Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

Допускаю к защите

Руководитель

|  |  |  |
| --- | --- | --- |
|  |  | В.Л.Аршинский |
| подпись |  | И.О. Фамилия |

|  |
| --- |
| **Разработка приложения с использованием объектно-ориентированного подхода** |

наименование темы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

|  |
| --- |
| Объектно-ориентированное программирование |
| 1.011.00.00 - ПЗ  **Ю** |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИСТБ-19-2 |  |  |  | Ю.В. Комогорцева |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | В.Л. Аршинский |
|  |  |  |  | подпись |  | И.О. Фамилия |

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2021 г.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| По курсу | Объектно-ориентированное программирование | | | |
| Студенту | Комогорцевой Ю. В. | | | |
|  | (фамилия, инициалы) | | | |
| Тема работы: | | Разработка приложения с использованием | | |
| объектно-ориентированного подхода | | | | |
| Исходные данные | | | Разработать Desktop-приложение для салонов красоты | |
| предназначенное для автоматизации стратегий взаимодействия с клиентами, | | | | |
| путём сохранения информации о клиентах и истории взаимоотношений с ним | | | | |
| Рекомендуемая литература: | | | |  |
| 1. Васильев А.Н. Java . Объектно-ориентированное программирование для | | | | |
| магистров и бакалавров. Базовый курс по объектно-ориентированному | | | | |
| программированию / А.Н. Васильев. – СПб.: Питер,2012. – 395 с. | | | | |
| 2. Рамбо Джеймс. UML: Руководство пользователя: спец. справ.: пер. с англ. | | | | |
| / Джеймс Рамбо, Айвар Якобсон, Грэди Буч . – СПб.: Питер, 2002. – 652 с. | | | | |
| 3. Грэди Буч. Объектно-ориентированный анализ и проектирование. Второе | | | | |
| издание, перевод с английского под редакцией И. Романовского и Ф. | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Графическая часть на |  | | | листах. | | | | | | | | |
|  |  |  |  |  | | | |  | | | | |
| Дата выдачи задания | " | 01 | " | марта | | | | 2021 г. | | | |  |
|  | | | | |  | | | | |  |  | |
| Задание получил | | | | |  | | | | |  | Ю.В. Комогорцева | |
|  | | | | | подпись | | | | |  | И.О. Фамилия | |
| Дата представления проекта руководителю | | | | | | " | 29 | | " | | мая 2021 г. | |
|  | | | | |  | | | | |  |  | |
| Руководитель курсового проекта | | | | |  | | | | |  | В.Л. Аршинский | |
|  | | | | | подпись | | | | |  | И.О. Фамилия | |

**Содержание**

[Введение 4](#_Toc74307057)

[1 Анализ задания и описание предметной области 5](#_Toc74307058)

[1.1 Постановка задач 5](#_Toc74307059)

[1.2 Функциональные требования 5](#_Toc74307060)

[1.3 Выбор средств реализации 7](#_Toc74307061)

[2 Проектирование приложения 8](#_Toc74307062)

[2.1 Проектирование интерфейса 8](#_Toc74307063)

[2.2 Проектирование классов 15](#_Toc74307064)

[2.3 Проектирование базы данных 22](#_Toc74307065)

[3 Реализация приложения 27](#_Toc74307066)

[3.1 Спецификация реализованных классов 27](#_Toc74307067)

[4 Тестирование 29](#_Toc74307068)

[4.1 Методика тестирования 29](#_Toc74307069)

[4.2 Разработанные тесты 29](#_Toc74307070)

[4.3 Разработанные сценарии использования 34](#_Toc74307071)

[4.4 Результаты тестирования 41](#_Toc74307072)

[Заключение 60](#_Toc74307073)

[Список использованных источников 61](#_Toc74307074)

# Введение

Целью курсового проекта является закрепление теоретических знаний полученных в процессе изучения курса «Объектно-ориентированное программирование» и развитие практических навыков применения объектного подхода в ходе решения конкретной практической задачи – разработки приложения с использованием объектно-ориентированного подхода, а также развитие способностей к обоснованному принятию самостоятельных решений в ходе проектной деятельности и получение базового опыта разработки программного обеспечения.

В рамках курсового проекта решается задача создания Desktop-приложения «EasyBeauty» — это информационная система для автоматизированного взаимодействия с клиентами и с возможностью хранить и редактировать данные об услугах, клиентах, сотрудниках, выполненных работах и записях. Вся информация хранится в локальной базе данных.

1. Анализ задания и описание предметной области

## Постановка задач

В ходе разработки необходимо решить следующие задачи:

1. Выработать функциональные требования к приложению
2. Выполнить проектирование интерфейса
3. Осуществить проектирование классов
4. Произвести проектирование базы данных
5. Разработать приложение, основываясь на результатах проделанной работы вышеперечисленных пунктов
6. Определить методику тестирования и проверить работу приложения
7. Резюмировать результаты курсового проекта
   1. Функциональные требования

Проведя анализ предметной области, было выделено следующее.

В работе салона красоты учувствуют сотрудники и клиенты. Важным звеном является администратор(далее - Пользователь) салона красоты – это человек, который регулирует взаимодействие клиентов и сотрудников. В обязанности администратора входит: запись клиентов на выбранные услуги и осуществление оплаты услуги. Администратору необходимо знать данные о сотруднике и клиенте, а также об услугах. Часто салоны красоты делают бонусную систему для привлечения клиентов. Также необходимо реализовать логирование всех действий администратора в программе, чтобы была возможность отследить неправомерные действия сотрудника и его работу. На основе данных о выполненных работах, можно реализовать просмотр отчетов о деятельности салона красоты.

Приложение должно выполнять следующие функции:

1. Логирование всех действий пользователя
2. Просмотр отчетов о работе салона красоты
3. Изменение процентов начисления и списания бонусной системы
4. Работа с клиентами
   1. Вывод информации о клиентах в таблицу
      1. ФИО
      2. Номер телефона
      3. Дата рождения
      4. Количество бонусов
   2. Добавление информации о клиенте
   3. Редактирование информации о клиенте
   4. Удаление информации о клиенте
   5. Уведомление пользователя о дне рождения клиента
5. Работа с услугами
   1. Вывод информации об услугах таблицу
      1. Название услуги
      2. Стоимость
   2. Добавление информации об услуге
   3. Редактирование информации об услуге
   4. Удаление информации об услуге
6. Работа с сотрудниками
   1. Вывод информации о сотрудниках в таблицу
      1. ФИО
      2. Номер телефона
      3. Дата рождения
      4. Количество бонусов
   2. Добавление информации о сотруднике
   3. Редактирование информации о сотруднике
   4. Удаление информации о сотруднике
   5. Уведомление пользователя о дне рождения сотрудника
7. Работа с информацией о записях
   1. Вывод информации о записях в таблицу
      1. Данные о клиенте
      2. Данные о выбранной услуге
      3. Данные о сотруднике
      4. Дата и время записи
   2. Добавление новой записи
   3. Редактирование существующей записи
   4. Удаление информации о записи
   5. Автоматическое удаление прошедших по времени записей
   6. Возможность просмотра записей на день
   7. Уведомление пользователя о предстоящей записи
8. Работа с информацией о выполненных работах
   1. Вывод информации о выполненных работах в таблицу
      1. Данные о клиенте
      2. Данные о сотруднике
      3. Данные о выбранной услуге
      4. Сумма дохода и использованные баллы
      5. Дата и время выполнения
   2. Добавление новой выполненной работы
   3. Выполнение возврата средств
   4. Выбор средств реализации

Были использованы следующие технологии:

* + - 1. IntelliJ IDEA – IDE для разработки приложения
      2. Драйвер JDBC для SQLite — JDBC (Java Data Base Connectivity)— это API для работы с базой данных на Java для SQLite
      3. Библиотека Swing – относится к библиотеке классов JFC (Java Foundation Classes), которая представляет собой набор библиотек для разработки графических оболочек.

# Проектирование приложения

* 1. Проектирование интерфейса

На основе анализа функциональных требований были выделены следующие окна приложения:

1. «Главное окно»
2. «Записи на день»
3. «Все записи»
4. «Выполненные работы»
5. «Услуги»
6. «Клиенты»
7. «Сотрудники»
8. «Новая услуга»
9. «Редактирование услуги»
10. «Новый клиент»
11. «Редактирование клиента»
12. «Новый сотрудник»
13. «Редактирование сотрудника»
14. «Новая выполненная работа»
15. «Новая запись»
16. «Редактировать запись»
17. «Отчеты»
18. «Логирование»
19. «Бонусная программа»

Все окна можно разделить на 3 группы:

Окна № 1-7 – это основные окна, содержащие важную информацию.

Окна № 8-16 – это вспомогательные окна для добавления и редактирования информации из основных окон.

Окна № 17-19 – это окна для управления и анализа работы информационной системы.

Для удобства реализации были созданы шаблоны для некоторых окон. Ниже представлены шаблоны с описанием.

Шаблон главного окна (см. рис. 1). Содержит панели с уведомлениями и кнопки (выделены голубым цветом). С помощью кнопок в центральной части пользователь открывает другие окна с таблицами и добавлениями записей указанной информации. В верхнем правом углу находятся кнопки, которые позволяют просмотреть логирование и отчеты о работе салона красоты, а также открыть окно «Бонусная система».

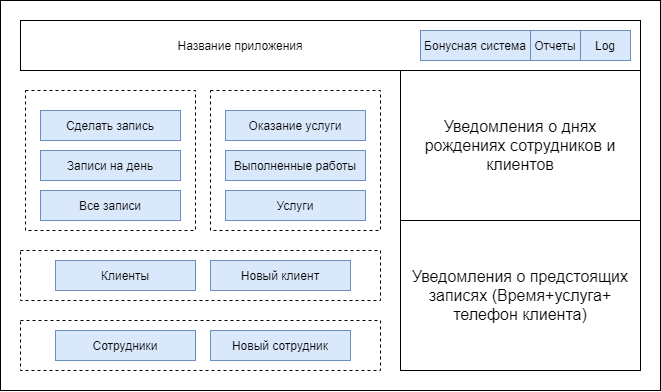


Рисунок 1 – Шаблон главного окна

Шаблон, содержащий таблицу (см. рис.2), подходит сразу для нескольких окон: «Клиенты», «Записи на день», «Все записи», «Выполненные работы», «Услуги», «Клиенты», «Сотрудники». В верхней части окна расположен примерный набор кнопок позволяющий работать с данными в таблице. Набор кнопок может меняться в зависимости от данных в таблице.

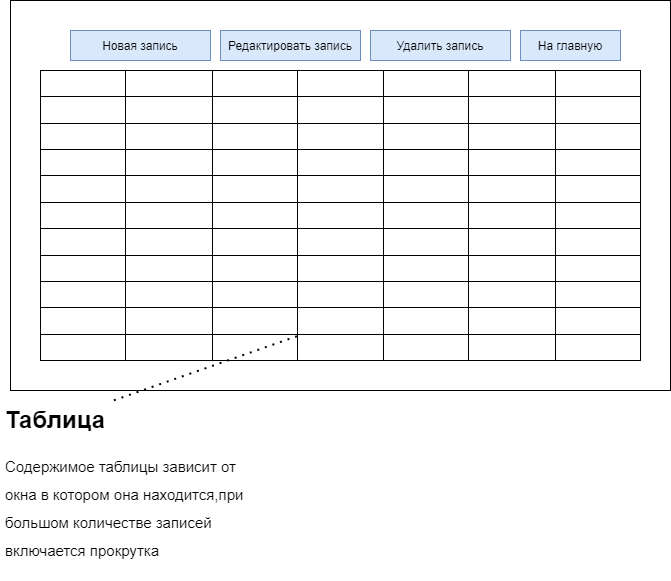


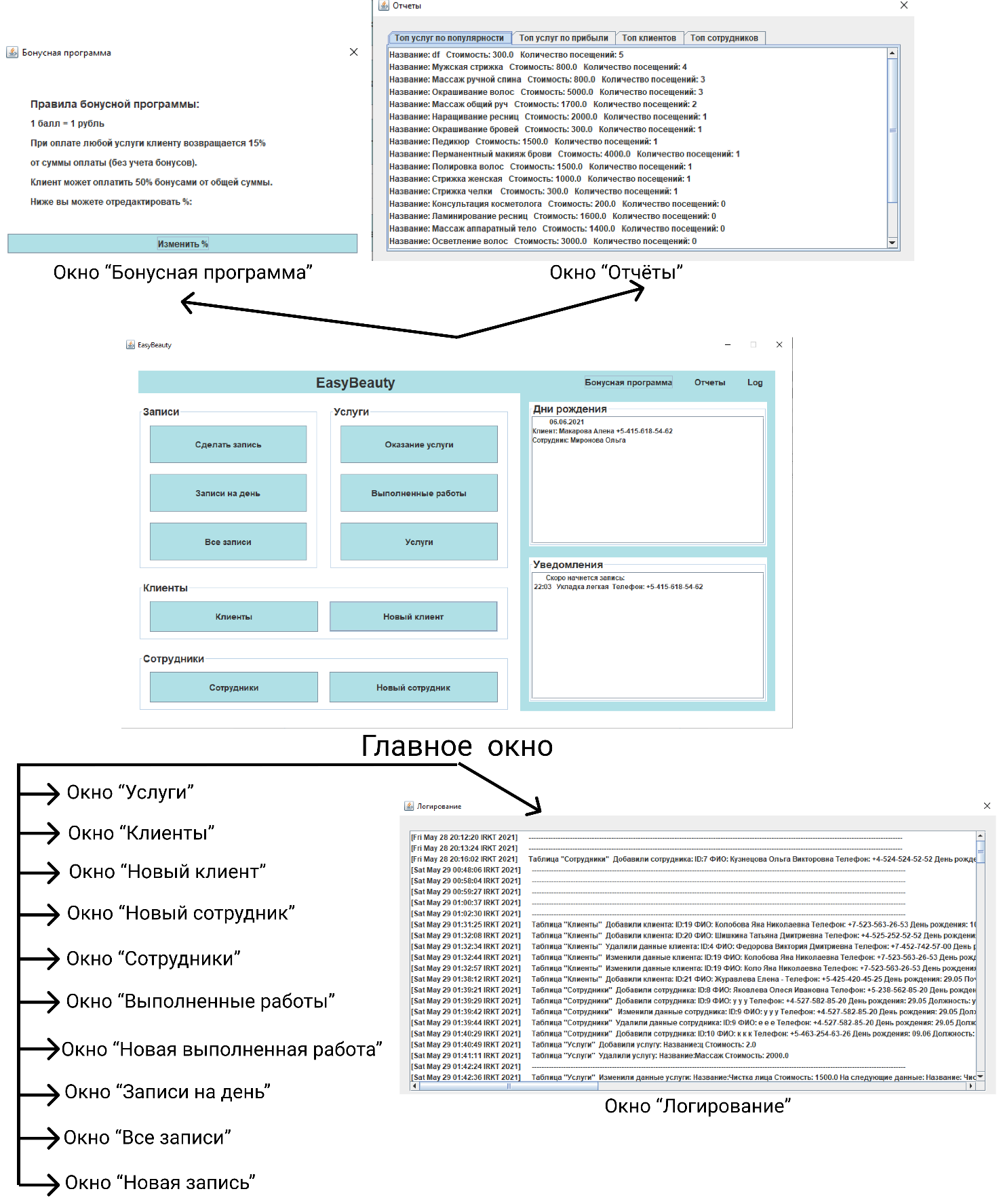
Рисунок 2 – Шаблон для окон, содержащих таблицу

Шаблон для окон добавления и редактирования записей (см. рис.3), подходит сразу для нескольких окон: «Новый клиент», «Редактирование клиента» , «Новый сотрудник», «Редактирование сотрудника», «Новая услуга», «Редактирование услуги», «Новая запись», «Редактирование записи», «Новая выполненная работа» .При редактирование записи ,данные выбранной записи автоматически заполняются в окно редактирования. Набор полей для ввода может меняться в зависимости от данных.



Рисунок 3 – Шаблон для окон добавления и редактирования

Для правильной реализации интерфейса была определена схема переходов, (см. рис. 4, рис. 5, рис. 6, рис. 7). В связи с большим количеством окон и для удобства просмотра схема была разбита на несколько рисунков, и по необходимости картинка заменялась названием окна.

Рисунок 4 – Схема переходов с «Главного окна» в другие окна приложения

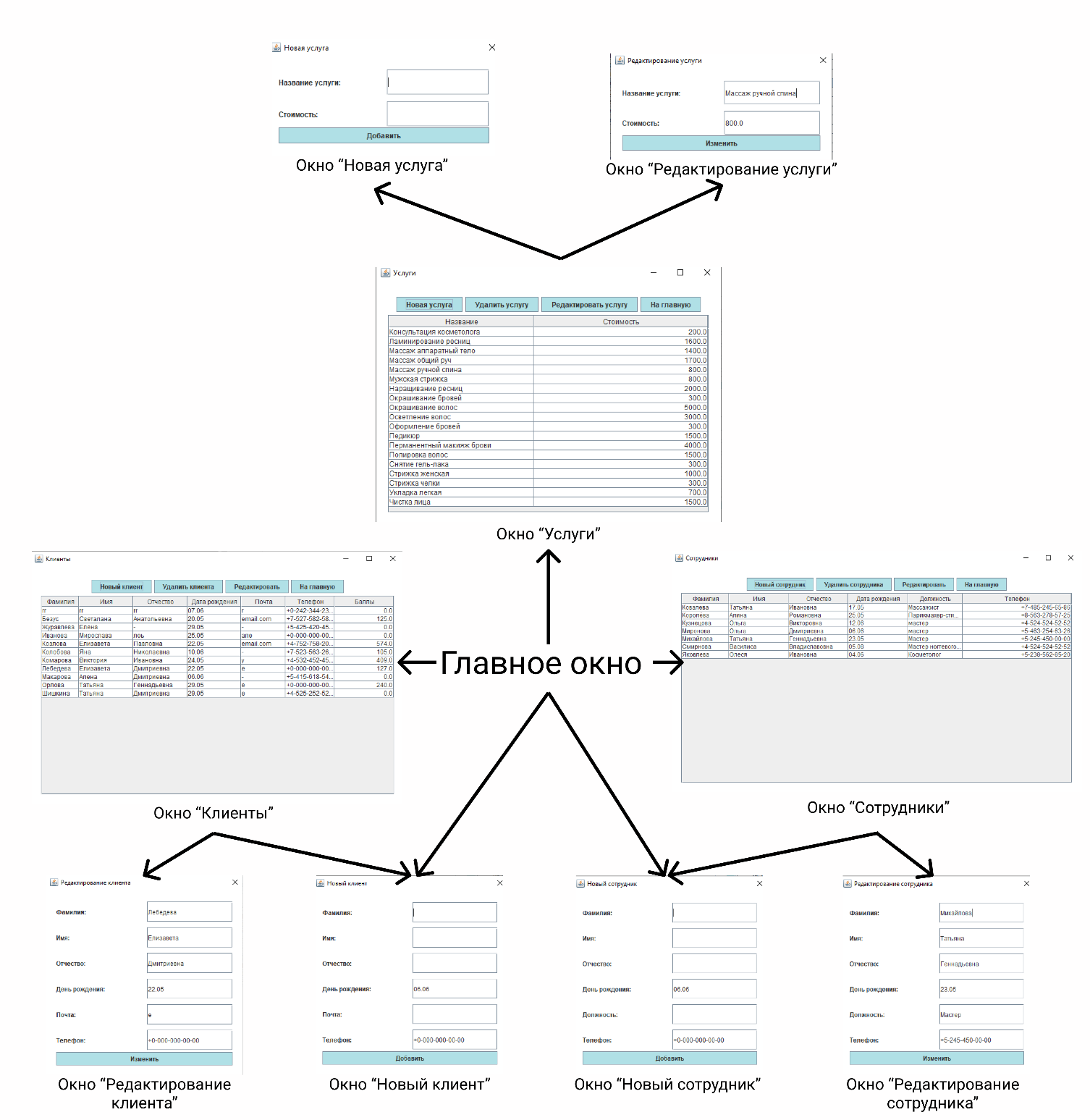


Рисунок 5 – Схема переходов между окнами приложения

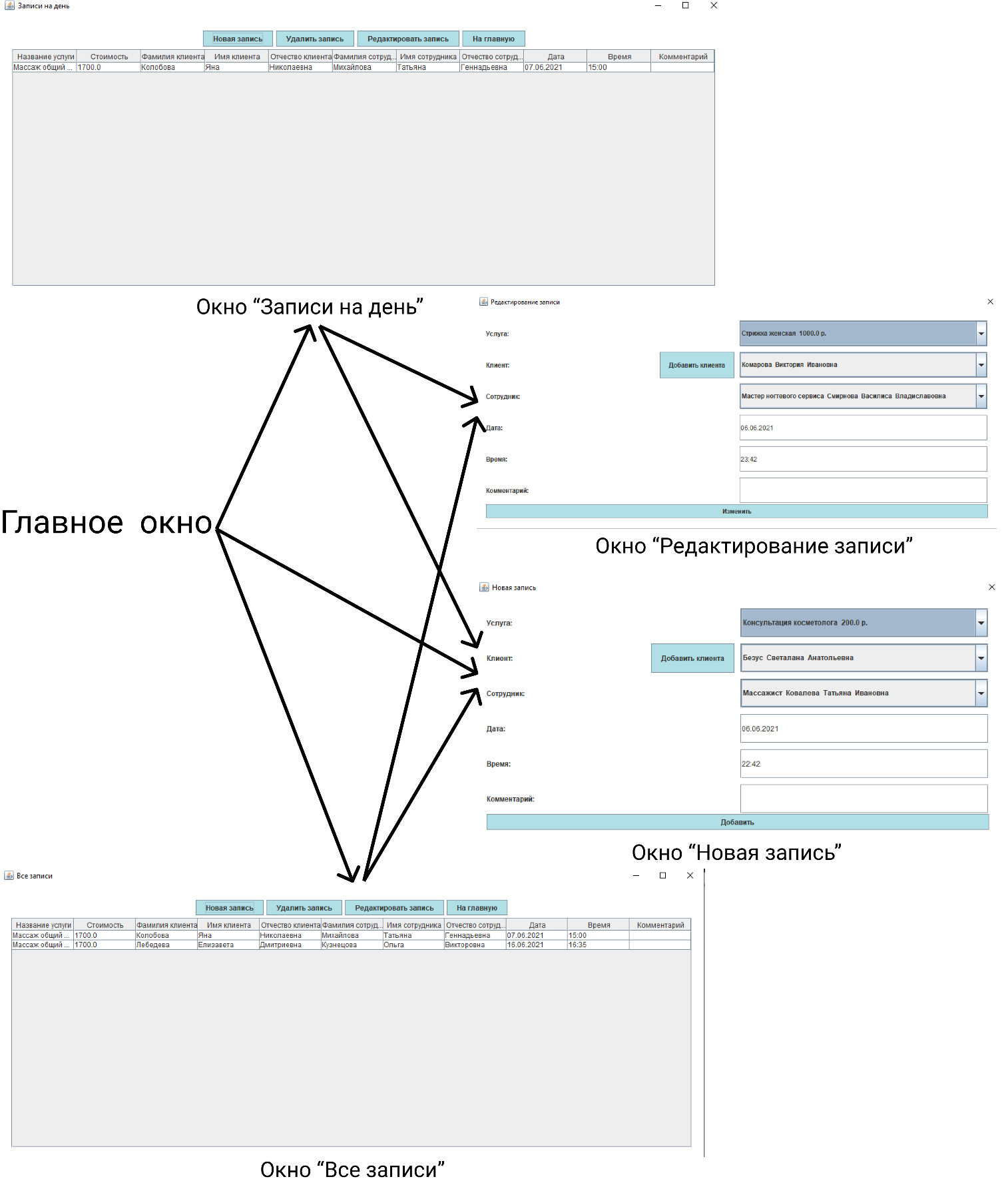


Рисунок 6 – Схема переходов между окнами приложения

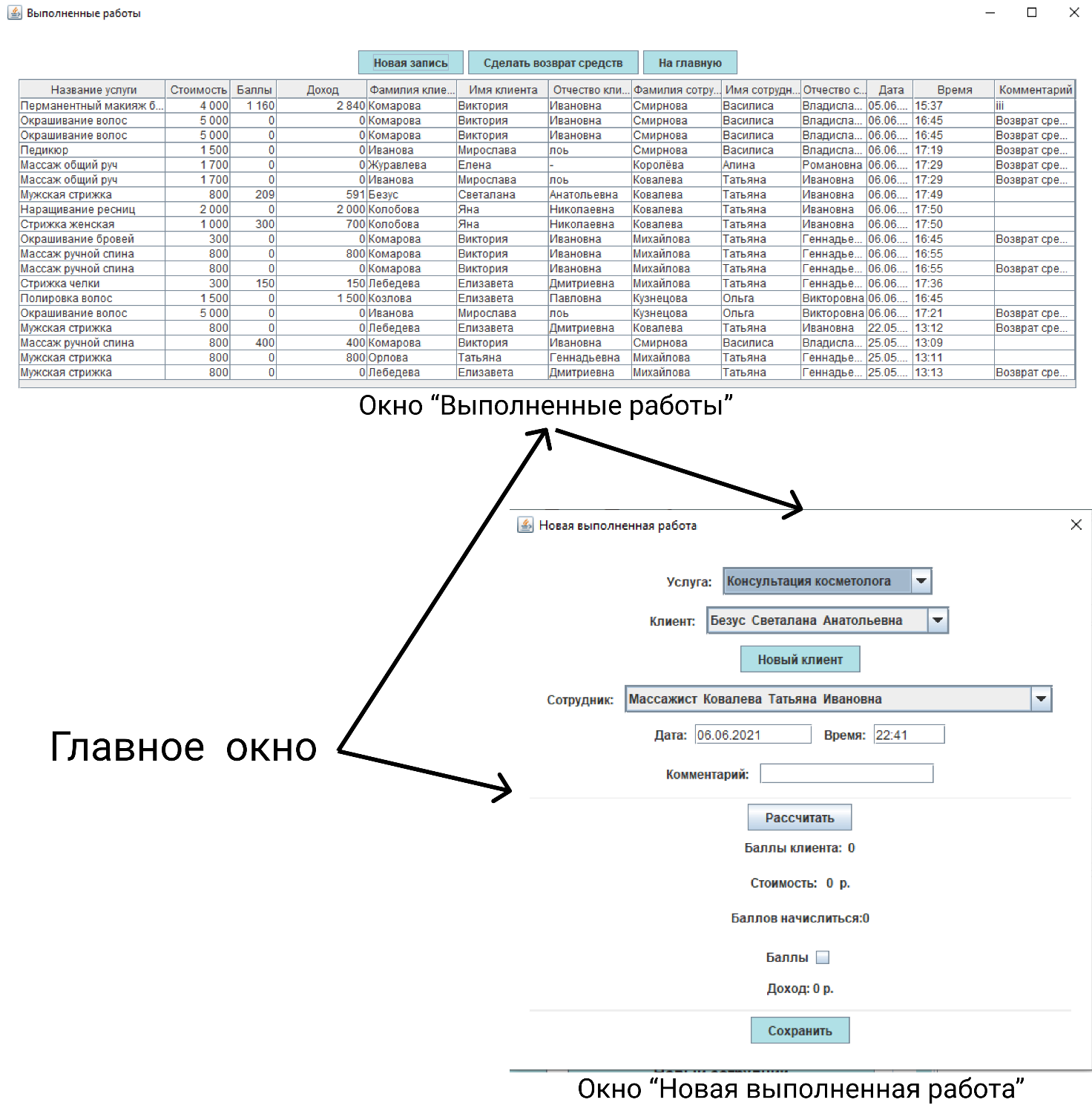


Рисунок 7 – Схема переходов между окнами приложения

* 1. Проектирование классов

Для реализации работы приложения были определены следующие классы-сущности (см рис.):

1. Clients – клиент
2. Employee – сотрудник
3. Record – запись
4. Services – услуга
5. Work – выполненная работа

Так же, исходя из спроектированных окон интерфейса, были определены следующие классы:

1. FrameRecord, FrameClients, FrameEmployee, FrameRecordDay, FrameServices, FramePerformedWork – классы для окон, содержащих таблицы с данными из БД
2. FrameReport – класс для отображения отчетов работы
3. RewardsProgram – класс для бонусной системы

Для «соединения» работы в БД и интерфейса определены следующие классы:

1. TableModel… – классы для формирования модели данных для классов интерфейса
2. DBWorker – класс для создания соединения с БД и выполнения запросов
3. Log – класс для логирования
4. UpdateTM – класс, содержащий метод для обновления всех моделей данных

Таким образом, все классы можно разделить на группы:

1. Essence (Классы-сущности) (см. рис. 7)
2. Interface (Классы графического интерфейса) (см. рис. 8)
3. TableModel&Data (Классы для работы с данными и их представления) (см. рис. 9 и рис. 10)

Для наглядности спроектированных классов была построена UML-диаграмма классов.

Взаимосвязь группы классов TableModel&Data и Interface показана на рис. 11. Предполагается, что группа классов Essence используется во всех остальных классах (при передаче между методами и пр.), поэтому показывать связи группы классов Essence нецелесообразно. Так же стоит отметить, что группа классов TableModel&Data содержит в себе такие классы как «UpdateTM», «DBWorker» и «Log», связь которых с другими классами этой группы никак не отражена на диаграмме классов UML. Это связано с тем, что, во время выполнения любых действий с данными, методы этих классов постоянно вызываются, однако объекты этих классов нигде не создаются.

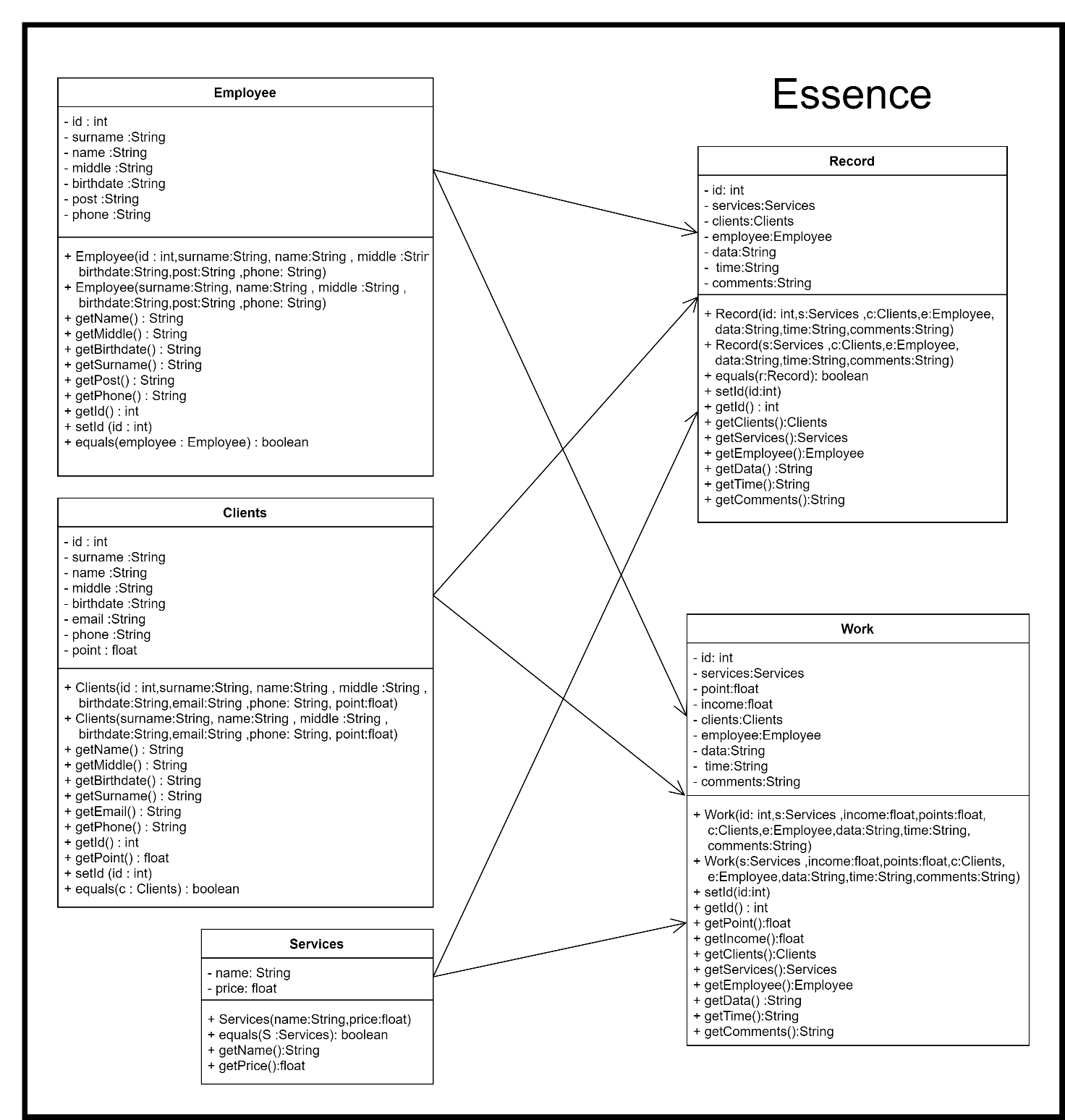


Рисунок 7 – Диаграмма UML, группа классов «Essence»

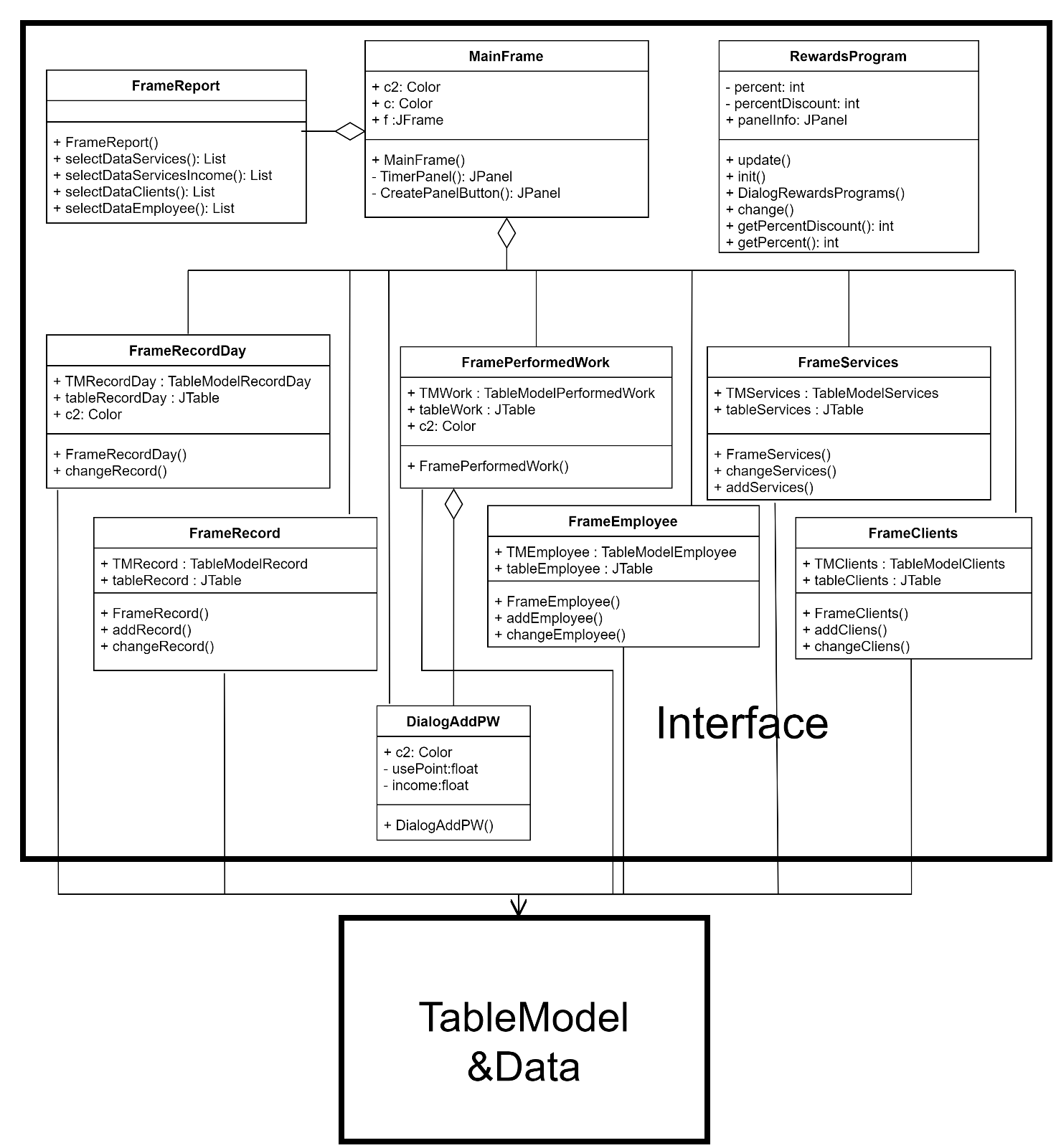
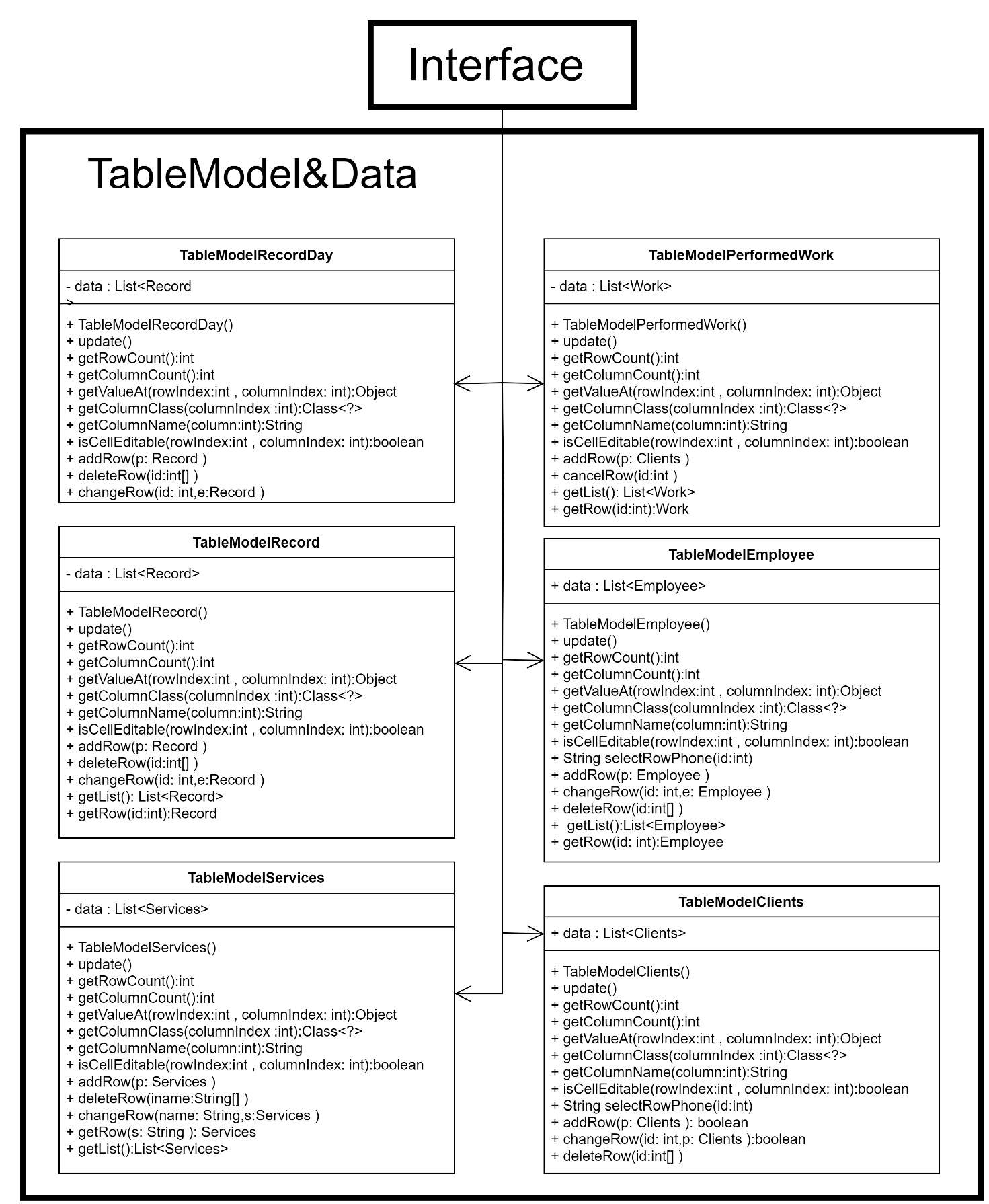


Рисунок 8 – Диаграмма UML, группа классов «Interface»



%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%26lt%3Bfont%20style%3D%26quot%3Bfont-size%3A%2038px%26quot%3B%26gt%3BInterface%26lt%3B%2Ffont%26gt%3B%22%20style%3D%22text%3Bhtml%3D1%3BstrokeColor%3Dnone%3BfillColor%3Dnone%3Balign%3Dcenter%3BverticalAlign%3Dmiddle%3BwhiteSpace%3Dwrap%3Brounded%3D0%3BfontFamily%3DHelvetica%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22550%22%20y%3D%22410%22%20width%3D%22260%22%20height%3D%2270%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

Рисунок 9 – Диаграмма UML, группа классов «TableModel&Data», первый фрагмент

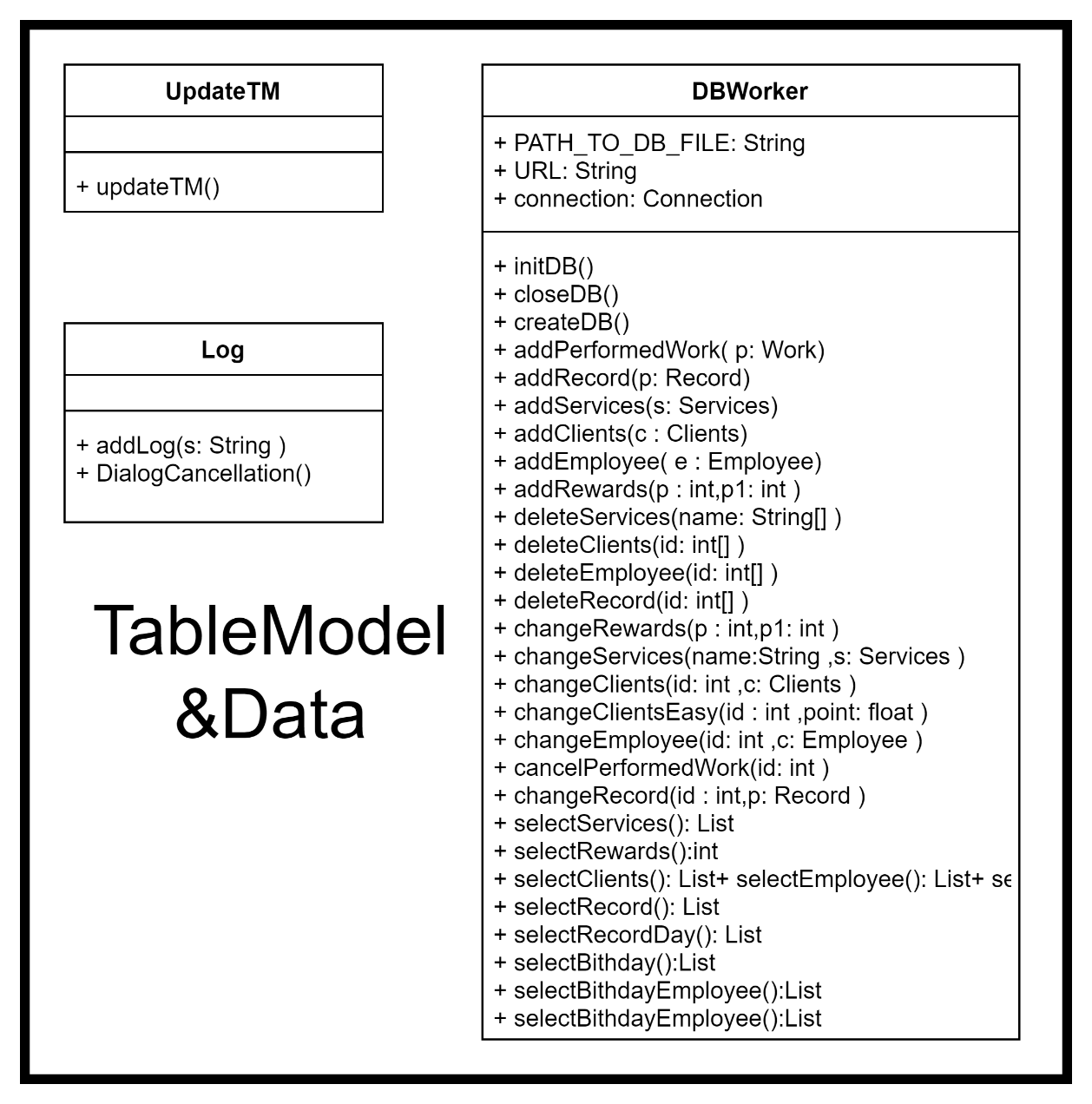


Рисунок 10 – Диаграмма UML, группа классов «TableModel&Data», второй фрагмент

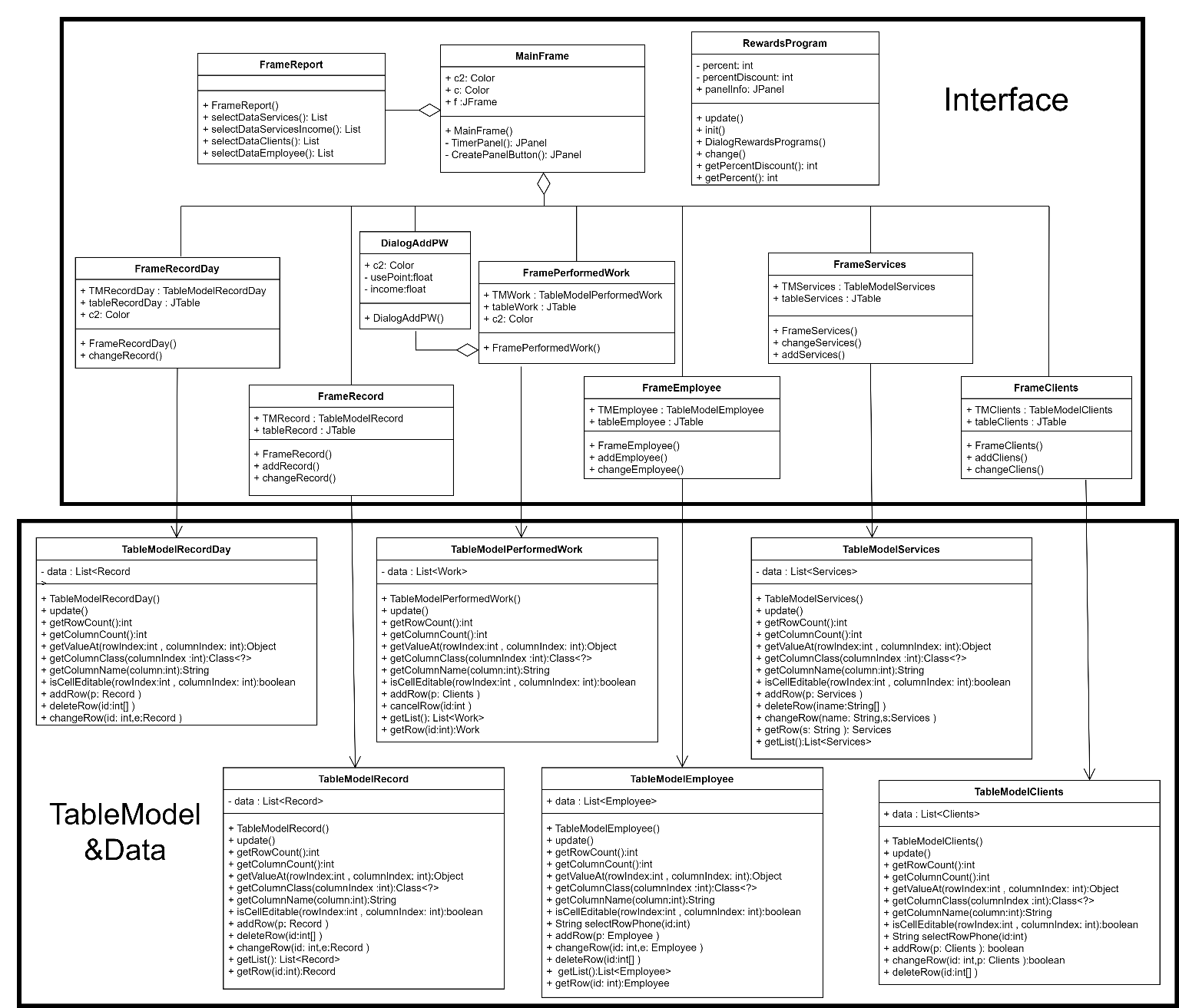


Рисунок 11 – Диаграмма UML, связь «TableModel&Data» и «Interface»

* 1. Проектирование базы данных

Для работы с базой данных были созданы следующие таблицы (см. рис. 12).

Первичным ключом для таблиц «clients», «employee», «record», «performedWork» является их уникальный идентификатор id. Для таблицы «services» первичным ключом является название услуги.

Отдельно существует таблица «rewards», предназначенная для хранения процентов, составляющих бонусную систему. Эта таблица содержит 2 поля и лишь одну запись, в ней и хранятся проценты. Она не взаимодействует ни с какой другой таблицей.

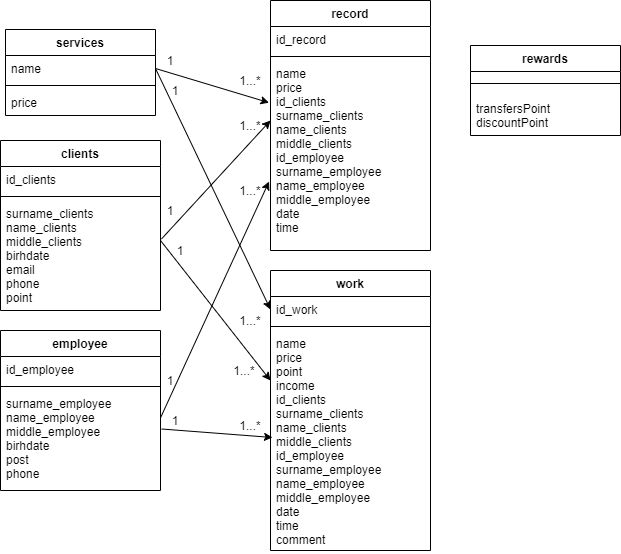


Рисунок 12 – диаграмма таблиц базы данных

Таблица 1 – Описание таблиц БД

|  |  |  |  |
| --- | --- | --- | --- |
| Имя таблицы | Имя атрибута | Тип | Описание |
| services | name | text | Идентификатор услуги в виде её названия |
| price | float | Стоимость услуги |
| clients | id\_clients | integer autoincrement | Идентификатор клиента |
| surname\_clients | text | Фамилия |
| name\_clients | text | Имя |
| middle\_clients | text | Отчество |
| birhdate | text | День рождения |
| email | text | Почта |
| phone | text not null | Телефон клиента |
| point | float | Количество баллов |
| employee | id\_employee | integer autoincrement | Идентификатор сотрудника |
| surname\_employee | text | Фамилия |

Таблица 1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| employee | name\_employee | text | Имя |
| middle\_employee | text | Отчество |
| birhdate | text | День рождения |
| post | text | Должность |
| phone | text not null | Телефон сотрудника |
| performedWork | id\_work | integer autoincrement | Идентификатор записи о выполненной работе |
| name | text | Название услуги |
| price | float | Стоимость услуги |
| point | float | Сколько баллов было использовано |
| income | float | Доход=price-point |
| id\_clients | integer | Идентификатор клиента |
| surname\_clients | text | Фамилия |
| name\_clients | text | Имя |
| middle\_clients | text | Отчество |
| id\_employee | integer | Идентификатор сотрудника |
| surname\_employee | text | Фамилия |
| name\_employee | text | Имя |
| middle\_employee | text | Отчество |
| date | text | Дата выполнения услуги |
| time | text | Время выполнения услуги |
| comment | text | Комментарий |
| record | id\_record | integer autoincrement | Идентификатор записи |
| name | text | Название услуги |
| price | float | Стоимость услуги |
| id\_clients | integer | Идентификатор клиента |
| surname\_clients | text | Фамилия |
| name\_clients | text | Имя |
| middle\_clients | text | Отчество |
| id\_employee | integer | Идентификатор сотрудника |
| surname\_employee | text | Фамилия |
| name\_employee | text | Имя |
| middle\_employee | text | Отчество |
| date | text | Дата записи |
| time | text | Время записи |
| comment | text | Комментарий |

Таблица 1 – Продолжение

|  |  |  |  |
| --- | --- | --- | --- |
| rewards | transfersPoint | int not null | Количество зачисляющихся бонусов клиенту с каждой покупки в процентах |
| discountPoint | int not null | Количество скидки из имеющихся у клиента бонусов при каждой покупки в процентах |

Реализованные SQL-запросы к базе данных:

services :

1. CREATE TABLE if not exists 'services' ('name' text PRIMARY KEY NOT NULL, ‘price' float NOT NULL); – создание таблицы в БД
2. INSERT INTO services ('name', 'price') VALUES(?,?) – Добавляет новую запись об услуге в БД
3. DELETE FROM services WHERE name=? – удаление определенной записи об услуге

DELETE FROM performedWork WHERE name=? – каскадное удаление из таблицы performedWork

DELETE FROM record WHERE name=? – каскадное удаление из таблицы record

1. UPDATE services SET 'name'=? ,'price'=? WHERE name=? –обновление записи в таблице

UPDATE record SET 'name'=? ,'price'=? WHERE name=? – каскадное обновление записи в таблице record

UPDATE performedWork SET 'name'=? WHERE name=? – каскадное обновление записи в таблице performedWork

1. SELECT \* FROM services ORDER BY name – возвращает все записи

clients :

1. CREATE TABLE if not exists 'clients' ('id\_clients' INTEGER PRIMARY KEY AUTOINCREMENT, 'surname\_clients' text, 'name\_clients' text NOT NULL, 'middle\_clients' text, 'birhdate' text, 'email' text, 'phone' text NOT NULL, 'point' float ); – создание таблицы в БД
2. INSERT INTO clients ('surname\_clients', 'name\_clients','middle\_clients','birhdate','email','phone','point') VALUES(?,?,?,?,?,?,?)" – добавление в БД
3. DELETE FROM clients WHERE id\_clients=? – удаление записи

DELETE FROM performedWork WHERE id\_clients=? – каскадное удаление в таблице performedWork

DELETE FROM record WHERE id\_clients=? – каскадное удаление в таблице record

1. UPDATE clients SET 'surname\_clients'=?, 'name\_clients'=?, 'middle\_clients'=?,'birhdate'=?, 'email'=?, 'phone'=? WHERE id\_clients =? – обновление записи

UPDATE record SET 'surname\_clients'=?,'name\_clients'=?,'middle\_clients'=?WHERE id\_record =? – каскадное обновление записи

UPDATE performedWork SET 'surname\_clients'=?,'name\_clients'=?,'middle\_clients'=?WHERE id\_clients =? – каскадное обновление записи

1. UPDATE clients SET 'point'=? WHERE id\_clients =? – обновление количества бонусов клиента
2. SELECT \* FROM clients ORDER BY surname\_clients,name\_clients – возвращает всех клиентов сортируя по фамилии и имени

employee :

1. CREATE TABLE if not exists 'employee' ( 'id\_employee' INTEGER PRIMARY KEY AUTOINCREMENT, 'surname\_employee' text, 'name\_employee' text NOT NULL, 'middle\_employee' text, 'birhdate' text, 'post' text, 'phone' text NOT NULL); – создание таблицы employee
2. INSERT INTO employee ('surname\_employee', 'name\_employee','middle\_employee','birhdate','post','phone’) VALUES(?,?,?,?,?,?) – добавление записи о сотруднике
3. DELETE FROM employee WHERE id\_employee=? – удаление сотрудника из таблицы employee

DELETE FROM record WHERE id\_employee=? – каскадное удаление

DELETE FROM performedWork WHERE id\_employee=? – каскадное удаление

1. UPDATE employee SET 'surname\_employee'=?, 'name\_employee'=?,'middle\_employee'=?,'birhdate'=?,'post'=?,'phone'=? WHERE id\_employee =? – обновление записи о сотруднике

UPDATE record SET 'surname\_employee'=?,'name\_employee'=?,'middle\_employee'=?WHERE id\_employee =? – каскадное обновление

UPDATE performedWork SET 'surname\_employee'=?,'name\_employee'=?,'middle\_employee'=?WHERE id\_employee =? – каскадное обновление

1. SELECT \* FROM employee ORDER BY surname\_employee,name\_employe – возвращает всех сотрудников сортируя по фамилии и имени

performedWork :

1. CREATE TABLE if not exists 'performedWork' ( 'id\_work' INTEGER PRIMARY KEY AUTOINCREMENT, 'name' text NOT NULL, 'price' float NOT NULL, 'point' float , 'income' float , 'id\_clients' INTEGER , 'surname\_clients' text, 'name\_clients' text NOT NULL, 'middle\_clients' text, 'id\_employee' INTEGER , 'surname\_employee' text, 'name\_employee' text NOT NULL, \_employee' text, 'date' text, text, 'comment' text ); – создание таблицы performedWork
2. INSERT INTO performedWork ('name','price', 'point' ,'income' ,'id\_clients','surname\_clients','name\_clients','middle\_clients','id\_employee','surname\_employee','name\_employee','middle\_employee','date','time','comment') VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?) – добавление записи о выполненной работе
3. UPDATE performedWork SET 'income'=?,'point'=?,'comment'=? WHERE id\_work =? –
4. SELECT \* FROM performedWork ORDER BY date,id\_employee,time – возращает все записи в таблице performedWork сортированные по дате,сотруднику и времени

record:

1. INSERT INTO record ('name','price','id\_clients','surname\_clients','name\_clients','middle\_clients','id\_employee','surname\_employee','name\_employee','middle\_employee','date','time','comment') VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)
2. DELETE FROM record WHERE id\_record=? – удаляет запись
3. UPDATE record SET 'name'=?,'price'=?,'id\_clients'=?, 'surname\_clients'=?,'name\_clients'=?,'middle\_clients'=?,'id\_employee'=?,'surname\_employee'=?,'name\_employee'=?,'middle\_employee'=?,'date'=?,'time'=?,'comment'=?WHERE id\_record =? – обновляет данные о записи
4. SELECT \* FROM record ORDER BY date,id\_employee,time – возращает все записи в таблице record сортированные по дате,сотруднику и времени

rewards:

1. CREATE TABLE if not exists 'rewards' ( 'transfersPoint' int NOT NULL, 'discountPoint' int NOT NULL); – создание таблицы
2. INSERT INTO rewards ('transfersPoint', 'discountPoint') VALUES(?,?) – добавление записи
3. UPDATE rewards SET 'transfersPoint'=? ,'discountPoint'=? – обновления данных в записи

# Реализация приложения

* 1. Спецификация реализованных классов

Для удобства работы с классами разработаны пакеты, которые делят классы по функциям.

Пакет main:

1. Пакет data – содержит классы и файлы с данными и все необходимое для работы с ними
   1. Пакет database – пакет для работы с БД
      1. BeautyShop.db – файл локальной БД
      2. DBWorker – класс, содержащий в себе методы для работы с БД (запросы, соединение и пр.)
   2. Пакет log – пакет для работы логирования
      1. log – класс с методами, которые добавляют записи в файл логирования, а также с окном показывающее логирование в программе
      2. Log.txt – txt-файл с данными логирования
   3. Пакет tableModel – пакет для работы классов наследуемых от AbstractTableModel
      1. TableModelClients – класс, представляющий модель данных о клиентах
      2. TableModelEmployee – класс, представляющий модель данных о сотрудниках
      3. TableModelPerformedWork – класс, представляющий модель данных о выполненных записях
      4. TableModelRecord – класс, представляющий модель данных о записях
      5. TableModelRecordDay – класс, представляющий модель данных о записях на день
      6. TableModelServices – класс, представляющий модель данных об услугах
      7. UpdateTM – класс, представляющий обновление всех моделей данных
2. Пакет essence – пакет с классами-сущностями необходимыми для работы с данными
   1. Clients – класс-сущность клиента
   2. Employee – класс-сущность сотрудника
   3. Record – класс-сущность записи
   4. Services – класс-сущность услуги
   5. Work – класс-сущность выполненной работы
3. Пакет userInterface – пакет с классами представляющие интерфейс программы
   1. Пакет table – пакет с классами интерфейса связанного с таблицами
      1. DialogAddPW – класс для окна добавления новой выполненной работы
      2. FrameClients – класс с интерфейсом окна «Клиенты», «Новый клиент», «Редактировать клиента»
      3. FrameEmployee – класс с интерфейсом окна «Сотрудники», «Новый сотрудник», «Редактировать сотрудник»
      4. FramePerformedWork – класс с интерфейсом окна «Выполненные работы»
      5. FrameRecord – класс с интерфейсом окна «Записи», «Новая запись», «Редактирование записи»
      6. FrameRecordDay – класс с интерфейсом окна «Записи на день»
      7. FrameServices – класс с интерфейсом окна «Услуги», «Новая услуга», «Редактирование услуги»
   2. FrameReport – класс с интерфейсом окна «Отчеты»
   3. MainFrame – класс с интерфейсом окна «Главное окно»
   4. RewardsProgram – класс с интерфейсом окна «Бонусная система»
4. Main – класс, содержащий точку входа в программу

Пакет test:

1. Пакет tableModel – содержит классы необходимые для Unit-тестирования классов, наследуемых от AbstractTableModel
   1. TableModelClientsTest – класс для тестирования класса TableModelClients
   2. TableModelEmployeeTest – класс для тестирования класса TableModelEmployee
   3. TableModelPerformedWorkTest – класс для тестирования класса TableModelPerformedWork
   4. TableModelRecordTest – класс для тестирования класса TableModelRecord
   5. TableModelServicesTest – класс для тестирования класса TableModelServices
2. FrameReportTest – класс для тестирования класса FrameReport
3. LogTest – класс для тестирования класса Log (Логирование)

# Тестирование

## Методика тестирования

В связи с большим объёмом предполагаемых тестов, при тестирование решено было использовать 2 метода:

Тестирование белого ящика – для нашей программы это юнит-тестирование, при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции - работоспособны и устойчивы, до определенной степени.

Тестирование черного ящика с помощью критических путей (сценариев использования) – это тестирование, функционального поведения объекта на предмет правильности работы при стандартном использовании программы.

## Разработанные тесты

Все разработанные тесты были разделены по функциональным требованиям на группы (Таблицы).

Таблица 2 – Тесты для главного окна

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 1 | Пользователь запускает систему | Открытие Главного окна |
| 2 | Пользователь в главном окне нажимает кнопку «Услуги» | Открытие окна Услуги |
| 3 | Пользователь в главном окне нажимает кнопку «Клиенты» | Открытие окна Клиенты |
| 4 | Пользователь в главном окне нажимает кнопку «Новый клиент» | Открытие окна Новый клиент |
| 5 | Пользователь в главном окне нажимает кнопку «Сотрудники» | Открытие окна Сотрудники |
| 6 | Пользователь в главном окне нажимает кнопку «Новый сотрудник» | Открытие окна Новый сотрудник |
| 7 | Пользователь в главном окне нажимает кнопку «Все записи» | Открытие окна Все записи |
| 8 | Пользователь в главном окне нажимает кнопку «Записи на день» | Открытие окна Записи на день |
| 9 | Пользователь в главном окне нажимает кнопку «Сделать запись» | Открытие окна Новая запись |
| 10 | Пользователь в главном окне нажимает кнопку «Выполненные работы» | Открытие окна Выполненные работы |
| 11 | Пользователь в главном окне нажимает кнопку «Оказание услуги» | Открытие окна Новая выполненная работа |

Таблица 2 – Продолжение

|  |  |  |
| --- | --- | --- |
| 12 | Пользователь в главном окне нажимает кнопку «Бонусная программа» | Открытие окна Бонусная программа |
| 13 | Пользователь в главном окне нажимает кнопку «Отчеты» | Открытие окна Отчеты |
| 14 | Пользователь в главном окне нажимает кнопку «Log» | Открытие окна Логирование |

Таблица 3 – Тесты для окна Услуги, Новая услуга и Редактирование услуги

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 15 | Пользователь выбирает «Новая услуга» | Открытие окна Новая услуга |
| 16 | Пользователь заполняет поля Название и Стоимость, нажимает «Добавить».  Название: Стрижка  Стоимость: 1000 | Сообщение по поводу успешного добавления услуги  Закрытие окна Новая услуга |
| 17 | Пользователь заполняет поля Название и Стоимость, нажимает «Добавить».  Название: Стрижка  Стоимость: ккк | Сообщение «В поле стоимость можно вводить только цифры!» |
| 18 | Пользователь заполняет поля Название и Стоимость, нажимает «Добавить».  Название: Стрижка  Стоимость: -100 | Сообщение «В поле стоимость можно вводить только положи-тельные числа!» |
| 19 | Пользователь заполняет поля Название и Стоимость нажимает «Добавить».  (Данная услуга уже существует)  Название: Педикюр  Стоимость: 1500 | Сообщение «Уже существует такая услуга!»  Закрытие окна Новая услуга |
| 20 | Пользователь выбирает услугу и нажимает «Редактировать услугу» | Открытие окна Редактирование услуги с заполненными полями |
| 21 | Тест 16, но для редактирования | Сообщение по поводу успешного редактирования услуги  Закрытие окна Редактирование услуги |
| 22 | Тест 17, но для редактирования | См. тест 17 |
| 23 | Тест 18, но для редактирования | См. тест 18 |
| 24 | Тест 19, но для редактирования | См. тест 19 |

Таблица 3 – Продолжение

|  |  |  |
| --- | --- | --- |
| 25 | Пользователь выбирает одну или несколько услуг и нажимает «Удалить услугу» | Сообщение «Запись успешно удалена!» |
| 26 | Пользователь нажимает «На главную» | Закрытие окна Услуги |

Таблица 4 – Тесты для окна Клиенты, Новый клиент и Редактирование клиента

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 27 | Пользователь нажимает «Новый клиент» | Открытие окна Новая услуга |
| 28 | Пользователь заполняет поля, нажимает «Добавить».  Фамилия: Иванова  Имя: Мария  Отчество: Ивановна  День рождение: 22.06  Почта: e  Телефон: +8-545-454-45-07 | Сообщение по поводу успешного добавления клиента  Закрытие окна Новый клиент |
| 29 | Пользователь заполняет не все поля, нажимает «Добавить».  Фамилия:  Имя:  Отчество:  День рождение: 22.06  Почта: e  Телефон: +0-000-000-00-00 | Сообщение «Поля Фамилия, Имя, Отчество и телефон должны быть заполнены!» |
| 30 | Пользователь выбирает клиента и нажимает «Редактировать» | Открытие окна Редактирование клиента с заполненными полями |
| 31 | Тест 28, но для редактирования | Сообщение по поводу успешного редактирования клиента  Закрытие окна Редактирование клиента |
| 32 | Тест 29, но для редактирования | См. тест 29 |
| 33 | Пользователь выбирает одного или несколько клиентов и нажимает «Удалить клиента» | Сообщение «Клиент успешно удален!» |
| 34 | Пользователь нажимает «На главную» | Закрытие окна Клиенты |

Таблица 5 – Тесты для окна Сотрудники, Новый сотрудник и Редактирование сотрудника

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 35 | Пользователь нажимает «Новый сотрудник» | Открытие окна Новый сотрудник |
| 36 | Пользователь заполняет поля, нажимает «Добавить».  Фамилия: Иванова  Имя: Мария  Отчество: Ивановна  День рождение: 22.06  Должность: e  Телефон: +8-545-454-45-07 | Сообщение по поводу успешного добавления сотрудника  Закрытие окна Новый сотрудник |
| 37 | Пользователь заполняет не все поля, нажимает «Добавить».  Фамилия:  Имя:  Отчество:  День рождение: 22.06  Должность:  Телефон: +0-000-000-00-00 | Сообщение «Поля Фамилия, Имя, Отчество, Должность и телефон должны быть заполнены!» |
| 38 | Пользователь выбирает сотрудника и нажимает «Редактировать» | Открытие окна Редактирование сотрудника с заполненными полями |
| 39 | Тест 36, но для редактирования | Сообщение по поводу успешного редактирования сотрудника  Закрытие окна Редактирование сотрудника |
| 40 | Тест 37, но для редактирования | См. тест 37 |
| 41 | Пользователь выбирает одного или несколько сотрудников и нажимает «Удалить сотрудника» | Сообщение «Сотрудник успешно удален!» |
| 42 | Пользователь нажимает «На главную» | Закрытие окна Сотрудники |

Таблица 6 – Тесты для окна Выполненные работы, Новая выполненная работа

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 42 | Пользователь нажимает «Новая запись»/ «Оказание услуги» (для главного окна) | Открытие окна Новая выполненная работа |
| 43 | Пользователь заполняет поля, нажимает «Сохранить».  Данные введены верно | Сообщение по поводу успешного добавления записи  Закрытие окна Новая выполненная работа |

Таблица 6 – Продолжение

|  |  |  |
| --- | --- | --- |
| 44 | Пользователь заполняет поля, нажимает «Добавить».  Дата и время указаны как будущее время | Сообщение «Нельзя выполнять работы в будущем времени !» |
| 45 | Пользователь выбирает одну или несколько записей и нажимает «Сделать возврат средств» | Сообщение «Возврат средств выполнен!» |
| 46 | Тест 45, но для выбранной записи уже был выполнен возврат средств | Сообщение «Уже был выполнен возврат средств!» |
| 47 | Пользователь не выбирает ни одной записи и нажимает «Сделать возврат средств» | Сообщение «Вы не выбрали запись» |
| 48 | Пользователь нажимает «На главную» | Закрытие окна Выполненные работы |

Таблица 7 – Тесты для окна Записи на день, Новая запись, Редактирование записи, Все записи

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 49 | Пользователь нажимает «Новая запись» | Открытие окна Новая запись |
| 50 | Пользователь заполняет поля, нажимает «Сохранить».  Данные введены верно | Сообщение по поводу успешного добавления записи  Закрытие окна Новая запись |
| 51 | Пользователь заполняет поля, нажимает «Добавить».  Дата и время указаны как прошлое время | Сообщение «Нельзя записывать на прошлое !» |
| 52 | Пользователь выбирает одну или несколько записей и нажимает «Удалить запись» | Сообщение «Запись успешно удалена!» |
| 53 | Пользователь не выбирает ни одной записи и нажимает «Удалить запись» | Сообщение «Вы не выбрали запись» |
| 54 | Тест 50, но для редактирования | Сообщение по поводу успешного редактирования записи  Закрытие окна Редактирование записи |
| 55 | Тест 51, но для редактирования | См. тест 37 |
| 56 | Пользователь нажимает «На главную» | Закрытие окна Записи на день/Все записи |

Таблица 8 – Тесты для окна Отчеты, Логирование, Бонусная программа и панели уведомления на главном окне

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 57 | Пользователь в окне Отчеты нажимает на вкладки | Меняется содержимое |
| 58 | Пользователь добавляет сотрудника/клиента с датой рождения, совпадающей с сегодняшней | В панели уведомлений присутствуют те, кого добавил пользователь |
| 59 | Пользователь добавляет запись на ближайший час | В панели уведомлений добавленная запись отображается |

Таблица 9 – Unit – тесты для проверки работы логики внутри системы

|  |  |  |
| --- | --- | --- |
| **№** | **Совершенное действие** | **Ожидаемый результат** |
| 60 | Добавление услуги | Тест пройден |
| 61 | Удаление услуги | Тест пройден |
| 62 | Редактирование услуги | Тест пройден |
| 63 | Добавление клиента | Тест пройден |
| 64 | Удаление клиента | Тест пройден |
| 65 | Редактирование клиента | Тест пройден |
| 66 | Добавление сотрудника | Тест пройден |
| 67 | Удаление сотрудника | Тест пройден |
| 68 | Редактирование сотрудника | Тест пройден |
| 69 | Добавление записи | Тест пройден |
| 70 | Удаление записи | Тест пройден |
| 71 | Редактирование записи | Тест пройден |
| 72 | Добавление выполненной работы | Тест пройден |
| 73 | Возврат средств | Тест пройден |
| 74 | Редактирование записи | Тест пройден |
| 75 | Формирование отчета по клиентам | Тест пройден |
| 76 | Формирование отчета по услугам | Тест пройден |
| 77 | Формирование отчета по сотрудникам | Тест пройден |
| 78 | Формирование отчета по доходу от услуг | Тест пройден |
| 79 | Добавление лога о добавление услуги | Тест пройден |
| 80 | Добавление лога об удаление услуги | Тест пройден |
| 81 | Добавление лога об изменение услуги | Тест пройден |

## Разработанные сценарии использования

Сценарии использования программы необходимы для формирования тестов в группы, так как между многими из них есть последовательная связь.

Результаты тестирования будут сгруппированы по выполненным сценариям, а в сценарии указывается номер теста.

Для всех сценариев действующие лица: Пользователь и Система.

**Сценарий №1** Добавление услуги.

Успешный сценарий:

* + 1. Пользователь выбирает в главном окне «Услуги». (Тест 2)
    2. Система открывает окно «Услуги»
    3. Пользователь выбирает «Новая услуга» (Тест 15)
    4. Система открывает окно добавления услуги.
    5. Пользователь заполняет поля Название и Стоимость, нажимает «Добавить». (Тест 16)
    6. Система проверяет корректность введенных данных
    7. Система добавляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 60)
    8. Система выдает пользователю сообщение по поводу успешного добавления услуги.

Результат: Пользователь успешно добавил новую услугу.

Расширения:

6а В поле стоимость записаны не цифры (Тест 17)

Система выдает сообщение «В поле стоимость можно вводить только цифры!»

Переход на шаг 6

6б В поле стоимость записано отрицательное число (Тест 18)

Система выдает сообщение «В поле стоимость можно вводить только положительные числа!»

Переход на шаг 6

6в Услуга, которую хочет добавить пользователь уже существует. (Тест 19)

Система выдает сообщение «Уже существует такая услуга!»

Переход на шаг 3

**Сценарий №2** Редактирование услуги.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Услуги». (Тест 2)
2. Система открывает окно «Услуги»
3. Пользователь выбирает услугу и нажимает «Редактировать услугу» (Тест 20)
4. Система открывает окно редактирования услуги.
5. Пользователь заполняет поля Название и Стоимость, нажимает «Изменить». (Тест 21)
6. Система проверяет корректность введенных данных
7. Система изменяет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 62)
8. Система выдает пользователю сообщение по поводу успешного изменения услуги.

Результат: Пользователь успешно изменил услугу.

Расширения:

6а В поле стоимость записаны не цифры (Тест 22)

Система выдает сообщение «В поле стоимость можно вводить только цифры!»

Переход на шаг 6

6б В поле стоимость записано отрицательное число (Тест 23)

Система выдает сообщение «В поле стоимость можно вводить только положительные числа!»

Переход на шаг 6

6в Услуга, которую хочет добавить пользователь уже существует. (Тест 24)

Система выдает сообщение «Уже существует такая услуга!»

Переход на шаг 3

**Сценарий №3** Удаление услуги.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Услуги». (Тест 2)
2. Система открывает окно «Услуги»
3. Пользователь выбирает услугу и нажимает «Удалить услугу» (Тест 25)
4. Система удаляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 61)
5. Система выдает пользователю сообщение по поводу успешного добавления услуги.

Результат: Пользователь успешно удалил услугу.

**Сценарий №4** Добавление клиента.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Клиенты». (Тест 3)
2. Система открывает окно «Клиенты»
3. Пользователь выбирает «Новый Клиент» (Тест 27)
4. Система открывает окно добавления клиента
5. Пользователь заполняет поля, нажимает «Добавить». (Тест 28)
6. Система проверяет корректность введенных данных
7. Система добавляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 63)
8. Система выдает пользователю сообщение по поводу успешного добавления клиента

Результат: Пользователь успешно добавил нового клиента

Расширения:

1а Пользователь выбирает в главном окне «Новый клиент» (Тест 4)

Переход на шаг 4

6а В любом из полей Поля Фамилия, Имя, Отчество или телефон нет данных (Тест 29)

Система выдает сообщение «Поля Фамилия, Имя, Отчество и телефон должны быть заполнены!»

Переход на шаг 6

**Сценарий №5** Редактирование клиента.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Клиенты». (Тест 3)
2. Система открывает окно «Клиенты»
3. Пользователь выбирает клиента и нажимает «Редактировать» (Тест 30)
4. Система открывает окно редактирования клиента
5. Пользователь заполняет поля, нажимает «Изменить». (Тест 31)
6. Система проверяет корректность введенных данных
7. Система изменяет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 65)
8. Система выдает пользователю сообщение по поводу успешного изменения клиента.

Результат: Пользователь успешно изменил клиента.

Расширения:

6а В любом из полей Поля Фамилия, Имя, Отчество или телефон нет данных (Тест 32)

Система выдает сообщение «Поля Фамилия, Имя, Отчество и телефон должны быть заполнены!»

Переход на шаг 6

**Сценарий №6** Удаление клиента.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Клиенты». (Тест 3)
2. Система открывает окно «Клиенты»
3. Пользователь выбирает клиента и нажимает «Удалить клиента» (Тест 33)
4. Система удаляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 64)
5. Система выдает пользователю сообщение по поводу успешного удаления клиента.

Результат: Пользователь успешно удалил клиента.

**Сценарий №7** Добавление сотрудника.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Сотрудники». (Тест 5)
2. Система открывает окно «Сотрудники»
3. Пользователь выбирает «Новый сотрудник» (Тест 35)
4. Система открывает окно добавления сотрудника
5. Пользователь заполняет поля, нажимает «Добавить». (Тест 36)
6. Система проверяет корректность введенных данных
7. Система добавляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 66)
8. Система выдает пользователю сообщение по поводу успешного добавления сотрудника

Результат: Пользователь успешно добавил нового сотрудника

Расширения:

1а Пользователь выбирает в главном окне «Новый сотрудник» (Тест 6)

Переход на шаг 4

6а В любом из полей Поля Фамилия, Имя, Отчество, Должность или телефон нет данных (Тест 37)

Система выдает сообщение «Поля Фамилия, Имя, Отчество, Должность и телефон должны быть заполнены!»

Переход на шаг 6

**Сценарий №8** Редактирование сотрудника.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Сотрудники». (Тест 5)
2. Система открывает окно «Сотрудники»
3. Пользователь выбирает сотрудника и нажимает «Редактировать» (Тест 38)
4. Система открывает окно редактирования сотрудника
5. Пользователь заполняет поля, нажимает «Изменить». (Тест 39)
6. Система проверяет корректность введенных данных
7. Система изменяет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 68)
8. Система выдает пользователю сообщение по поводу успешного изменения сотрудника.

Результат: Пользователь успешно изменил сотрудника.

Расширения:

6а В любом из полей Поля Фамилия, Имя, Отчество или телефон нет данных (Тест 40)

Система выдает сообщение «Поля Фамилия, Имя, Отчество и телефон должны быть заполнены!»

Переход на шаг 6

**Сценарий №9** Удаление сотрудника.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Сотрудники». (Тест 5)
2. Система открывает окно «Сотрудники»
3. Пользователь выбирает сотрудника и нажимает «Удалить сотрудника» (Тест 41)
4. Система удаляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 67)
5. Система выдает пользователю сообщение по поводу успешного удаления сотрудника.

Результат: Пользователь успешно удалил сотрудника.

**Сценарий №10** Добавление записи.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Все записи». (Тест 7)
2. Система открывает окно «Все записи»
3. Пользователь выбирает «Новая запись» (Тест 49)
4. Система открывает окно добавления записи
5. Пользователь заполняет поля, нажимает «Добавить». (Тест 50)
6. Система проверяет корректность введенных данных
7. Система добавляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 69)
8. Система выдает пользователю сообщение по поводу успешного добавления записи

Результат: Пользователь успешно добавил новую запись

Расширения:

1а Пользователь выбирает в главном окне «Сделать запись» (Тест 9)

Переход на шаг 4

1б Пользователь выбирает в главном окне «Записи на день» (Тест 8)

Система открывает окно «Записи на день»

Переход на шаг 3

5а Пользователь не находит нужного клиента, выбирает «Добавить клиента»

Переход на Сценарий №4, шаг 4

После добавления клиента пользователь возвращается к окну «Новая запись»

Переход на шаг 5

6а Введенные пользователем дата и время являются в прошлом (Тест 51)

Система выдает сообщение «Нельзя записывать на прошлое!»

Переход на шаг 5

**Сценарий №11** Редактирование записи.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Все записи». (Тест 7)
2. Система открывает окно «Все записи»
3. Пользователь выбирает «Редактировать запись» (Тест 54)
4. Система открывает окно редактирования записи
5. Пользователь заполняет поля, нажимает «Изменить». (Тест 54)
6. Система проверяет корректность введенных данных
7. Система изменяет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 71)
8. Система выдает пользователю сообщение по поводу успешного изменения записи.

Результат: Пользователь успешно изменил запись.

Расширения:

1а Пользователь выбирает в главном окне «Записи на день» (Тест 8)

Система открывает окно «Записи на день»

Переход на шаг 3

5а Пользователь не находит нужного клиента, выбирает «Добавить клиента»

Переход на Сценарий №4, шаг 4

После добавления клиента пользователь возвращается к окну «Новая запись»

Переход на шаг 5

6а Введенные пользователем дата и время являются в прошлом (Тест 55)

Система выдает сообщение «Нельзя записывать на прошлое!»

Переход на шаг 5

**Сценарий №12** Удаление записи.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Все записи». (Тест 7)
2. Система открывает окно «Все записи»
3. Пользователь выбирает записи и нажимает «Удалить записи» (Тест 52)
4. Система удаляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 70)
5. Система выдает пользователю сообщение по поводу успешного удаления записи.

Результат: Пользователь успешно удалил записи.

1а Пользователь выбирает в главном окне «Записи на день» (Тест 8)

Система открывает окно «Записи на день»

Переход на шаг 3

3а Пользователь не выбрал запись (Тест 53)

Система выдает сообщение «Вы не выбрали запись»

Переход на шаг 3

**Сценарий №13** Добавление выполненной работы.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Выполненные работы». (Тест 10)
2. Система открывает окно «Выполненные работы»
3. Пользователь выбирает «Новая запись» (Тест 42)
4. Система открывает окно добавления выполненной работы
5. Пользователь заполняет поля, нажимает «Сохранить». (Тест 43)
6. Система проверяет корректность введенных данных
7. Система добавляет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 72)
8. Система выдает пользователю сообщение по поводу успешного добавления выполненной работы

Результат: Пользователь успешно добавил новую запись о выполненной работе

Расширения:

1а Пользователь выбирает в главном окне «Оказать услугу» (Тест 11)

Переход на шаг 4

5а Пользователь не находит нужного клиента, выбирает «Добавить клиента»

Переход на Сценарий №4, шаг 4

После добавления клиента пользователь возвращается к окну «Новая выполненная работа»

Переход на шаг 5

6а Введенные пользователем дата и время являются в будущем (Тест 44)

Система выдает сообщение «Нельзя выполнять работы в будущем времени!»

Переход на шаг 5

**Сценарий №14** Выполнение возврата средств для выполненной работы.

Успешный сценарий:

1. Пользователь выбирает в главном окне «Выполненные работы». (Тест 10)
2. Система открывает окно «Выполненной работы»
3. Пользователь выбирает выполненную работу и нажимает «Сделать возврат средств» (Тест 45)
4. Система изменяет запись в БД, обновляет необходимые таблицы, записывает Log (Тест 73)
5. Система выдает пользователю сообщение по поводу успешного возврата средств.

Результат: Пользователь успешно выполнил возврат средств.

3а Пользователь не выбрал запись (Тест 47)

Система выдает сообщение «Вы не выбрали запись»

Переход на шаг 3

4а Пользователь выбрал запись, для которой уже был выполнен возврат средств(Тест 46)

Система выдает сообщение «Уже был выполнен возврат средств!»

Переход на шаг 3

## Результаты тестирования

Предпологается,что результат нескольких тестов, где ожидаемое действие Пользователь нажимает «На главную», не отображён графически, однако с помощью кнопки «На главную» были проведены тесты 1-11 , следовательно кнопка работает корректно и тесты 26, 34, 42, 56,48 пройдены.



Рисунок 13 – Результат выполнения теста 1

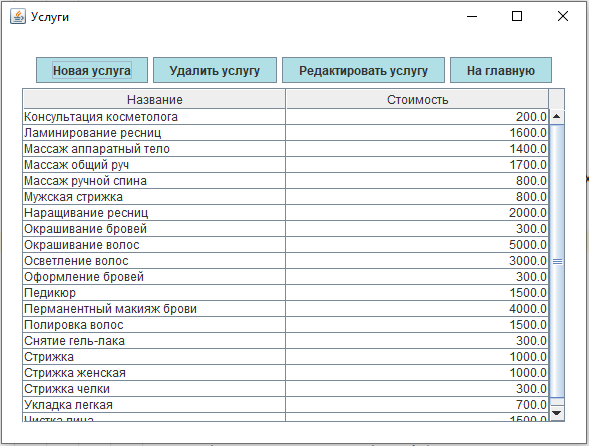


Рисунок 14 – Результат выполнения теста 2

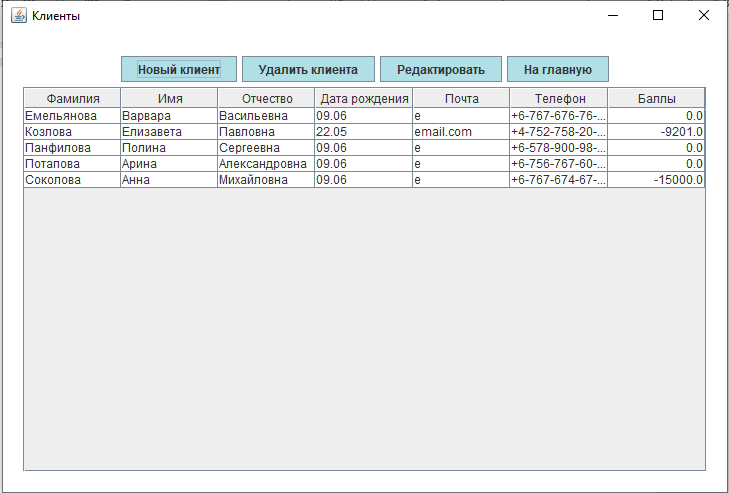


Рисунок 15 – Результат выполнения теста 3

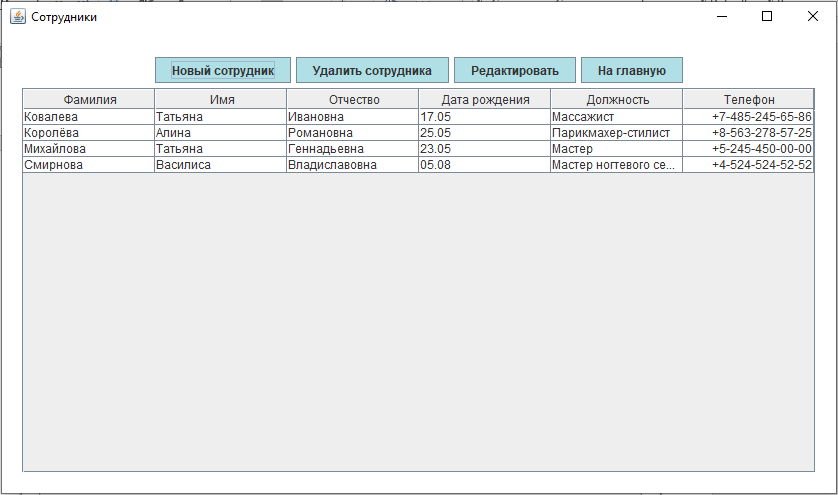


Рисунок 16 – Результат выполнения теста 5

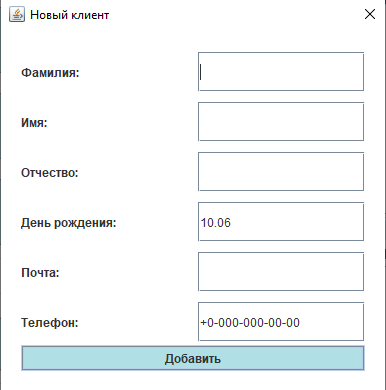
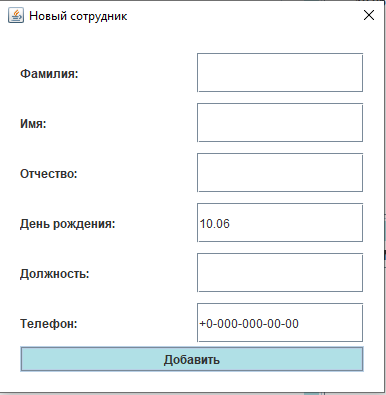
 

Рисунок 17 – а) результат теста 4; б) результат теста 6

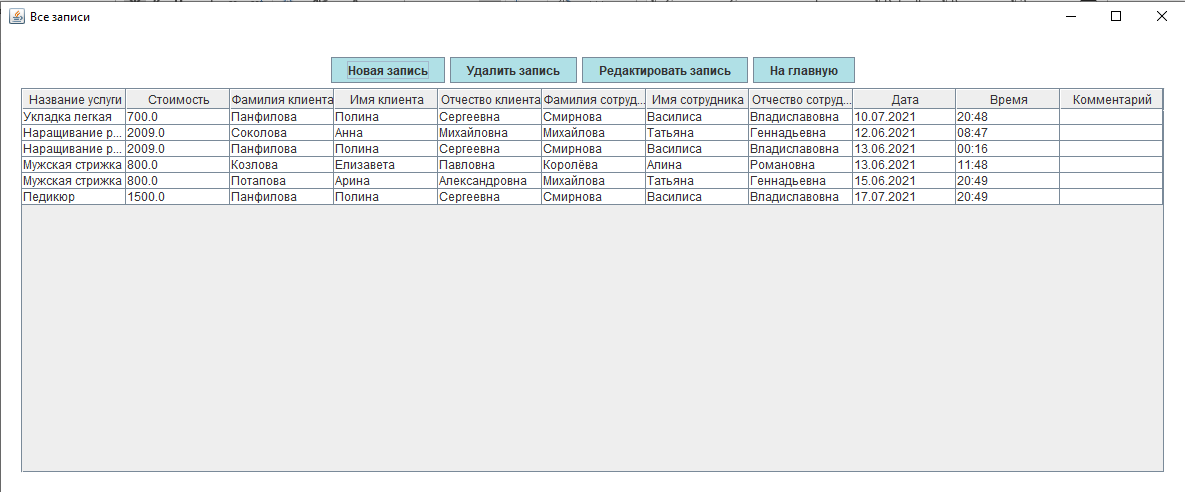


Рисунок 18 – Результат выполнения теста 7

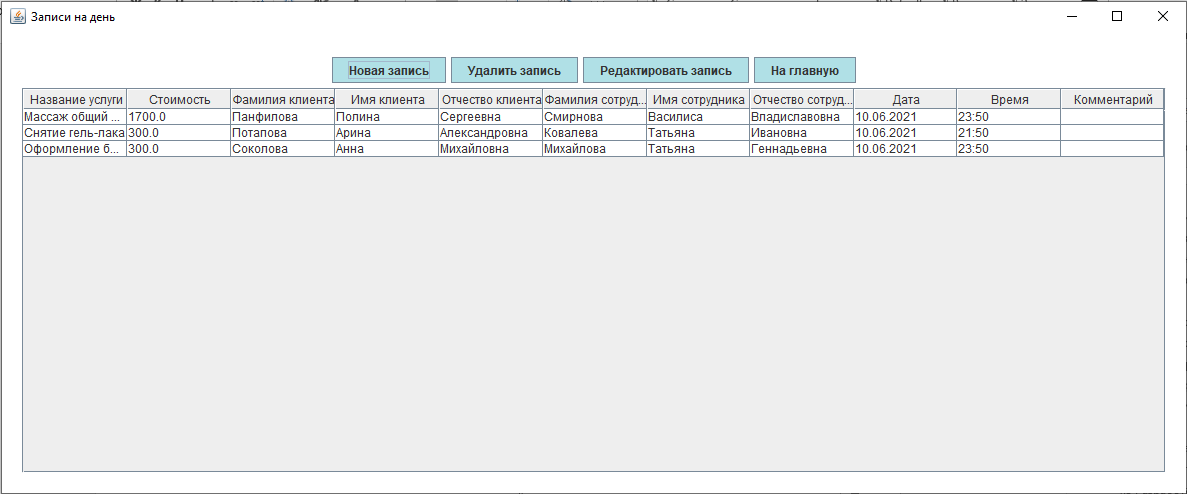


Рисунок 19 – Результат выполнения теста 8



Рисунок 20 – Результат выполнения теста 9

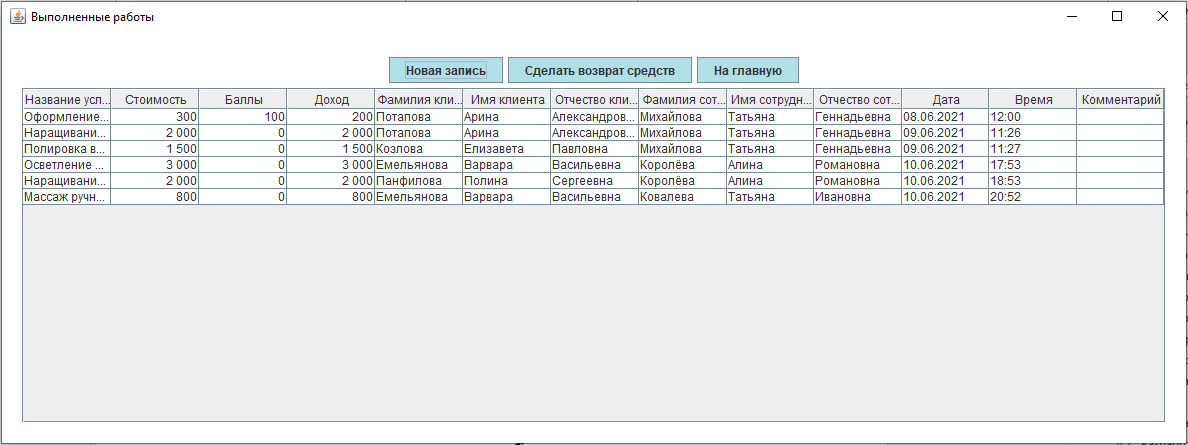


Рисунок 21 – Результат выполнения теста 10

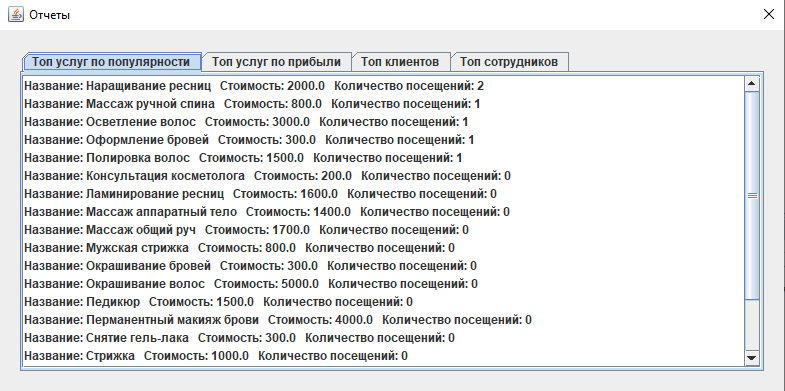


Рисунок 22 – Результат выполнения теста 13

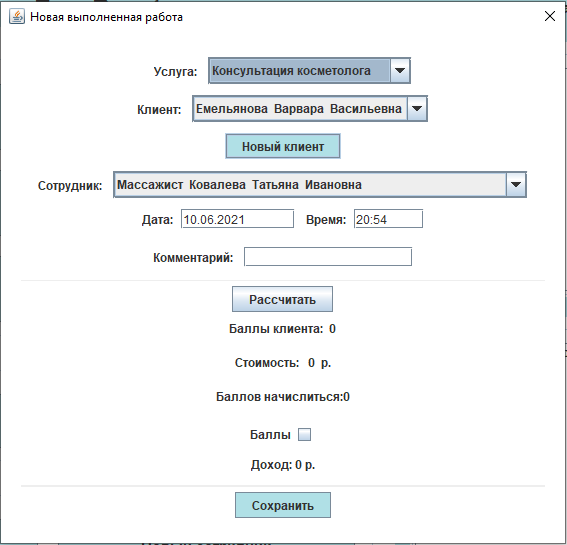


Рисунок 23 – Результат выполнения теста 11

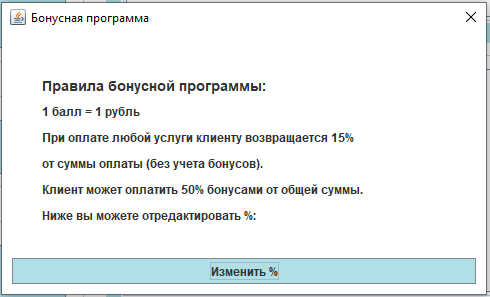


Рисунок 24 – Результат выполнения теста 12

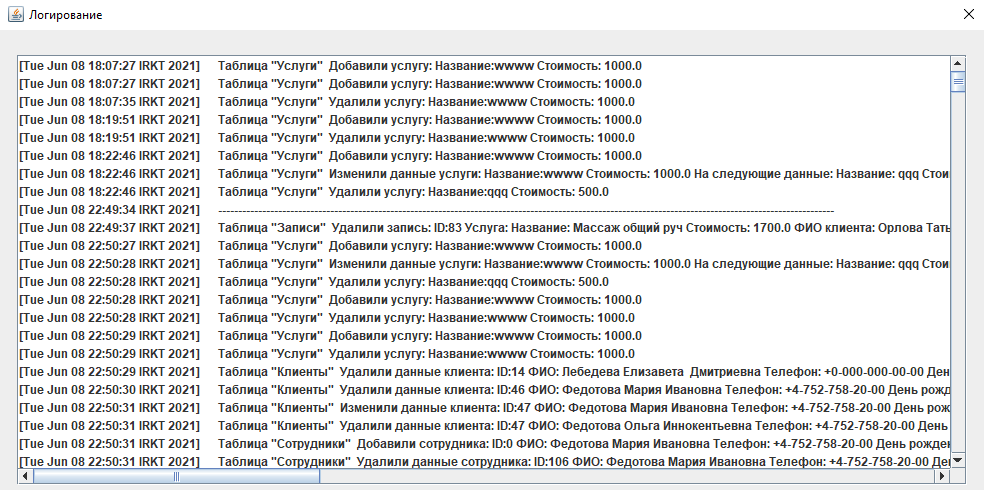


Рисунок 25 – Результат выполнения теста 14

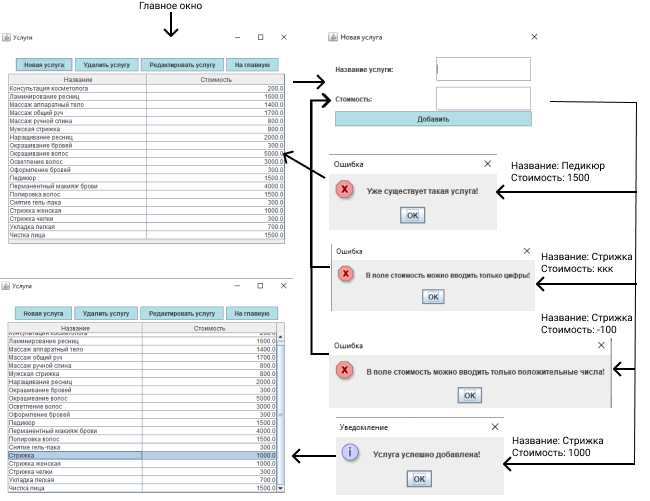


Рисунок 26 – Результат выполнения Сценария №1 Добавление услуги



Рисунок 27 – Результат выполнения Сценария №2 Редактирование услуги

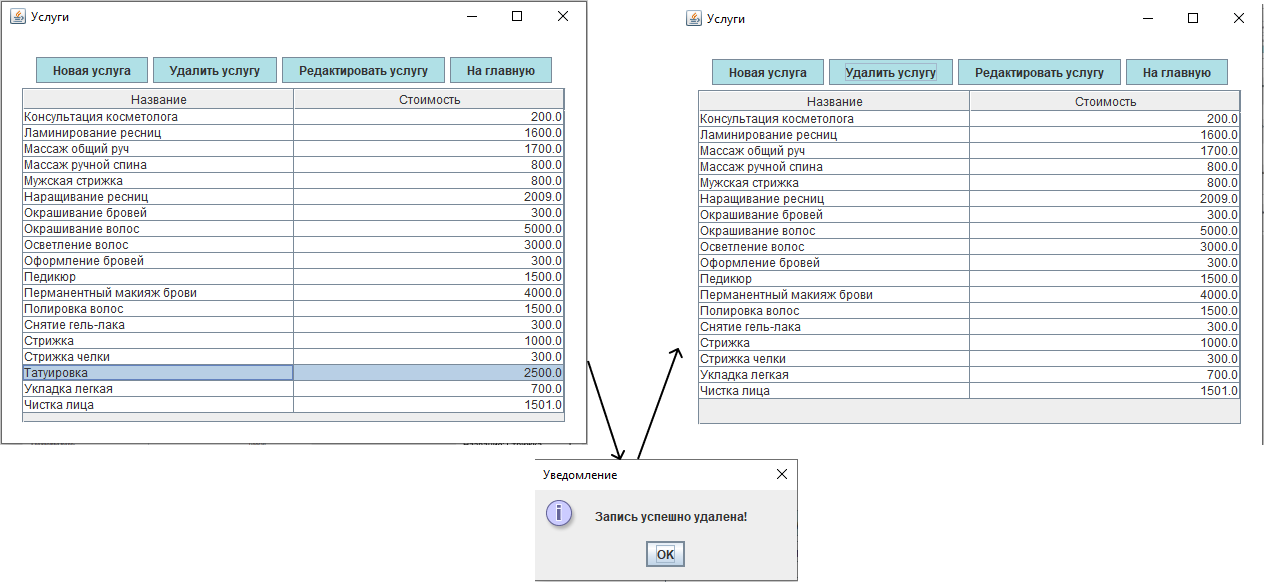


Рисунок 28 – Результат выполнения Сценария №3 Удаление услуги

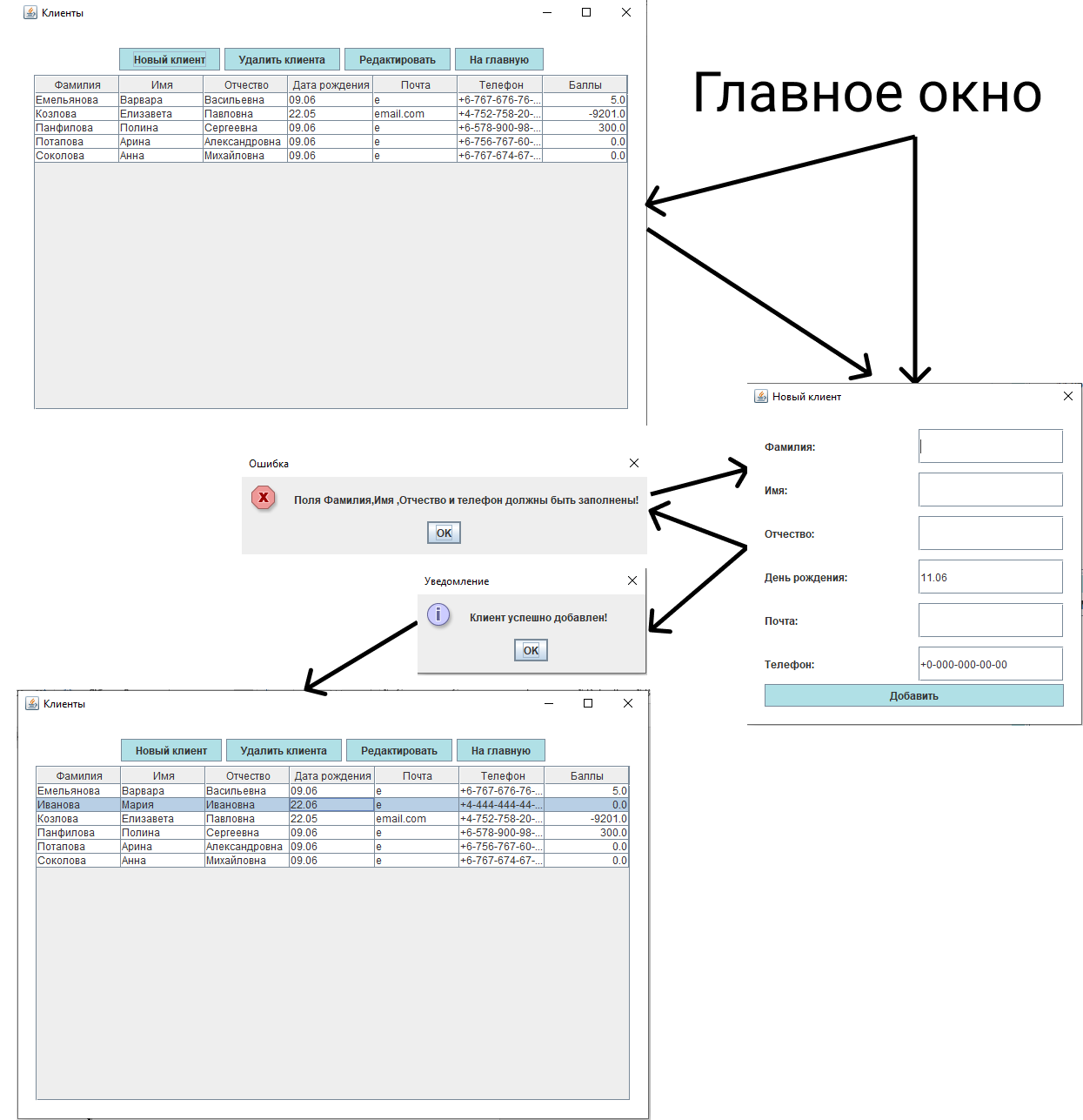


Рисунок 29 – Результат выполнения Сценария №4 Добавление клиента

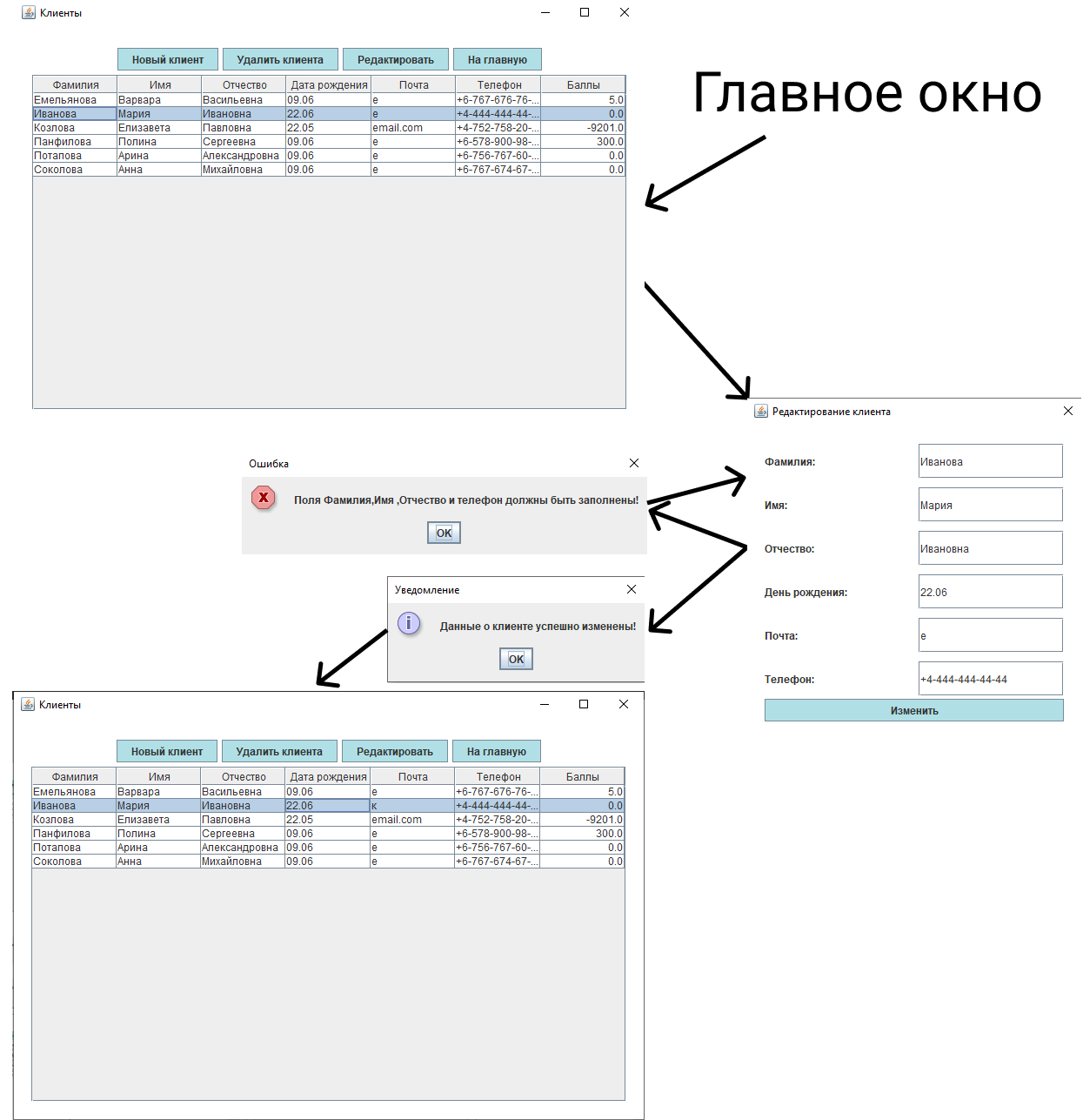


Рисунок 30 – Результат выполнения Сценария №5 Редактирование клиента

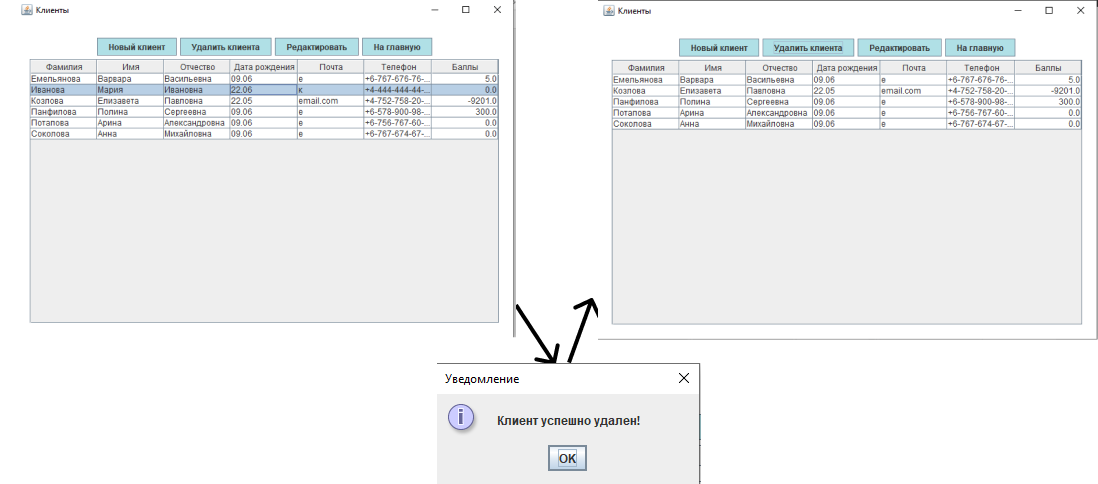


Рисунок 31 – Результат выполнения Сценария №6 Удаление клиента

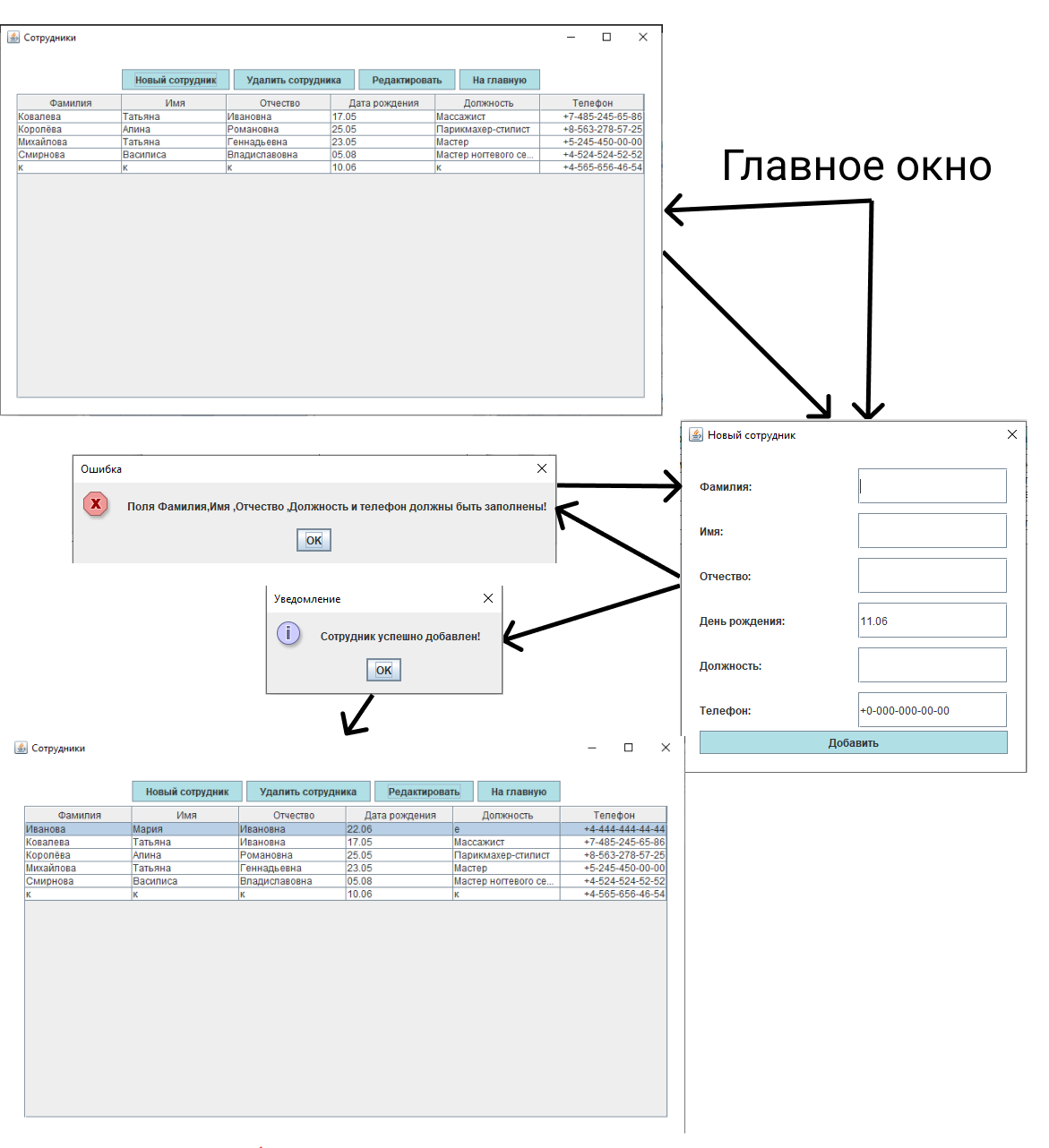


Рисунок 32 – Результат выполнения Сценария №7 Добавление сотрудника

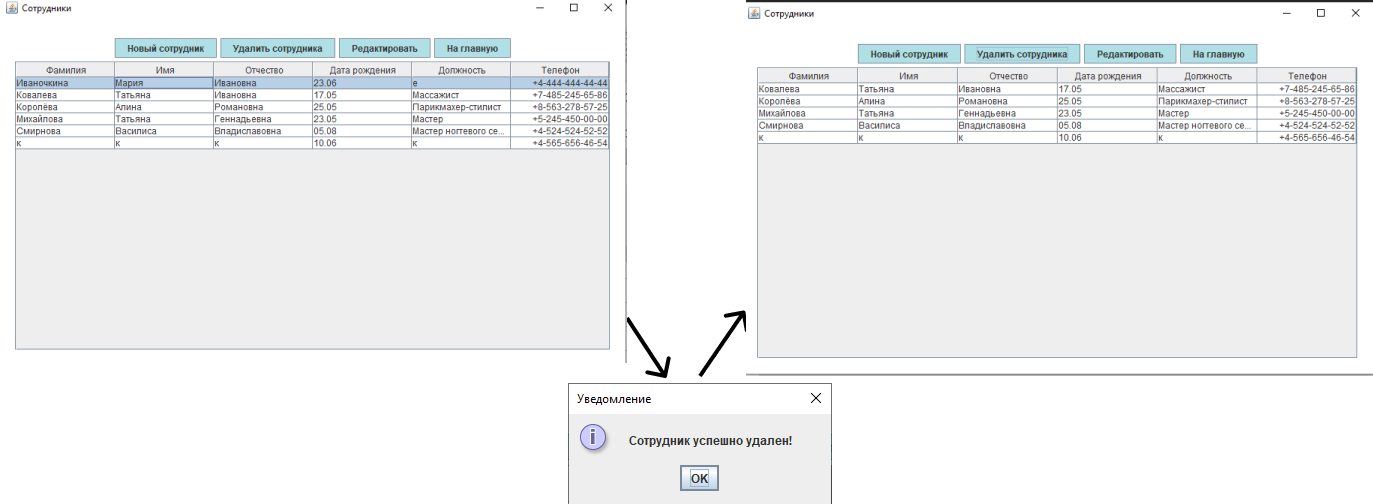


Рисунок 33 – Результат выполнения Сценария №9 Удаление сотрудника

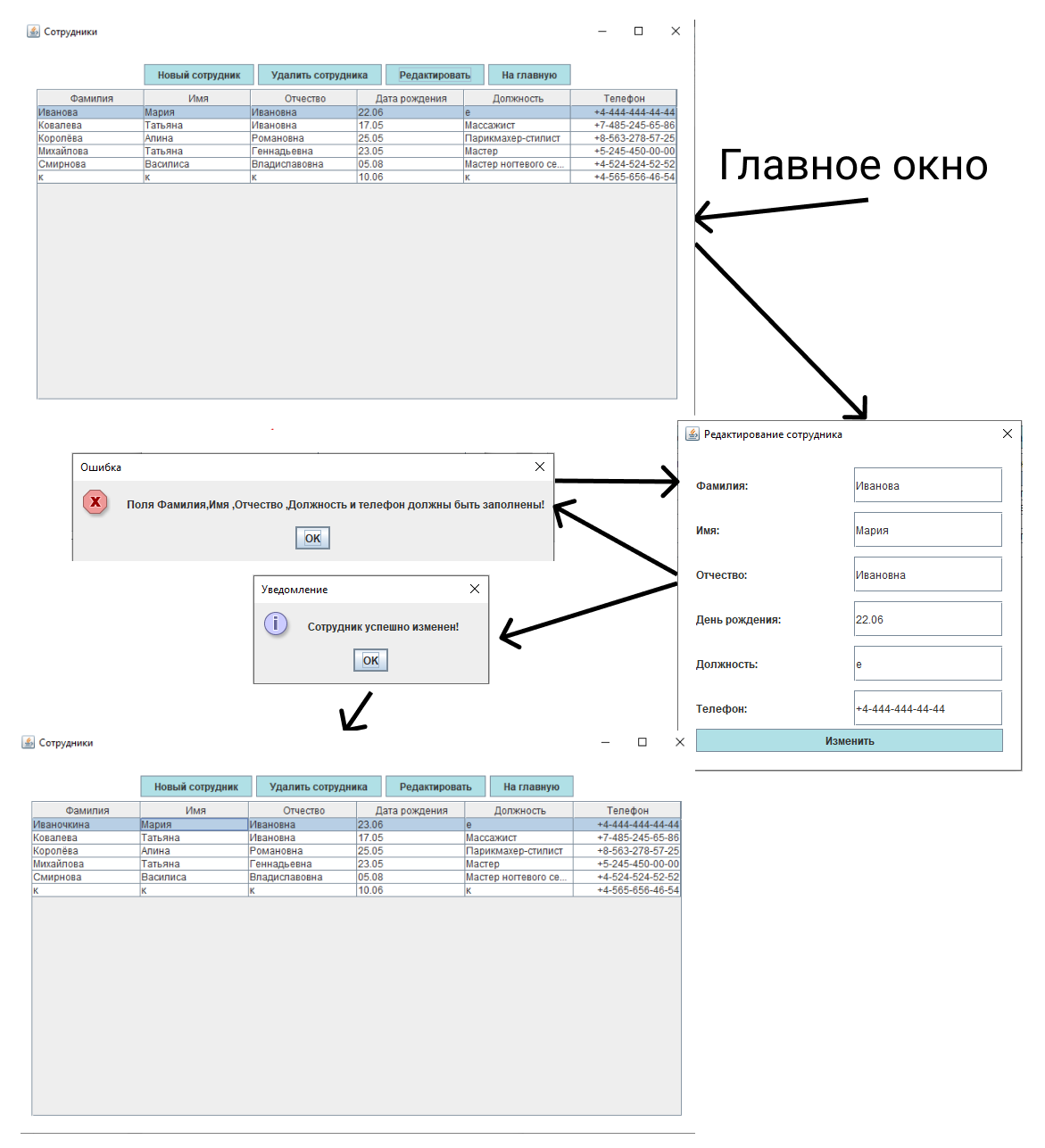


Рисунок 34 – Результат выполнения Сценария №8 Редактирование сотрудника

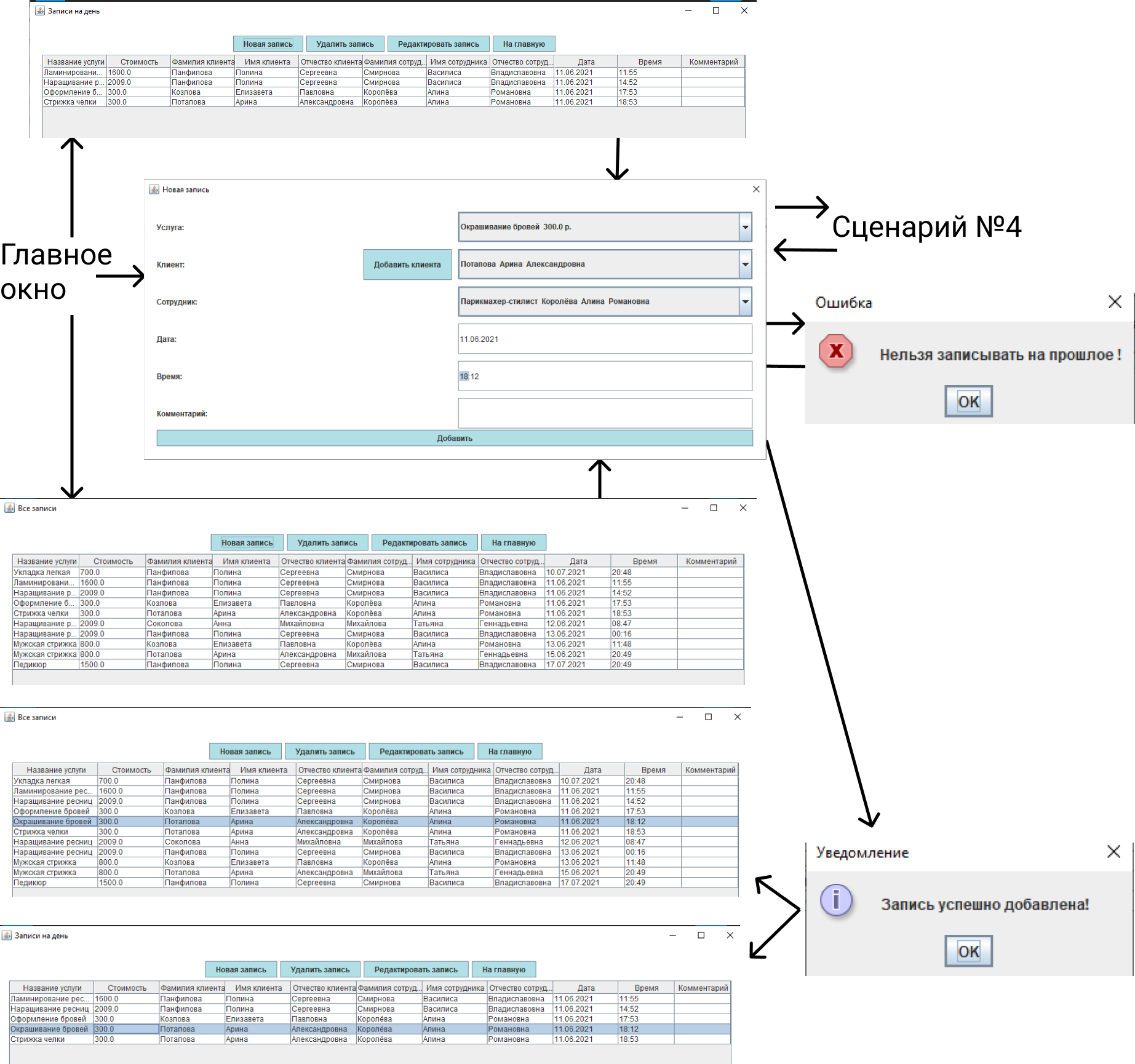


Рисунок 35 – Результат выполнения Сценария №10 Добавление записи

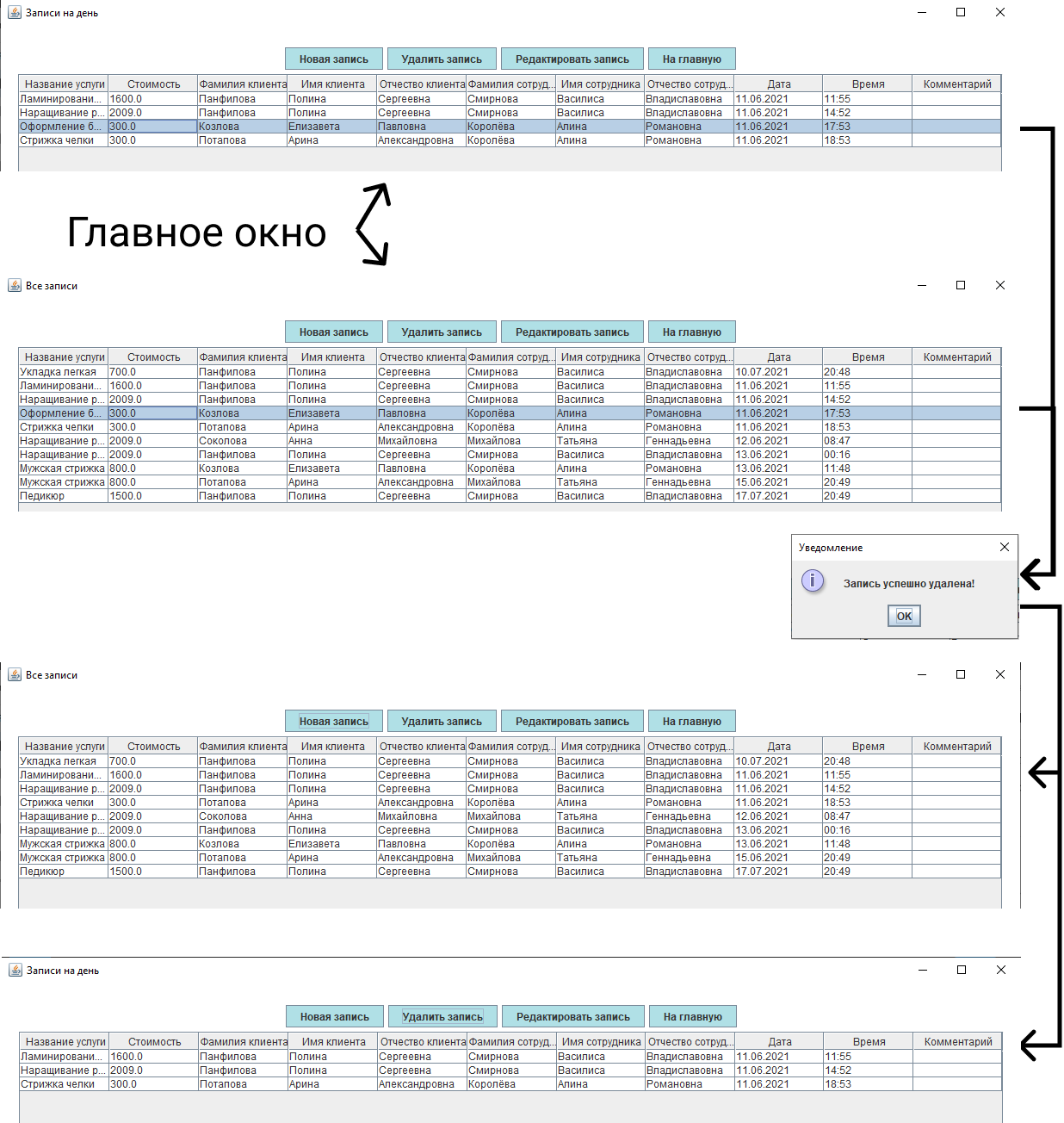


Рисунок 36 – Результат выполнения Сценария №12 Удаление записи

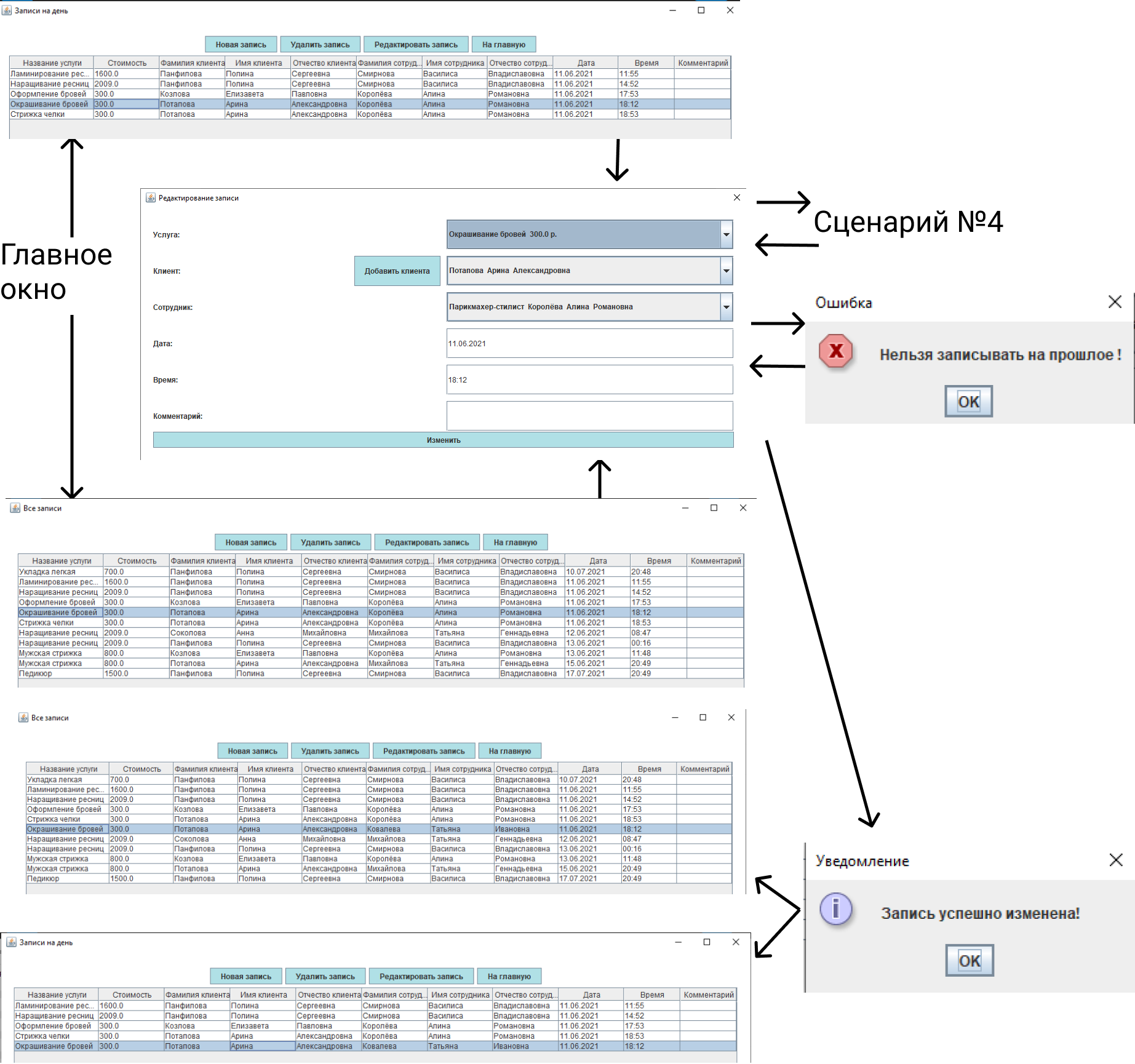


Рисунок 37 – Результат выполнения Сценария №11 Редактирование записи



Рисунок 38 – Результат выполнения Сценария №13 Добавление выполненной работы

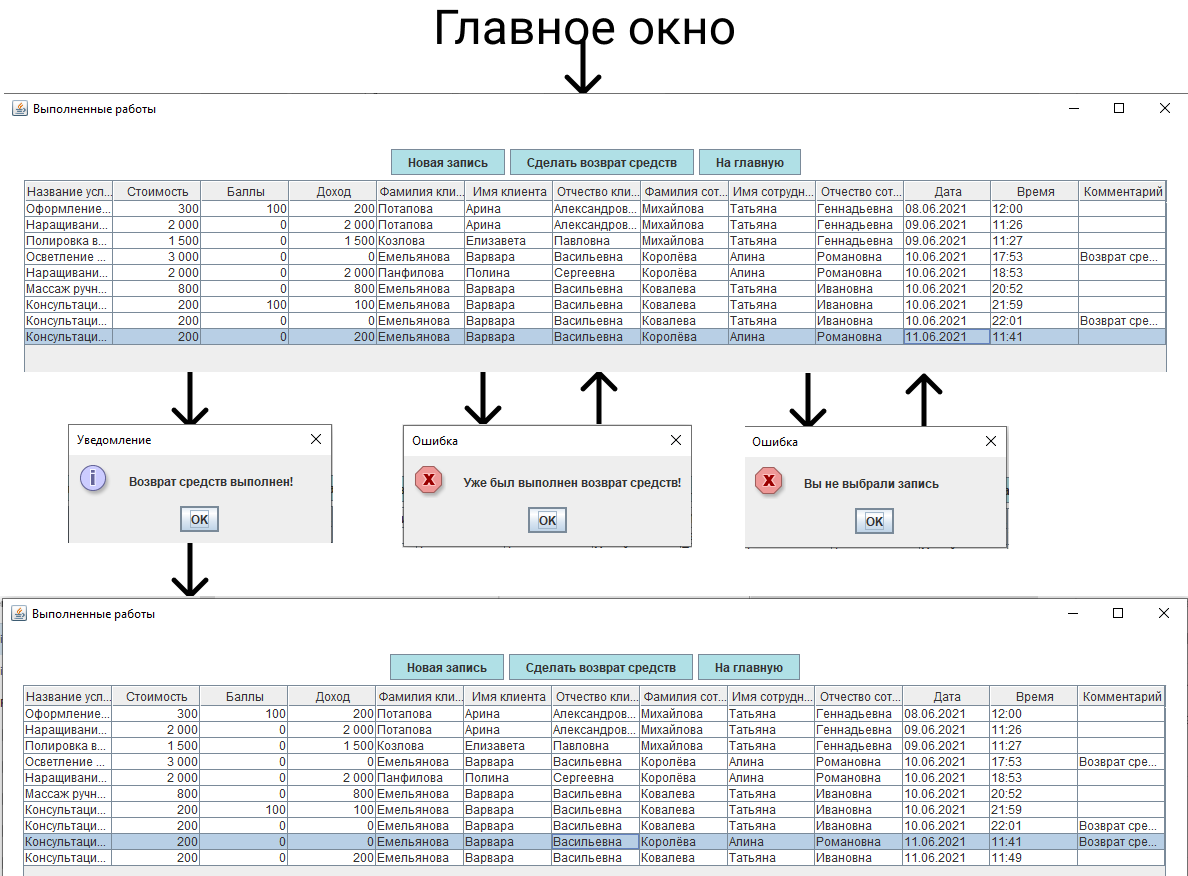


Рисунок 39 – Результат выполнения Сценарий №14 Выполнение возврата средств для выполненной работы.

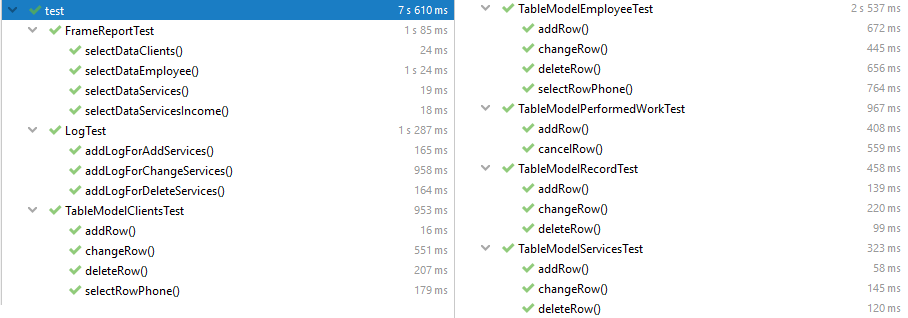


Рисунок 40 – Результат выполнения Unit – тестов 60 – 81

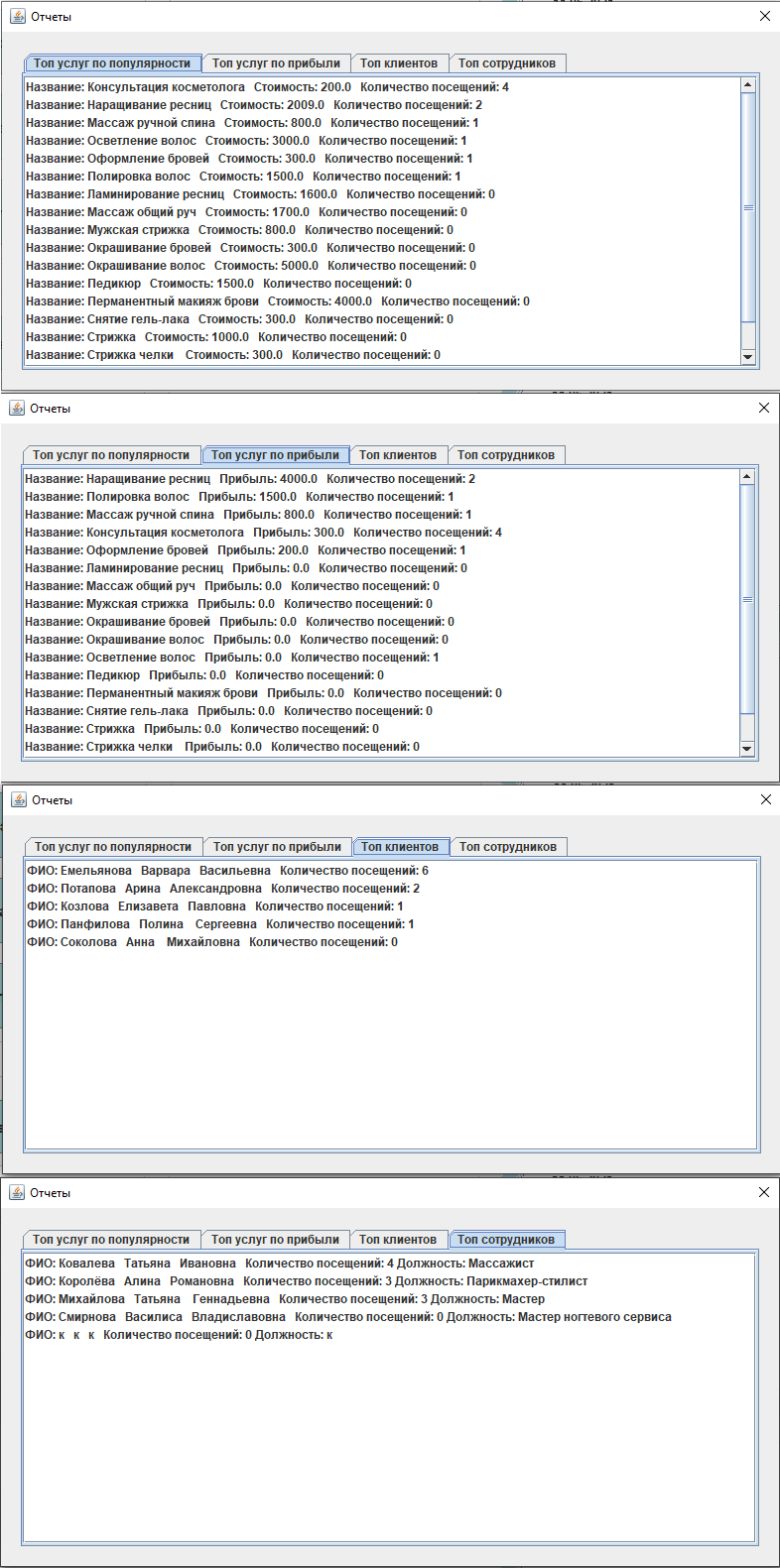


Рисунок 41 – Результат выполнения теста 57



Рисунок 42 – Результат выполнения теста 58 и 59

# Заключение

В процессе разработки данного приложения были получены и закреплены следующие навыки:

1. Проектирования и применения объектного подхода в реализации информационной системы
2. Проектирования пользовательского интерфейса и компоновки элементов взаимодействия с пользователем
3. Создания и проектирования локальной базы данных с помощью СУБД SQLite
4. Программирование Desktop-приложения

Были решены следующие задачи:

1. Выработаны функциональные требования приложения
2. Спроектированы интерфейс, классы и локальная база данных
3. Разработано приложение согласно реализованным требованиям и проектированию
4. Приложение было протестировано с помощью нескольких методов тестирования
5. Составлена документация по всей проделанной работе

Исходный код приложения опубликован в публичном репозитории на сервисе GitHub [8].

Цель курсового проекта достигнута.

Список использованных источников

1. Библиотека Swing [электронный ресурс] // URL: <http://java-online.ru/libs-swing.xhtml> (дата обращения: 17.04.2021)
2. Блог по Java – список статей на JavaRush [электронный ресурс], URL: <https://javarush.ru/groups/posts> (дата обращения: 03.05.2021)
3. Trail: Creating a GUI With JFC/Swing [официальный сайт] , URL: <https://docs.oracle.com/javase/tutorial/uiswing/index.html> (дата обращения: 25.04.2021)
4. Отношения классов — от UML к коду [Электронный ресурс] // Хабр: [сайт]. [2006]. URL: <https://habr.com/ru/post/150041/> (дата обращения: 20.05.2021)
5. Язык программирования Java [Электронный ресурс] // Метанит: [сайт]. [2018]. URL: <https://metanit.com/java/tutorial/> (дата обращения: 15.04.2021)
6. SQLite учебник - ORACLE PL/SQL •MYSQL •MARIADB •SQL SERVER •SQLITE [Электронный ресурс] : [сайт]. URL <https://oracleplsql.ru/sqlite-manual.html>(Дата обращения: 20.04.2021)
7. SQLite JDBC Driver [Электронный ресурс] // GitHub: [сайт]. URL: <https://github.com/xerial/sqlite-jdbc.git> (Дата обращения: 05.05.2021)
8. Репозиторий с исходных кодом проекта [Электронный ресурс] // GitHub: [сайт]. URL<https://github.com/KomogortsevaYulia/BeautyShop_OOP.git>