует.");

}

if(idServiceSelected==null)

{

flagS=0;

JOptionPane.showMessageDialog(null, "Выберете услугу!");

}

if(idWorkerSelected==null)

{

flagW=0;

JOptionPane.showMessageDialog(null, "Выберете услFFFугу!");

}

int t=checkDateTime();

if (flagD==1 && flagC==1 && flagS==1 &&flagW==1 && t==0) {

PreparedStatement statement = connection.prepareStatement("SELECT `Status` FROM `record` where "

+ "`id\_Customer`='"+idCustomerSelected+"' "

+ "and `id\_Service`='"+idServiceSelected+"' and `id\_Worker`= '"+idWorkerSelected+"'and `Date` = '"+d+"'and `Hour`='"+timeSelected+"'");

ResultSet rest = statement.executeQuery();

while(rest.next()) {

flagForCheckExistenceRecord=rest.getString(1);

}

if(flagForCheckExistenceRecord==null) {

insert it = new insert();

it.insertRow("INSERT INTO `salon`.`record` \r\n" +

"(`id\_Customer` , `id\_Service` , `id\_Worker` , `Date` , `Hour` , `Status`)\r\n" +

"VALUES (\r\n" +

"'"+idCustomerSelected+"', '"+idServiceSelected+"', '"+idWorkerSelected+"', '"+d+"', '"+timeSelected+"', '0')");

String str []=serviceNameWorkerNameSelected.split(" ");

String serviceTemp=str[0];

String workerSNameTemp=str[1];

String workerNameTemp=str[2];

JTextPane textPane = new JTextPane();

textPane.setBounds(10, 108, 447, 80);

contentPane.add(textPane);

textPane.setText("Клиент:"+textFieldSecondNameCustomer.getText()+" "+textFieldNameCustomer.getText()

+" \nУслуга: "+serviceTemp+" \n"+"Сотрудник: "+workerSNameTemp+" "

+workerNameTemp+"\n Дата:"+textFieldDay.getText()

+"\nВремя:"+timeSelected);

}

else if(flagForCheckExistenceRecord!=null) {

JOptionPane.showMessageDialog(null, "Это время занято! Выберете другое.");

}

}

else {

JOptionPane.showMessageDialog(null, "Это время занято! Выберете другое345434r3t.");

}

} catch (Exception ex) {

JOptionPane.showMessageDialog(null, "Connection failed..."+ex);

System.out.println(ex);

}

dispose();

}

});

contentPane.add(btnWriteRecord);

}

}

worker java

**package** ud;

**public** **class** worker **implements** Comparable<worker> {

**protected** **int** id;

**protected** String secondName;

**protected** String name;

**protected** String fatherName;

**protected** String phone;

**protected** **float** prossent;

**protected** String date;

**public** worker(**int** id\_, String secondName\_, String name\_, String fatherName\_, String phone\_, **float** prossent\_,

String date\_) {

id = id\_;

secondName = secondName\_;

name = name\_;

fatherName = fatherName\_;

phone = phone\_;

prossent = prossent\_;

date=date\_;

}

**public** **int** getID() {

**return** id;

}

**public** String getSecondName() {

**return** secondName;

}

**public** String getName() {

**return** name;

}

**public** String getFatherName() {

**return** fatherName;

}

**public** String getPhone() {

**return** phone;

}

**public** **float** getProssent() {

**return** prossent;

}

**public** String getDate() {

**return** date;

}

**public** **void** setID(**int** id) {

**this**.id = id;

}

**public** **void** setSecondName(String secondName) {

**this**.secondName = secondName;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setFatherName(String fatherName) {

**this**.fatherName = fatherName;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** **void** setProssent(**float** prossent) {

**this**.prossent = prossent;

}

**public** **void** setDate(String date) {

**this**.date = date;

}

@Override

**public** **int** compareTo(worker c) {

**return** **this**.secondName.compareTo(c.secondName);

}

@Override

**public** String toString() {

**return** secondName +" "+name;

}

}

WorkerTableModel.java

package ud;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.table.AbstractTableModel;

public class WorkerTableModel extends AbstractTableModel {

private static final long serialVersionUID = 7927259757559420606L;

private LinkedList<worker> WorkerT\_obj;

public WorkerTableModel(LinkedList<worker> WorkerT\_obj) {

this.WorkerT\_obj = WorkerT\_obj;

}

public Class<?> getColumnClass(int columnIndex) {

return getValueAt(0, columnIndex).getClass();

}

public int getColumnCount() {

return 7;

}

public String getColumnName(int columnIndex) {

switch (columnIndex) {

case 0:

return "ID";

case 1:

return "Фамилия";

case 2:

return "Имя";

case 3:

return "Отчество";

case 4:

return "Телефон";

case 5:

return "Процент";

case 6:

return "ДтПрнт";

}

return "";

}

public void addRow(worker nr) { // добавление строки в таблицу

WorkerT\_obj.add(nr);

fireTableDataChanged();// вызываем для обновления таблицы

// выделяем добавленную строку - иначе после обновления таблицы теряет выделение

int index = WorkerT\_obj.indexOf(nr);

tablePanelForWorker.table.setRowSelectionInterval(index, index);

}

public int getRowCount() {

return WorkerT\_obj.size();

}

public Object getValueAt(int rowIndex, int columnIndex) {

worker p = WorkerT\_obj.get(rowIndex);

switch (columnIndex) {

case 0:

return p.getID();

case 1:

return p.getSecondName();

case 2:

return p.getName();

case 3:

return p.getFatherName();

case 4:

return p.getPhone();

case 5:

return p.getProssent();

case 6:

return p.getDate();

}

return "";

}

public boolean isCellEditable(int rowIndex, int columnIndex) {

return false;

}

public void setValueAt(Object value, int rowIndex, int columnIndex) {

}

public void updateRow(int index, worker nr) {

WorkerT\_obj.set(index, nr);

fireTableDataChanged();

tablePanelForWorker.table.setRowSelectionInterval(index, index);

}

public void deleteRow(int id) {

Iterator<worker> isph = WorkerT\_obj.iterator();

boolean flag = false;

while (isph.hasNext()) {

if (isph.next().getID()==id) {

isph.remove();

flag = true;

break;

}

}

if (flag) {

fireTableDataChanged();

} else

;

}

public void deleteAll() { // удаление строки из таблицы

int size = WorkerT\_obj.size();

WorkerT\_obj.clear();

fireTableRowsDeleted(0, size);

}

}

# **ПРИЛОЖЕНИЕ Б**

**Запросы к базе данных**

Примеры запросов к базе данных:

* запрос на вывод фамилии и имени сотрудников;

SELECT `second\_name` , `name`

FROM `worker`

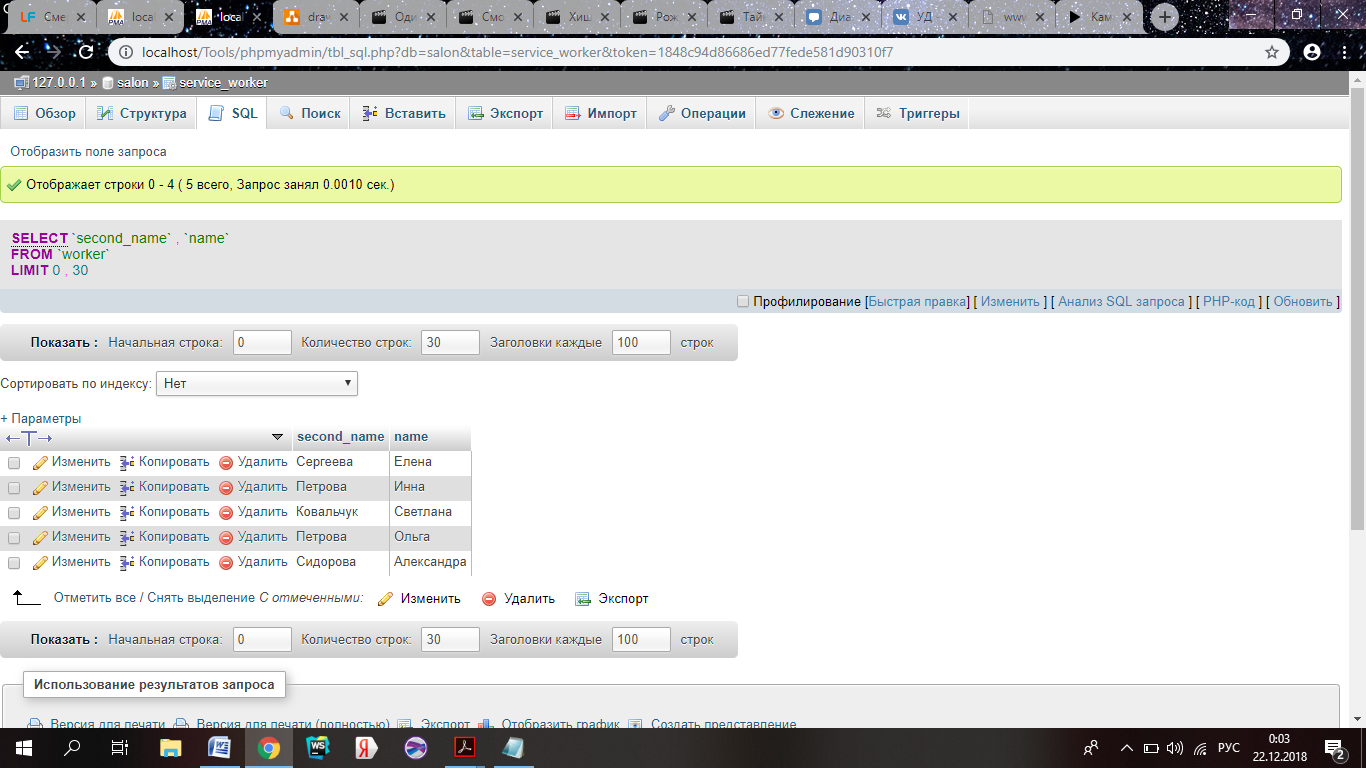


Рисунок Б.1 –Запрос на вывод фамилии и имени сотрудников

* запрос на выборку всех сотрудников, принятых на работу с 2018 года;

SELECT \* FROM `worker`

WHERE `Start\_Day` > '2018-01-01'

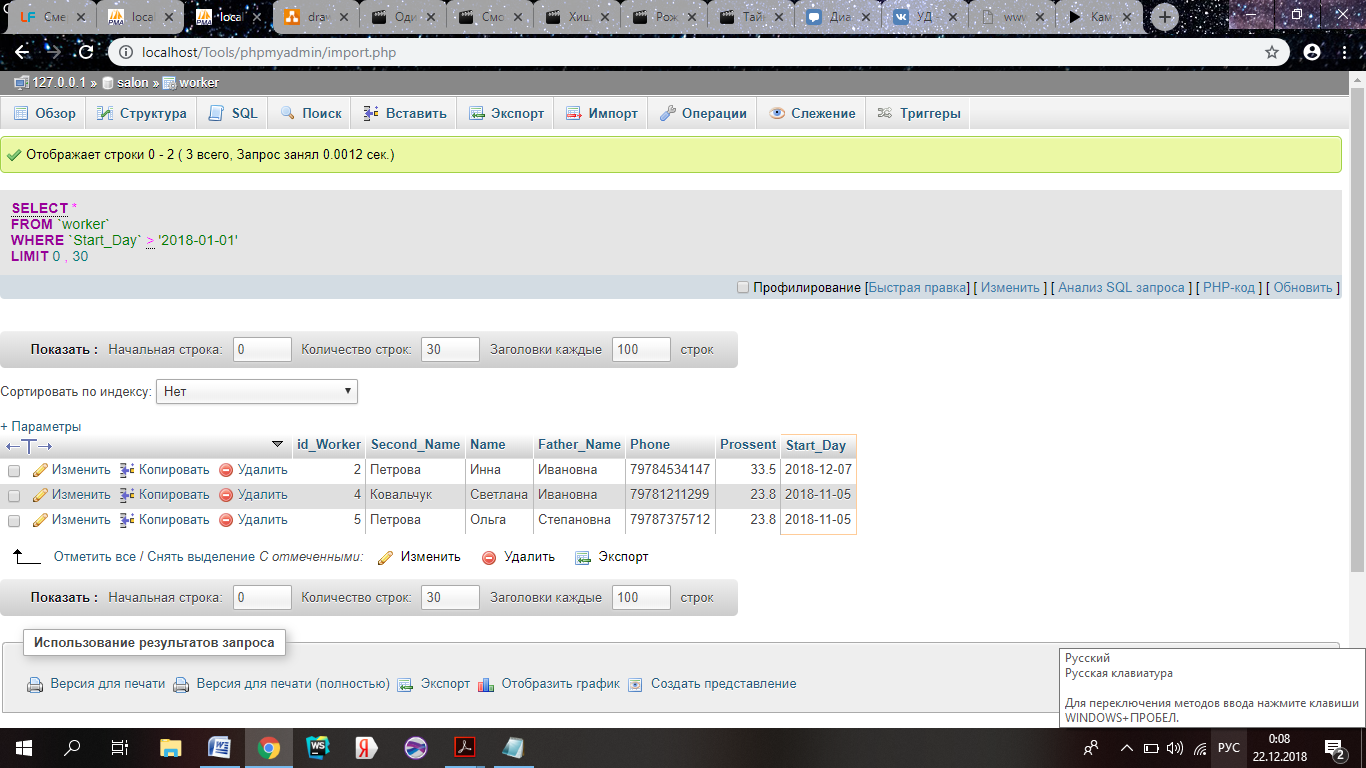


Рисунок Б.2 – Запрос на выборку всех сотрудников, принятых на работу после 2018-01-01

* запрос на выборку услуг стоимостью больше 1000 рублей и длительностью больше 90 минут;

SELECT \* FROM `sevice` WHERE `Cost` >1000 AND `Time` >90

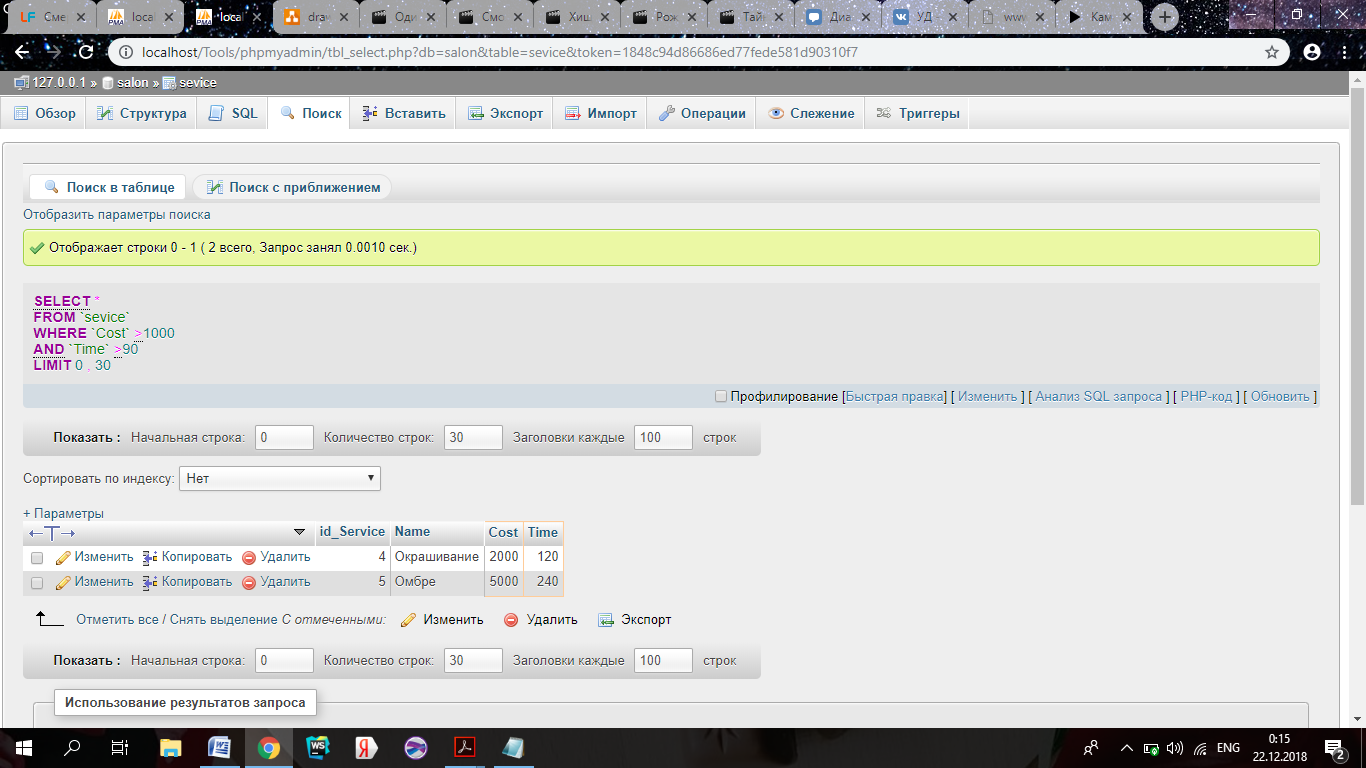


Рисунок Б.3 – Запрос на выборку услуг стоимостью больше 1000 рублей и длительностью больше 90 минут

* запрос на выборку сотрудников, осуществляющих услугу Маникюр;

SELECT `worker`.`Second\_Name`, `worker`.`Name` FROM `worker`

join `service\_worker` on

(`worker`.`id\_worker`=`service\_worker`.`id\_worker`)

join `sevice` on

(`service\_worker`.`id\_service`=`sevice`.`id\_service` and `sevice`.`Name`='Маникюр')

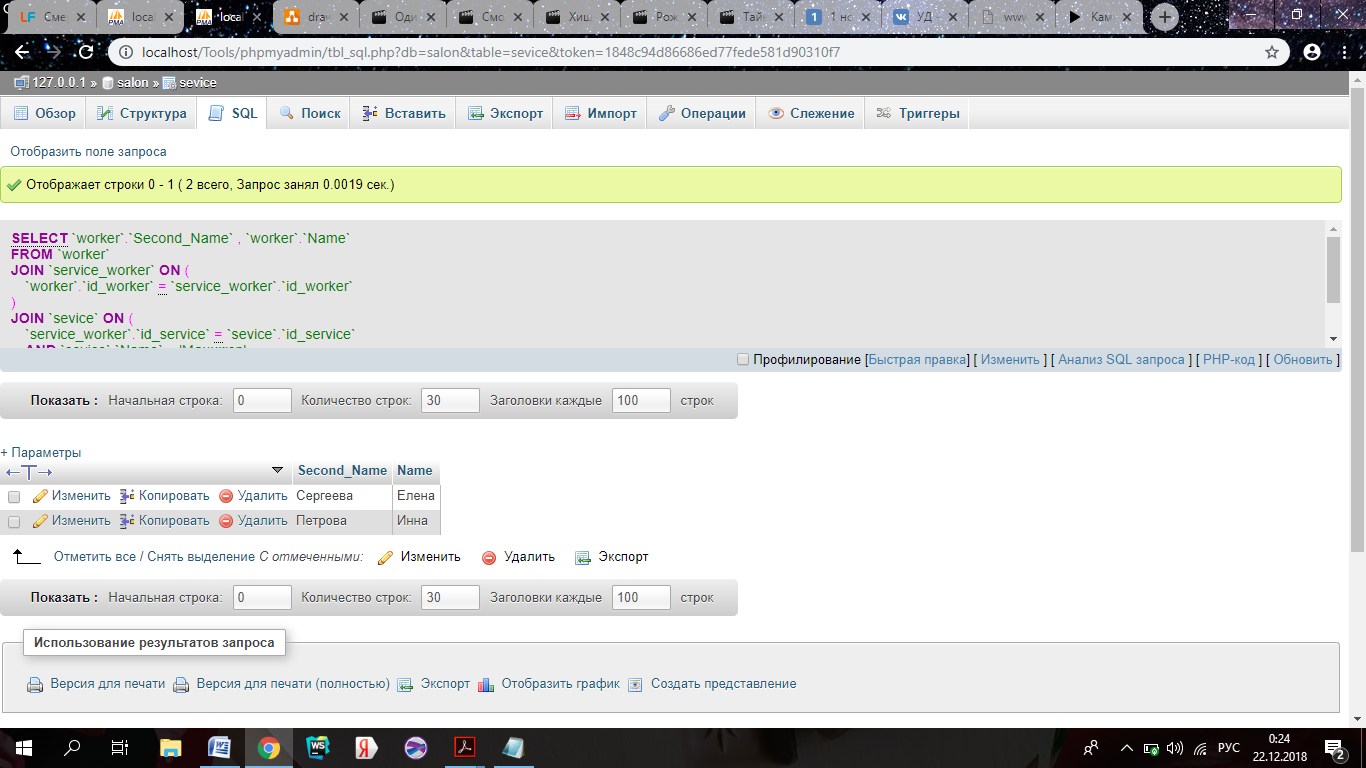


Рисунок Б.4 – запрос на выборку сотрудников, осуществляющих услугу Маникюр

* запрос на выборку запланированных посещений на 24 декабря 2018 года;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`,`worker`.`Father\_Name`,`worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`

on(`customer`.`id\_Customer`=`record`.`id\_Customer`and`record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker`and`record`.`id\_Service` = `service\_worker`.`id\_Service`)

inner join `worker`

on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

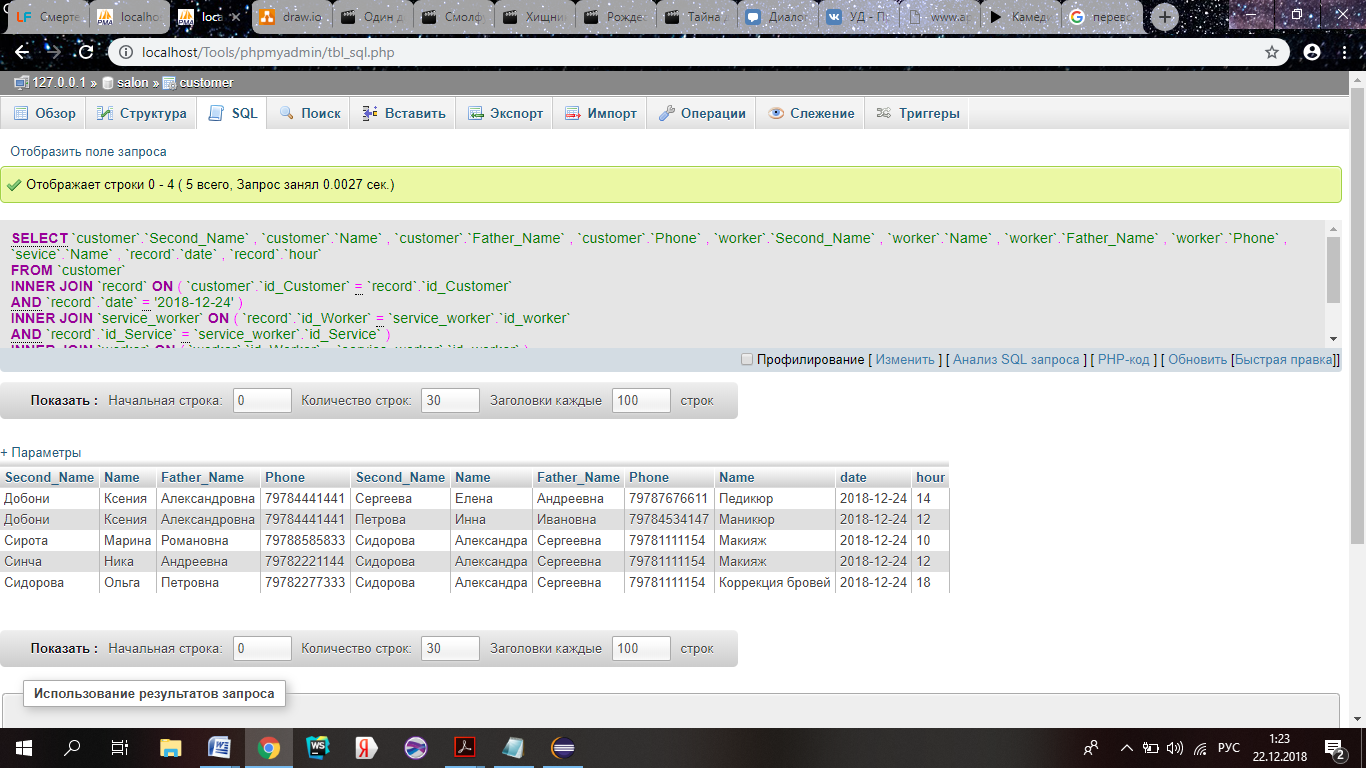


Рисунок Б.5 – Запрос на выборку запланированных посещений на 24 декабря 2018 года

* запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`,

`worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`,

`sevice`.`Name`,

`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`

on(`customer`.`id\_Customer`=`record`.`id\_Customer`and`record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker`and record`.`id\_Service` = `service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`and `worker`.`second\_name` = 'Сидорова')

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

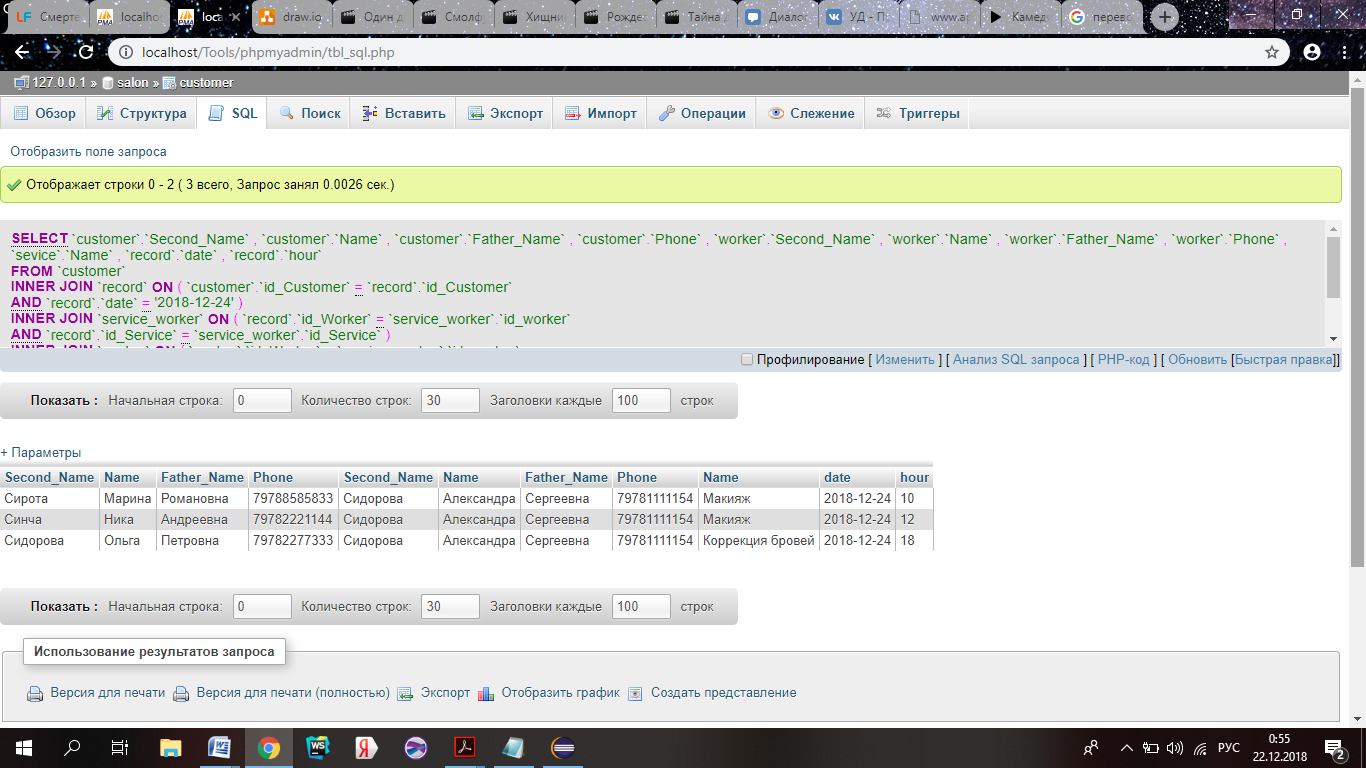


Рисунок Б.6 – Запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой

* запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой на услугу Макияж

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer` inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer` and `record`.`date`='2018-12-24')

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`

and `worker`.`second\_name`='Сидорова')

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`

and `sevice`.`name`='макияж')

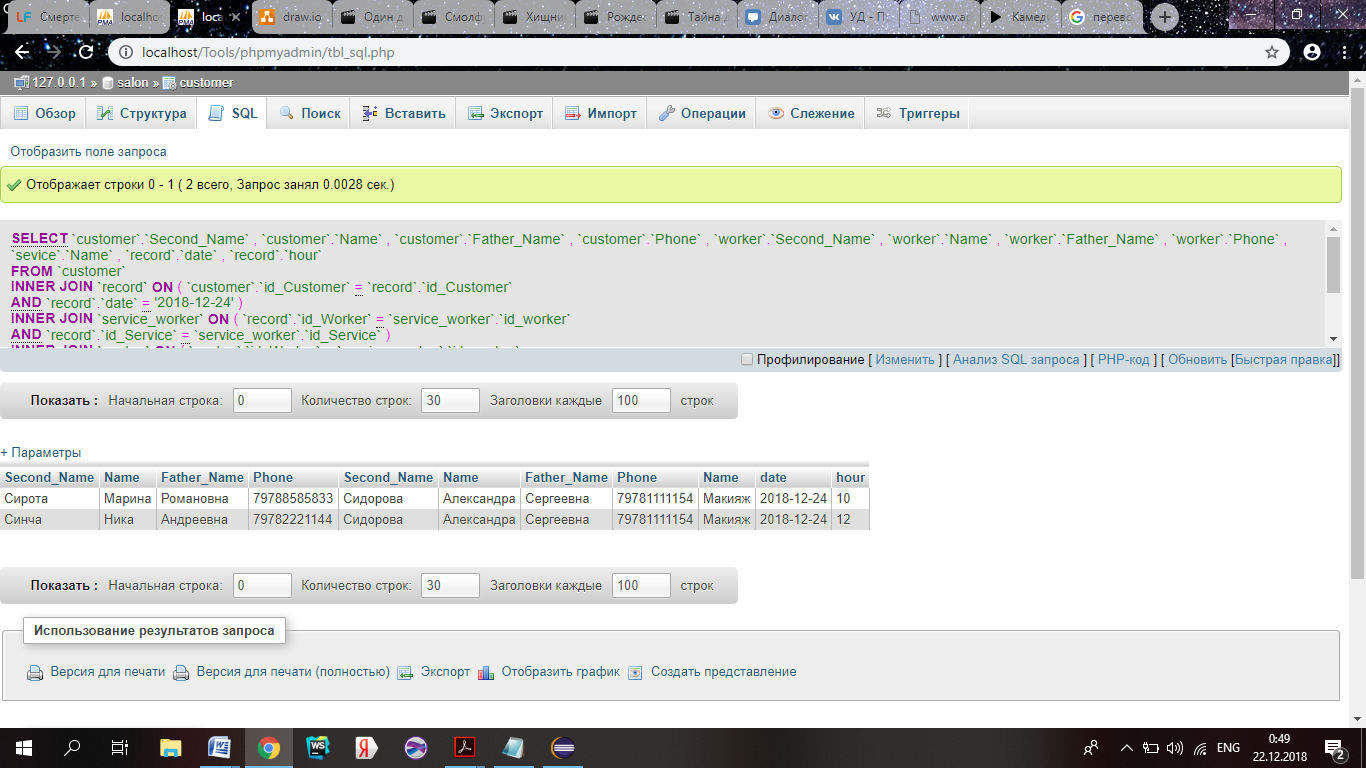


Рисунок Б.7 – Запрос на выборку запланированных посещений на 24 декабря 2018 года у сотрудника Сидоровой на услугу Макияж

* запрос на выбор всех данных обо всех посещениях;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `worker`.`Second\_Name`, `worker`.`Name`, `worker`.`Father\_Name`, `worker`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer`

inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer`)

inner join `service\_worker`

on(`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker`

on(`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice`

on(`sevice`.`id\_Service`=`service\_worker`.`id\_Service`)

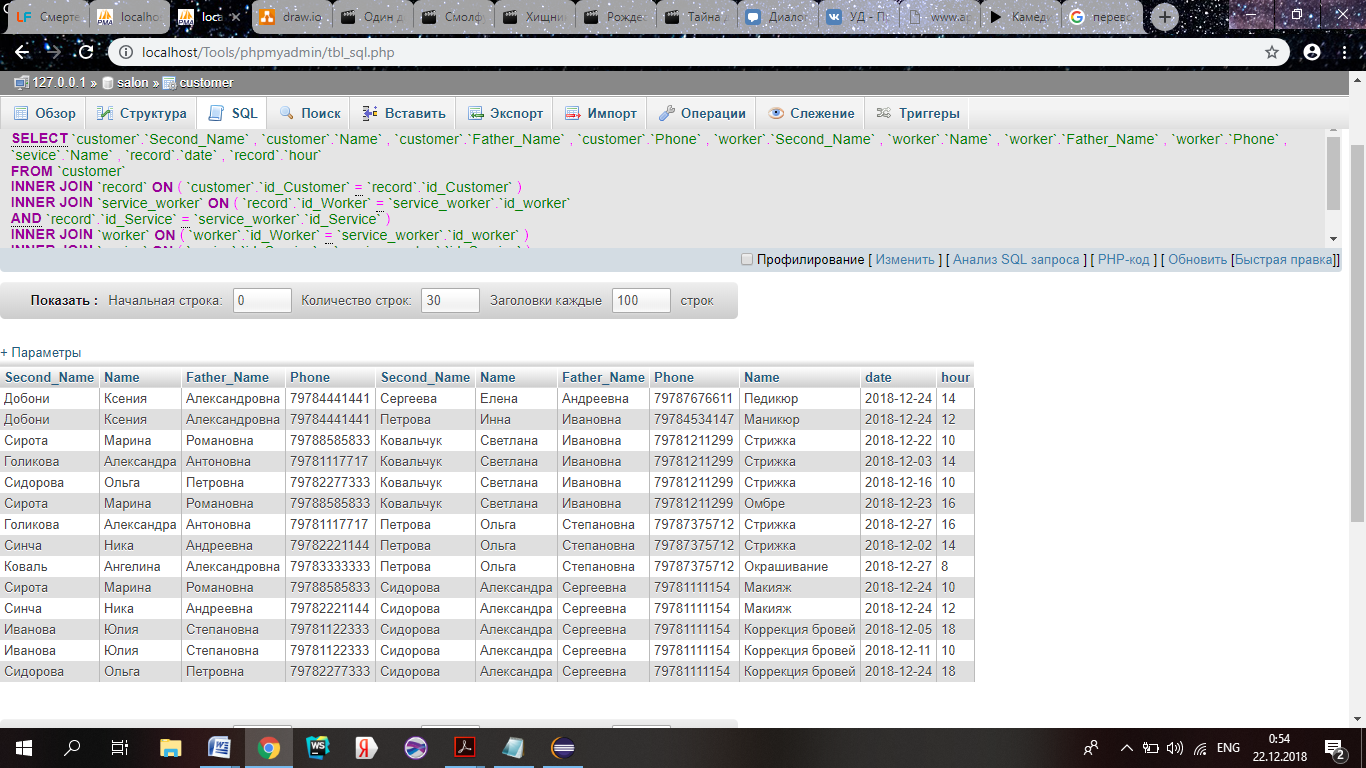


Рисунок Б.8 – Запрос на выбор всех данных обо всех посещениях

* запрос на выбор информации о посещениях услуги Стрижка клиентом Сирота;

SELECT `customer`.`Second\_Name`, `customer`.`Name`, `customer`.`Father\_Name`, `customer`.`Phone`, `sevice`.`Name`,`record`.`date`,`record`.`hour`

FROM `customer` inner join `record`on (`customer`.`id\_Customer`=`record`.`id\_Customer` and `customer`.`Second\_name`='Сирота')

inner join `service\_worker`on (`record`.`id\_Worker`=`service\_worker`.`id\_worker` and`record`.`id\_Service`=`service\_worker`.`id\_Service`)

inner join `worker` on (`worker`.`id\_Worker`=`service\_worker`.`id\_worker`)

inner join `sevice` on (`sevice`.`id\_Service`=`service\_worker`.`id\_Service` and `sevice`.`name`='Стрижка')

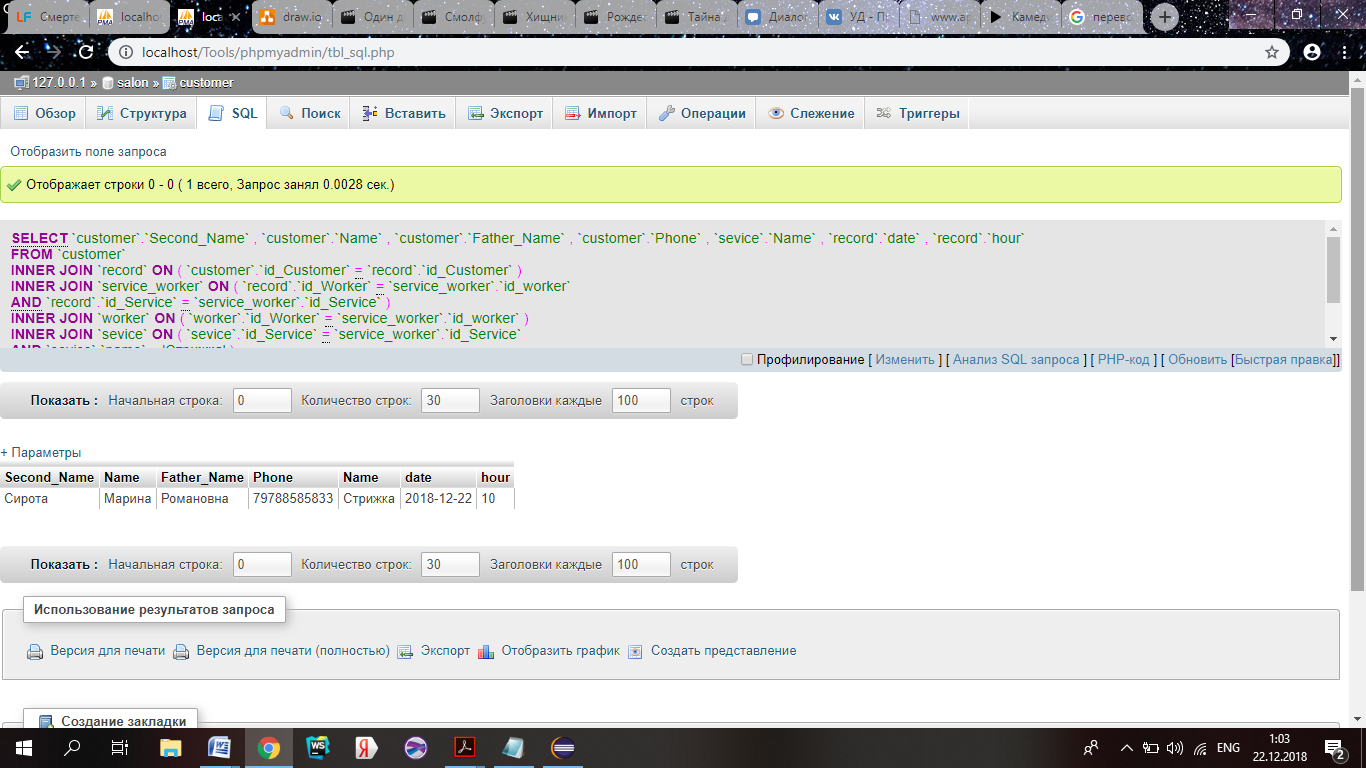


Рисунок Б.8 – Запрос на выбор информации о посещениях услуги Стрижка клиентом Сирота