Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

|  |  |
| --- | --- |
| Факультет | инженерно-экономический |
|  |  |
| Кафедра | экономической информатики |
|  |  |
| Дисциплина «Программирование сетевых приложений» | |

|  |  |
| --- | --- |
|  | *К защите допустить:*  Руководитель курсового проекта  ассистент кафедры ЭИ  M.P. Архиева |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

курсового проекта

на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АНАЛИЗА ДАННЫХ ОБ ОКАЗАННЫХ УСЛУГАХ, ПОСЕЩАЕМОСТИ И ЭФФЕКТИВНОСТИ РАБОТЫ СОТРУДНИКОВ БАРБЕРШОПА**

БГУИР КП 6-05-0611-01 218 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 373601  Хрущев Евгений Сергеевич    (подпись студента) |
|  | Курсовая работа представлена на  проверку . .2025    (подпись студента) |

Минск 2025

**РЕФЕРАТ**

БГУИР КП 6-05-0611-01 171 ПЗ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АНАЛИЗА ДАННЫХ ОБ ОКАЗАННЫХ УСЛУГАХ, ПОСЕЩАЕМОСТИ И ЭФФЕКТИВНОСТИ РАБОТЫ СОТРУДНИКОВ БАРБЕРШОПА : курсовой проект / Е.С. Хрущев. – Минск : БГУИР, 2025, – п.з. – 89 с., чертежей (плакатов) – 5 л. формата А4.

Пояснительная записка 89 с., 41 рис., 6 табл., 11 источников, 3 приложения.

**Ключевые слова**: клиент, организация взаимодействия с пользователями, барбершоп, посещаемость, эффективность работы сотрудников, учет оказанных услуг, веб-приложение, метрика.

*Целью курсового проекта* является повышение качества управления деятельностью барбершопа путем автоматизации процессов учета оказанных услуг, анализа посещаемости и эффективности работы сотрудников.

*Объект исследования* – процесс организации взаимодействия между барбершопом и его клиентами, а также управление работой сотрудников.

*Предмет исследования* – методы и способы сбора, обработки и анализа данных о посещаемости, предоставляемых услугах и результативности работы сотрудников барбершопа.

*Методология проведения работы*: в процессе разработки системы использованы современные подходы к обработке данных, функциональный анализ процессов, принципы построения баз данных, моделирование системы с помощью UML-диаграмм.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку. Программный продукт разработан на языке *Java* с применением *Intellij IDE*.

*Результаты работы*: рассмотрены основные бизнес-процессы предметной области. В результате было создано веб-приложение, которое упрощает учет оказанных услуг, позволяет анализировать посещаемость и эффективность работы сотрудников. Приложение обладает удобным интерфейсом, интуитивно понятно в использовании.

**СОДЕРЖАНИЕ**

[Перечень условных обозначений, символов и терминов 6](#_Toc191739160)

[Введение 6](#_Toc191739161)

[1 Анализ литературных источников и программных решений 8](#_Toc191739162)

[1.1 Описание и анализ предметной области 8](#_Toc191739163)

[1.2 Обзор функциональности аналогов программного средства 10](#_Toc191739164)

[2 Моделирование предметной области и разработка требований к программному средству 13](#_Toc191739165)

[2.1 Анализ и формализация бизнес-процессов предметной области 13](#_Toc191739166)

[2.2 Анализ требований к программному средству и разработка их спецификации 19](#_Toc191739167)

[2.3 Образ предлагаемого решения 22](#_Toc191739168)

[3 Проектирование и разработка программного средства 24](#_Toc191739169)

[3.1 Архитектурные решения и технологии реализации программного средства 24](#_Toc191739170)

[3.2 Проектирование и разработка пользовательского интерфейса 24](#_Toc191739171)

[3.3 Разработка модели данных 24](#_Toc191739172)

[3.4 Описание статических и динамических аспектов поведения программных объектов 25](#_Toc191739173)

[3.5 Разработка и описание алгоритмов, реализующих бизнес-логику программного средства 25](#_Toc191739174)

[3.6 Механизмы обеспечения информационной безопасности 25](#_Toc191739175)

[4 Тестирование и проверка работоспособности программного средства 27](#_Toc191739176)

[5 Руководство по установке (развертыванию) и использованию программного средства 28](#_Toc191739177)

[5.1 Руководство по установке (развертыванию) программного средства 28](#_Toc191739178)

[5.2 Руководство пользователя 28](#_Toc191739179)

[Заключение 29](#_Toc191739180)

[Список использованных источников 29](#_Toc191739181)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 30](#_Toc191739182)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих бизнес-логику приложения 32](#_Toc191739183)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 32](#_Toc191739184)

Ведомость документов курсового проекта 19

# Перечень условных обозначений, символов и терминов

БД (база данных) – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

IDEF – методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов.

Барбершоп – это место, где мужчины могут получить полный комплекс услуг по уходу за внешностью.

Проблема – это еще пока неизвестная основная причина, вызвавшая один или несколько инцидентов.

Учет оказанных услуг – это важнейший процесс, позволяющий организовать систематизированное хранение информации о проведенных процедурах (стрижках, бритье, уходовых процедурах и т.д.).

Таблица Excel – это приложение Windows, предназначенное для обработки электронных таблиц.

Эффективность работы сотрудников – параметр, отражающий результативность деятельности каждого мастера.

Метрика – значение, которое помогает выявить слабые места в работе барбершопа, оценивать эффективность маркетинга и уровень сервиса, а также принимать управленческие решения для повышения прибыли.

CRM-система (Система управления взаимоотношениями с клиентами) – это прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами).

Основная целевая аудитория – это группа людей или компаний, которая имеет непосредственный интерес к продукту или услуге.

BI-системы (сокр. от Business Intelligence) – это программы и инструменты, с помощью которых можно собирать и анализировать информацию о компании.

KPI (Key Performance Indicators) – это ключевые показатели эффективности, используемые для оценки результативности деятельности компании, отдела или отдельного сотрудника.

**YCLIENTS** – одна из самых известных платформ в сфере бьюти-индустрии, предлагающая полный комплекс услуг для автоматизации работы салонов и барбершопов.

# Введение

В современном мире индустрия обслуживания, включая барбершопы, сталкивается с ростом конкуренции и возрастающими ожиданиями клиентов по качеству предоставляемых услуг. Актуальность темы автоматизации анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников обусловлена необходимостью повышения эффективности управления барбершопами, оптимизации внутренних процессов и улучшения взаимодействия с клиентами. В условиях цифровизации и активного внедрения современных технологий организации должны адаптироваться к новым требованиям рынка, чтобы сохранять конкурентоспособность и удовлетворять запросы клиентов.

Целью данного курсового проекта является повышение качества управления деятельностью барбершопа путем автоматизации процессов учета оказанных услуг, анализа посещаемости и эффективности работы сотрудников. Внедрение такой системы позволит минимизировать количество ошибок в учете, оперативно получать информацию о загруженности сотрудников, анализировать популярность услуг и в целом повысить удовлетворенность клиентов. Объект исследования – процесс организации взаимодействия между барбершопом и его клиентами, а также управление работой сотрудников. Предмет исследования – методы и способы сбора, обработки и анализа данных о посещаемости, предоставляемых услугах и результативности работы сотрудников барбершопа.

Чтобы достигнуть поставленной цели потребуется решение следующих задач:

* выполнить обзор и анализ предметной области, а также существующих программных решений;
* осуществить моделирование предметной области и разработать требования к программному средству;
* спроектировать и разработать программное средство;
* протестировать и проверить работоспособность программного средства;
* написать руководство по установке и использованию программного средства.

Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 92,53%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А (рисунок А.1).

# 1 Анализ литературных источников и программных решений

## 1.1 Описание и анализ предметной области

В данном разделе рассматривается предметная область – деятельность барбершопа, связанная с оказанием парикмахерских и сопутствующих услуг, а также анализ данных о работе сотрудников и посещаемости клиентов.

Барбершопы, как предприятия сферы услуг, активно развиваются и конкурируют за привлечение и удержание клиентов. Для обеспечения высокого уровня обслуживания и эффективной работы персонала требуется системный подход к сбору, обработке и анализу данных о предоставляемых услугах, графиках работы мастеров, загрузке рабочих мест и предпочтениях клиентов.

Барбершоп – это не только место, где мужчина станет выглядеть ухоженно и стильно, отгородившись от любопытных глаз и разговоров. Это своеобразный клуб, где можно почувствовать себя комфортно, помимо парикмахерских услуг пообщаться с людьми на интересные темы, а не слушать женские сплетни. Можно закурить сигару, выпить виски, одним словом отдохнуть.

Современные барбершопы – это заведения, где мужчина может снять стресс. Если женщины ходят в салоны красоты, чтобы не просто поухаживать за собой, а релаксировать, пообщаться с мастерами и между собой, то для мужчин такое кажется лишним. Барбершоп в строгом понимании – это парикмахеры, которые стригут, а не болтают. Если в заведении есть журналы, то они мужские. Вообще вся атмосфера – мужская, строгая, даже брутальная. Могут работать и женщины, но они ведут себя более сдержано, чем в женских салонах красоты [1].

Учет оказанных услуг – это важнейший процесс, позволяющий организовать систематизированное хранение информации о проведенных процедурах (стрижках, бритье, уходовых процедурах и т.д.). Такой учет включает регистрацию сведений о клиенте, дате визита, предоставленных услугах, мастере, а также сумме оплаты. Анализ этой информации позволяет выявлять популярные услуги, оценивать работу сотрудников и разрабатывать программы лояльности для клиентов.

Посещаемость барбершопа – один из ключевых показателей эффективности работы предприятия. Она определяется количеством клиентов за определенный период времени и позволяет оценивать популярность заведения, эффективность рекламных кампаний, а также сезонные колебания спроса. Учет посещаемости также дает возможность оптимизировать расписание работы сотрудников и минимизировать простои.

Эффективность работы сотрудников – параметр, отражающий результативность деятельности каждого мастера. Он включает такие показатели, как количество обслуженных клиентов, средний чек, уровень клиентской удовлетворенности и количество возвращающихся клиентов. Анализ этих данных позволяет выявлять сильные стороны персонала, определять потребности в обучении и принимать управленческие решения о мотивации сотрудников.

Традиционные способы учета клиентов и услуг, такие как бумажные журналы или базовые таблицы Excel, уже не соответствуют современным требованиям управления. Они не позволяют быстро обрабатывать большие объемы информации, предоставлять оперативные отчеты, проводить аналитику посещаемости и результативности мастеров. В таблице 1.1 представлены основные метрики, которые используются для анализа эффективности работы барбершопа: среднее количество клиентов в день, среднее время обслуживания клиента, процент возвратных клиентов, средний чек, нагрузка на сотрудника.

Таблица 1.1 – Метрики для анализа эффективности работы барбершопа

|  |  |  |
| --- | --- | --- |
| Показатель | Значение | Возможные проблемы при ручном учете |
| Среднее количество клиентов в день | 30–70 человек | Ошибки в подсчете, потеря данных |
| Среднее время обслуживания клиента | 30–60 минут | Несвоевременное планирование загрузки мастеров |
| Процент возвратных клиентов | 40–60% | Нет достоверной статистики без автоматизации |
| Средний чек | 30–70 руб. | Нет аналитики динамики выручки |
| Нагрузка на сотрудника | 6–10 клиентов в смену | Перегрузка или недогрузка без точного учета |

На сегодняшний день многие барбершопы осуществляют учет вручную, что приводит к следующим проблемам:

* высокая вероятность ошибок при вводе данных;
* потеря или искажение информации о клиентах и услугах;
* невозможность оперативного анализа посещаемости и результатов работы сотрудников;
* затруднения при формировании отчетности для руководства.

Для решения данных проблем необходимо внедрение программного средства, которое позволит автоматизировать процессы учета оказанных услуг, регистрации посещений и анализа эффективности работы сотрудников.

Например, в Минске **более 120 барбершопов**, расположенных во всех районах города. На рисунке 1.1 представлена карта со всеми барбершопами в городе.

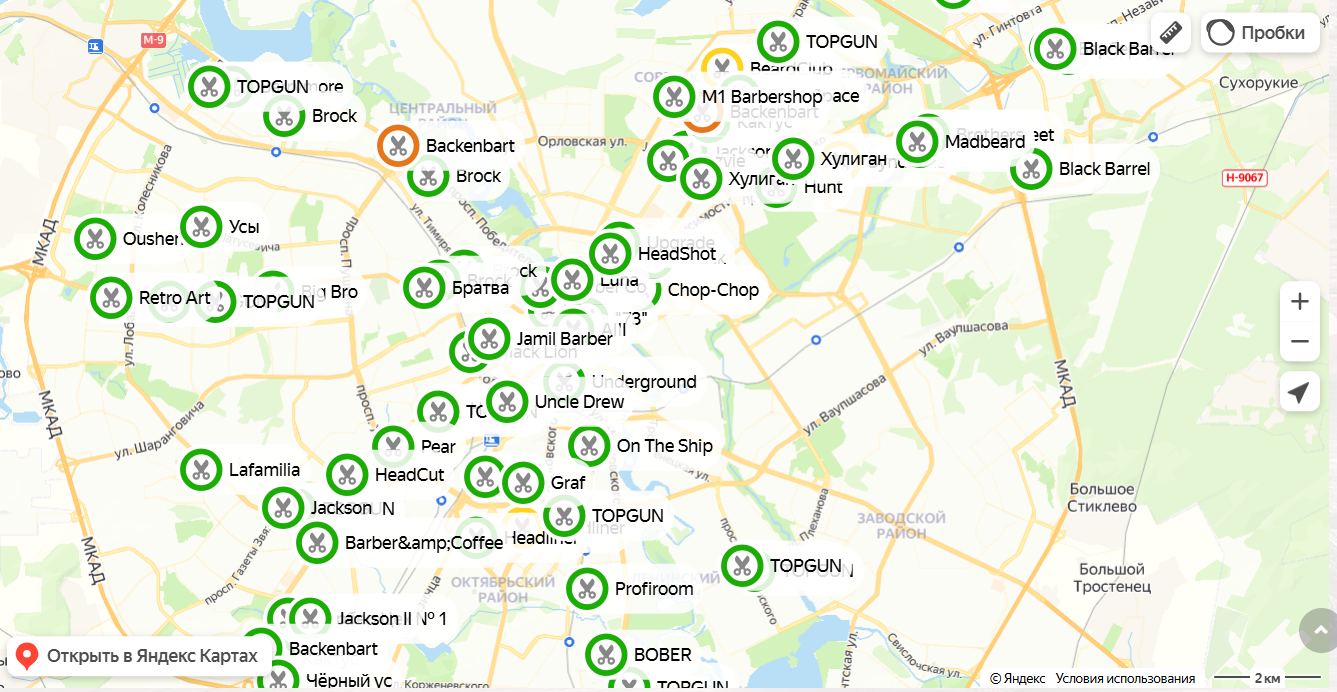


Рисунок 1.1 – Расположение барбершопов в Минске

CRM-системы позволяют записываться в барбершоп через онлайн-форму – на сайте, в мессенджерах или соцсетях. Клиент выбирает услугу, дату, барбера или другого мастера и получает SMS-уведомление о предстоящем визите. Его запись автоматически попадает в электронный журнал. Администратор следит за созданными записями, отвечает на входящие звонки и напоминает клиентам о визитах.

Сценарий разговора с новым клиентом будет отличаться от сценария беседы с уже бывавшим в барбершопе гостем. На каждый случай должен быть свой скрипт – план, шаблон.

Если клиент новый, например, он заполнил форму «Перезвоните мне» на сайте, спросите, какая услуга его интересует. Расскажите, что предлагает наш барбершоп и в чём наши преимущества: скидка при первом посещении, профессиональные барберы, умеющие работать с любой длиной волос и бороды. Резюмируйте договорённости: дату визита, стоимость услуг, имя мастера. В конце разговора спросите, как клиент узнал про наш барбершоп. Это позволит оценить эффективность рекламных каналов. Если клиент уже знаком, достаточно уточнить процедуру, мастера и время записи [2].

Ниже приведена таблица 1.2 с описанием ситуаций, которые могут возникнуть при записи клиентов. Оптимально для каждого такого случая создать скрипт. Очевидно, что администратор должен быть в курсе цен, предоставляемых услуг и специализации мастеров.

Таблица 1.2 – Запись клиента: часто встречающиеся ситуации

|  |  |
| --- | --- |
| Ситуация | Решение |
| Клиент не уверен, подходит ли ему барбершоп/услуга/мастер | Пригласите клиента в барбершоп, чтобы он смог оценить сервис и познакомиться с мастерами |
| Нет желаемого времени для записи | Предложите другое время или запишите клиента в лист ожидания |
| У вас нет услуги или косметического средства, которое нужно клиенту | Предложите альтернативу самостоятельно или назначьте консультацию с мастером |
| Клиент хочет записаться на услугу, но его смущает цена | Предложите более доступную альтернативу, уточните, есть ли в барбершопе скидки, расскажите, чем обусловлена цена |
| Клиент сомневается в качестве конкретной услуги | Оперируйте фактами: сколько клиентов получили услугу и остались довольны, какие сертификаты и лицензии есть у мастера, какие гарантии вы предоставляете |
| Клиент прежде обслуживался и остался недоволен, но хочет попробовать вновь | Выразите сожаление и уточните, в чём была проблема. Оцените, насколько реально сейчас изменить мнение клиента, есть ли на это ресурсы. Если такая возможность есть, предложите записаться |

Барбершоп отличается от обычной парикмахерской не только перечнем услуг, но и особым стилем оформления, продвижением и ценообразованием. Такое заведение не просто место, где можно привести себя в порядок, но и клуб по интересам. Клиент может обсудить любые проблемы с барбером, посоветоваться или посмотреть футбольный матч. Барбершопы являются элитными заведениями, соответственно цены на простую стрижку могут быть выше, чем в обычной парикмахерской на 40-45%. Однако этот факт не влияет на частоту и готовность мужчин посещать заведение, так как большая часть посетителей является состоятельными людьми.

Целевая аудитория барбершопов – мужчины в возрасте от 26 до 45 лет с уровнем доходов от 2000 рублей. Как правило, это люди, ведущие активный образ жизни, готовые к переменам и уделяющие внимание своей внешности. Они готовы воспользоваться услугами барбера раз в 2 недели и заплатить сумму от 50 до 150 рублей. Люди старшего поколения (46-55 лет) реже пользуются услугами барберов, они более консервативны и не готовы к переменам. Как правило, это руководители среднего и высшего звена с уровнем дохода от 3000 рублей в месяц. Они не так активно пользуются интернетом, а информацию получают от друзей и знакомых. Однако они готовы тратить большие суммы, средний чек в этом сегменте – от 200 рублей.

Еще один сегмент – молодые замужние женщины. Именно они чаще всего покупают подарочные сертификаты и ВИП-абонементы для супругов, а также приводят подстричься детей. Доля этой категории невелика – порядка 1-2%. На рисунке 1.2 представлена диаграмма возрастной структуры посетителей барбершопа.

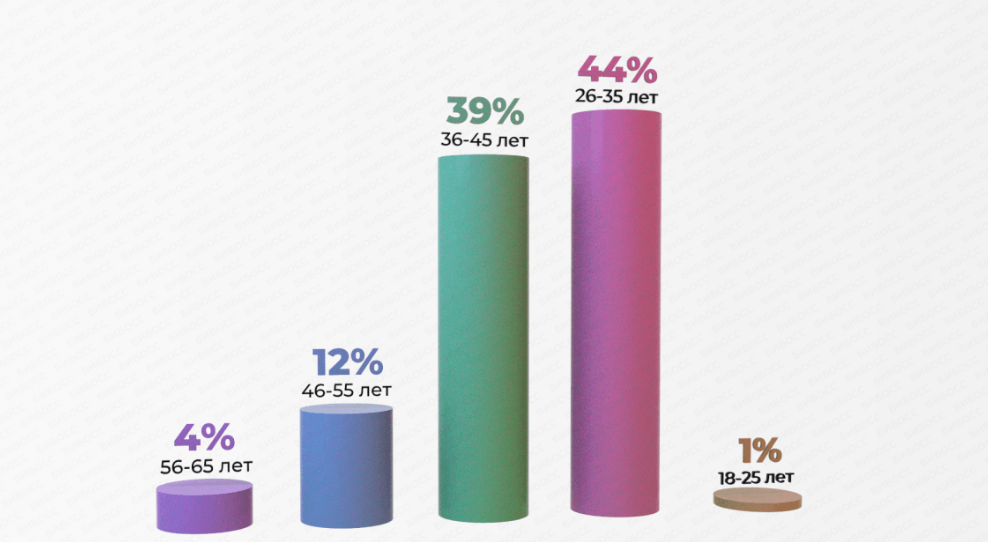


Рисунок 1.2 – Возрастная структура посетителей барбершопа

Для принятия управленческих решений барбершопы используют BI-инструменты (Power BI, Tableau, Google Data Studio). Примеры применения:

* **прогнозирование спроса (**анализ сезонности: пиковые дни/часы, влияние праздников, прогноз загрузки мастеров на основе исторических данных**);**
* **анализ эффективности услуг (**рейтинг популярных услуг (например, стрижка и бритье), выявление убыточных или низкомаржинальных процедур**);**
* **оценка работы мастеров (**средний чек, количество клиентов в смену, процент возвратов, сравнение показателей между филиалами (для сетевых барбершопов)**);**
* **маркетинговая аналитика (**ROI рекламных каналов (соцсети, контекстная реклама), сегментация клиентов по частоте посещений и среднему чеку**);**
* **пример дашборда в Power BI** (визуализация динамики выручки по неделям, тепловая карта загрузки мастеров по времени суток);
* график возвратных клиентов и новых.

**Преимущества**:

* автоматизация отчетности (экономия времени);
* гибкость: настройка под конкретные KPI бизнеса;
* возможность интеграции с CRM и системами онлайн-записи.

Эти технологии не только упрощают рутинные операции, но и помогают барбершопам оставаться конкурентоспособными в условиях растущего рынка.

Таким образом, автоматизация процессов учета данных о клиентах, услугах и работе сотрудников является необходимым шагом для повышения эффективности деятельности барбершопа, повышения уровня клиентского сервиса и оптимизации управления предприятием.

## 1.2 Обзор функциональности аналогов программного средства

На рынке существуют различные программные решения для автоматизации учета оказанных услуг, анализа посещаемости и оценки эффективности работы сотрудников в индустрии красоты. Рассмотрим несколько популярных аналогов, их функциональность, а также их преимущества и недостатки применительно к деятельности барбершопа.

**YCLIENTS** – одна из самых известных платформ в сфере бьюти-индустрии. Она предлагает полный комплекс услуг для автоматизации работы салонов и барбершопов: запись клиентов онлайн, учет визитов, формирование отчетов о работе сотрудников, контроль финансов, управление складом и маркетинговые инструменты. Преимущества: широкий функционал, интеграция с сайтами и соцсетями, мобильные приложения для администраторов и клиентов. Недостатки: высокая стоимость лицензии при расширении функционала, сложность освоения для небольших команд. На рисунке 1.3 изображена главная страница платформы.

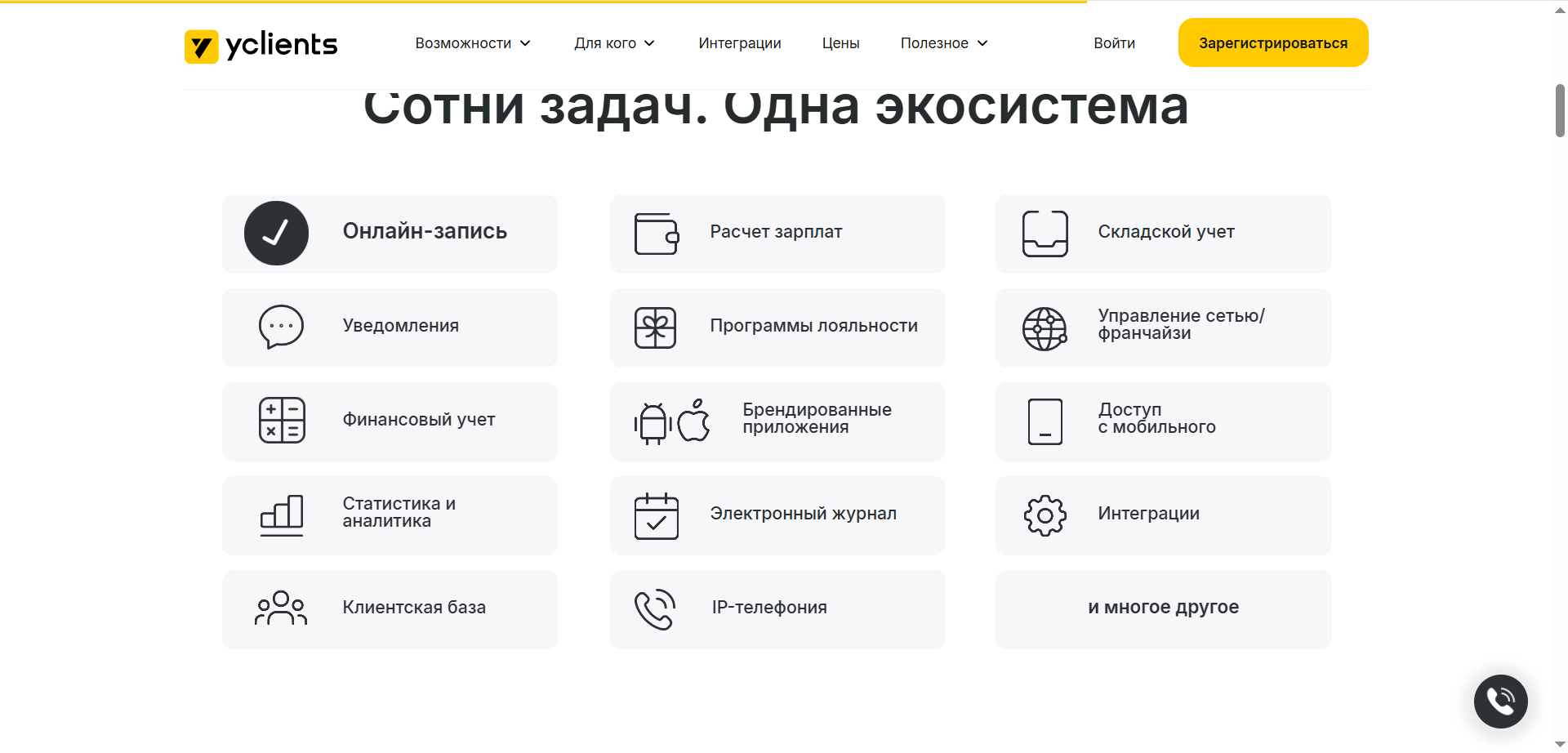


Рисунок 1.3 – Главная страница платформы YCLIENTS

SimpleSpa – облачное решение для автоматизации салонов и барбершопов, акцентирующее внимание на удобстве работы с расписанием мастеров, онлайн-записи клиентов и расчетах эффективности сотрудников.  
Преимущества: интуитивно понятный интерфейс, быстрый запуск системы, доступность мобильных версий. Недостатки: ограниченный функционал для проведения глубокого анализа данных и построения расширенной отчетности. На рисунке 1.4 изображена главная страница сайта SimpleSpa.

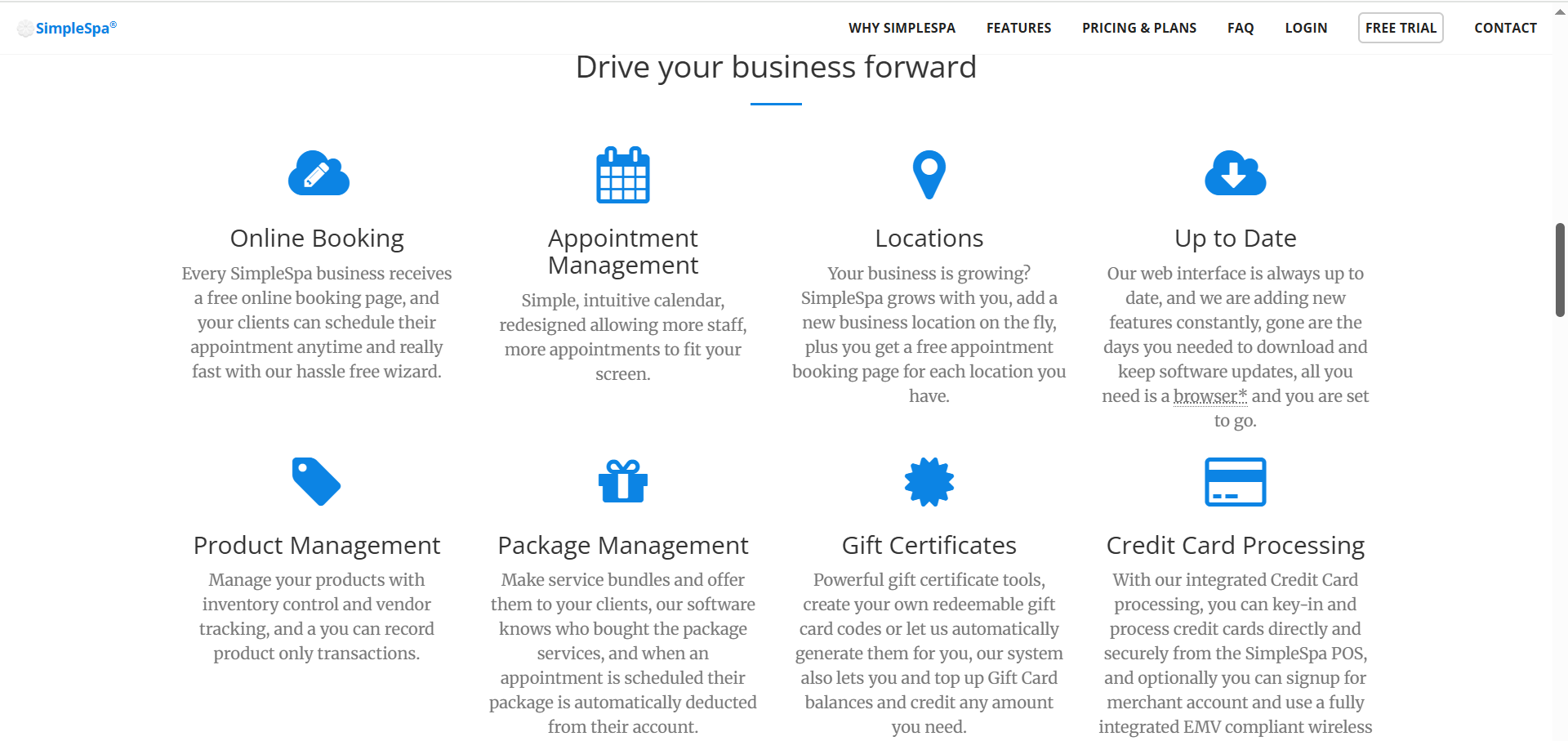


Рисунок 1.4 – Главная страница платформы SimpleSpa

В таблице 1.3 представлено сравнение моей системы, SimpleSpa и YCLIENTS.

Таблица 1.3 – Сравнение систем

|  |  |  |  |
| --- | --- | --- | --- |
| Функционал | YCLIENTS | SimpleSpa | Разрабатываемое ПО |
| Учет оказанных услуг | Да | Да | Да |
| Анализ посещаемости | Да | Да | Да |
| Оценка эффективности работы сотрудников | Да | Частично | Да |
| Генерация подробных аналитических отчетов | Да | Ограничено | Да |
| Удобство интерфейса | Высокое | Высокое | Высокое |
| Возможность кастомизации | Средняя | Низкая | Высокая |
| Стоимость | Высокая | Средняя | Доступная |

Существующие решения предлагают широкий набор функций, однако они часто избыточны для небольших барбершопов или не обеспечивают необходимую гибкость настройки под конкретные бизнес-процессы. Разрабатываемое программное средство будет ориентировано на потребности небольших и средних барбершопов: упрощённый интерфейс, полный контроль над учетными и аналитическими процессами и возможность оперативной настройки без значительных затрат.

В данном разделе было проведено сравнение трех систем автоматизации учета оказанных услуг, посещаемости и эффективности работы сотрудников: YCLIENTS, SimpleSpa и разрабатываемого нами программного средства. Каждая из этих систем обладает уникальным набором функций и характеристик, что позволяет барбершопам выбирать наиболее подходящее решение в зависимости от своих потребностей и масштаба бизнеса.

Платформа YCLIENTS отличается широкой функциональностью и возможностью интеграции с различными внешними сервисами, что делает её привлекательной для крупных барбершопов. SimpleSpa, напротив, ориентировано на быстрое внедрение и удобство работы, однако имеет ограниченные возможности по глубокой аналитике данных. Разрабатываемое нами программное средство нацелено на упрощение процессов учета и аналитики, удобство использования и доступность, что делает его особенно подходящим для небольших и средних барбершопов, которые хотят минимизировать затраты на обучение сотрудников.

Таким образом, выбор между этими системами должен основываться на реальных задачах предприятия, финансовых возможностях и уровне готовности к цифровой трансформации. Каждое из решений имеет свои преимущества и ограничения, и правильный выбор программного средства способен существенно повысить эффективность работы барбершопа, качество обслуживания клиентов и конкурентоспособность на рынке.

# 2 Моделирование предметной области и разработка требований к программному средству

## 2.1 Анализ и формализация бизнес-процессов предметной области

Формализованность и регламентированность бизнес-процессов на сегодняшний день является одним из ключевых факторов успеха любого предприятия. Описание и оптимизация бизнес-процессов прямым образом влияет на операционную и стратегическую эффективность, что в итоге влияет на показатели прибыльности и рейтинги [3].

Рассмотрим детально процесс анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа и построим для него диаграмму в нотации IDEF0. Диаграмма контекстного уровня представлена на рисунке 2.1.

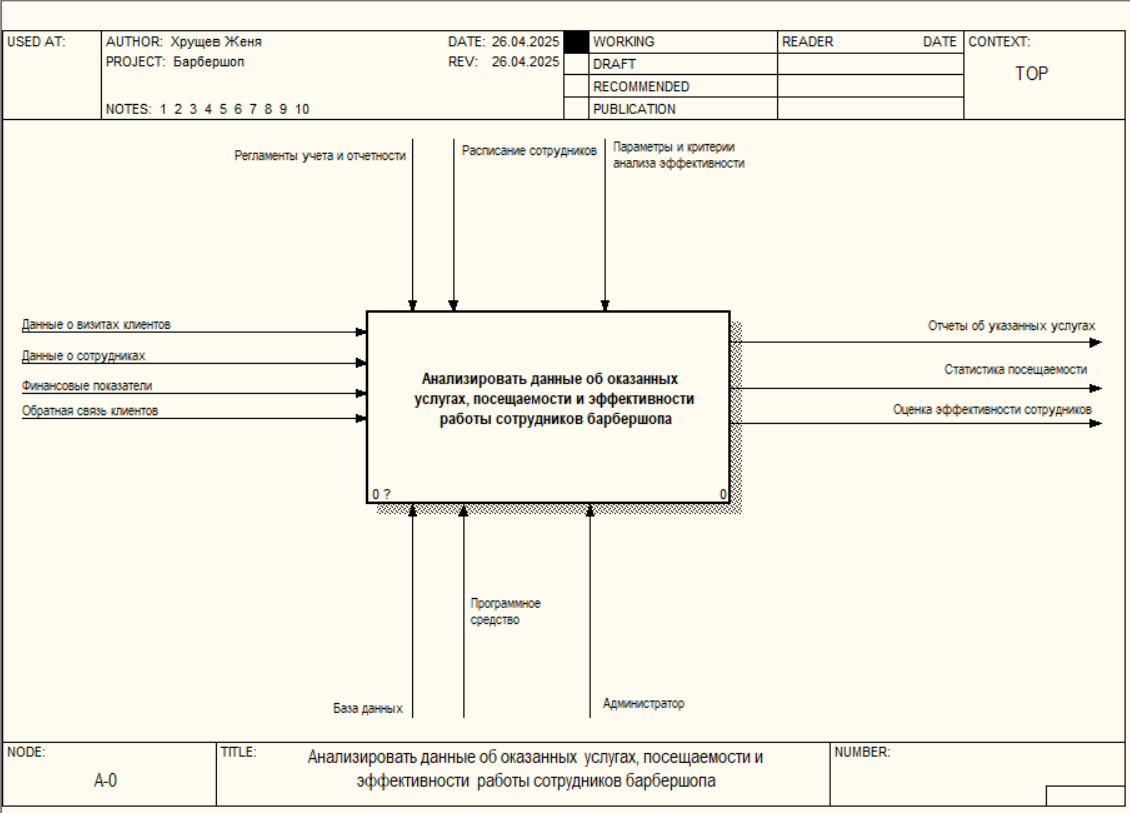


Рисунок 2.1 – Контекстная диаграмма модели «AS-IS»

В качестве входных данных используются данные о визитах клиентов, данные о сотрудниках, финансовые показатели, обратная связь клиентов. Как результат должны быть получены отчеты об указанных услугах, статистика посещаемости, оценка эффективности сотрудников. Работа осуществляется администратором, базой данной и программным средством, в своей работе они должны руководствоваться регламентом учета и отчетности, параметры и критерии анализа эффективности, расписание работы сотрудников.

Для детального рассмотрения была проведена декомпозиция контекстного уровня, в результате чего процесс был разбит ещё на четыре подпроцесса (рисунок 2.2).

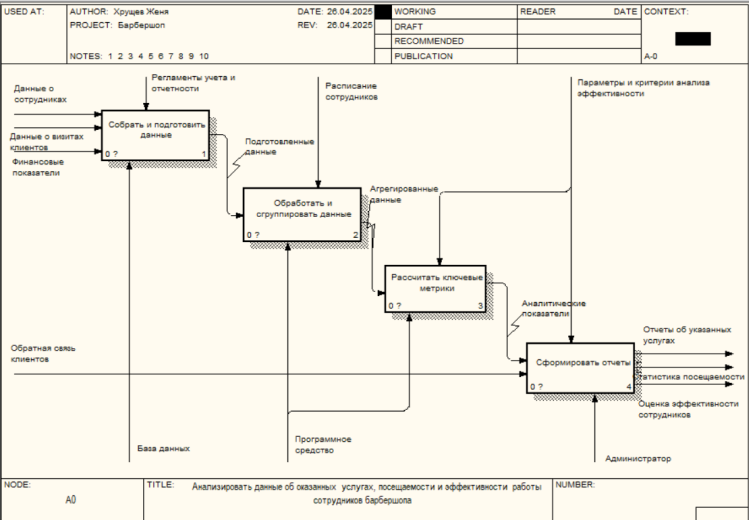


Рисунок 2.2 – Декомпозиция контекстного уровня модели «AS-IS»

В качестве подпроцессов были выделены следующие:

* собрать и подготовить данные;
* обработать и сгруппировать данные;
* рассчитать ключевые метрики;
* сформировать отчеты.

Все работы по полученным подпроцессам выполняются в основном программным средством, на первом и последнем шаге к нему присоединяются базой данной и администратором. Первый подпроцесс заключается в сборе необходимых данных и их подготовке, другими словами в базу данных заносятся данные, получая подготовленные данные для обработки, руководствуясь регламентом учета и отчестности.

В свою очередь этот подпроцесс может быть разбит еще на несколько этапов:

* извлечь данные;
* очистить данные;
* актуализировать расписание сотрудников;
* формировать наборы данных.

Декомпозиция подпроцесса «Собрать и подготовить данные» приведена на рисунке 2.3.

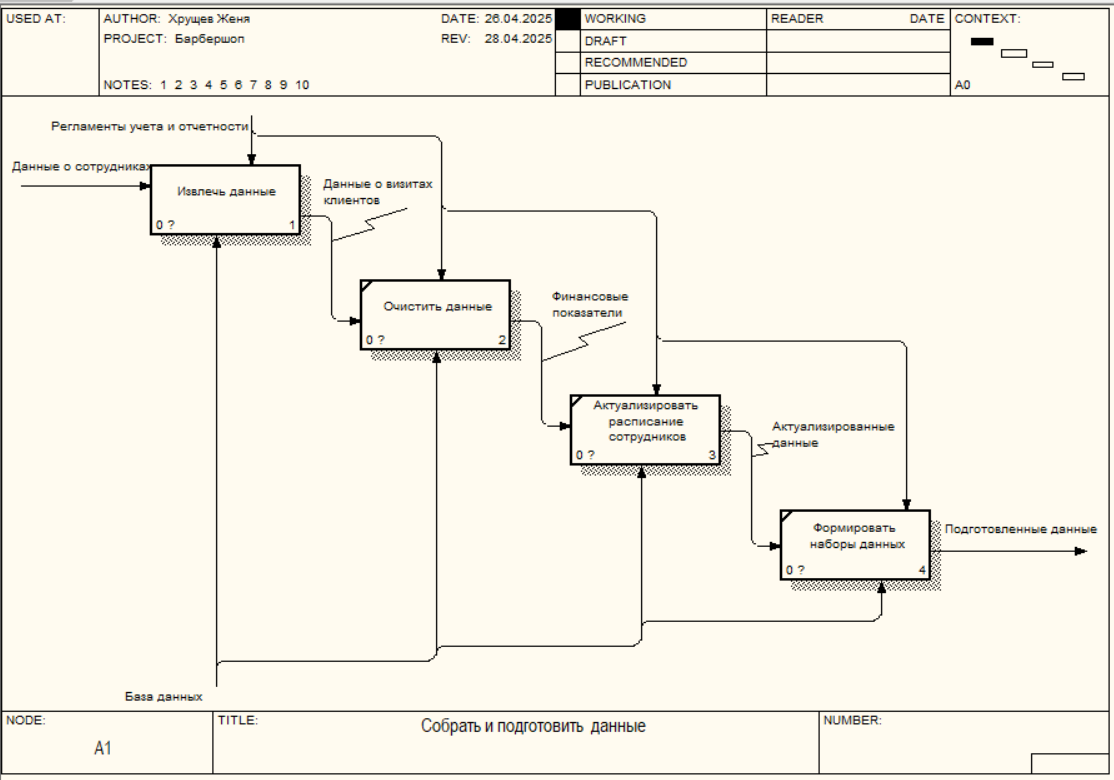


Рисунок 2.3 – Декомпозиция подпроцесса «Собрать и подготовить данные» модели «AS-IS»

Процесс начинается со сбора и учета информации, на котором осуществляется сбор данных от сотрудников. Затем происходит сбор информации о работниках, что позволяет получить необходимые данные. На этапе очистки данных производится проверка и удаление некорректных или лишних записей. После этого формируются основные показатели, которые будут использоваться в дальнейшем анализе. Следующий шаг – актуализация набора данных, где обновляются и корректируются собранные данные для обеспечения их актуальности. Завершающим этапом является формирование окончательного набора данных, готового для дальнейшего использования.

Этот подпроцесс может быть также декомпозирован на несколько шагов:

* собрать данных;
* анализировать и сформировать показатели;
* сравнить данные;
* сформировать окончательные наборы.

Декомпозиция подпроцесса «Извлечь данные» приведена на рисунке 2.4.

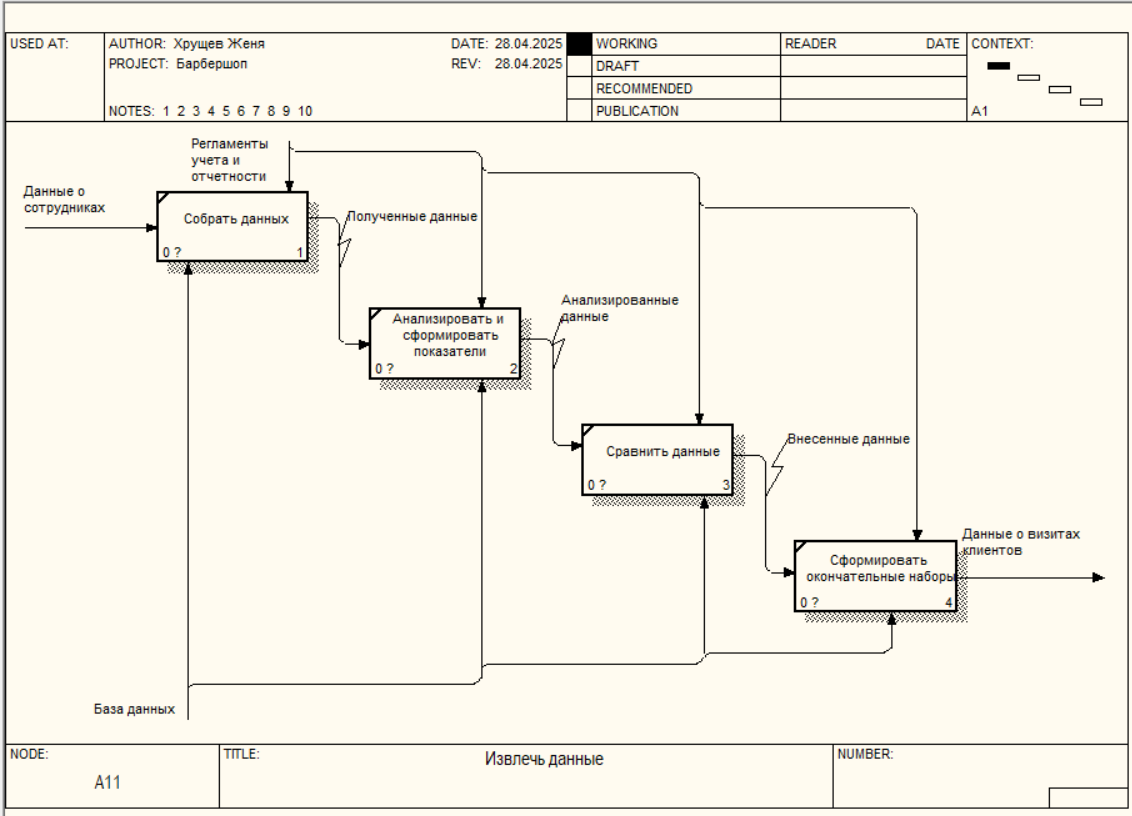


Рисунок 2.4 – Декомпозиция подпроцесса «Вносить учетные данные» модели «AS-IS»

Подпроцесс «Анализировать и сформировать показатели» может быть также декомпозирован на несколько шагов:

* получить информацию;
* очистить и подготовить данные;
* рассчитать ключевые показатели;
* визуализировать данные.

Декомпозиция подпроцесса «Анализировать и сформировать показатели» приведена на рисунке 2.5.

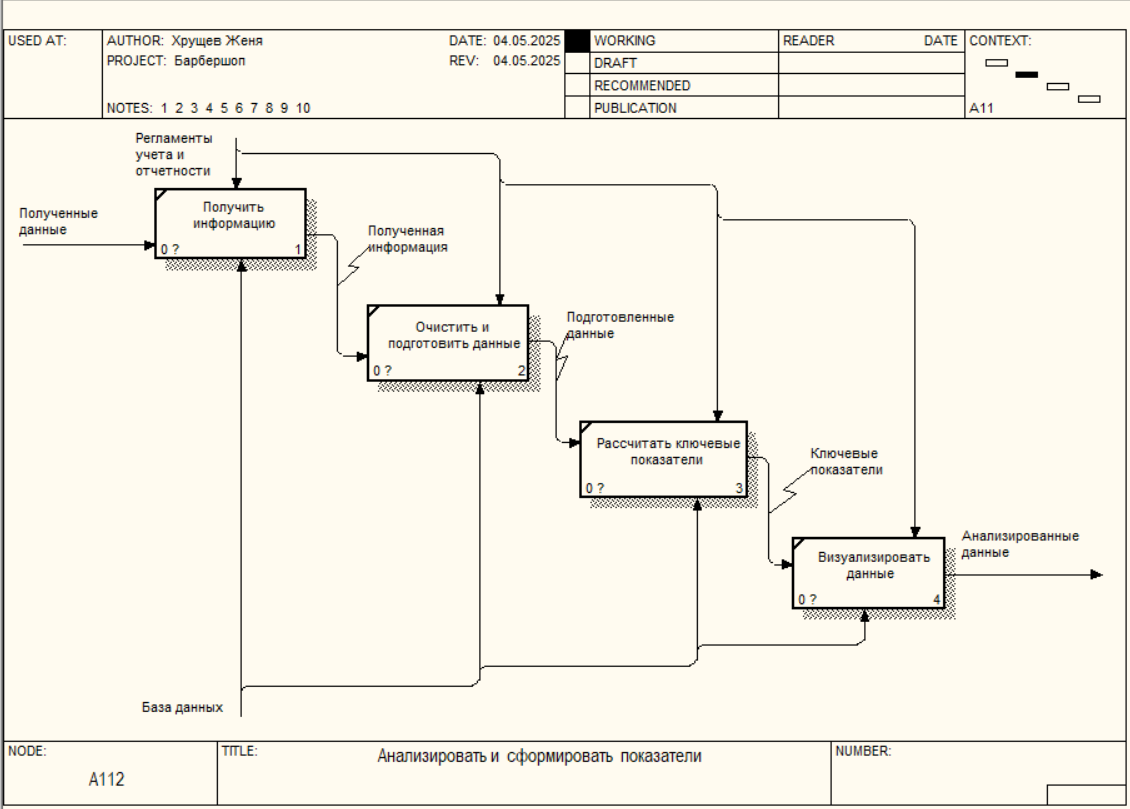


Рисунок 2.5 – Декомпозиция подпроцесса «Анализировать и сформировать показатели» модели «AS-IS»

Следующий подпроцесс – обработать и сгруппировать данные. Классификация данных – сотрудники классифицируют данные по установленным критериям, чтобы упростить дальнейшую обработку. Структурировать по периодам времени – данные организуются на основе четырех времен года, чтобы создать базовую структуру. Декомпозиция этого подпроцесса приведена на рисунке 2.6.

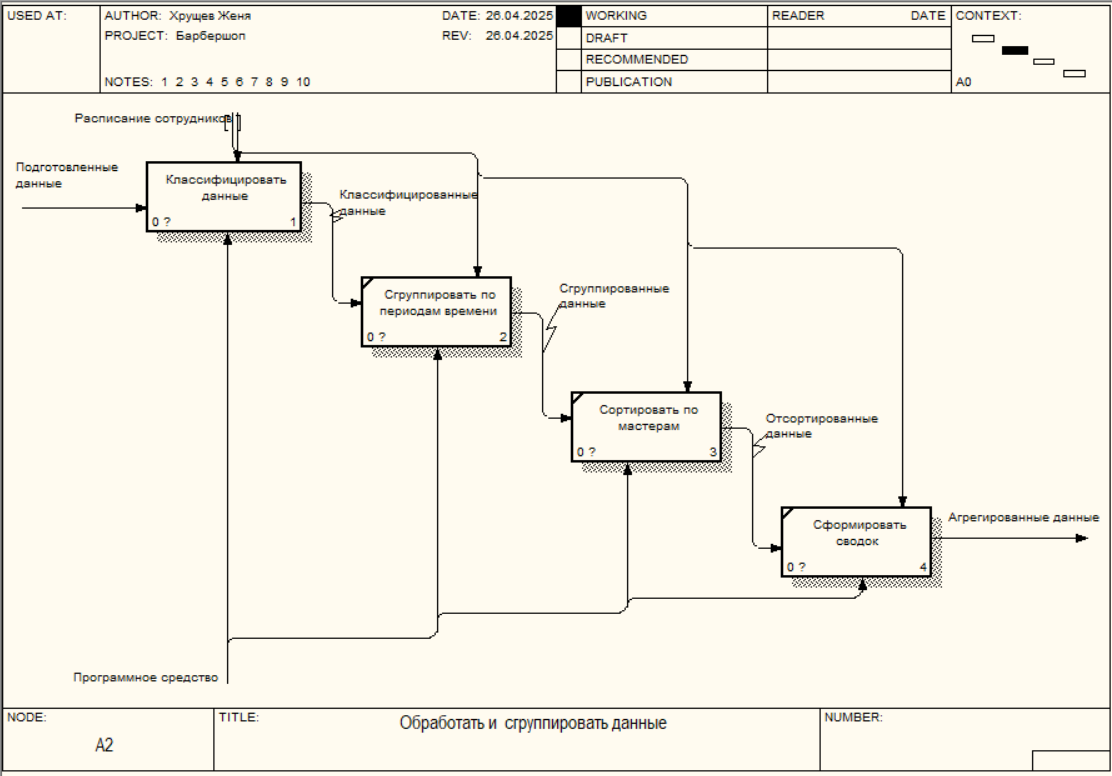


Рисунок 2.6 – Декомпозиция подпроцесса «Обработать и сгруппировать данные» модели «AS-IS»

Подпроцесс «Классифицировать данные» может быть также декомпозирован на несколько шагов:

* классифицировать данные по типу услуг;
* классифицировать данные по типу клиентов;
* классифицировать данные по времени;
* классифицировать данные по сотрудникам.

Декомпозиция подпроцесса «Классифицировать данные» приведена на рисунке 2.7.

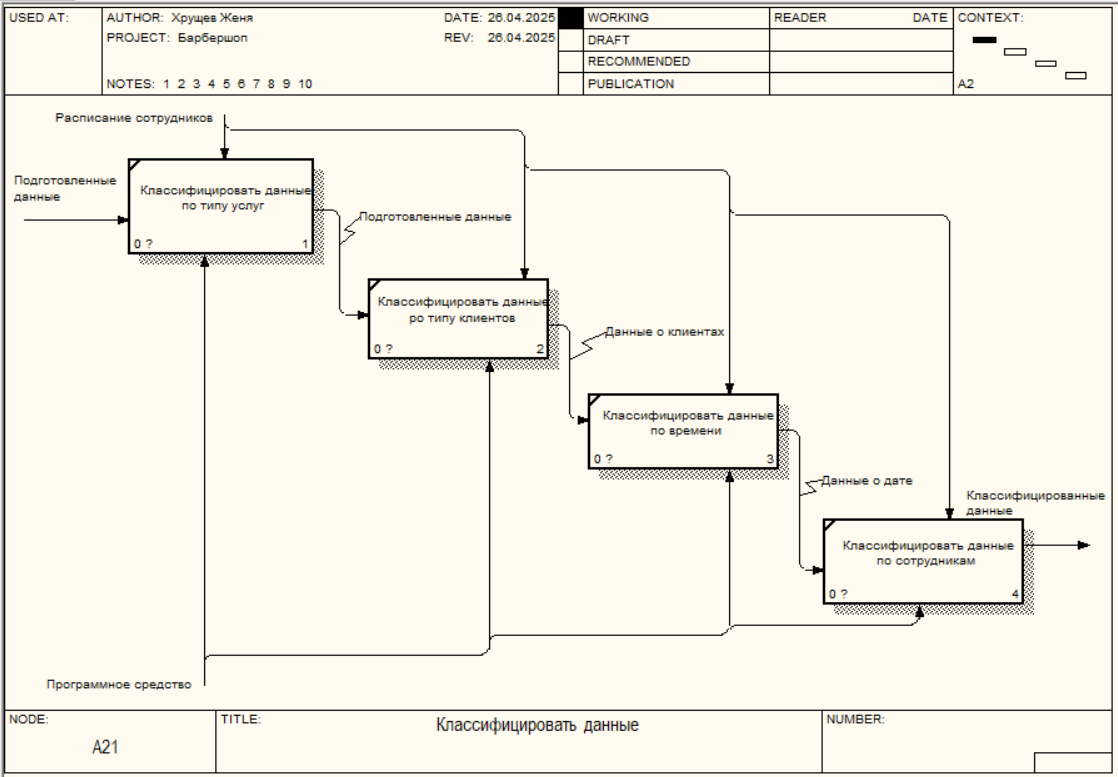


Рисунок 2.7 – Декомпозиция подпроцесса «Классифицировать данные» модели «AS-IS»

Прежде чем перейти к третьему подпроцессу, отметим, что он является объемным, однако он имеет большое значение, поскольку от того насколько ответственно подойдут к его выполнению зависит анализ данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа.

**Расчет общей посещаемости** – на этом этапе вычисляется сводный показатель, который отражает общую эффективность. Оценка загруженности сотрудников – анализируется, насколько эффективно используются ресурсы в процессе. Анализ возврата клиентов – проводится оценка возврата клиентов, чтобы понять, насколько эффективны.

На входе подпроцесса «Рассчитать ключевые метрики», представленного на рисунке 2.8, имеются агрегированные данные. В выполнении этой части принимает участие программное средство. Учитывая параметры и критерии анализа эффективности, в результате будут получены аналитические показатели для метрик.

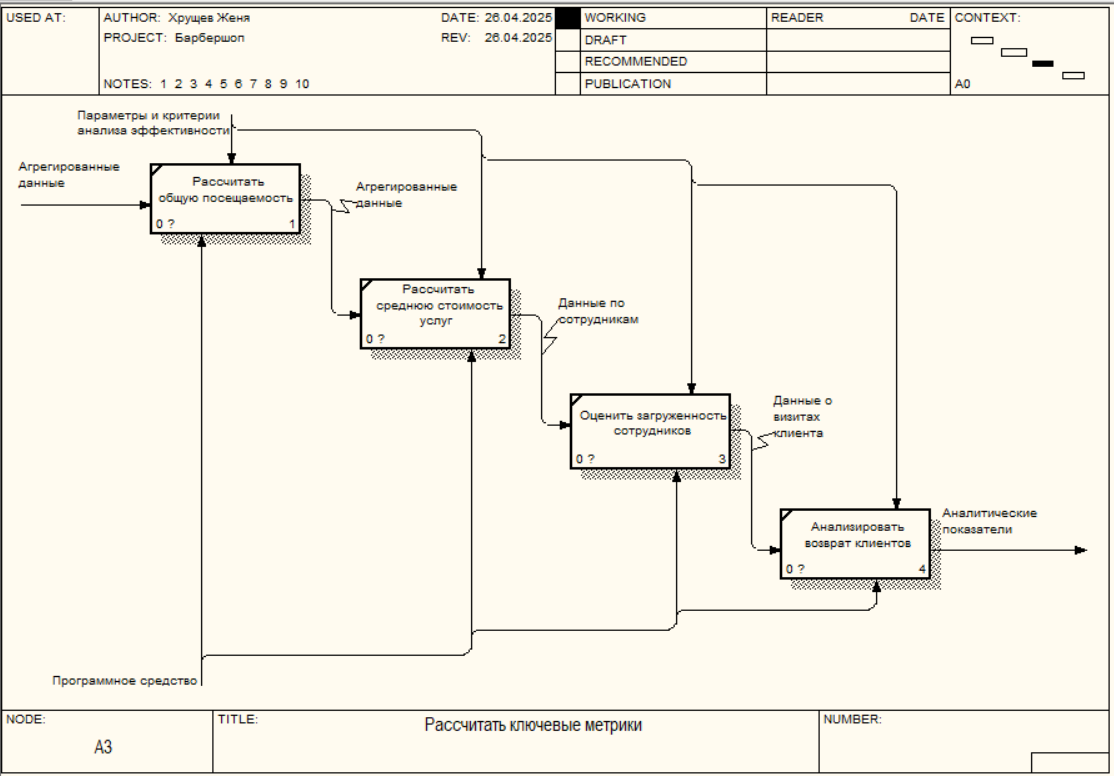


Рисунок 2.8 – Декомпозиция подпроцесса «Рассчитать ключевые метрики» модели «AS-IS»

Первый блок – сформировать отчеты об услугах – на этом этапе создаются отчеты, которые отображают достижения и результаты работы. **Генерировать рекомендации** – формируются отчеты, касающиеся конкретных сотрудников и их эффективности. Сформировать отчеты о эффективности – осуществляется анализ эффективности работников, на основе собранных данных и отчетов.

Декомпозиция подпроцесса «Сформировать отчеты» представлена на рисунке 2.9.

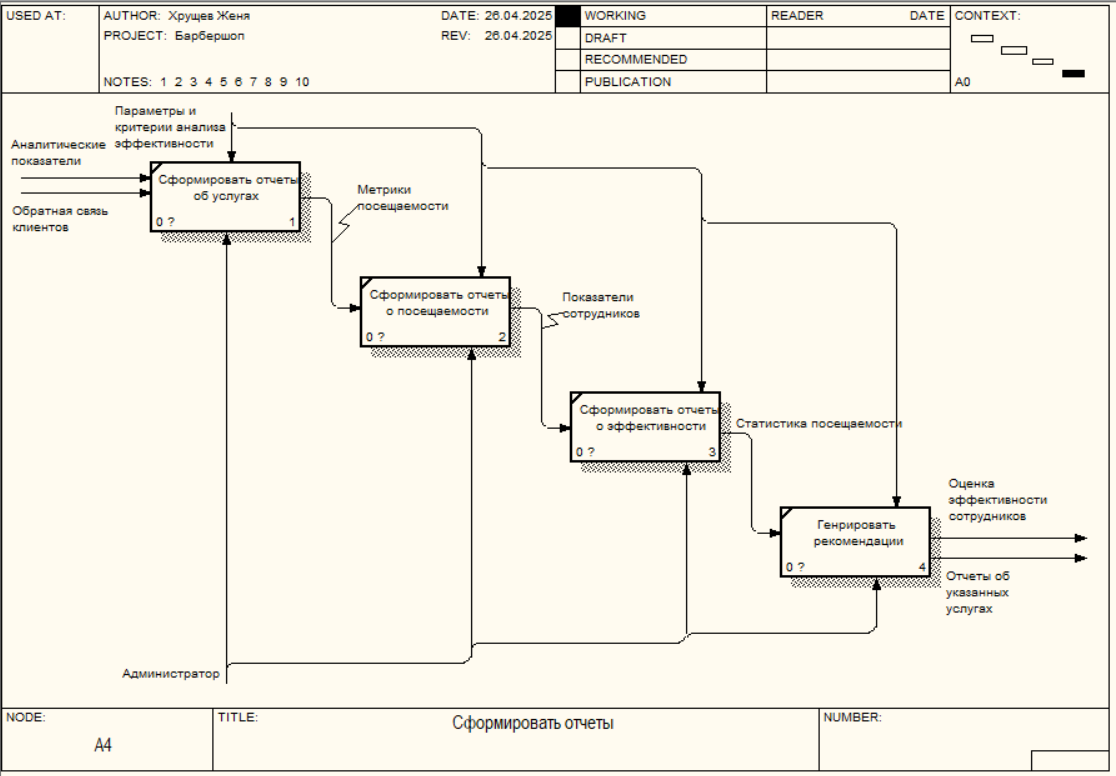


Рисунок 2.9 – Декомпозиция подпроцесса «Сформировать отчеты» модели «AS-IS»

Таким образом, происходит анализ данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа. И рассмотрев детально диаграмму, можно заметить, что главную роль здесь играет администратор. Система способствует повышению операционной и стратегической эффективности, что в конечном итоге положительно сказывается на прибыльности и репутации барбершопа.

## 2.2 Анализ требований к программному средству и разработка их спецификации

Для успешной разработки программного обеспечения необходимо понять принципы работы барбершопа. В настоящее время учет данных об оказанных услугах и посещаемости осуществляется частично вручную, что требует непосредственного участия сотрудников в обработке запросов. Это увеличивает трудоемкость работы и затрудняет параллельную обработку информации.

Каждое новое поступление данных будет зарегистрировано в базе данных, что облегчит поиск информации и снизит риск потери данных. Для корректной работы системы необходимо предусмотреть возможность добавления, удаления и редактирования сведений о новых услугах и посещениях. В качестве базы данных будет использоваться PostgreSQL, с подключением при авторизации пользователей. В зависимости от уровня прав, пользователи будут иметь различные возможности для выполнения своих задач [4].

Сотрудники барбершопа смогут обращаться за информацией о оказанных услугах и статистике посещаемости. Для удобства в системе будет реализована возможность формирования отчетов по заданным критериям, что упростит процесс анализа данных.

Для ведения отчетности барбершоп будет собирать информацию о предоставленных услугах и запросах клиентов. Эти данные помогут анализировать эффективность работы сотрудников и тенденции использования услуг, что, в свою очередь, будет полезно для оптимизации работы заведения.

Для повышения удобства работы сотрудников необходимо организовать функционал для управления данными, который позволит им эффективно вносить информацию о новых услугах и посещениях. Сотрудники будут ответственны за регистрацию данных и их привязку к соответствующим категориям.

Для ускорения и упрощения регистрации данных и формирования отчетов было принято решение о разработке программного средства для автоматизации анализа данных об услугах, посещаемости и эффективности работы сотрудников барбершопа.

Выделены следующие бизнес-цели проекта:

* автоматизировать процесс учета оказанных услуг и посещаемости, включая регистрацию данных о клиентах и услугах;
* обеспечить оперативный доступ сотрудников к информации о посещаемости и эффективности работы без участия администратора;
* реализовать возможность получения клиентами актуальной информации о предоставляемых услугах и статистике посещаемости.

Для сотрудников, осуществляющих анализ и учет данных, продукт будет представлять собой веб-приложение, автоматизирующее рутинные операции. В отличие от конкурентов, наше решение позволит регистрировать услуги и предоставлять клиентам оперативную информацию, значительно упрощая доступ к данным и повышая эффективность работы барбершопа.

Для наглядного представления возможностей разработанного программного продукта будет использоваться диаграмма вариантов использования (use case-диаграмма) в нотации UML, представленная на рисунке 2.10.

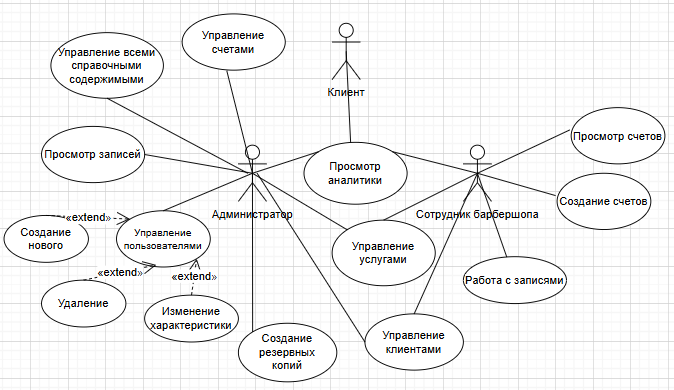


Рисунок 2.10 – Диаграмма вариантов использования

В системе представлено три роли: клиент, сотрудник барбершопа, администратор. Клиент – это обычный пользователь барбершопа, не обладающий административными правами; сотрудник барбершопа – работник, который взаимодействует с системой для регистрации услуг и клиентами; администратор – имеет возможность управлять аккаунтами сотрудников (добавлять новые, удалять старые, изменять роли).

Рассмотрение диаграммы начнем с обзора возможностей администратора во взаимодействии с системой. Исключая процесс авторизации, для него доступны еще восемь действий в системе. Рассмотрим каждое подробнее:

* «Управление пользователями» – возможность добавление нового библиотекаря или пользователя (гостя барбершопа), а также удаление старого, и при возможности изменение роли;
* «Просмотр аналитики» – просмотр посещаемости барбершопа и просмотр календаря;
* «Создание резервных копий» – создание резервных копий базы данных;
* «Управление клиентами» – возможность регистрации новых клиентов, обновление информации о клиентах;
* «Управление услугами» – добавление, редактирование и удаление услуг;
* «Просмотр записей» – просмотр всех записей клиентов на услуги;
* «Управление всеми справочными содержимыми» – управление списками должностей, типов услуг, статусов записей;
* «Управление счетами» – создание счетов для клиентов на основе оказанных услуг.

Что касается других действующих лиц, то поскольку система разрабатывалась в большей степени для администратора и сотрудника барбершопа, функционал для клиента уже. Сотрудник, также исключая авторизацию, имеет возможность просматривать аналитику, управлять услугами и клиентами, создавать и просматривать счета, работать с записями. Клиент может также просматривать аналитику.

В заключение, диаграмма вариантов использования четко иллюстрирует взаимодействие между администратором, сотрудником барбершопа и клиентом, акцентируя внимание на их ключевых функциях. Это подчеркивает необходимость ясного разделения ролей и обязанностей, что способствует оптимизации процессов, качества повышению внутренней организации. Таким образом, система ориентирована на удобство пользователей и эффективное администрирование.

## 2.3 Образ предлагаемого решения

В рамках разработки информационной системы для учета оказанных услуг, регистрации посещений и анализа эффективности работы сотрудников барбершопа реализуется **приложение с клиент-серверной архитектурой**. Такая архитектура обеспечивает централизованное хранение и обработку данных, разделение бизнес-логики и пользовательского интерфейса, а также гибкость масштабирования [5].

Система создается с использованием языка программирования Java, с применением JavaFX для разработки графического интерфейса клиента. Серверная часть представляет собой многопоточный HTTP-сервер, способный обрабатывать параллельные запросы от клиентов. В качестве хранилища используется реляционная база данных PostgreSQL, состоящая не менее чем из шести взаимосвязанных таблиц, нормализованных до третьей нормальной формы (3НФ). Для обеспечения ссылочной целостности реализованы внешние ключи и каскадные зависимости между таблицами.

Обмен данными между клиентом и сервером осуществляется через HTTP-запросы. Все операции по работе с данными – авторизация, регистрация новых услуг, учет посещений, формирование статистики, управление счетами – реализуются на серверной стороне, которая взаимодействует с базой данных через DAO-классы и слой SQL-абстракции (LibrarySQLFactory и др.).

Для конфигурации серверной части предусмотрены внешние настройки (порт, параметры подключения к БД и другие), что позволяет изменять их без перекомпиляции кода – через конфигурационные файлы или аргументы командной строки.

В реализации будут использованы ключевые приемы объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм, переопределение и перегрузка методов, использование абстрактных классов и интерфейсов, обработка исключений и сериализация объектов. Будут реализованы как два паттерна проектирования, например, Strategy для ввода текста с настраиваемой стратегией форматирования и Utility Class для общих методов без создания экземпляров класса.

Программное средство системы учета новых поступлений в библиотеку создается в следующих целях:

* автоматизация процессов учета оказанных услуг и посещений;
* повышение эффективности работы сотрудников за счет оптимизации рутинных процессов;
* создание систематизированного каталога услуг, доступного для фильтрации и поиска по различным параметрам;
* предоставление пользователям информации о посещаемости и статистике без необходимости обращения к администратору.

Таким образом, исследование текущей организации учета данных в барбершопе позволило выявить ряд недостатков: фрагментарность учета, отсутствие автоматизации, сложности в получении информации о посещениях и ограниченные аналитические возможности. После внедрения системы ожидается значительное повышение прозрачности и оперативности процессов. Были определены ключевые роли пользователей, составлены варианты использования, а также сформированы функциональные и нефункциональные требования к системе. В результате была выполнена постановка задачи и обоснован выбор клиент-серверной архитектуры с реляционной базой данных и защищенной моделью доступа.

# 3 Проектирование и разработка программного средства

## 3.1 Архитектурные решения и технологии реализации программного средства

Программное средство для системы анализа данных барбершопа разрабатывается на основе **клиент-серверной архитектуры**, обеспечивающей чёткое разделение ответственности между клиентской и серверной частями. Такой подход позволяет легко масштабировать систему, обеспечивать одновременную работу нескольких пользователей и безопасно хранить данные в централизованной базе.

**Серверная часть** отвечает за реализацию бизнес-логики, управление ролями пользователей, взаимодействие с базой данных и обработку клиентских запросов через протокол TCP/IP. Сервер написан с использованием многопоточности, что обеспечивает высокую производительность и параллельную обработку входящих соединений. Также реализована гибкая система конфигурации: параметры подключения к базе данных и сетевые настройки могут задаваться через конфигурационные файлы или аргументы командной строки, без необходимости перекомпиляции проекта [6].

**Клиентская часть** реализована с использованием JavaFX и представляет собой настольное графическое приложение, обеспечивающее интуитивно понятный интерфейс для различных категорий пользователей. В зависимости от роли (администратор, сотрудник, клиент), пользователь получает доступ к соответствующему функционалу: авторизация, регистрация услуг, управление записями, просмотр и анализ статистики, работа с клиентами, продуктами и услугами.

**Хранилище данных** представлено реляционной базой данных PostgreSQL, спроектированной в соответствии с требованиями третьей нормальной формы (3НФ) для минимизации избыточности и повышения целостности данных. Структура базы включает не менее шести взаимосвязанных таблиц, таких как users, role, clients, services, products, invoices, events, каждая из которых связана по принципам ссылочной целостности. В рамках проекта также поддерживается журналирование действий (таблица log) и ведение информации об администраторах, сотрудниках и клиентах.

Система обеспечивает высокий уровень безопасности: все пароли пользователей хранятся в зашифрованном виде с использованием современных хеш-функций. Кроме того, реализована функция резервного копирования базы данных с возможностью восстановления, что обеспечивает сохранность информации даже в случае сбоя.

Для улучшения читаемости, расширяемости и поддержки кода в проекте используются следующие паттерны проектирования:

* Strategy для создания настраиваемой стратегии форматирования текстового ввода, позволяющей гибко обрабатывать ввод пользователя (например, номера телефонов);
* Utility Class реализует набор статических методов для валидации и форматирования, что избавляет от необходимости создавать экземпляры вспомогательных классов.

Дополнительно, в проекте используются такие принципы объектно-ориентированного программирования, как инкапсуляция, наследование, полиморфизм, переопределение и перегрузка методов. Применяются также интерфейсы и абстрактные классы, реализована обработка исключений и сериализация объектов. В архитектуре проекта задействован **паттерн Singleton** для управления подключением к базе данных и **Factory Method** – для создания объектов пользователей разных ролей.

Таким образом, предложенное архитектурное решение соответствует современным требованиям к информационным системам: оно надёжно, расширяемо, безопасно и эффективно. Система автоматизирует процессы в барбершопе, упрощает учет оказанных услуг и посещений, предоставляет расширенные возможности аналитики и управления, тем самым повышая общую эффективность и прозрачность бизнес-процессов.

## 3.2 Проектирование и разработка пользовательского интерфейса

Система дизайна пользовательского интерфейса программного средства.Система дизайна имеет единый стиль и привлекательный интерфейс. Разработанная система дизайна в основном выполнена в белых, голубых цветах.

В программном средстве для кнопок были выбраны серые и белые цветы. Была выбрана такая цветовая палитра потому, что это цвета соответствуют фирменному стилю барбершопа, что создает элегантный и профессиональный вид. Для текста выбран шрифт Ellipsis. Также на системе дизайна можно увидеть, как в программном средстве представлены поисковая строка, а также выпадающий список, календари, карточки клиентов и сотрудников [7].

Пользовательский интерфейс программного средства. На рисунке 3.1 представлен дизайн формы авторизации.

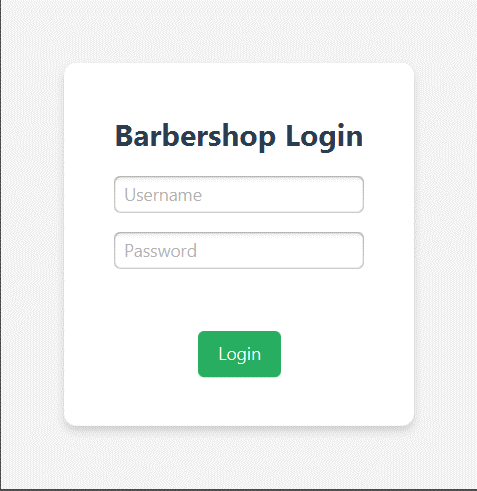


Рисунок 3.1 – Пользовательский интерфейс формы авторизации

Пользователь с ролью «Администратор» после входа в аккаунт может выполнять действия «Добавить нового клиента», «Просмотр аналитики», «Добавить нового сотрудника», «Просмотр статистики», «Просмотр всех клиентов», «Просмотр всех сотрудников». Окно логики действия «Добавить нового клиента» представлена на рисунке 3.2.

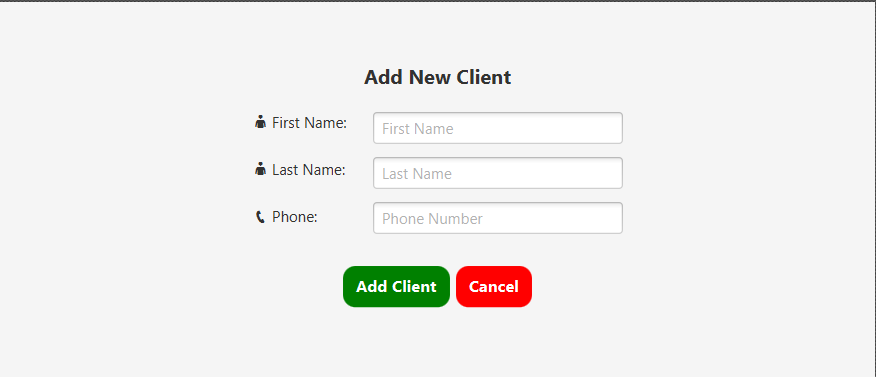


Рисунок 3.2 – Окно «Добавить нового клиента»

На рисунке 3.3 показана страница, которая открывается при нажатии на просмотр аналитики.

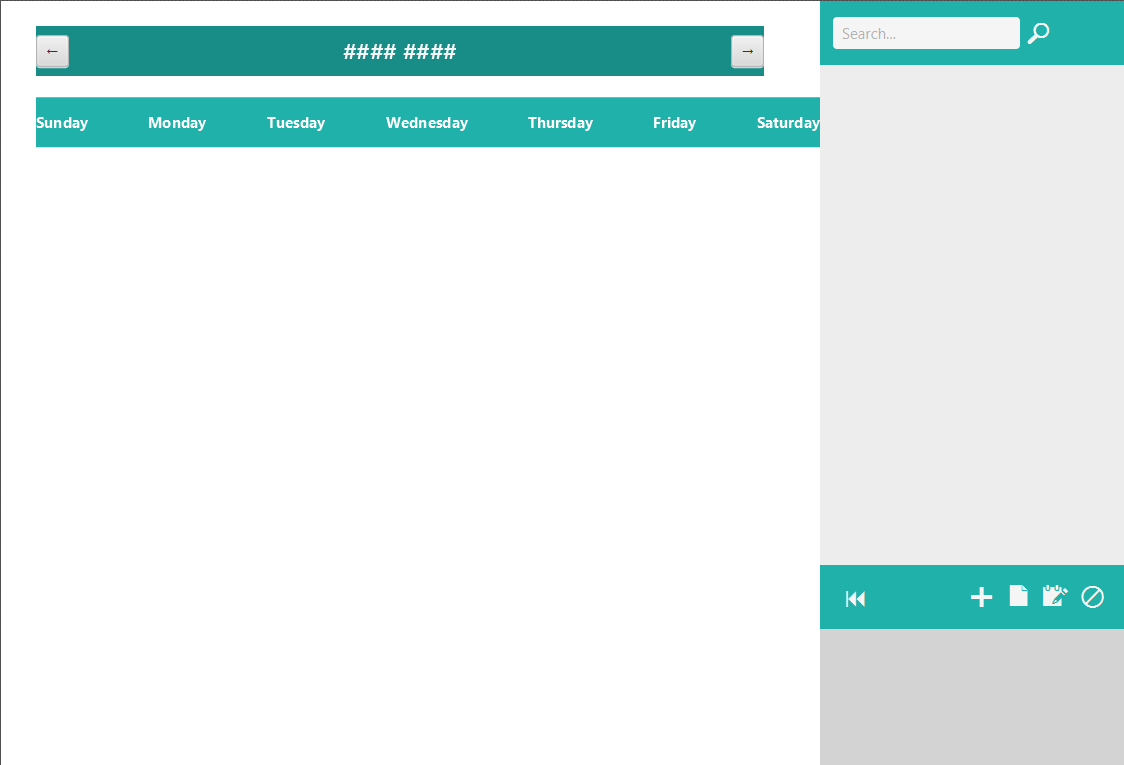


Рисунок 3.3 – Окно «Просмотр аналитики»

На рисунке 3.4 представлен пользовательский интерфейс страницы «Добавить нового сотрудника».

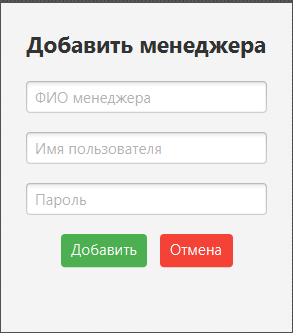


Рисунок 3.4 – Окно «Добавить нового сотрудника»

На рисунке 3.5 показан пользовательский интерфейс страницы «Просмотр статистики».

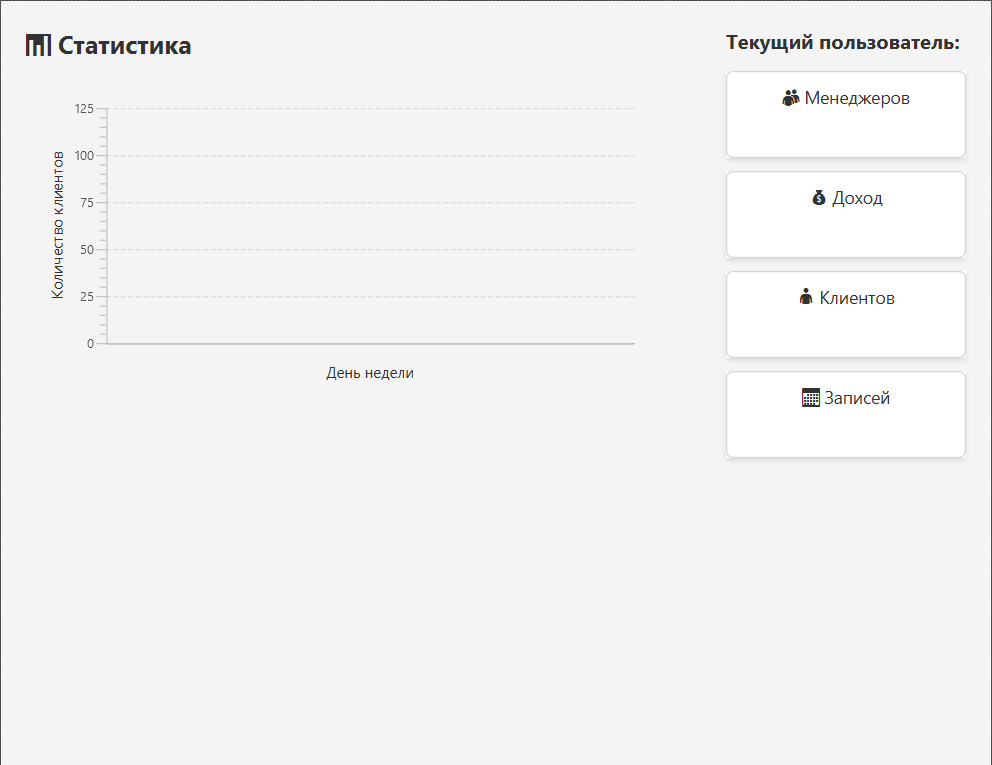


Рисунок 3.5 – Окно «Просмотр статистики»

На рисунке 3.6 представлен пользовательский интерфейс страницы «Просмотр всех сотрудников».

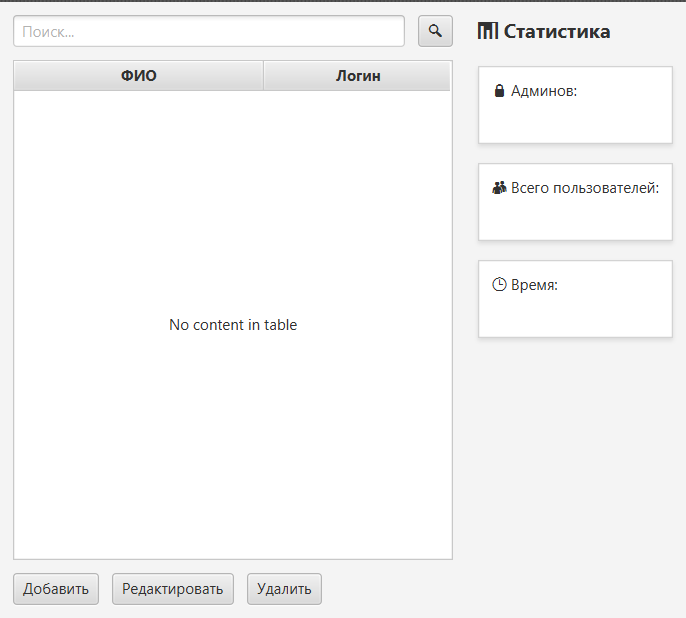


Рисунок 3.6 – Окно «Просмотр всех сотрудников»

На рисунке 3.7 представлен пользовательский интерфейс страницы «Просмотр всех клиентов».

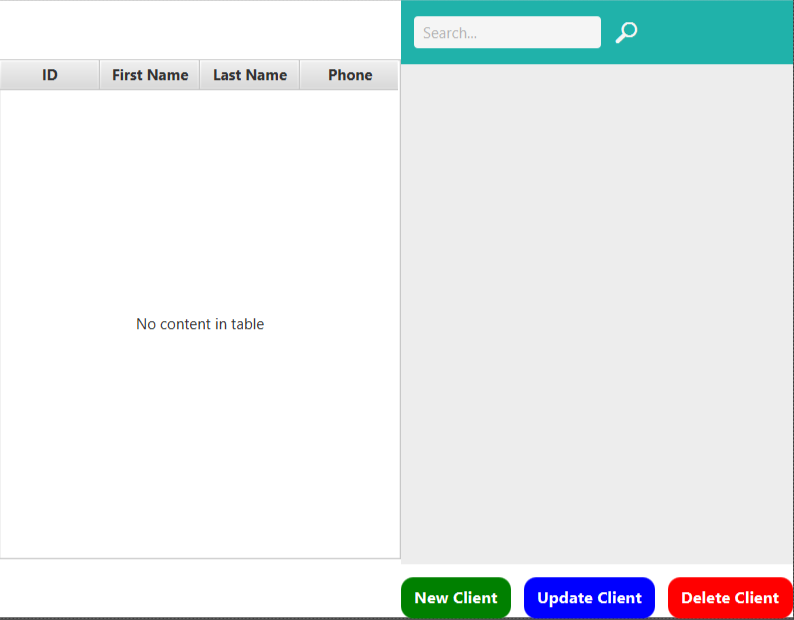


Рисунок 3.7 – Окно «Просмотр всех клиентов»

На рисунке 3.8 представлен пользовательский интерфейс страницы «Создание нового продукта».

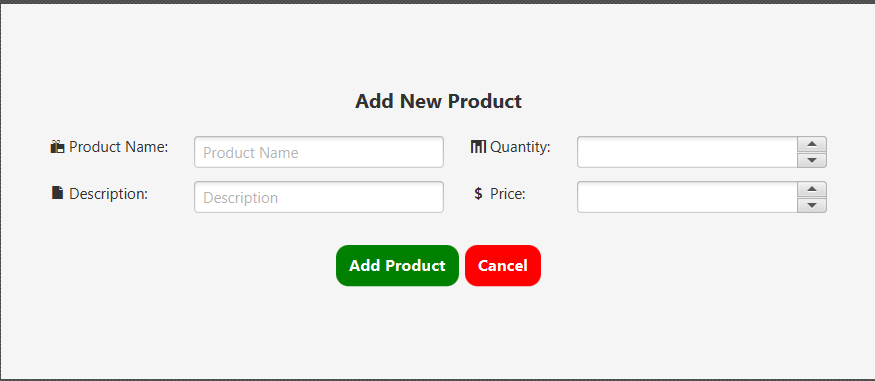


Рисунок 3.8 – Окно «Создание нового продукта»

На рисунке 3.9 представлен пользовательский интерфейс страницы «Создание нового события».

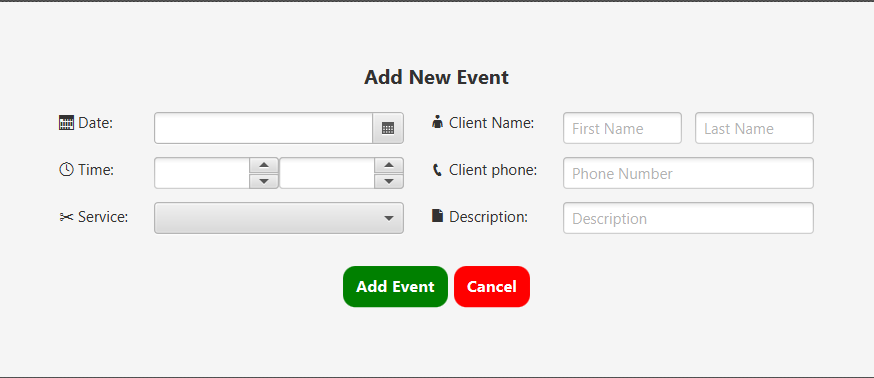


Рисунок 3.9 – Окно «Создание нового события»

На рисунке 3.10 представлен пользовательский интерфейс страницы «Создание нового сервиса».

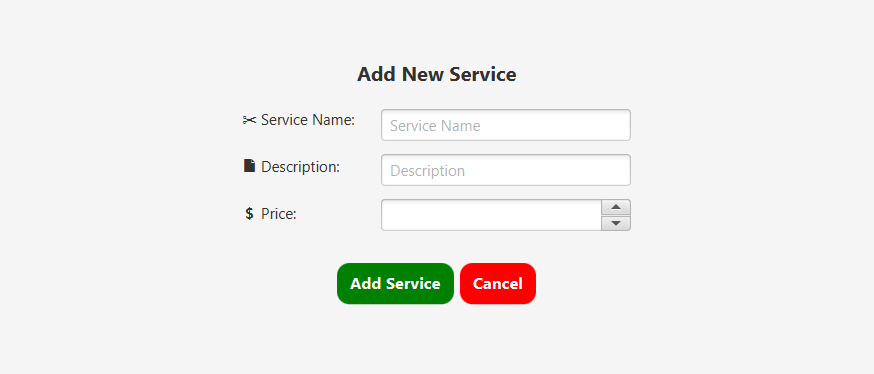


Рисунок 3.10 – Окно «Создание нового сервиса»

Таким образом, была разработана система дизайна пользовательского интерфейса сервиса для программного средства для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа. Также была построена схема логики действий пользователя в интерфейсе программного средства. Были разработаны планы расположения элементов на странице, после чего был реализован пользовательский интерфейс.

## 3.3 Разработка модели данных

Для поддержки реализации программного средства необходимо хранить информацию о пользователях, ролях пользователей, клиентах, услугах, записях (посещениях), счетах, продуктах, журналах действий [8].

Рассмотрим, из каких частей состоит эта информация. Сущность «Пользователь» можно описать следующим образом:

* уникальный идентификатор пользователя;
* логин;
* пароль;
* полное имя пользователя;
* телефон;
* email;
* дата регистрации;
* роль.

Сущность «Роль» можно описать следующим образом:

* уникальный идентификатор;
* название роли.

Сущность «Клиент» можно описать следующим образом:

* уникальный идентификатор клиента;
* ФИО;
* номер телефона;
* email;
* дата регистрации.

Сущность «Услуги» можно описать следующим образом:

* уникальный идентификатор услуги;
* название услуги;
* описание;
* стоимость;
* продолжительность в минутах.

Сущность «Записи» можно описать следующим образом:

* уникальный идентификатор записи;
* клиент;
* услуга;
* сотрудник;
* дата и время записи
* статус.

Сущность «Счета» можно описать следующим образом:

* уникальный идентификатор счета;
* клиент;
* общая сумма;
* дата и время создания;
* флаг оплаты.

Сущность «Продукты» можно описать следующим образом:

* уникальный идентификатор продукта;
* название товара;
* описание;
* цвет;
* количество на складе.

Сущность «Журнал действий» можно описать следующим образом:

* уникальный идентификатор события;
* кто выполнил действие;
* действие;
* время события.

У атрибутов каждой сущности есть определенный тип данных, описание которых представлено в таблице 3.1.

Таблица 3.1 – Описание типов данных

|  |  |  |
| --- | --- | --- |
| Тип данных | Описание | Атрибуты |
| *Integer* | Хранение целых чисел | Кто выполнил действие, количество на складе, цена, уникальный ID продукта, общая сумма, клиент, уникальный ID счёта, сотрудник, услуга, уникальный ID записи, продолжительность в минутах, стоимость, уникальный ID услуги, уникальный ID клиента, уникальный идентификатор пользователя. |
| *String* | Хранение строковых данных | Действие, описание, название товара, статус, описание, название услуги, email, номер телефона, фио клиента, название роли, пароль, логин. |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Тип данных | Описание | Атрибуты |
| *Date*/ *Timestamp* | Хранение даты и времени | Время события, дата и время создания, дата и время записи, дата регистрации. |
| *Boolean* | Хранение булевых значений | Флаг оплаты. |

На этапе физического проектирования необходимо преобразовать логическую модель данных в физическую схему базы данных. Полученная физическая модель базы данных представлена на рисунке 3.11.

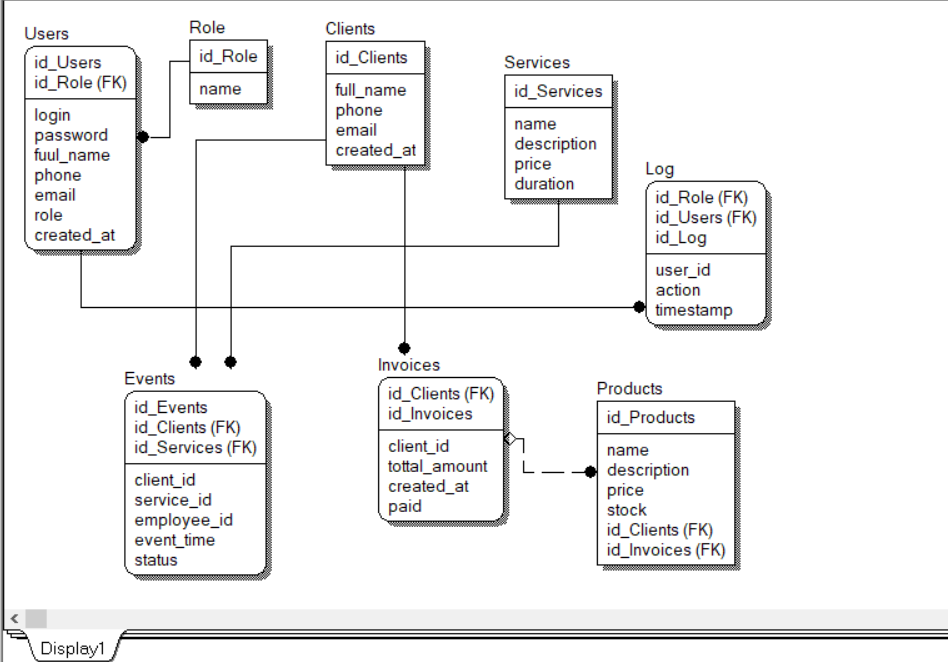


Рисунок 3.11 – ER-диаграмма для барбершопа

В ER-диаграмме также отражены взаимосвязи между сущностями. Один тип роли может быть назначен многим пользователям, что обозначается связью 1:M между сущностями «Role» и «Users», также и с клиентом, у одного клиента может быть много записей на услуги, «Clients» и «Events». Каждый сотрудник может выполнять много записей, что также отображает связь 1:M между «Users» и «Events». Одна услуга может фигурировать в нескольких записях, поэтому между «Services» и «Events» также установлена связь 1:M. Наконец, у одного клиента может быть несколько счетов, что отражает связь 1:M между сущностями «Clients» и «Invoices», а также один пользователь может совершить много действий, что отображается через связь «Users» и «Log». Также имеется связь M:M, если в счете может быть несколько товаров, что отображает связь между «Invoices» и «Products».

В данном подразделе были описаны сущности программного средства, после чего была построена *ER*-модель предметной области программного средства. Также была построена логическая модель данных и описаны ее сущности. После этого была разработана физическая схема базы данных. В конце подраздела был приведет скрипт генерации базы данных для разрабатываемого программного средства.

## 3.4 Описание статических и динамических аспектов поведения программных объектов

Язык *UML* представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

В данном курсовом проекте представлены пять диаграмм:

1 Диаграмма классов

На рисунках 3.12 и 3.13 представлена диаграмма классов пакета *Barbershop*.

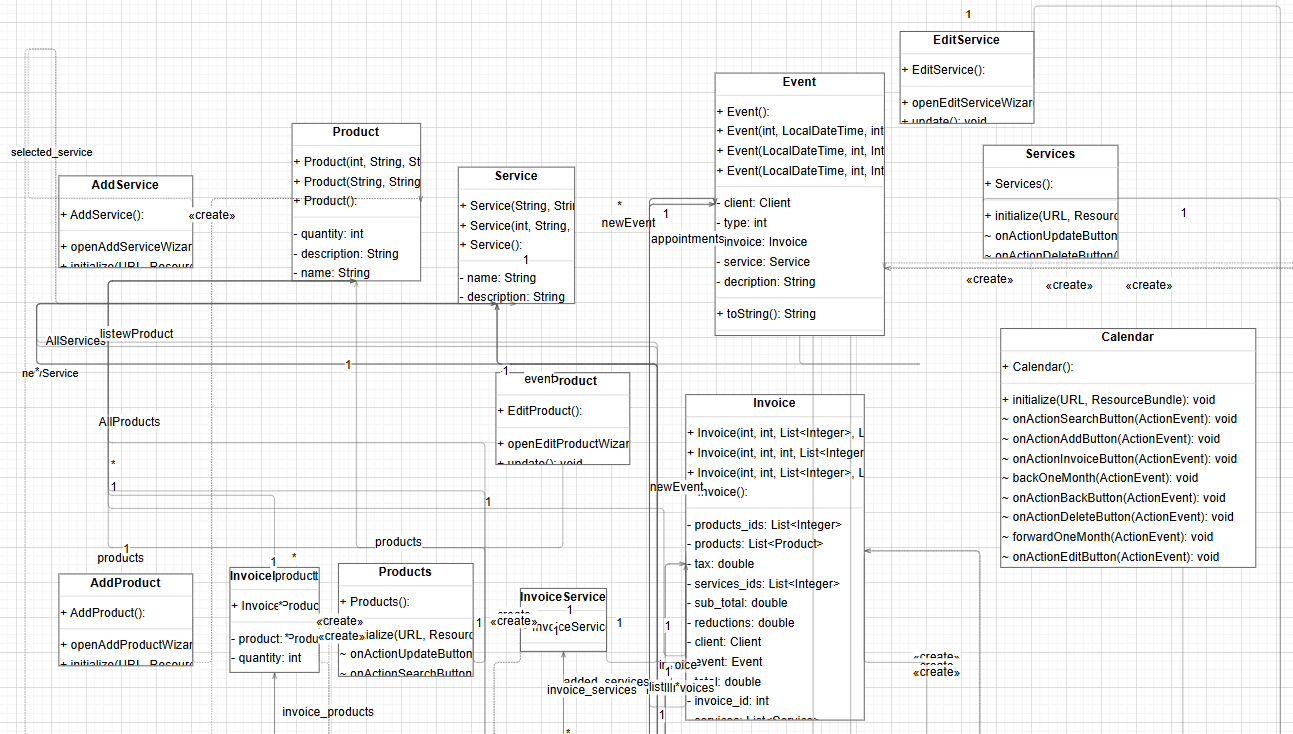


Рисунок 3.12 – Диаграмма классов пакета *Barbershop*

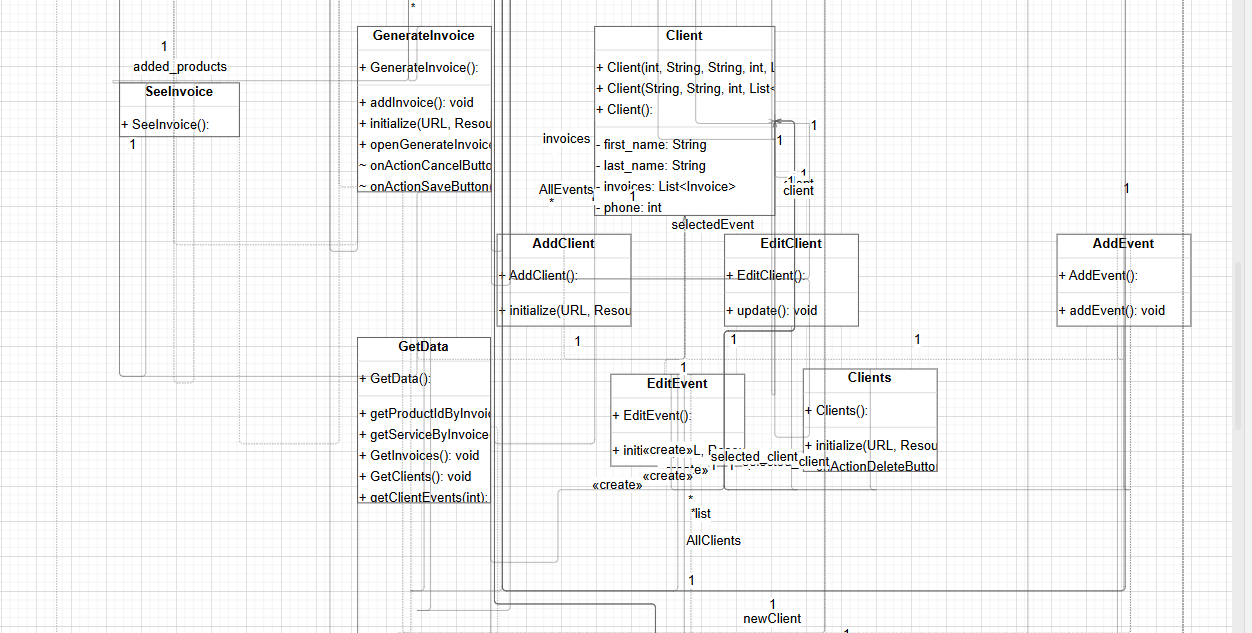


Рисунок 3.13 – Продолжение диаграммы классов пакета *Barbershop*

Представленная диаграмма классов имеет основные сущности: класс Client (инкапсуляция: все поля приватные, доступ через getters/setters, агрегация: содержит коллекции связанных событий и счетов), класс Event (ассоциация: связь с Client и Service, композиция: не может существовать без клиента), класс Invoice (использует агрегацию для товаров/услуг, реализует принцип единственной ответственности). Также имеется наследование: классы AddService, EditService, DeleteService наследуют общий интерфейс модальных окон, абстрактный класс BaseController может содержать общую логику; полиморфизм: перегрузка методов в PaternController, переопределение: контроллеры календаря могут переопределять методы отображения; абстракция: интерфейс DatabaseOperations (неявный) для всех классов работы с БД, абстракция окон через Modality.APPLICATION\_MODAL. В диаграмме классов представлены ключевые взаимосвязи: ассоциации (Client → Event (один клиент – много записей), Invoice → Product (счёт содержит много товаров)), зависимости (LoginController зависит от UserSession), композиции (InvoiceProduct полностью принадлежит Invoice) [9].

2 Диаграмма компонентов

Диаграмма компонентов используется для представления организации совокупности компонентов и существующих между ними зависимостей. Данный тип диаграмм описывает систему с точки зрения реализации. Диаграмма может быть связана с диаграммой классов т.к. обычно один компонент отображается на одном или нескольких интерфейсах, классах. Эта диаграмма представлена на рисунке 3.14.

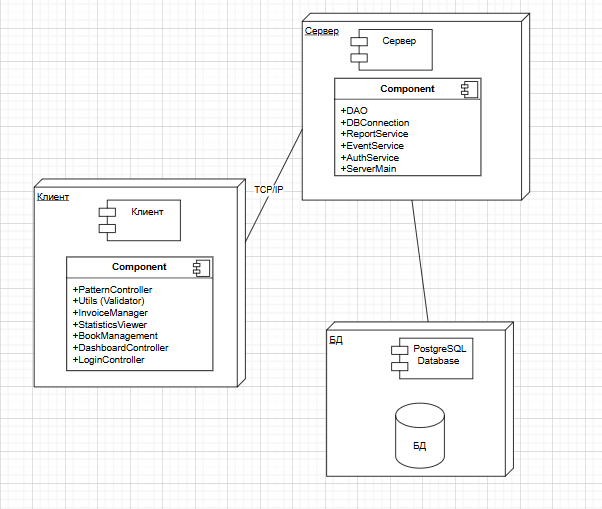


Рисунок 3.14 – Диаграмма компонентов

Мое программное средство для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа, включает JavaFX-клиент, многопоточный сервер и PostgreSQL-базу данных.

Клиентское приложение разрабатывается с использованием JavaFX, включает компоненты:

* LoginController – окно авторизации, отправляет логин/пароль на сервер;
* **DashboardController** – главная панель с навигацией;
* **BookManagement / StatisticsViewer / InvoiceManager** – модули для управления книгами, статистикой, счетами;
* **PatternController & Utils** – утилиты: валидация текста, маски ввода, форматирование.

Серверная часть запускается как многопоточное приложение, включающая компоненты:

* **ServerMain** – точка входа, запускает сервер и слушает порт;
* **RequestHandler** – обрабатывает запросы клиентов в отдельных потоках;
* **AuthService** – отвечает за аутентификацию и авторизацию;
* **EventService, ReportService** – бизнес-логика (работа с посещениями, отчетами и т.п.);
* **DBConnection (Singleton)** – подключение к БД с помощью JDBC;
* **DAO-классы** – слои доступа к данным.

PostgreSQL-база данных содержит нормализованные таблицы (до 3НФ), связанные внешними ключами:

* users, roles – для авторизации;
* clients, services, events – основа модели посещений;
* products, invoices – продажи и счета;
* log – журнал действий.

Пароли хранятся в виде хешей, резервные копии поддерживаются. Сервер контролирует доступ, проверяет роли (админ, сотрудник, клиент) и отправляет только нужную информацию клиенту. Клиент посылает TCP-соединения на сервер.

3 Диаграмма состояний

Диаграмма состояний для процесса «Анализ данных об указанных услугах» представлена на рисунке 3.15.

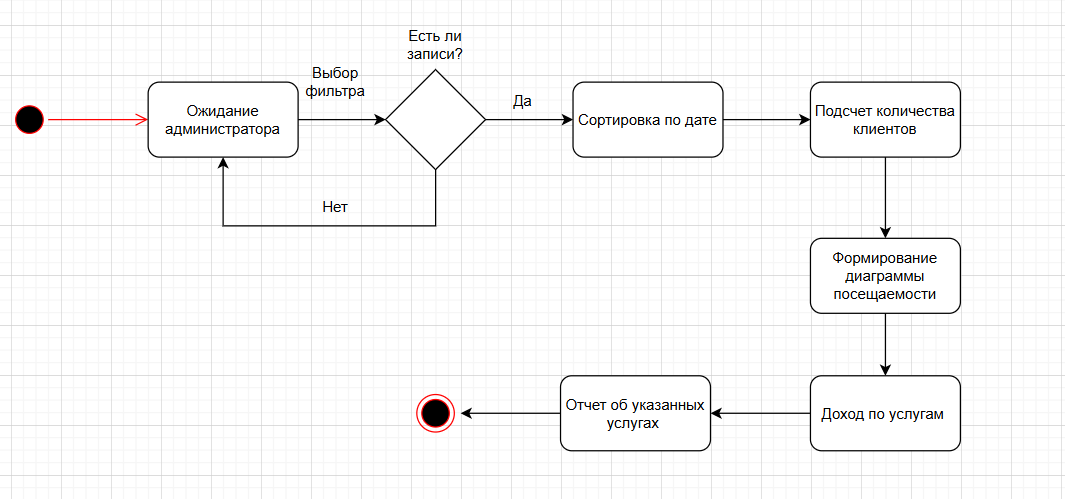


Рисунок 3.15 – Диаграмма состояний процесса «Анализ данных об указанных услугах»

Процесс анализа данных об оказанных услугах в информационной системе барбершопа представлен в виде набора логически связанных состояний, отражающих поведение системы от начала анализа до представления и завершения отчёта. Такой подход обеспечивает последовательную и наглядную обработку пользовательских запросов, а также адаптацию к различным условиям.

Система находится в режиме ожидания, пока администратор или менеджер не инициирует анализ. Пользователь выбирает необходимые фильтры:

* временной интервал (определённый день, неделя, месяц, произвольный диапазон);
* категория или тип услуги (стрижка, укладка, окрашивание и пр.);
* фамилия сотрудника, оказывающего услугу;
* возраст или пол клиента;
* стоимость услуги и длительность визита.

После выбора фильтров система обращается к базе данных PostgreSQL и выполняет SQL-запросы для получения соответствующих записей. В этот момент происходит проверка:

* если данные найдены → переход к следующему состоянию;
* если по заданным параметрам ничего не найдено → переход в состояние «Нет данных для анализа».

Затем система обрабатывает полученные данные:

* выполняет сортировку, агрегацию (например, суммирование выручки);
* группирует данные по дням, сотрудникам или типам услуг;
* вычисляет KPI (например, среднее время обслуживания, доход на одного клиента, частота обращений);
* подготавливает таблицы и промежуточные результаты.

На основе обработанных данных формируются графики и диаграммы посещаемости, доход по услугам. Готовые отчеты отображаются в клиентском приложении. Пользователь видит структурированные данные, в виде отчетов или таблиц. После выполнения всех операций и при завершении работы с отчетами, система возвращается в исходное состояние ожидания нового действия от пользователя.

Таким образом, данная диаграмма состояний отражает полный жизненный цикл процесса анализа данных: от инициации до вывода результатов. Она обеспечивает модульность, повторяемость анализа и возможность гибкой настройки в зависимости от целей пользователя.

4 Диаграмма последовательности

Диаграмма последовательности варианта использования процесса «Анализ данных о посещаемости барбершопа» представлена на рисунке 3.16.

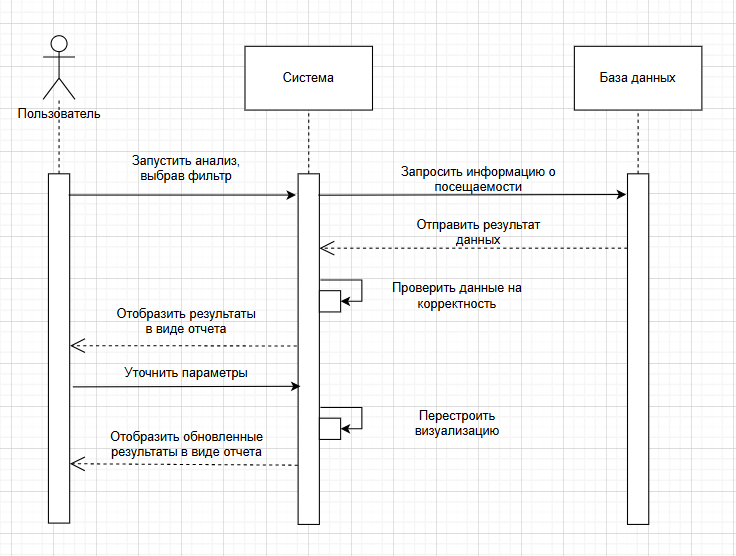


Рисунок 3.16 – Диаграмма последовательности варианта   
использования «Анализ данных о посещаемости барбершопа»

Диаграмма последовательности отражает взаимодействие между объектами пользователь, система и база данных в процессе анализа посещаемости. Они фиксируют взаимодействие между объектами в контексте сотрудничества. Диаграммы ориентированы на время и визуально показывают порядок взаимодействия, используя вертикальную ось диаграммы для представления времени, когда и какие сообщения отправляются. Описание сообщений представлено в таблице 3.2.

Таблица 3.2 – Описание сообщений

|  |  |  |  |
| --- | --- | --- | --- |
| Объект-отправитель | Объект-получатель | Имя сообщения | Описание сообщения |
| Пользователь | Система | Запустить анализ посещаемости, выбрав фильтр | Пользователь инициирует анализ, выбирая параметры. |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Объект-отправитель | Объект-получатель | Имя сообщения | Описание сообщения |
| Система | База данных | Запросить информацию о посещаемости | Система формирует SQL-запрос для получения информации о прошедших записях клиентов. |
| База данных | Система | Отправить результат данных | База возвращает список посещений по заданным критериям. |
| Система | Система | Проверить данные на корректность | Система проверяет, чтобы были отправлены те данные. |
| Система | Пользователь | Отобразить результаты анализа | Система визуализирует результаты в пользовательском интерфейсе. |
| Пользователь | Система | Уточнить параметры | Пользователь может изменить параметры анализа или экспортировать результат. |
| Система | Система | Перестроить визуализацию | Система визуализирует новые результаты. |
| Система | Пользователь | Отобразить новые результаты анализа | Система визуализирует новые результаты в пользовательском интерфейсе. |

5 Диаграмма развертывания

Диаграмма развертывания представляет конфигурацию узлов и размещенных в них компонентов. Диаграмма связана с диаграммой компонентов т.к. в узле размещается один либо несколько компонентов. Для каждой модели создается только одна диаграмма развертывания, отображающая процессоры, устройства и их соединения. Эта диаграмма представлена на рисунке 3.17.

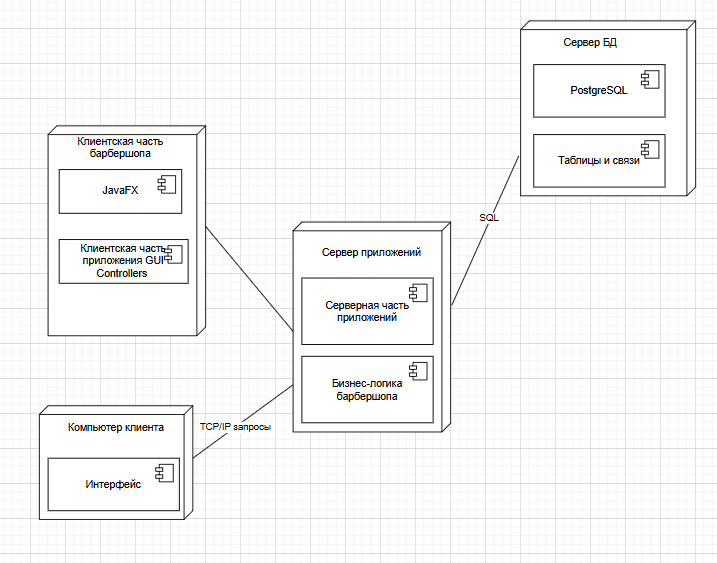


Рисунок 3.17 – Диаграмма развертывания

Клиентское устройство – компьютер или ноутбук пользователя, на котором установлено Java-приложение с графическим интерфейсом, реализованным с помощью JavaFX. Пользователь взаимодействует с системой через это приложение. Клиент выполняет следующие функции:

* вводит данные (логин, регистрация, параметры поиска);
* управляет клиентами, услугами, счетами;
* просматривает статистику и аналитику;
* может экспортировать отчёты (в PDF, Excel и т.п.).

Сервер приложения – это узел, где развернут сервер на Java. Он принимает TCP/IP-запросы от клиентов, обрабатывает их и взаимодействует с базой данных. Основные компоненты:

* javaServer – базовая серверная структура;
* бизнес-логика – набор Java-классов, реализующих поведение системы (например, авторизация, управление клиентами, формирование статистики);
* dао-классы (DB, GetData, DeleteData, и другие) – слой доступа к данным, обеспечивающий безопасную и корректную работу с базой данных;
* контроллеры – компоненты, обрабатывающие действия, связанные с пользователями, услугами, событиями, отчетами.

Сервер поддерживает многопоточность, что позволяет обслуживать несколько клиентов одновременно, обеспечивая отклик и масштабируемость.

На сервере базы данных развернута СУБД PostgreSQL, в которой хранится вся информация о деятельности барбершопа. Между таблицами реализованы связи один-ко-многим и многие-к-одному, что обеспечивает целостность данных. Сервер обрабатывает SQL-запросы от приложения и возвращает результаты для анализа и отображения на клиенте.

Таким образом, в данном подразделе были построены пять диаграмм: диаграмма классов, диаграмма деятельности, диаграмма состояний и диаграмма последовательности и диаграмма развертывания.

## 3.5 Разработка и описание алгоритмов, реализующих бизнес-логику программного средства

**Алгоритм добавления записи в барбершоп.** Описание шагов алгоритма:

1 Пользователь (менеджер/админ) открывает форму добавления новой записи.

2 Система проверяет роль.

3 Если нет прав – показ ошибки.

4 Если такая запись есть, то проверяется существует ли клиент такой.

5 Если нет – заполнение и сохранение данных.

6 Если есть – поиск.

7 Загрузка списка услуг.

8 Возможность выбора нескольких услуг.

9 Выбор мастера.

10 Выбор времени.

11 Если нет свободного на это время – выбор другого.

12 Если есть – запись.

13 Сохранение записи.

Схема алгоритма добавления записи в барбершоп представлена на рисунке 3.18.

Код алгоритма представлен ниже.

package com.barbershop.models;  
import java.time.LocalDateTime;  
  
import com.barbershop.controllers.database.GetData;  
import com.barbershop.controllers.patterns.PaternController;  
  
public class Event {  
 private int event\_id;  
 private LocalDateTime date\_time;  
 private int client\_id;  
 private Client client;  
 private Integer invoice\_id;  
 private Invoice invoice;  
 private int service\_id;  
 private Service service;  
 private String decription;  
 private int type; //1: not done, 0: canceled, 2: done  
   
 public Event() {}  
 //get from database  
 public Event(int event\_id, LocalDateTime date\_time, int client\_id, Integer invoice\_id, int service\_id,  
 String decription, int type) {  
 this.event\_id = event\_id;  
 this.date\_time = date\_time;  
 this.client\_id = client\_id;  
 this.invoice\_id = invoice\_id;  
 this.service\_id = service\_id;  
 this.decription = decription;  
 this.type = type;  
 for (Client c : GetData.*AllClients*) {  
 if (c.getClient\_id() == client\_id){  
 this.client = c;  
 break;  
 }  
 }  
 if (invoice\_id != null) {  
 for (Invoice i : GetData.*AllInvoices*) {  
 if (i.getInvoice\_id() == invoice\_id){  
 this.invoice = i;  
 break;  
 }  
 }  
 }  
 for (Service s : GetData.*AllServices*) {  
 if (s.getServiceId() == service\_id){  
 this.service = s;  
 }  
 }  
 }  
 //insert in database  
 public Event(LocalDateTime date\_time, int client\_id, Integer invoice\_id, int service\_id, String decription,  
 int type) {  
 this.date\_time = date\_time;  
 this.client\_id = client\_id;  
 this.invoice\_id = invoice\_id;  
 this.service\_id = service\_id;  
 this.decription = decription;  
 this.type = type;  
 for (Client c : GetData.*AllClients*) {  
 if (c.getClient\_id() == client\_id){  
 this.client = c;  
 break;  
 }  
 }  
 if (invoice\_id != null) {  
 for (Invoice i : GetData.*AllInvoices*) {  
 if (i.getInvoice\_id() == invoice\_id){  
 this.invoice = i;  
 break;  
 }  
 }  
 }  
 for (Service s : GetData.*AllServices*) {  
 if (s.getServiceId() == service\_id){  
 this.service = s;  
 }  
 }  
 }  
 public Event(LocalDateTime date\_time, int client\_id, Integer invoice\_id, int service\_id, String decription) {  
 this.date\_time = date\_time;  
 this.client\_id = client\_id;  
 this.invoice\_id = invoice\_id;  
 this.service\_id = service\_id;  
 this.decription = decription;  
 this.type = 1;   
 }  
   
 public int getEventId() {  
 return event\_id;  
 }  
 public void setEventId(int event\_id) {  
 this.event\_id = event\_id;  
 }  
 public LocalDateTime getDateTime() {  
 return date\_time;  
 }  
 public void setDateTime(LocalDateTime date\_time) {  
 this.date\_time = date\_time;  
 }  
 public String getDate(){  
 return this.date\_time.getDayOfMonth()+" "+this.date\_time.getMonth()+" "+this.date\_time.getYear();  
 }  
 public String getTime(){  
 int min = this.date\_time.getMinute();  
 int hour = this.date\_time.getHour();  
 if ( min < 10){  
 if (hour < 10) {  
 return this.date\_time.getHour()+"0"+":"+min+"0";  
 }  
 return this.date\_time.getHour()+":"+min+"0";  
 }  
 return this.date\_time.getHour()+":"+min;  
 }  
 public String getDateAndTime(){  
 return getDate()+" "+getTime();  
 }  
 public int getClientId() {  
 return client\_id;  
 }  
 public void setClientId(int client\_id) {  
 this.client\_id = client\_id;  
 }  
 public Integer getInvoiceId() {  
 return invoice\_id;  
 }  
 public void setInvoiceId(Integer invoice\_id) {  
 this.invoice\_id = invoice\_id;  
 }  
 public int getServiceId() {  
 return service\_id;  
 }  
 public void setServiceId(int service\_id) {  
 this.service\_id = service\_id;  
 }  
 public String getDecription() {  
 return decription;  
 }  
 public void setDecription(String decription) {  
 this.decription = decription;  
 }  
 public int getType() {  
 return type;  
 }  
 public void setType(int type) {  
 this.type = type;  
 }  
 public Client getClient() {  
 return client;  
 }  
 public void setClient(Client client) {  
 this.client = client;  
 }  
 public Invoice getInvoice() {  
 return invoice;  
 }  
 public void setInvoice(Invoice invoice) {  
 this.invoice = invoice;  
 }  
 public Service getService() {  
 return service;  
 }  
 public void setService(Service service) {  
 this.service = service;  
 }  
  
 @Override  
 public String toString() {  
 return this.getDate()+"\n"+this.getTime()+"\n"+PaternController.*capitalize*(this.getClient().getFirst\_name())+" "+this.getClient().getLast\_name().toUpperCase()+"\n"+this.getDecription()+"\n"+this.getService().getDescription();  
 }  
/\* public String toString() {  
 return "Event [event\_id=" + event\_id + ", date\_time=" + date\_time + ", client\_id=" + client\_id + ", invoice\_id="  
 + invoice\_id + ", service\_id=" + service\_id + ", decription=" + decription + ", type=" + type + "]";  
 } \*/  
}

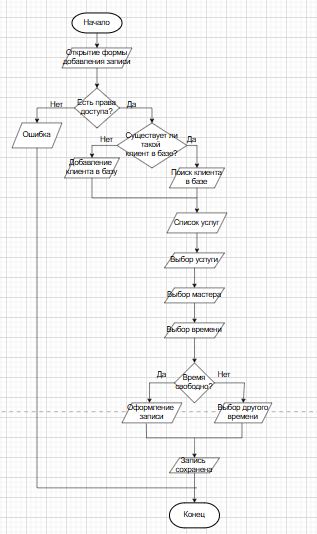


Рисунок 3.18 – Схема алгоритма добавление нового события

**Алгоритм редактирования менеджера.** Описание шагов алгоритма:

1 Проверить наличие выбранного пользователя в базе данных.

2 Если пользователь не выбран – ошибка.

3 Если выбран – ввести ФИО.

4 Проверить заполнение фамилии.

5 Если пустые – ошибка.

6 Если заполнены – перейти к изменению.

7 Ввод нового пароля.

8 Ввод нового логина.

9 Подключение к бд.

10 Сохранить изменения.

Схема алгоритма выгрузки освобождения представлена на рисунке 3.19.

Код данного алгоритма представлен ниже.

package com.barbershop.controllers.admin;  
  
import com.barbershop.models.User;  
import javafx.fxml.FXML;  
import javafx.scene.control.Alert;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
  
public class EditManagerController {  
  
 @FXML private TextField fullNameField;  
 @FXML private TextField usernameField;  
 @FXML private PasswordField passwordField;  
  
 public static User *selectedUser*;  
  
 @FXML  
 public void initialize() {  
 if (*selectedUser* != null) {  
 fullNameField.setText(*selectedUser*.getFullName());  
 usernameField.setText(*selectedUser*.getUsername());  
 }  
 }  
  
 @FXML  
 private void handleSave() {  
 String fullName = fullNameField.getText().trim();  
 String username = usernameField.getText().trim();  
 String password = passwordField.getText().trim();  
  
 if (fullName.isEmpty() || username.isEmpty()) {  
 showAlert("ФИО и логин обязательны для заполнения.");  
 return;  
 }  
  
 try (Connection conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/barbershop", "postgres", "1234")) {  
 String sql;  
 PreparedStatement stmt;  
 if (!password.isEmpty()) {  
 sql = "UPDATE users SET full\_name = ?, username = ?, password = ? WHERE user\_id = ?";  
 stmt = conn.prepareStatement(sql);  
 stmt.setString(1, fullName);  
 stmt.setString(2, username);  
 stmt.setString(3, password);  
 stmt.setInt(4, *selectedUser*.getUserId());  
 } else {  
 sql = "UPDATE users SET full\_name = ?, username = ? WHERE user\_id = ?";  
 stmt = conn.prepareStatement(sql);  
 stmt.setString(1, fullName);  
 stmt.setString(2, username);  
 stmt.setInt(3, *selectedUser*.getUserId());  
 }  
 stmt.executeUpdate();  
 close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 showAlert("Ошибка при сохранении изменений.");  
 }  
 }  
  
 @FXML  
 private void handleCancel() {  
 close();  
 }  
  
 private void close() {  
 Stage stage = (Stage) fullNameField.getScene().getWindow();  
 stage.close();  
 }  
  
 private void showAlert(String message) {  
 Alert alert = new Alert(Alert.AlertType.*WARNING*);  
 alert.setTitle("Ошибка");  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
}

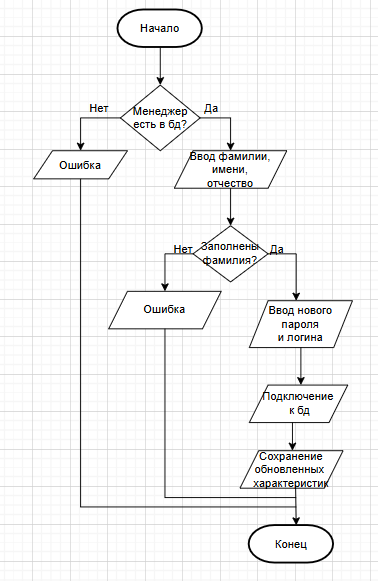


Рисунок 3.19 – Схема алгоритма редактирования менеджера

На рисунке 3.20 представлена схема алгоритма всей программы. Данная схема алгоритма отображает процесс «Анализ данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа». При входе в программу перед пользователем открывается две основные опции: «Авторизация» и «Регистрация». Если пользователь выбирает регистрацию, ему необходимо ввести логин и пароль, а также выбрать роль. Система проверяет уникальность логина, и, если он свободен, создается новая учетная запись. После успешной регистрации пользователь получает уведомление. При выборе авторизации пользователю предлагается ввести свои логин и пароль. Система проверяет введенные данные, и при их корректности определяется роль пользователя: либо это администратор, либо менеджер, либо клиент. В зависимости от роли пользователь перенаправляется в соответствующее меню [10].

Администратор управляет персоналом (просмотр списка менеджеров, добавление, редактирование, удаление) и аналитикой (просмотр статистики, формирование отчетов). Пациент входит, вводя свои данные тем самым создавая аккаунт, а дальше он может просматривать аналитику менеджеров. Менеджер имеет доступ к более широкому функционалу. В его меню предусмотрены управление записями (просмотр календаря, добавление записи, а также ее удаление или редактирование), управление клиентами (просмотр списка, добавление, редактирование, удаление), управление услугами (просмотр, добавление, редактирование, удаление), управление товара (просмотр, добавление, редактирование, удаление), финансы (генерация счетов, просмотр счетов).

Кроме того, в любой момент пользователь или администратор может выйти из системы, завершив сессию.

В общем, данная схема отображает процесс авторизации и регистрации пользователей с разделением функционала на три роли. Она описывает, как каждый тип пользователя взаимодействует с системой и какие функции доступны в зависимости от их роли.

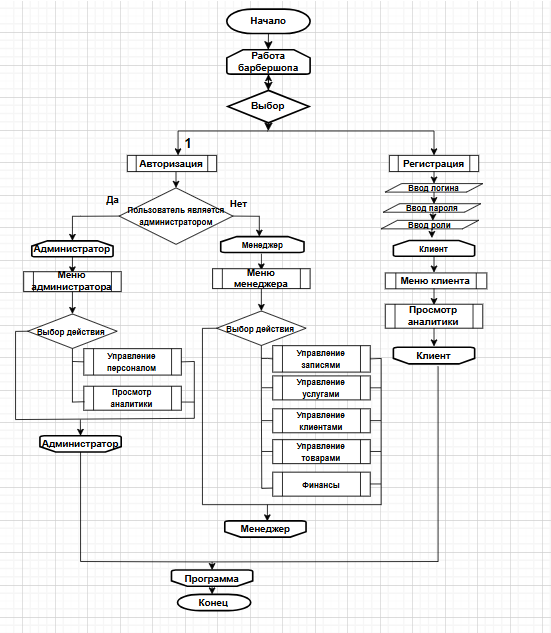


Рисунок 3.20 – Схема алгоритма всей программы

В данном подразделе были приведены алгоритмы процессов создания мероприятия и формирования освобождения. Также были построены схемы алгоритмов, соответствующие данным алгоритмам. Кроме того, был представлен код, показывающий реализацию данных алгоритмов.

## 3.6 Механизмы обеспечения информационной безопасности

Информационная безопасность программного средства для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа, обеспечивает защиту данных от несанкционированного доступа, модификации и утечки при передаче и хранении. Для этого в проекте реализованы следующие меры: шифрование и хранение паролей, аутентификация и управление сессиями, ограничение прав доступа, безопасность передачи данных.

Для хранения пользовательских паролей пользователей используется хеширование с солью с применением криптографического алгоритма SHA-256. Это исключает возможность восстановления исходного пароля даже при компрометации базы данных. Соль (уникальная случайная строка) добавляется к паролю перед хешированием, что повышает устойчивость к атакам методом радужных таблиц. При регистрации пользователя проводится валидация пароля, включая минимальную длину и наличие хотя бы одной цифры [11].

Аутентификация пользователей осуществляется при помощи ввода логина и пароля. После успешной авторизации создаётся уникальный идентификатор сессии, по которому осуществляется дальнейшее взаимодействие клиента с сервером, что предотвращает повторную передачу логина и пароля при каждом запросе.

В системе предусмотрены три уровня доступа: администратор, сотрудник барбершопа, клиент. Каждая роль имеет доступ только к своему набору функций: администратор управляет пользователями и резервными копиями; сотрудник барбершопа регистрирует услуги, фиксирует посещения и просматривает статистику, создает счеты и просматривает их, работает с записями; клиент может только просматривать аналитику.

Контроль доступа реализован на стороне сервера – сервер проверяет роль авторизованного пользователя и разрешает или запрещает выполнение определённых операций. Это позволяет исключить несанкционированный доступ к данным и действиям вне рамок полномочий. При передаче данных между клиентом и сервером применяется протокол TCP/IP, что защищает от перехвата и подмены данных в сетевой среде.

Таким образом, в ходе проектирования и разработки программного средства была спроектирована архитектура приложения, выбраны компоненты и технологии для реализации, разработаны макеты пользовательского интерфейса, схемы логической и физической моделей базы данных, реализована модель данных, разработаны диаграммы UML и алгоритмы. Определены механизмы обеспечения информационной безопасности, что гарантирует надежную защиту данных и функциональность системы.

# 4 Тестирование и проверка работоспособности программного средства

Функциональные спецификации программного средства приведены в разделе 2.2 и описываются вариантами использования.

Для того чтобы убедиться, что программное средство может корректно выполнять работу и устойчиво к ошибкам пользователя было проведено тестирование, отражающие все возможные исключительные ситуации.

Во-первых, было рассмотрена возможность не корректной авторизации пользователя, а именно:

* попытка входа без указания логина;
* попытка входа без указания пароля;
* ввод неверных данных.

Программа реагирует на все три ситуации, и пользователю возвращается окно с указанием ошибки.

Далее была протестирована работа с записями администраторов. Программа реагирует на ошибки при регистрировании, а именно: добавление администратора с логином, который уже существует, попытка регистрации без ввода логина/пароля, ввод неверного пароля при подтверждении. Программа реагирует на все ситуации и возвращает пользователю сообщение об ошибке.

Также было проведено тестирование программы при попытке изменить пароль администратора. При вводе пароля короче, чем 5 символов, или вводе отличного пароля при подтверждении программа возвратит пользователю сообщение об ошибке.

При изменении пароля регистратора предусмотрена обработка исключительных ситуаций, когда пользователь ввел разные пароль и пароль подтверждения, а также ввод пароля, не удовлетворяющего требованию длины. Провели тестирование функционала системы при редактировании данных менеджеров. При добавлении нового, если не заполнить одно из обязательных полей, программа вернет ошибку пользователю.

При попытке редактирования либо удаления записи без выбора её из списка программа возвращает соотвествующие сообщение пользователю.

По результатам тестирования можно сделать вывод, что разработанное программное средство удовлетворяет функциональным требованиям и функции выполняются корректно.

Тест-кейсы для проверки уровня базовых пользовательских требований приведены в таблице 4.1.

Таблица . – Тест-кейсы для проверки уровня пользовательских требований

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор тест-кейса | Название тест-кейса | Шаги тест-кейса | Ожидаемый результат |
| **TC-1** | Авторизоваться | 1. Открыть веб-приложение/клиентское приложение барбершопа. 2. Нажать кнопку «Войти». 3. Ввести логин и пароль, нажать «Войти». | 1. Отображение главной страницы. 2. Открытие формы авторизации. 3. Переход на главную страницу с меню «Записи» или сообщение об ошибке. |
| **TC-2** | Создать записи клиента | 1. Выбрать пункт меню «Записи». 2. Нажать «Новая запись». 3. Заполнить данные клиента (имя, услуга, мастер, время). 4. Подтвердить запись. | 1. Открытие формы списка записей. 2. Отображение формы создания записи. 3. Добавление записи в систему с подтверждением или сообщением об ошибке. |
| **TC-3** | Редактировать записи клиента | 1. Выбрать пункт меню «Записи». 2. Найти запись клиента (поиск по имени/дате). 3. Нажать «Изменить». 4. Внести изменения, сохранить. | 1. Отображение списка записей. 2. Найденная запись выделена. 3. Открытие формы редактирования. 4. Обновленные данные сохраняются или выводится ошибка. |
| **TC-4** | Отменить записи клиента | 1. Выбрать пункт меню «Записи». 2. Найти запись клиента. 3. Нажать «Отменить». 4. Подтвердить отмену. | 1. Отображение списка записей. 2. Запись выделена. 3. Появление диалога подтверждения. 4. Запись исчезает из системы или выводится ошибка. |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор тест-кейса | Название тест-кейса | Шаги тест-кейса | Ожидаемый результат |
| **TC-5** | Добавить нового мастера | 1. Выбрать пункт меню «Мастера». 2. Нажать «Добавить мастера». 3. Заполнить данные (ФИО, специализация, график). 4. Сохранить. | 1. Открытие списка мастеров. 2. Отображение формы добавления. 3. Новый мастер появляется в списке или выводится ошибка. |
| **TC-6** | Назначить услуги мастеру | 1. Выбрать пункт меню «Услуги». 2. Выбрать услугу (например, «Стрижка»). 3. Нажать «Назначить мастеру». 4. Выбрать мастера из списка, сохранить. | 1. Открытие списка услуг. 2. Услуга выделена. 3. Открытие формы выбора мастера. 4. Услуга привязывается к мастеру или выводится ошибка. |
| **TC-7** | Формировать отчеты по выручке | 1. Выбрать пункт меню «Отчеты». 2. Указать период (например, «За месяц»). 3. Нажать «Сформировать». | 1. Открытие формы отчетов. 2. Отображение отчета с выручкой за выбранный период или сообщение об ошибке. |
| **TC-8** | Проверить графики работы мастера | 1. Выбрать пункт меню «Мастера». 2. Выбрать мастера, нажать «График». 3. Указать дату, нажать «Показать». | 1. Открытие списка мастеров. 2. Отображение графика выбранного мастера. 3. Показ занятых/свободных слотов на указанную дату. |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор тест-кейса | Название тест-кейса | Шаги тест-кейса | Ожидаемый результат |
| **TC-9** | Добавить отзыва клиента | 1. Выбрать пункт меню «Клиенты». 2. Найти клиента, нажать «Оставить отзыв». 3. Ввести текст, оценку, сохранить. | 1. Открытие списка клиентов. 2. Отображение формы отзыва. 3. Отзыв сохраняется в системе или выводится ошибка. |
| **TC-10** | Редактировать прайс-листа | 1. Выбрать пункт меню «Услуги». 2. Нажать «Редактировать прайс». 3. Изменить цены, сохранить. | 1. Открытие списка услуг. 2. Отображение формы редактирования. 3. Обновленные цены сохраняются или выводится ошибка. |

Все тест-кейсы программного средства пройдены успешно. Полученные в ходе тестирования результаты соответствуют ожидаемым. Все функциональные требования к программному средству выполняются. Следовательно, разработанное программное средство удовлетворяет требованиям задания на разработку.

# 5 Руководство по установке (развертыванию) и использованию программного средства

## 5.1 Руководство по установке (развертыванию) программного средства

1 Необходимые программы и компоненты. Для успешной установки и работы программного средства для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа, необходимо наличие следующих программ:

* операционная система семейства *Windows*;
* сервер баз данных *PostgreSQL*;
* среда разработки *Intellij IDEA*.

2 Последовательность установки. Для установки разработанного программного средства необходимо выполнить следующие шаги:

– назначить права для пользователя, под которым будет проводиться установка и настройка;

* установить и настроить *Java*;
* установить и настроить сервер баз данных *PostgreSQL*;
* установить и настроить среду разработки *Intellij IDEA*;
* установить и настроить разработанное программное средство.

3 Состав дистрибутива. В поставляемый конечному пользователю дистрибутив входят следующие элементы:

* скрипт генерации пустой базы данных;
* установочный пакет сервера баз данных Postgre*SQL*;
* установочный пакет *Intellij IDEA*;
* файл программного средства с расширением «.*java*».

4 Распаковка дистрибутива. Распаковка дистрибутива имеет место быть только в случае, если установка программного средства осуществляется в определенные системные директории операционной системы. Если распаковка дистрибутива осуществляется в новую или создаваемую при установке дистрибутива директорию, то данный раздел руководства можно опустить.

В данном случае предполагается, что установка и запуск программного средства не требуют распаковки дистрибутива в системные директории.

5 Восстановление базы данных из резервной копии. Для восстановления базы данных из резервной копии необходимо выполнить следующие действия:

* открыть командную строку и подключится к базе данных *PostgreSQL* с помощью учетных данных суперпользователя;
* необходимо создать пустую базу данных;

CREATE DATABASE barbershop;

* необходимо выполнить команду для восстановления базы данных из скрипта, предполагая, что файл скрипта находится в указанной директории;

pg\_restore -d barbershop -U root barbershop.dump

* ввести пароль и дождаться завершения процесса восстановления.

После успешного выполнения скрипта база данных «*barbershop*» будет восстановлена.

6 Проверка работоспособности программного средства. Для проверки работоспособности программного обеспечения необходимо:

–  убедиться, что сервис *PostgreSQL* запущен;

–  используя *Intellij IDEA* подключить бд, с помощью имени пользователя, пароля, названия бд.

В случае успеха при запуске приложения откроется главная страница программного средства.

## 5.2 Руководство пользователя

Данный раздел представляет собой руководство по эксплуатации разработанной системы, для ее установки требуется операционная система Windows 7 и выше, интерфейс программы и все данные предоставлены на русском языке.

При успешном запуске программы появится страница регистрации/авторизации. Форма авторизации представлена на рисунке 5.1.

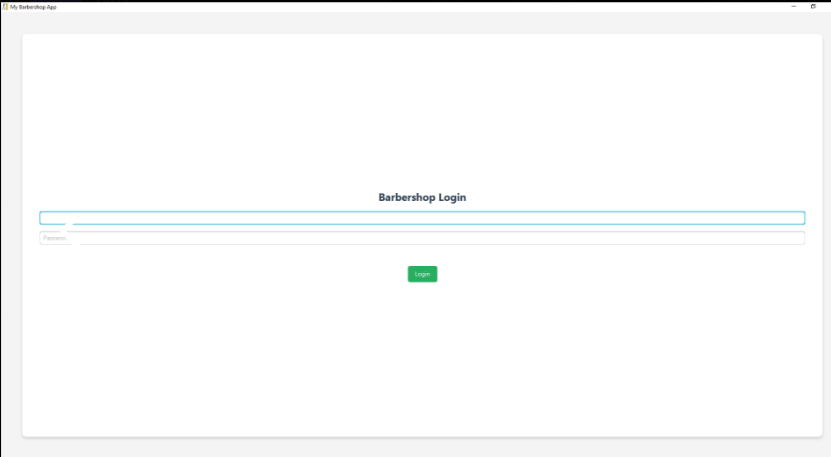


Рисунок 5.1 – Форма авторизации

При входе как администратор, он имеет права добавлять менеджера и редактировать его характеристики, что представлено на рисунке 5.2 и 5.3.

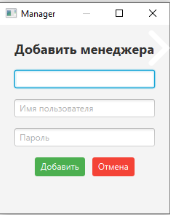


Рисунок 5.2 – Форма добавить менеджера

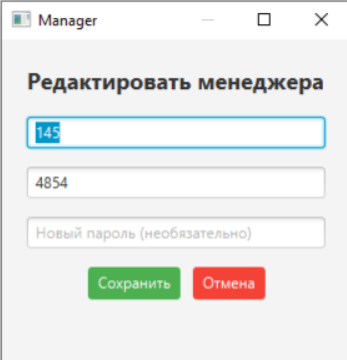


Рисунок 5.3 – Форма редактировать менеджера

Также, администратор имеет право управлять пользователями, а точнее добавлять нового, удалять и редактировать их характеристики (логин, пароль), что можно увидеть на рисунке 5.4. Также он может посмотреть сколько всего пользователей зарегистрировано и дату входа.

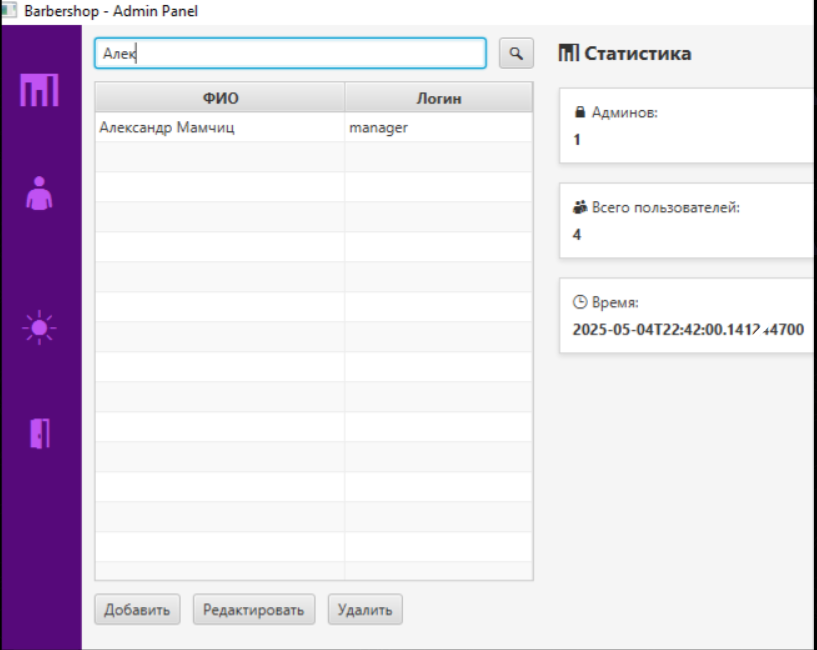


Рисунок 5.4 – Форма управления пользователями

При добавлении нескольких пользователей, можно заметить изменения в статистике, представленные на рисунке 5.5.

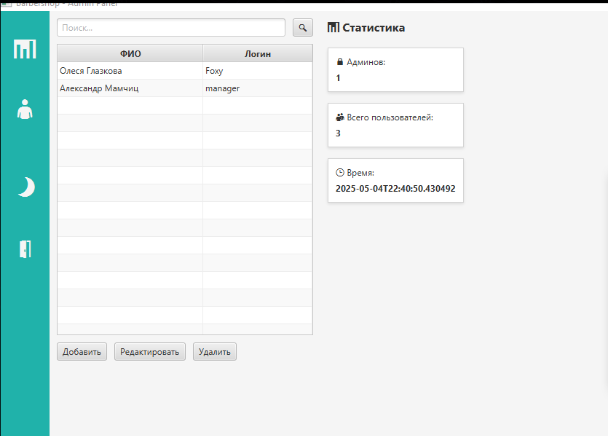


Рисунок 5.5 – Форма статистики

При входе как сотрудник барбершопа можно увидеть статистику, в нем изображается количество менедженров и клиентов, количнство записей в барбершоп, доход за неделю, а также график количества клиентов по дням недели, и диаграмма, которая показывает процент посещаемости той или иной услуги, все это показано на рисунке 5.6.

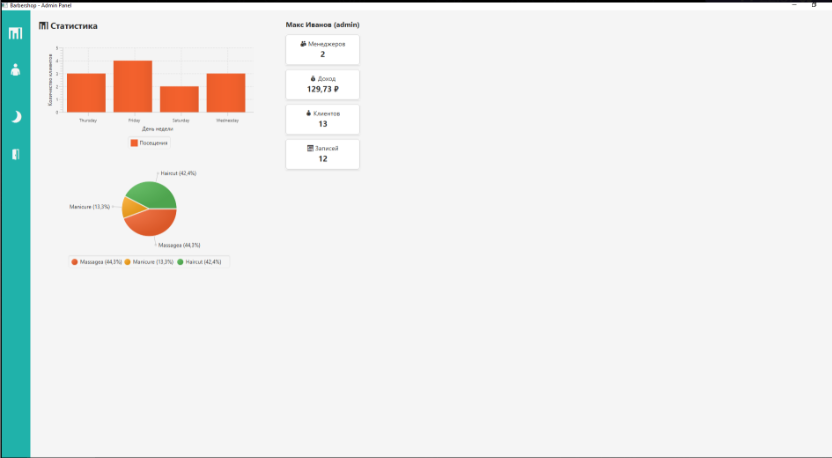


Рисунок 5.6 – Форма статистики у менеджера

Таким образом в главе были описаны возможности разработанной системы, использованной с разных сторон администратора и менеджера. Как было подмечено их функционал довольно отличается. Функционал данного приложения удобен благодаря проработанному интерфейсу.

Таким образом, выполнены проектирование и разработка программного средства для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа. Выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработана информационная модель системы, представленная в виде схемы базы данных; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование системы.

# Заключение

В результате работы над курсовым проектом, целью которого было улучшение качества взаимодействия с пользователями барбершопа за счет сокращения времени, затрачиваемого на регистрацию данных об оказанных услугах и посещаемости, был проведен всесторонний анализ ключевых элементов взаимодействия с клиентами. Рассмотренные принципы и подходы подтвердили актуальность выбранной темы.

В ходе работы были сформулированы функциональные и нефункциональные требования, разработана архитектура системы, реализующая клиент-серверную модель взаимодействия с использованием многопоточного сервера, базы данных PostgreSQL и клиентского интерфейса на JavaFX.

Также был проведен анализ функциональных требований к создаваемому программному средству, разработана информационная модель системы и различные архитектурные решения.

Разработанный программный продукт для автоматизации учёта новых поступлений в библиотеку обладает следующим рядом преимуществ:

* выполнен в архитектуре клиент-сервер;
* имеет понятный и простой в использовании интерфейс;
* предусматривает возможность осуществления авторизации в системе;
* разделяет функционал для пользователей в соответствии с их ролями;
* адаптирован под специфику работы барбершопа и учитывает его потребности.

Программное средство протестировано и готово к использованию, для него также было составлено подробное руководство пользователя.

Разработанный программный продукт представляет практическую ценность для барбершопов, так как позволяет сократить трудозатраты на учет данных, повысить точность хранения информациив, ускорить обработку информации и тем самым увеличить продуктивность работы сотрудников.

# Список использованных источников

1. Ашихин, П. 12 нот «Парфюмера», 2018. – 52-100 с.
2. Гоуинг, Т. Борода и философия, 2016. – 101 с.
3. IDEF0. Знакомство с нотацией и пример использования [Электронный ресурс]. − Режим доступа : <https://www.trinion.org/blog/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya>
4. Фаулер, М. Архитектура корпоративных программных приложений / М. Фаулер. – М.: Вильямс, 2016 – 544 с.
5. Лукин, И.К. Сравнительный анализ языков программирования для разработки веб-приложений / И.К. Лукин. – Самара: Поволожский Государственный Университет Телекоммуникаций и Информатики, 2017. – 381-384 с.
6. Сергеев, В. И., Григорьев, М. Н., Уваров, С. А. Логистика: информационные системы и технологии / В. И. Сергеев, М. Н. Григорьев, С. А. Уваров. – М.: Альфа-Пресс, 2018 – 355 с
7. ТРПО – Курсовое проектирование (работа) – Пособие [Электронный ресурс].
8. И.Н. Блинов, В.С. Романчик, «Java. Методы программирования» Минск: издательство «Четыре четверти», 2013. – 896 с.
9. Диаграмма классов [Электронный ресурс]. – Режим доступа: <https://flexberry.github.io/ru/gpg_class-diagram.html>
10. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин; пер. с англ. – СПб.: Питер, 2019. – 432 с.
11. Жоголев, Е.А. Технология программирования / Е.А. Жоголев – М.: Научный мир, 2004. – 216 с.

# Приложение А

**(обязательное)**

**Отчет о проверке на заимствования в системе «Антиплагиат»**

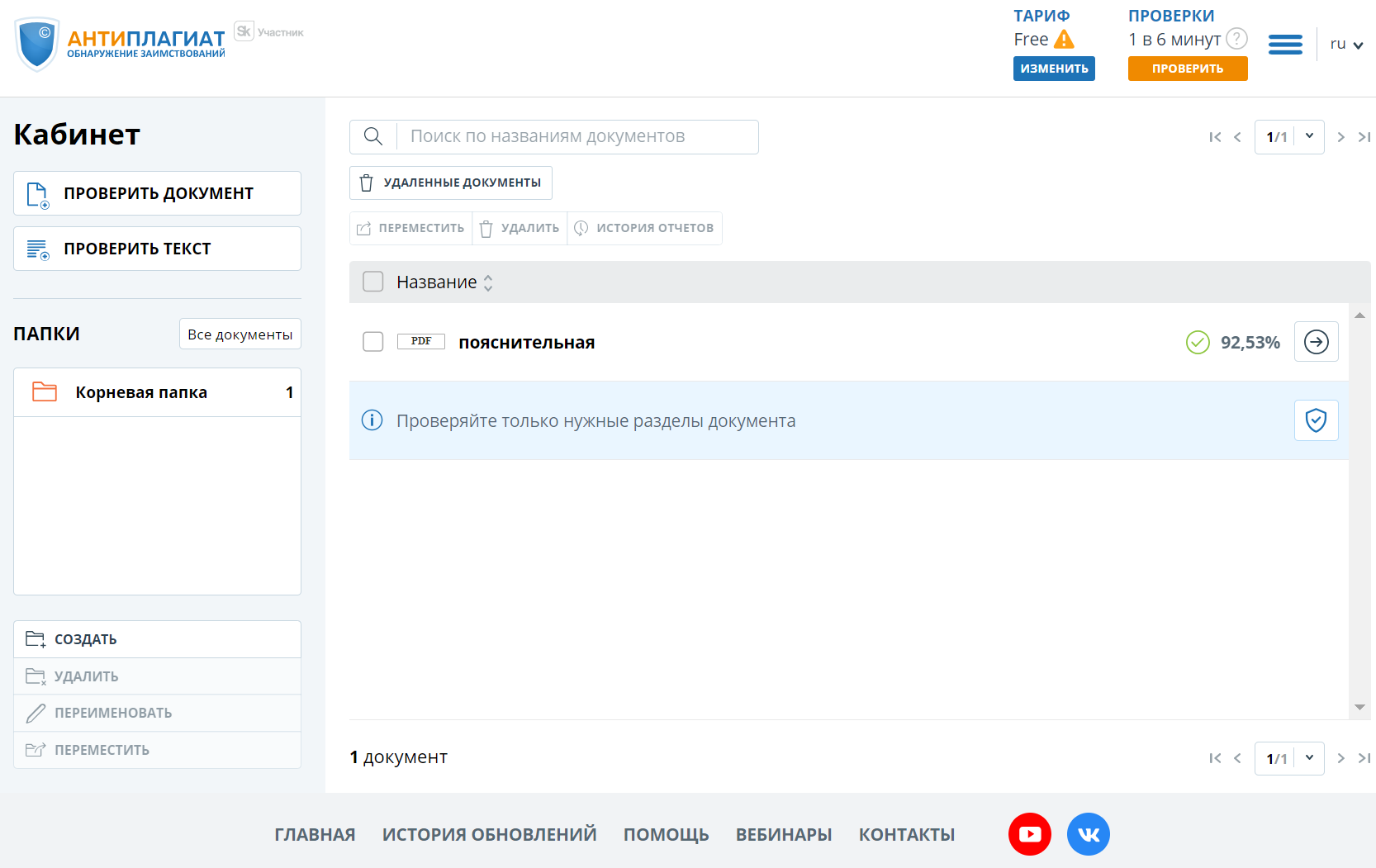


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# Приложение Б

**(обязательное)**

**Листинг кода алгоритмов, реализующих бизнес-логику приложения**

**Файл AddmanagerModalController**

package com.barbershop.controllers.admin;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
  
public class AddManagerModalController {  
  
 @FXML private TextField fullNameField;  
 @FXML private TextField usernameField;  
 @FXML private PasswordField passwordField;  
 @FXML private Button addButton;  
 @FXML private Button cancelButton;  
  
 @FXML  
 public void initialize() {  
 addButton.setOnAction(e -> handleAdd());  
 cancelButton.setOnAction(e -> ((Stage) cancelButton.getScene().getWindow()).close());  
 }  
  
 private void handleAdd() {  
 String fullName = fullNameField.getText();  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
  
 if (fullName.isEmpty() || username.isEmpty() || password.isEmpty()) {  
 showAlert("Ошибка", "Все поля должны быть заполнены");  
 return;  
 }  
  
 try (Connection conn = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/barbershop", "postgres", "1234")) {  
 PreparedStatement stmt = conn.prepareStatement("INSERT INTO users (full\_name, username, password, role) VALUES (?, ?, ?, 'manager')");  
 stmt.setString(1, fullName);  
 stmt.setString(2, username);  
 stmt.setString(3, password);

Продолжение приложения Б

stmt.executeUpdate();  
  
 ((Stage) addButton.getScene().getWindow()).close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 showAlert("Ошибка", "Не удалось добавить менеджера");  
 }}  
 private void showAlert(String title, String message) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
}

**Файл DB**

package com.barbershop.controllers.database;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
import static java.sql.DriverManager.*getConnection*;  
  
public class DB {  
 private static final String *DB\_URL* = "jdbc:postgresql://localhost:5432/barbershop";  
 private static final String *DB\_USER* = "postgres";  
 private static final String *DB\_PASSWORD* = "1234";  
  
 private static Connection *transactionConnection* = null;  
 public static Connection *conn*;  
  
  
  
 */\*\* CREATE a DATABASE.  
 \* @param dbName Database name  
 \*/* public static void createDB(String dbName) {  
 // В PostgreSQL база данных обычно создается через отдельное соединение  
 try (Connection connection = *getConnection*("jdbc:postgresql://localhost:5432/postgres", *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
 statement.executeUpdate("CREATE DATABASE " + dbName.toLowerCase());  
 System.*out*.println("Database created successfully.");  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());

Продолжение приложения Б

}  
 }  
  
  
 public static void startTransaction() throws SQLException {  
 if (*transactionConnection* != null) {  
 throw new SQLException("Transaction already in progress");  
 }  
 *transactionConnection* = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 *transactionConnection*.setAutoCommit(false);  
 }  
  
 public static void commitTransaction() throws SQLException {  
 if (*transactionConnection* != null) {  
 *transactionConnection*.commit();  
 *transactionConnection*.close();  
 *transactionConnection* = null;  
 }  
 }  
  
 public static void rollbackTransaction() {  
 try {  
 if (*transactionConnection* != null) {  
 *transactionConnection*.rollback();  
 *transactionConnection*.close();  
 *transactionConnection* = null;  
 }  
 } catch (SQLException e) {  
 System.*out*.println("Error rolling back transaction: " + e.getMessage());  
 }  
 }  
  
 */\*\* CREATE a TABLE in a database  
 \* @param dbName database name  
 \* @param tableName table name  
 \* @param attributes table attributes  
 \*/* public static void createTable(String dbName, String tableName, String attributes) {  
 String query = "CREATE TABLE IF NOT EXISTS " + tableName.toLowerCase() + " (" + attributes + ");";  
 try (Connection connection = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
 statement.execute(query);  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());  
 }  
 }  
  
 */\*\* INSERT a ROW in a specific table in a database.  
 \* @param dbName database name*

Продолжение приложения Б

*\* @param tableName table name  
 \* @param values values to be inserted  
 \* @return last inserted id  
 \*/* public static int insertRow(String dbName, String tableName, String values) {  
 // Изменено "RETURNING id" на "RETURNING client\_id" для таблицы clients  
 String returningColumn = tableName.equalsIgnoreCase("clients") ? "client\_id" : "id";  
 String query = "INSERT INTO " + tableName.toLowerCase() + " VALUES " + values + " RETURNING " + returningColumn + ";";  
  
 try (Connection connection = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet rs = statement.executeQuery(query)) {  
 if (rs.next()) {  
 return rs.getInt(1);  
 }  
 return -1;  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());  
 return -1;  
 }  
 }  
  
 */\*\* UPDATE a ROW in a specific table in a database.  
 \* @param dbName database name  
 \* @param tableName table name  
 \* @param setClause new values  
 \* @param conditions where the values will be updated condition  
 \*/* public static void updateRow(String dbName, String tableName, String setClause, String conditions) {  
 String query = "UPDATE " + tableName.toLowerCase() + " SET " + setClause + " WHERE " + conditions + ";";  
 try (Connection connection = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
 statement.executeUpdate(query);  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());  
 }  
 }  
  
 */\*\* DELETE a ROW from a specific table in a database.  
 \* @param dbName database name  
 \* @param tableName table name  
 \* @param conditions conditions to identify the row(s) to delete  
 \*/* public static void deleteRow(String dbName, String tableName, String conditions) {

Продолжение приложения Б

String query = "DELETE FROM " + tableName.toLowerCase() + " WHERE " + conditions + ";";  
 try (Connection connection = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement()) {  
 statement.executeUpdate(query);  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());  
 }  
 }  
  
 */\*\* SELECT values from a specific table in a database.  
 \* @param dbName database name  
 \* @param tableName table name  
 \* @param attributes attibutes to be selected from the table  
 \* @param conditions where condition  
 \* @return List of values selected  
 \*/* public static List<List<String>> selectRow(String dbName, String tableName, String attributes, String conditions) {  
 List<List<String>> resultList = new ArrayList<>();  
 String query = "SELECT " + attributes + " FROM " + tableName.toLowerCase();  
 if (!conditions.isEmpty()) {  
 query += " " + conditions;  
 }  
 query += ";";  
  
 try (Connection connection = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement statement = connection.createStatement();  
 ResultSet resultSet = statement.executeQuery(query)) {  
  
 ResultSetMetaData metaData = resultSet.getMetaData();  
 int columnCount = metaData.getColumnCount();  
  
 while (resultSet.next()) {  
 List<String> row = new ArrayList<>();  
 for (int i = 1; i <= columnCount; i++) {  
 row.add(resultSet.getString(i));  
 }  
 resultList.add(row);  
 }  
 } catch (SQLException e) {  
 System.*out*.println("PostgreSQL ERROR: " + e.getMessage());  
 }  
 return resultList;  
 }  
  
 public static int executeInsertWithReturnId(String sql, Object... params) throws SQLException {  
 try (Connection conn = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 PreparedStatement pstmt = conn.prepareStatement(sql)) {

Продолжение приложения Б

for (int i = 0; i < params.length; i++) {  
 pstmt.setObject(i + 1, params[i]);  
 }  
  
 try (ResultSet rs = pstmt.executeQuery()) {  
 if (rs.next()) {  
 return rs.getInt(1);  
 }  
 }  
 }  
 throw new SQLException("Failed to get generated ID");  
 }  
  
 public static ResultSet select(String query) {  
 try {  
 Connection conn = *getConnection*(*DB\_URL*, *DB\_USER*, *DB\_PASSWORD*);  
 Statement stmt = conn.createStatement();  
 return stmt.executeQuery(query);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 return null;  
 }  
 }  
  
}

**Файл Service**

package com.barbershop.models;  
  
public class Service {  
 private int service\_id;  
 private String name;  
 private String description;  
 private double price;  
  
 public Service() {}  
 //get from database  
 public Service(int service\_id, String name, String description, double price) {  
 this.service\_id = service\_id;  
 this.name = name;  
 this.description = description;  
 this.price = price;  
 }  
 //insert in database  
 public Service(String name, String description, double price) {  
 this.name = name;  
 this.description = description;  
 this.price = price;  
 }

Продолжение приложения Б

public int getServiceId() {  
 return service\_id;  
 }  
 public void setServiceId(int service\_id) {  
 this.service\_id = service\_id;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getDescription() {  
 return description;  
 }  
 public void setDescription(String description) {  
 this.description = description;  
 }  
 public double getPrice() {  
 return price;  
 }  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return "Name: "+ name + "\nDescription: "+ description + "\nPrice: "+ price;  
 }  
/\* public String toString() {  
 return "Service [service\_id=" + service\_id + ", name=" + name + ", description=" + description + ", price=" + price + "]";  
 } \*/  
}

**Файл UserSession**

package com.barbershop.models;  
  
public class UserSession {  
 private static String *username*;  
 private static String *role*;  
 private static String *fullName*;  
  
 public static void set(String user, String r, String name) {  
 *username* = user;  
 *role* = r;  
 *fullName* = name;  
 }  
  
 public static String getUsername() {

Продолжение приложения Б

return *username*;  
 }  
  
 public static String getRole() {  
 return *role*;  
 }  
  
 public static String getFullName() {  
 return *fullName*;  
 }  
  
 public static void clear() {  
 *username* = null;  
 *role* = null;  
 *fullName* = null;  
 }  
}

**Файл InvoiceProduct**

package com.barbershop.models;  
  
public class InvoiceProduct {  
 private Product product;  
 private int quantity;  
 private Double price;  
  
 public InvoiceProduct(Product product) {  
 this.product = product;  
 this.quantity = 1;  
 this.price = product.getPrice()\*quantity;  
 }  
  
 public Product getProduct() {  
 return product;  
 }  
  
 public void setProduct(Product product) {  
 this.product = product;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 this.price = this.product.getPrice()\*quantity;  
 }  
  
 public Double getPrice() {  
 return price;  
 }

Продолжение приложения Б

public void setPrice(Double price) {  
 this.price = price;  
 }  
  
 @Override  
 public String toString() {  
 return product.getName() + " "+ price +" DHs"  
 +"\n"+ product.getPrice() + " x"+ quantity;  
 }  
}

**Файл PatternController**

package com.barbershop.controllers.patterns;  
  
import java.util.function.UnaryOperator;  
import java.util.regex.Pattern;  
  
import javafx.scene.control.TextFormatter.Change;  
  
public class PaternController {  
 public static String clearText(String input) {  
 return input.replaceAll("\\D+", "");  
 }  
  
 @SuppressWarnings("exports")  
 public static UnaryOperator<Change> createPatternFilter(String pattern) {  
 Pattern digitsPattern = Pattern.*compile*("\\d\*");  
 final int maxDigits = pattern.replaceAll("[^#]\*", "").length();  
 return change -> {  
 String text = change.getText();  
 if (!digitsPattern.matcher(text).matches()) {  
 return null; // prevent inputs other than digits  
 }  
 if (change.getControlText().equals(change.getControlNewText())) {  
 return change; // allow all changes not modifying the text  
 }  
  
 String clearText = *clearText*(change.getControlNewText());  
 String clearPrefix = *clearText*(change.getControlNewText().substring(0, change.getAnchor()));  
 final int prefixLength = clearPrefix.length();  
 if (clearText.length() > maxDigits) {  
 if (prefixLength > maxDigits) {  
 return null; // cursor already positioned after the last digit placeholder  
 }  
 clearText = clearText.substring(0, maxDigits); // cut of excessive digits  
 }  
 StringBuilder resultText = new StringBuilder(pattern.length());  
 int index = 0;

Продолжение приложения Б

int prefixIndex = 0;  
  
 // copy parts digits before the cursor  
 while (prefixIndex < prefixLength) {  
 char c = pattern.charAt(index);  
 if (c == '#') {  
 resultText.append(clearPrefix.charAt(prefixIndex));  
 prefixIndex++;  
 } else {  
 resultText.append(c);  
 }  
 index++;  
 }  
  
 // deal with following non-digit placeholders  
 char c;  
 while (index < pattern.length() && (c = pattern.charAt(index)) != '#') {  
 resultText.append(c);  
 index++;  
 }  
  
 int newAnchor = resultText.length();  
 String clearSuffix = clearText.substring(prefixLength);  
 int suffixIndex = 0;  
  
 // copy remaining digits  
 while (index < pattern.length() && suffixIndex < clearSuffix.length()) {  
 c = pattern.charAt(index);  
 if (c == '#') {  
 resultText.append(clearSuffix.charAt(suffixIndex));  
 suffixIndex++;  
 } else {  
 resultText.append(c);  
 }  
 index++;  
 }  
  
 resultText.append(pattern.substring(index));  
 change.setRange(0, change.getControlText().length());  
 change.setText(resultText.toString());  
 change.selectRange(newAnchor, newAnchor);  
  
 return change;  
 };  
 }  
  
 // Validation methods  
 public static boolean isValidName(String name) {  
 // Validate that the name is not null or empty and does not contain digits, spaces, or special characters  
 return name != null && !name.isEmpty() && !name.matches(".\*[0-9 ].\*")

Продолжение приложения Б

&& !name.matches(".\*[^a-zA-Z0-9 ].\*");  
 }  
   
 public static boolean isValidPhoneNumber(String phone) {  
 // Validate that the phone number is not null, not empty, and is a 10-digit numeric string  
 return phone != null && !phone.isEmpty() && phone.matches("\\d{9}");  
 }  
  
 public static String capitalize(String str) {  
 if (str == null || str.isEmpty()) {  
 return str;  
 }  
 return str.substring(0, 1).toUpperCase() + str.substring(1).toLowerCase();  
 }  
}

# Приложение В

**(обязательное)**

**Листинг скрипта генерации базы данных**

-- Создание базы данных

CREATE DATABASE barbershop

WITH

OWNER = postgres

ENCODING = 'UTF8'

CONNECTION LIMIT = -1;

-- Подключение к базе данных

\c barbershop

-- Таблица пользователей системы

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(100) NOT NULL,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

phone VARCHAR(20),

role VARCHAR(20) NOT NULL CHECK (role IN ('ADMIN', 'MANAGER')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Таблица клиентов

CREATE TABLE clients (

client\_id SERIAL PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

phone VARCHAR(20) NOT NULL UNIQUE,

notes TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Таблица услуг

CREATE TABLE services (

service\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

description TEXT,

duration INTERVAL NOT NULL,

price NUMERIC(10, 2) NOT NULL,

active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Таблица товаров

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

Продолжение приложения В

description TEXT,

quantity INTEGER NOT NULL DEFAULT 0,

price NUMERIC(10, 2) NOT NULL,

active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Таблица записей (событий)

CREATE TABLE events (

event\_id SERIAL PRIMARY KEY,

client\_id INTEGER REFERENCES clients(client\_id),

user\_id INTEGER REFERENCES users(user\_id),

start\_time TIMESTAMP NOT NULL,

end\_time TIMESTAMP NOT NULL,

status VARCHAR(20) NOT NULL CHECK (status IN ('SCHEDULED', 'COMPLETED', 'CANCELLED', 'NOSHOW')),

notes TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Связь многие-ко-многим между событиями и услугами

CREATE TABLE event\_services (

event\_id INTEGER REFERENCES events(event\_id),

service\_id INTEGER REFERENCES services(service\_id),

PRIMARY KEY (event\_id, service\_id)

);

-- Таблица счетов

CREATE TABLE invoices (

invoice\_id SERIAL PRIMARY KEY,

event\_id INTEGER REFERENCES events(event\_id),

client\_id INTEGER REFERENCES clients(client\_id),

user\_id INTEGER REFERENCES users(user\_id),

invoice\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

subtotal NUMERIC(10, 2) NOT NULL,

tax NUMERIC(10, 2) NOT NULL,

discount NUMERIC(10, 2) DEFAULT 0,

total NUMERIC(10, 2) NOT NULL,

payment\_method VARCHAR(50),

status VARCHAR(20) NOT NULL CHECK (status IN ('PENDING', 'PAID', 'CANCELLED'))

);

-- Связь многие-ко-многим между счетами и услугами

CREATE TABLE invoice\_services (

invoice\_id INTEGER REFERENCES invoices(invoice\_id),

service\_id INTEGER REFERENCES services(service\_id),

quantity INTEGER DEFAULT 1,

price NUMERIC(10, 2) NOT NULL,

PRIMARY KEY (invoice\_id, service\_id)

);

-- Связь многие-ко-многим между счетами и товарами

Продолжение приложения В

CREATE TABLE invoice\_products (

invoice\_id INTEGER REFERENCES invoices(invoice\_id),

product\_id INTEGER REFERENCES products(product\_id),

quantity INTEGER NOT NULL,

price NUMERIC(10, 2) NOT NULL,

PRIMARY KEY (invoice\_id, product\_id)

);

-- Таблица для хранения сессий пользователей

CREATE TABLE user\_sessions (

session\_id VARCHAR(100) PRIMARY KEY,

user\_id INTEGER REFERENCES users(user\_id),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

expires\_at TIMESTAMP NOT NULL

);

-- Индексы для улучшения производительности

CREATE INDEX idx\_events\_client ON events(client\_id);

CREATE INDEX idx\_events\_user ON events(user\_id);

CREATE INDEX idx\_events\_time ON events(start\_time, end\_time);

CREATE INDEX idx\_invoices\_client ON invoices(client\_id);

CREATE INDEX idx\_invoices\_event ON invoices(event\_id);

-- Начальные данные

INSERT INTO users (username, password, first\_name, last\_name, role)

VALUES

('admin', '$2a$10$xJwL5vxZ4U3z7qOYQ3vj.eW.7XjZ7N6c1uJdJ6Y1wV2KkQ9m6bXe', 'Admin', 'Adminov', 'ADMIN'),

('manager1', '$2a$10$yH2eZ3vX4W5z6A7B8C9D.eF0G1H2I3J4K5L6M7N8O9P0Q1R2S3T4U', 'Ivan', 'Petrov', 'MANAGER');

INSERT INTO services (name, description, duration, price)

VALUES

('Стрижка мужская', 'Классическая мужская стрижка', '00:45:00', 30.00),

('Стрижка детская', 'Стрижка для детей до 12 лет', '00:30:00', 25.00),

('Бритье головы', 'Полное бритье головы с уходом', '00:40:00', 35.00);

INSERT INTO products (name, description, quantity, price)

VALUES

('Гель для волос', 'Укладочный гель 200мл', 50, 15.00),

('Воск для бороды', 'Воск для укладки бороды 50мл', 30, 20.00),

('Шампунь', 'Шампунь для волос 300мл', 40, 25.00);

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | | | *Дополнитель-ные сведения* | | |
|  | | | | | Текстовые документы | | |  | | |
|  | | | | |  | | |  | | |
| БГУИР КП 1-40 05 01-02 083 ПЗ | | | | | Пояснительная записка | | | 90 с. | | |
|  | | | | |  | | |  | | |
|  | | | | | Графические документы | | |  | | |
|  | | | | |  | | |  | | |
| ГУИР.425196.001ПД | | | | | Схема алгоритма клиент-серверного взаимодействия | | | Формат А4 | | |
| ГУИР.425196.002ПД | | | | | Описание процесса «Привлечь новых клиентов на пошив и поставку спецодежды» | | | Формат А4 | | |
| ГУИР.425196.003ПЛ | | | | | Плакат Диаграмма классов приложения для расчета стоимости изделий | | | Формат А4 | | |
| ГУИР.425196.004ПЛ | | | | | Плакат Модели представления программного средства для расчета стоимости изделий | | | Формат А4 | | |
| ГУИР.425196.005ПЛ | | | | | Плакат Результаты проектирования приложения для расчета стоимости изделий | | | Формат А4 | | |
|  | | | | |  | | |  | | |
|  | | | | |  | | |  | | |
|  |  |  |  |  | БГУИР КП 1-40 05 01-02 083 Д1 | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Программное средство для анализа данных об оказанных услугах, посещаемости и эффективности работы сотрудников барбершопа  Ведомость курсового  проекта | Лит | | | Лист | Листов |
| Разраб. | | Хрущев |  |  |  | Т |  | 1 | 1 |
| Пров. | | Снопок |  |  | Кафедра ЭИ  гр. 373601 | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |