МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Институт «Информатика и вычислительная техника»

Кафедра «Автоматизированные системы обработки информации и управления»

Работа защищена с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Программная инженерия»

на тему «Автоматизация деятельности салона красоты»

|  |  |
| --- | --- |
| Выполнил  студент гр. Б23-784-1 | Асанова К.А.  Зинатулина С.Р.  Сычугова А.А. |
|  |  |
| Принял  к.т.н., доцент каф. АСОИУ | Касимов Д.Р. |

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc185853602)

[ВВЕДЕНИЕ 4](#_Toc185853603)

[1. Выявление и анализ требований к разрабатываемому приложению 5](#_Toc185853604)

[1.1 Функции продукта. 5](#_Toc185853605)

[1.2 Обзор модели объектов. 5](#_Toc185853606)

[1.3 Характеристика пользователей. 5](#_Toc185853607)

[1.4 Обзор Use-Case модели. 6](#_Toc185853608)

[1.5 Специфические требования. 8](#_Toc185853609)

[1.5.1 Функциональные требования. 8](#_Toc185853610)

[1.5.2 Нефункциональные требования. 8](#_Toc185853611)

[1.6 Интерфейсы. 9](#_Toc185853612)

[2. Планирование программного проекта 11](#_Toc185853613)

[2.1 Бэклог продукта 11](#_Toc185853614)

[2.2 Планирование спринтов 13](#_Toc185853615)

[3. Проектирование программного продукта 15](#_Toc185853616)

[3.1 UML-диаграмма классов 15](#_Toc185853617)

[3.2 UML-диаграмма последовательности 16](#_Toc185853618)

[3.3 UML-диаграмма состояний 18](#_Toc185853619)

[4. Тестирование 20](#_Toc185853620)

[4.1 Чек-лист 20](#_Toc185853621)

[4.2 Дефекты (баги) 20](#_Toc185853622)

[4.2.1 Ожидаемое и реальное поведение. 21](#_Toc185853623)

[4.2.2 Приоритет и серьезность дефекта. 21](#_Toc185853624)

[4.2.3 Примеры багов 22](#_Toc185853625)

[4.2.4 Оценка срока исправления 23](#_Toc185853626)

[4.2.5 Отчет о дефекте. 23](#_Toc185853627)

[4.3 Тест-кейсы. 26](#_Toc185853628)

[4.4 Описание программного кода. 28](#_Toc185853629)

[5. Оценка качества проекта 30](#_Toc185853630)

[ЗАКЛЮЧЕНИЕ 31](#_Toc185853631)

[СПИСОК ЛИТЕРАТУРЫ 32](#_Toc185853632)

[Приложение 33](#_Toc185853633)

# **ВВЕДЕНИЕ**

Современные технологии играют важную роль в повышении эффективности работы бизнеса, включая салоны красоты. В условиях быстро меняющегося рынка и растущей конкуренции использование специализированного программного обеспечения становится не просто преимуществом, а настоящей необходимостью для успешного функционирования таких учреждений. Цель данной курсовой работы — разработка программного обеспечения (ПО) для автоматизации процессов управления клиентами и услугами в салоне красоты.

Система, которая будет разработана, предназначена для улучшения взаимодействия с клиентами, обеспечивая более высокий уровень сервиса и поддержки. Она также автоматизирует процесс записи на услуги, что, безусловно, значительно упрощает жизнь как клиентам, так и сотрудникам. Вдобавок к этому, система предоставит доступ к информации о мастерах и их портфолио, что дает клиентам возможность принимать более обоснованные решения при выборе специалиста. Основные функциональные возможности системы будут включать:

1. Запись на услуги: Клиенты смогут удобно и быстро выбирать интересующие их услуги и записываться на них в оптимальное для себя время, что значительно уменьшит время ожидания и упростит организацию рабочего процесса.
2. Портфолио мастеров: Возможность просмотра работ мастеров будет полезной для клиентов, позволяя им оценить качество выполняемых услуг и сделать более осознанный выбор, основываясь на реальных примерах.
3. Управление расписанием: Мастера смогут эффективно управлять своим временем и графиком работы, что позволит улучшить организацию рабочего процесса и избежать накладок или несоответствий в расписании.

Документ содержит спецификацию требований к разрабатываемой системе, включая функциональные и нефункциональные требования, технические характеристики и основные критерии качества. Это руководство предназначено для разработчиков, тестировщиков и других участников проекта, позволяя всем заинтересованным сторонам четко понимать, как должна функционировать система и какие результаты необходимо достичь.

Таким образом, разработка данной системы необходима для повышения качества обслуживания клиентов, оптимизации рабочих процессов и увеличения прибыли салона красоты. В конечном итоге, интеграция таких технологий поможет не только упростить работу сотрудников, но и создать более комфортные условия для клиентов, что в свою очередь приведет к росту лояльности и удовлетворенности потребителей.

1. **Выявление и анализ требований к разрабатываемому приложению**

На этом этапе происходит сбор и систематизация требований к приложению от всех заинтересованных сторон. Это критически важный процесс, так как правильно определенные требования являются основой для успешной разработки.

* 1. **Функции продукта**

1) Предложение услуг мастеров

2) Ознакомление клиента с портфолио мастера

3) Запись к мастеру

* 1. **Обзор модели объектов**

База данных должна предоставлять функции хранения, изменения и отображения сведений о клиентах и их записях (как самому клиенту, так и администратору). Общая схема работы системы представлена на общей схеме работы системы (рисунок 1.1)

База данных

Сервер приложения Пользователи

Рисунок 1.1 - Общая схема работы системы

* 1. **Характеристика пользователей**

Конечными пользователями разрабатываемого ПО являются клиент, администратор и мастер. Клиент может выбирать мастеров, выбирать услуги, просматривать портфолио мастеров и оставлять отзывы. Клиент не имеет доступ к базе данных клиентов. Мастер имеет возможность просматривать отзывы о своих работах и добавлять свои работы в портфолио. Администратор может отслеживать данные о клиентах, делать записи клиентов к мастерам, просматривать их и изменять при необходимости.

* 1. **Обзор Use-Case модели**

Use-cases, выявленные в процессе моделирования представлена на диаграмме вариантов использования (Рисунок 1.2). В системе есть только три роли – клиент, администратор и мастер, которые имеет полный доступ ко всей функциональности разработанной системы.

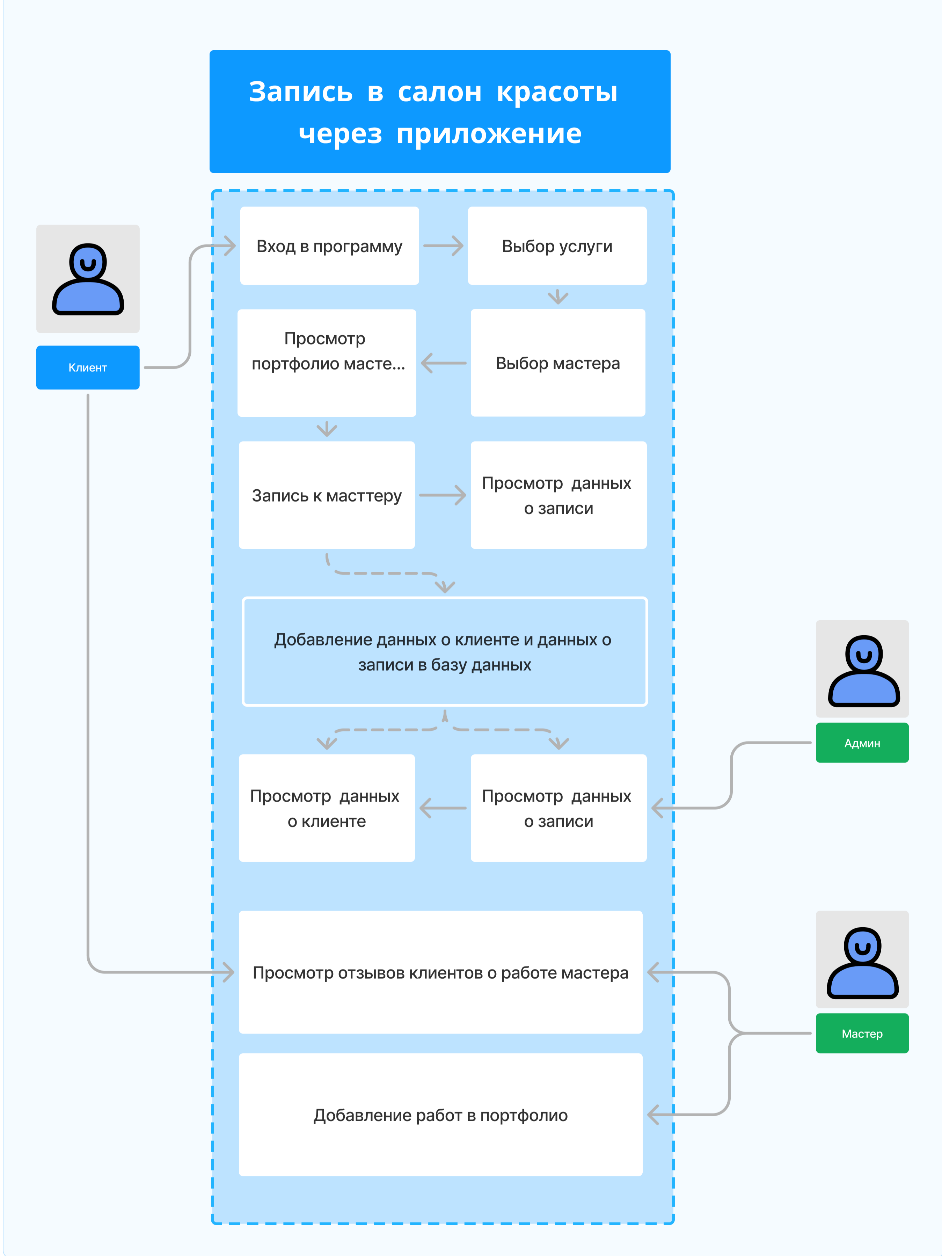
****

Рисунок 1.2 - Диаграмма вариантов использования

Таблица 1. Детализированное описание варианта использования «Запись в салон»:

|  |  |
| --- | --- |
| **Название прецедента:** | Запись в салон |
| **Действующие лицо:** | Клиент |
| **Цель:** | Записаться к мастеру на услугу |
| **Предусловия:** | Клиент входит в приложение салона |
| **Главная последовательность:** | 1) Клиент просматривает услуги салона и выбирает необходимую ему.  2) Система показывает клиенту мастеров.  3) Клиент выбирает мастера и начинает просматривать его портфолио.  4) Клиент нажимает кнопку «записаться к мастеру».  5) Клиент выбирает время и дату записи к мастеру.  6) Система добавляет запись клиента.  7) Клиент осуществляет запись, записывая свое имя, фамилию и номер телефона.  8) Система выводит уведомление клиенту о месте записи, времени и дате. |
| **Альтернативная последовательность** | 1) Клиент просматривает услуги салона и выбирает необходимую ему.  2) Система показывает клиенту мастеров.  3) Клиент выбирает мастера.  4) Клиент просматривает портфолио мастера.  5) Клиент нажимает кнопку «назад» и переходит снова к мастерам.  6) Клиент снова выбирает мастера и смотрит его портфолио.  7) Клиент нажимает кнопку «записаться к мастеру».  8) Клиент выбирает время и дату записи к мастеру.  9) Система добавляет запись клиента.  10) Клиент осуществляет запись, записывая свое имя, фамилию и номер телефона.  11) Система выводит уведомление клиенту о месте записи, времени и дате. |

* 1. **Специфические требования**
     1. **Функциональные требования**
* Система должна посылать данные пользователя на сервис админа
* Система должна выводить данные о записи пользователю
* Система должна предоставить пользователю портфолио мастеров
* Система должна предоставить пользователю выбор услуг
* Система должна ознакомить пользователя с ценовым сегментом
* Система должна отправлять уведомления об успешной записи на указанную электронную почту или смс сообщением.
* Пользователи должны иметь возможность просматривать историю своих записей.
* Пользователь может записываться на услуги.
  + 1. **Нефункциональные требования**
* Система должна обрабатывать не менее 3000 запросов в день без заметного увеличения времени отклика.
* Система должна быть совместима с основными браузерами: Chrome, Firefox, Safari.
* Система должна обеспечивать надежную защиту персональной информации клиентов, включая контактные данные, историю посещений, медицинские сведения, а также данные о платежных картах.
* Доступ к системе должен быть ограничен только авторизованным персоналом салона. Необходимо использовать пароли.
* Необходимо регулярно создавать резервные копии всех данных, чтобы обеспечить восстановление системы в случае сбоев или аварий.
* Система должна быть доступна 24/7, чтобы сотрудники могли работать без перебоев. Необходимо использовать отказоустойчивые технологии, чтобы обеспечить минимальное время простоя.
* Система должна работать быстро и эффективно, чтобы не задерживать сотрудников и клиентов. Необходимо оптимизировать производительность системы, минимизировать время загрузки страниц, оптимизировать запросы к базе данных.
* Система должна быть масштабируемой, чтобы удовлетворить потребности салона в будущем, в том числе в случае увеличения числа клиентов, сотрудников и предоставляемых услуг.
  1. **Интерфейсы**

Ниже представлена UI-схема приложения.

Изображение выглядит как текст, снимок экрана, смартфон

Автоматически созданное описание

Рисунок 1.3 - UI-схема

Описание UI-схемы, представленной на рисунке 1.3:

1. Приветствие клиента.

2. Предоставление клиенту выбор услуг.

3. Предоставление списка мастеров по выбранной услуге.

4. Просмотр портфолио выбранного мастера.

5. Предоставление прайса у выбранного мастера.

6. Ознакомление клиента со свободными датами для записи.

7. Запись клиента.

8. Завершение записи клиента.

1. **Планирование программного проекта**

Эффективное планирование является ключом к успешной реализации программного проекта.

* 1. **Бэклог продукта**

Mindmap – это ассоциативная карта, отвечающая на вопрос «что надо сделать, чтобы запустить проект».  
Данная ментальная карта отвечает на следующие вопросы:  
1. Кого нанимать в команду?  
В нашу команду были приглашены следующие специалисты: аналитик Роберт, senior fullstack Миша, middle fullstack Иван, junior fullstack Тихон, tester Соня, UI designer Мишель, tester Вадим, manager тимлид Женя.  
2. Откуда взять бюджет?  
Бюджет на разработку нашего продукты был обеспечен заказчиком.  
3. Куда распределить бюджет?  
Бюджет нашего проекта был распределен на: зарплату сотрудникам, разработку приложения, премии и сoffeebreak.  
4. С чего начинается разработка приложения?  
Разработка приложения начинается с анализа рынка конкурентов и анализа рынка пользователей.  
5. Какие функции имеет продукт?  
Приложение позволяет клиентам записываться на услуги, выбирать мастера, просматривать портфолио мастеров, оставлять отзывы и просматривать свои записи; мастера могут выкладывать и пополнять свое портфолио, просматривать отзывы к своим работам и услугам, просматривать записи, менять запись (в крайних случаях); админ может контролировать процесс отношений между мастерами и клиентами и использовать данные приложения для улучшения работы салона.



Рисунок 2.1 - Mindmap проекта

* 1. **Планирование спринтов**

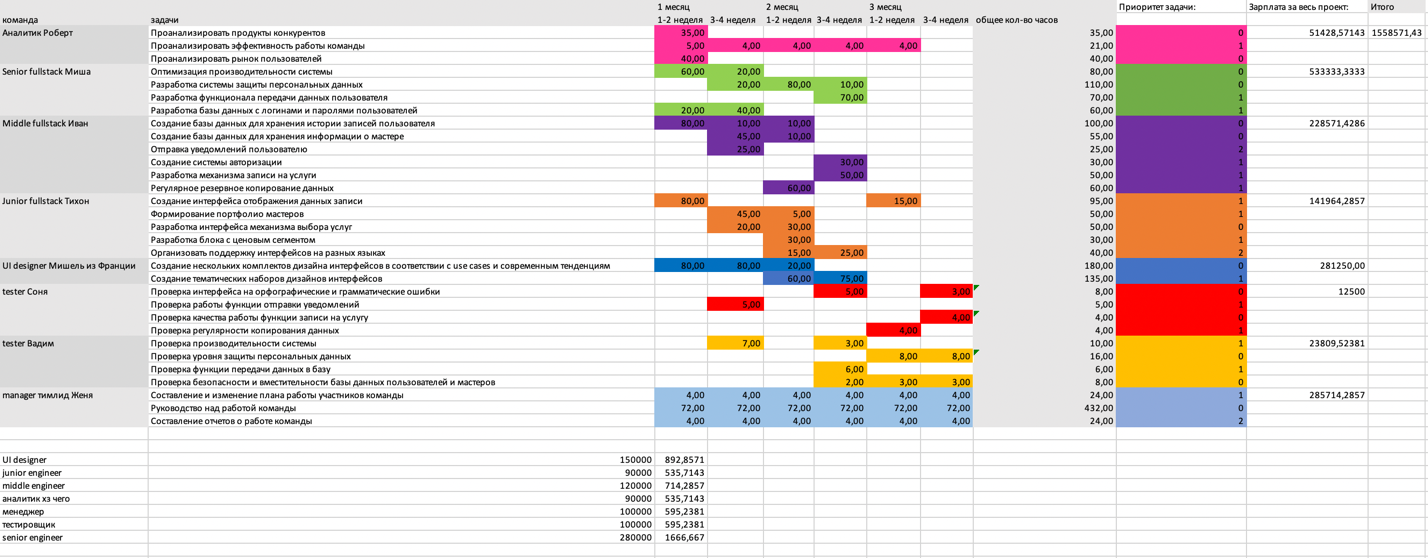


Рисунок 2.2 - Диаграмма Ганта

В данной диаграмме показаны важные аспекты нашего проекта: участники команды, задачи участников, распределение задач по спринтам (1 спринт – 2 недели), общее количество часов, приоритет заданий, зарплата для каждого участника и общая стоимость проекта.

Каждый участник команды был тщательно подобран для выполнения определенных задач в проекте. Почему именно такой состав команды?

Задача нашего проекта заключается в разработке приложения для записи в салон красоты. Для реализации проекта в нашей команде присутствуют следующие специалисты:

Аналитик Роберт – с него начинается работа нашего проекта. Роберт отвечает за анализ эффективности работы команды, сбор анализа целевой аудитории и конкурентов. Его опыт гарантирует нашей команде эффективную разработку конкурентноспособного продукта.

Senior fullstack Миша – ведущий разработчик. Миша отвечает за архитектуру приложения, сложные технические решения и написание важных элементов приложения. Также, он руководит командой разработчиков и обеспечивает высокое качество кода.

Middle fullstack Иван – разработчик уровня middle. Иван занимается задачами средней сложности.

Junior fullstack Тихон – младший разработчик. Тихон реализует простые задачи программного кода, под руководством Миши и Ивана.

UI designer Мишель из Франции – дизайнер. Привлекательный дизайн имеет огромную роль в успехе приложения, а опыт работы Мишель может обеспечить высокое качество, соответствие современным тенденциям и желаниям заказчика.

Tester Вадим – ведущий тестировщик. Вадим занимается тестированием приложения и выявляет баги высокого уровня. Его опыт гарантирует стабильность и качество приложения перед релизом.

Tester Соня – младший тестировщик. Соня выявляет баги среднего и низкого уровня.

Manager тимлид Женя - менеджер проекта. Женя отвечает за организацию и координацию работы команды, управление проектом и контролем сроков. Его работа является залогом успешного завершения проекта.

1. **Проектирование программного продукта**

Проектирование программного продукта — это ключевой этап разработки программного обеспечения, который включает в себя структурирование системы, определение ее компонентов и установление взаимодействия между ними. В данном разделе мы представим UML-диаграммы, которые помогут визуализировать архитектуру приложения для записи в салон красоты. Основные виды диаграмм, используемые в проекте, включают диаграммы классов, последовательностей и состояний.

* 1. **UML-диаграмма классов**

UML-диаграмма классов демонстрирует статическую структуру системы и описывает отношения между основными классами. В нашем приложении для записи в салон красоты выделены следующие ключевые классы:

* Пользователь: класс, общий для классов «клиент», «администратор» и «мастер», хранящий их общие атрибуты. Атрибуты включают: ФИО\_пользователя, телефон\_пользователя и почта\_пользователя.
* Клиент: представляет клиента, записывающегося на услугу. Атрибуты: дата\_рагистрации. Класс «клиент» ссылается на класс «сервис работы с клиентами», включающий методы: запись нового клиента, удаление существующего клиента, просмотр информации о клиенте, редактирование информации о клиенте.
* Мастер: представляет мастера, оказывающего услугу. Включает атрибуты: опыт\_работы. Хранит ссылку на класс «сервис для работы с мастерами», включающий методы: запись нового мастера, удаление существующего мастера, просмотр информации о мастере, редактирование информации о мастере. Также хранит ссылку на класс «портфолио», включающий атрибуты: работы, описание; методы: редактировать портфолио, просмотр портфолио.
* Администратор: представляет администратора салона. Методы включают: просмотр информации о клиенте, просмотр записи, редактирование записи.
* Запись: хранит данные о записи клиента. Атрибуты включают: дата\_и\_время, а также ссылки на клиента, услугу и мастера. Методы включают: изменить время, отменить запись.
* Услуга: описание услуг, предлагаемых салоном. Содержит атрибуты: название, описание, цена, фото. Содержит ссылку на класс «сервис для работы с услугами», включающий методы: редактировать услугу, добавить/удалить услугу.

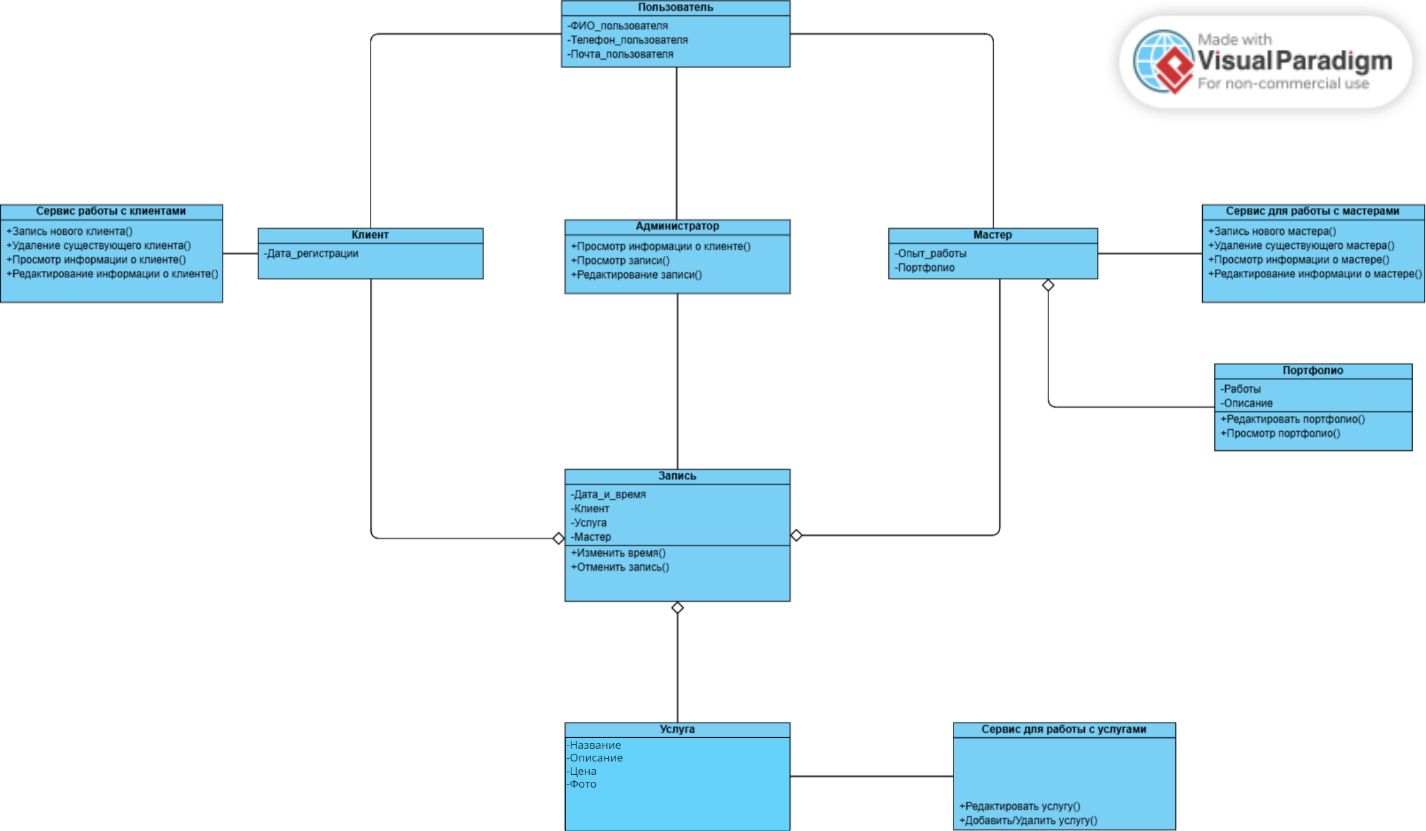


Рисунок 3.1 - Диаграмма классов

* 1. **UML-диаграмма последовательности**

UML-диаграмма последовательности описывает взаимодействие объектов в процессе записи на услугу. Рассмотрим сценарий, в котором пользователь записывается на прием в салон. Далее описаны шаги сценария (действия пользователя):

* Пользователь запускает приложение. Система запрашивает список услуг из б/д. Данные об услугах возвращаются в систему, и пользователь получает доступ к выбору услуг.
* Пользователь выбирает услугу. Система возвращает выбранную услугу в б/д и запрашивает список мастеров из б/д. Список мастеров возвращаются в систему, и пользователь получает доступ к списку мастеров.
* Пользователь выбирает мастера. Система возвращает выбранного мастера в б/д и запрашивает портфолио выбранного мастера из б/д. Портфолио выбранного мастера возвращается в систему, и пользователь получает доступ к портфолио мастера.
* Пользователь записывается к выбранному мастеру. Система запрашивает свободное время для записи из б/д. Список свободных окошек для записи возвращается в систему, и пользователь получает доступ к спуску окошек для записи.
* Пользователь выбирает время для проведения услуги. Система отправляет данные о времени в б/д и запрашивает окно подтверждения выбранной услуги из б/д. Окно возвращается в систему, и пользователь видит окно подтверждения.
* Пользователь подтверждает выбор услуги. Система запрашивает окно ввода персональных данных из б/д. Окно возвращается в систему, и пользователь видит окно ввода.
* Пользователь вводит персональные данные (почта, телефон). Система возвращает персональные данные пользователя в б/д и запрашивает окно подтверждения записи на услугу из б/д. Б/д возвращает окно подтверждения (итоговое), и пользователь видит окно.
* Пользователь подтверждает запись на услугу. Система создает новую запись в б/д, и б/д отображает ее в интерфейсе администратора. Б/д возвращает подтвержденную запись в систему, и пользователь видит подтверждение записи.

Эта диаграмма демонстрирует, как именно взаимодействуют классы, и в каком порядке выполняются операции.

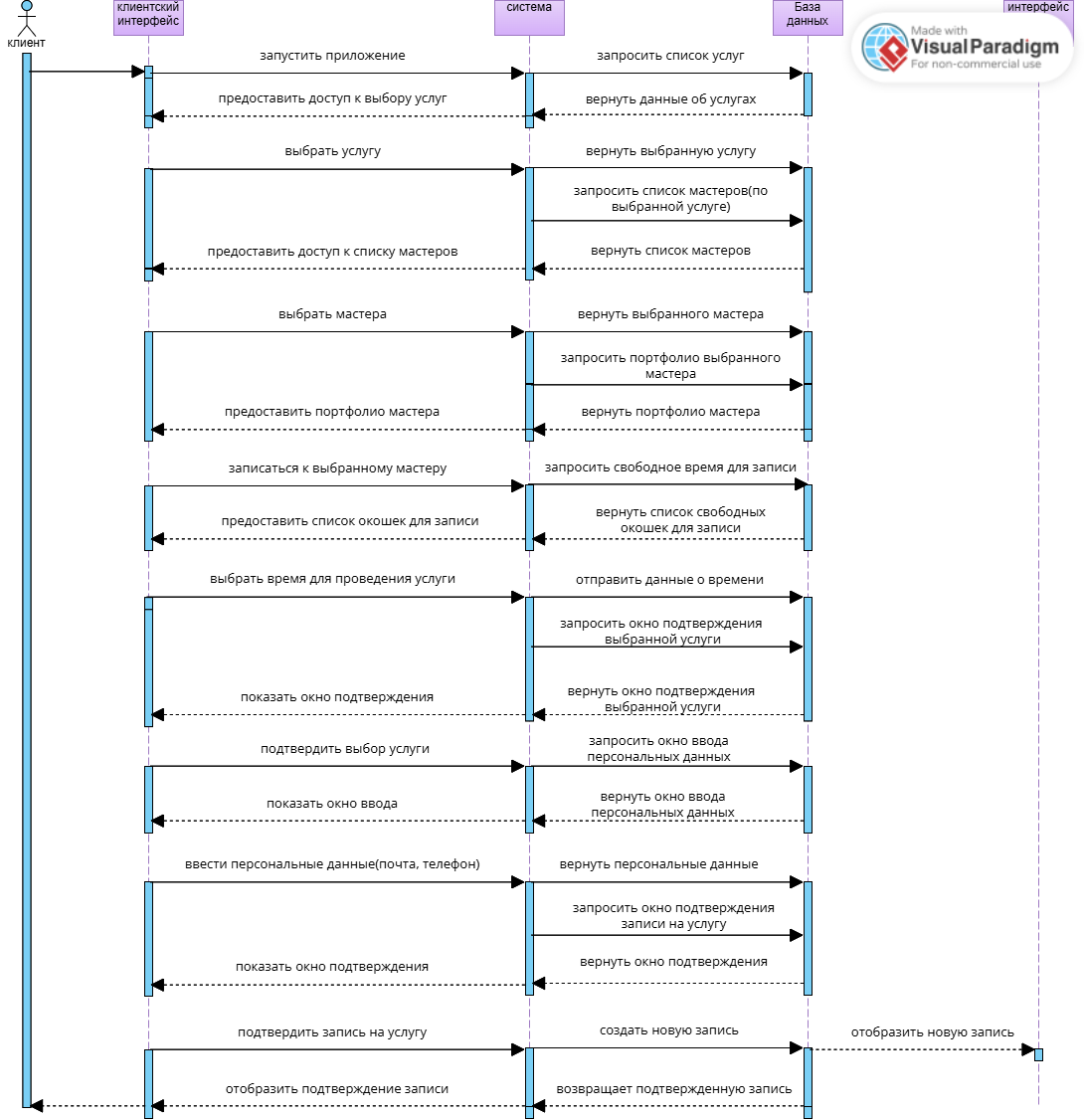


Рисунок 3.2 - Диаграмма последовательности

* 1. **UML-диаграмма состояний**

UML-диаграмма состояний иллюстрирует различные состояния объекта и переходы между ними в зависимости от событий. Для класса «Запись» определены следующие состояния:

Создана: запись была создана, но еще не подтверждена. На этом этапе клиент может вносить изменения или отменить запись.

* Подтверждена: запись подтверждена клиентом. Здесь клиент и администратор получают уведомление о том, что запись клиента принята.
* Отменена: запись может быть отменена как самим клиентом, так и администратором (например, если произошла ошибка или нет доступных мастеров).
* В процессе: на этом этапе клиент получает услугу, и запись в активном состоянии.
* Завершена: услуга была предоставлена, и запись закрыта.
* Переходы между состояниями:
* Создана → Подтверждена: клиент подтверждает запись, при условии, что запись создана.
* Подтверждена → Отменена: клиент отменил запись, при условии, что запись подтверждена.
* Подтверждена → В процессе: клиент пришел на запись, при условии, что запись подтверждена.
* В процессе → Завершена: услуга оказана, при условии, что статус записи «запись в процессе»
* Завершена → Конечное состояние: администратор подтвердил завершение записи, при условии, что запись существует, статус записи: подтверждена.
* Такое описание состояний и переходов позволяет четко понять, как будет работать система записей в салон красоты.

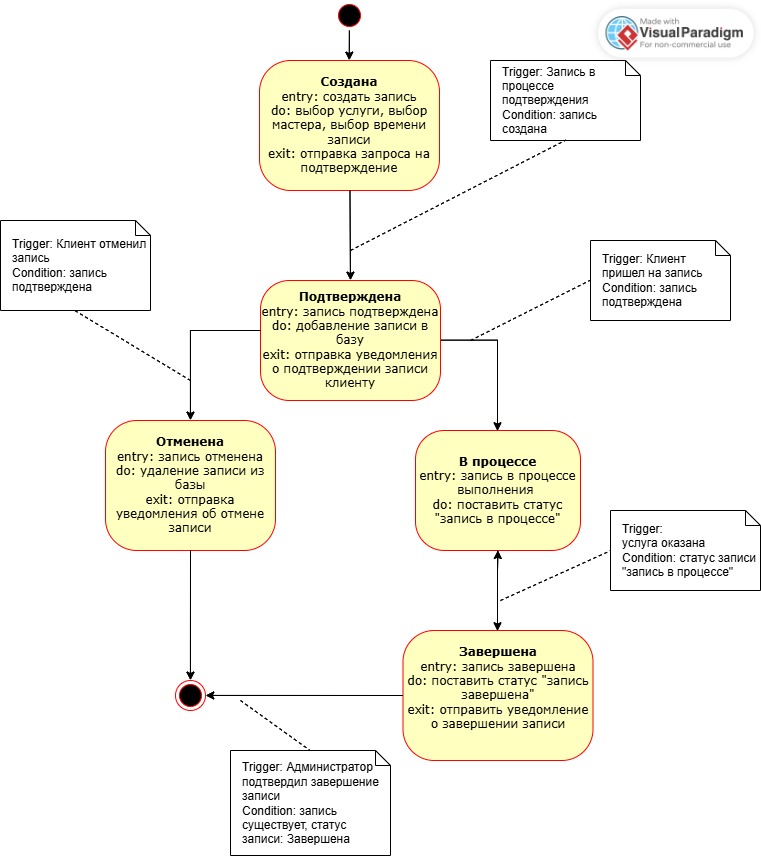


Рисунок 3.3 - Диаграмма состояний

1. **Тестирование**

Тестирование — это важный этап разработки программного обеспечения, на котором проверяется соответствие продукта заданным требованиям. Это позволяет выявить ошибки и недочеты до момента релиза, что, в свою очередь, гарантирует высокое качество конечного продукта.

* 1. **Чек-лист**

1) Проверить на корректность передачу данных пользователя на сервис админа.

2) Проверить возможность выбора даты и времени записи.

3) Проверить возможность выбора мастера.

4) Проверить возможность выбора услуги.

5) Проверить вывод данных о записи пользователю.

6) Проверить на корректное отображение цен.

7) Проверить систему на безопасную передачу данных (данные пользователей должны шифроваться).

8) Проверить поведение системы при критических ситуациях (например: отключение интернета, на финальном этапе записи.

9) Проверить систему на безопасную передачу хранение (данные пользователей должны шифроваться).

10) Проверить работу отправки уведомлений.

11) Проверить систему на сохранение истории записей.

12) Проверить систему на предоставление доступа истории записей самого пользователя.

13) Проверить систему на производительность (система должна обрабатывать не менее 1000 запросов в день без заметного увеличения времени отклика)

14) Проверить систему на совместимость с основными браузерами: Chrome, Firefox, Safari.

15) Проверить интерфейс на соответствие макету.

* 1. **Дефекты (баги)**

В процессе тестирования могут быть выявлены дефекты или баги, которые представляют собой ошибки в коде или несоответствие функциональности требованиям. Их выявление и исправление очень важны для обеспечения стабильной работы приложения.

* + 1. **Ожидаемое и реальное поведение**

1. Первый дефект: приложение не позволяет записаться, принудительно завершает работу.
2. Ожидаемое поведение: приложение завершает запись.
3. Реальное поведение: приложение вылетает при завершении записи.
4. Второй дефект: приложение не открывает окно записи на маникюр.
5. Ожидаемое поведение: приложение позволяет записаться на маникюр.
6. Реальное поведение: приложение не открывает запись на маникюр
7. Третий дефект: неправильное написание заголовка.
8. Ожидаемое поведение: текст не содержит орфографических или грамматических ошибок.
9. Реальное поведение: присутствуют орфографические и грамматические ошибки.
   * 1. **Приоритет и серьезность дефекта**

Первый дефект является критическим и высокоприоритетным, потому что не дает выполнить основную функцию приложения, а именно записать клиента на услугу, и требует оперативного исправления.

Второй дефект является высокоприоритетным и значительным, так как часть работы логики основного функционала нарушена, но в целом оно работает, и нужно исправить его после критических багов.

Третий дефект является тривиальным и низкоприоритетный, потому что не мешает работе основных функций приложения, но может портить впечатление о приложении у клиентов, и требует исправления в последнюю очередь.

* + 1. **Примеры багов**

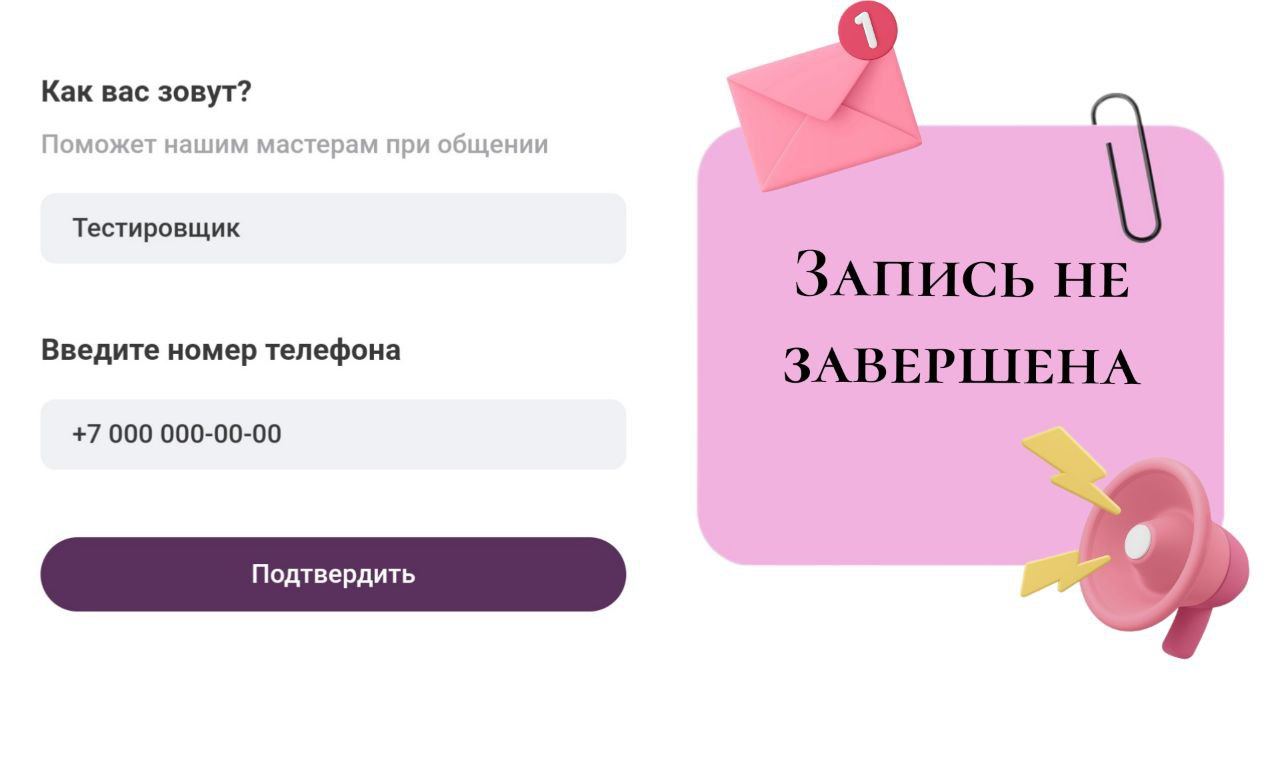


Рисунок 4.1 - Баг №1

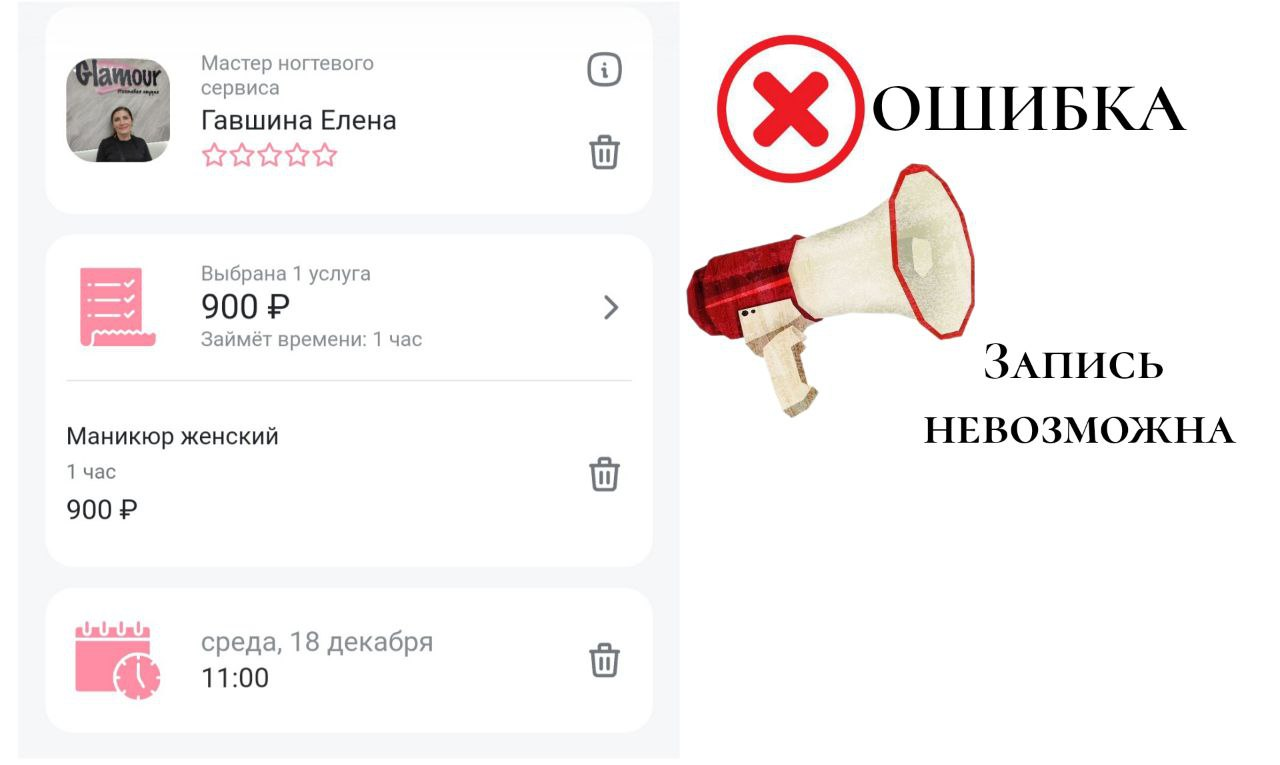


Рисунок 4.2 - Баг №2

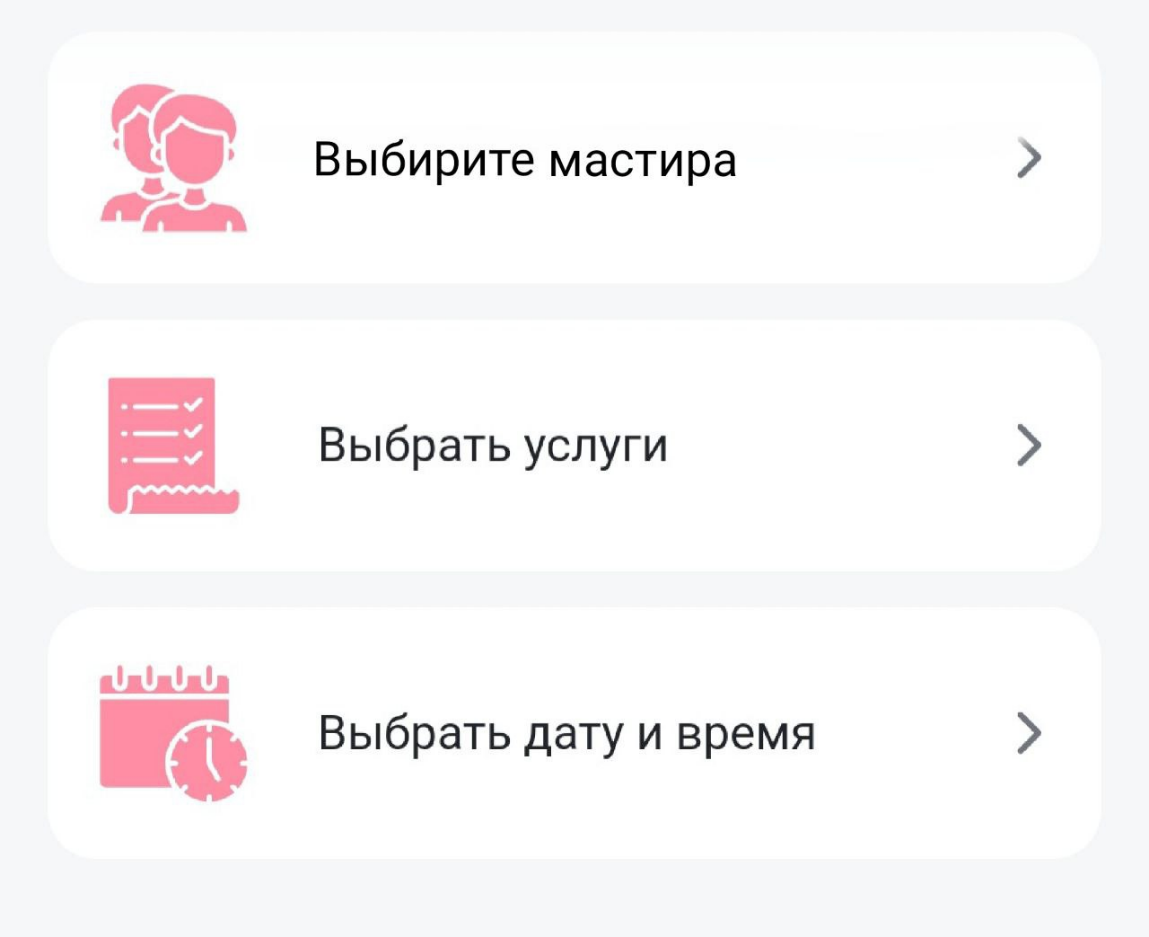


Рисунок 4.3 - Баг №3

* + 1. **Оценка срока исправления**

Ниже в таблице 2 показано кто и за сколько исправит каждый баг, который возник.

Таблица 4.1 - Сроки исправления багов

|  |  |  |
| --- | --- | --- |
| № бага | Cроки | Член команды |
| 1 | 3-4 неделя 1 месяц | Senior fullstack Миша |
| 2 | 3-4 неделя 2 месяц | Middle fullstack Иван |
| 3 | 1-2 неделя 3 месяц | Junior fullstack тихон |

* + 1. **Отчет о дефекте**

Ниже представлен отчет о вышерассмотренных дефектах.

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор  Таблица 4.2 - Отчет о дефекте | Краткое описание | Подробное описание | Шаги по воспроизведению |
| 1 | Приложение не позволяет завершить запись на услугу | Мобильное приложение не позволяет завершить запись на услугу, вылетает при попытке завершить запись  Ожидаемый результат: приложение завершает запись  Фактический результат: приложение вылетает при завершении записи | 1)открыть приложение  2)выбрать услугу  3)выбрать мастера  4)выбрать дату и время  5) ввести персональные данные  6) завершить запись  Дефект: приложение не позволяет записаться, принудительно завершает работу |
| 2 | Приложение не позволяет записаться на маникюр | Мобильное приложение не позволяет записаться на маникюр  Ожидаемый результат: приложение позволяет записаться на маникюр  Фактический результат: приложение не открывает запись на маникюр | 1)открыть приложение  2)выбрать услугу: маникюр  Дефект: приложение не открывает окно записи на маникюр |
| 3 | Орфографические/грамматические ошибки | Периодически можно наблюдать орфографические или грамматические ошибки  Ожидаемый результат: текст не содержит орфографических или грамматических ошибок  Фактический результат: присутствуют орфографические и грамматические ошибки | 1)открыть приложение  2)выбрать услугу  3)обнаружить ошибку  Дефект: неправильное написание заголовка |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Воспроизводимость | Важность | Срочность | Симптом | Возможность обойти | Комментарий | Приложения |
| всегда | Высокая | Срочная | Некорректная операция | нет | Запись невозможна, т.к. ошибка глобальная | - |
| иногда | Средняя | Срочная | Некорректная операция | нет | Не открывается окно записи на услугу маникюра | - |
| редко | Низкая | Обычная | Некорректное отображение | нет | Ошибка в тексте, при записи к мастеру: вместо «выберете мастера» написано «выбирите мастира» | - |

* 1. **Тест-кейсы**

Тест-кейсы представляют собой формализованные сценарии тестирования, которые описывают, как должно проходить тестирование отдельных функций приложения. Ниже в таблице 4.3 предоставлены некоторые тест-кейсы приложения.

Таблица 4.3 - Тест-кейсы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | Приоритет | Модуль приложения | Подмодуль приложения | Исходные данные, шаги | Ожидаемый результат по каждому шагу тест-кейса |
| 1 | A | Окно записи | Выбор услуг | Запись на услугу  1)Открыть приложение  2)Нажать кнопку «Услуги»  3)Нажать кнопку «Массаж»  4)Нажать кнопку «Мастера»  5)Нажать кнопку «Ольга»  6)Нажать кнопку «Выбрать дату и время»  7)Ввести персональные данные  8)Нажать кнопку «Завершить запись»  9)Не позволяет завершить запись | 1)Приложение открывается  2)Появляется кнопка «Услуги»  3) Появляется кнопка «Массаж»  4)Появляется кнопка «Мастера»  5) Появляется кнопка «Ольга»  6)Появляется кнопка «Выбрать дату и время»  7)Персональные данные вводятся  8) Появляется кнопка «Завершить запись»  9)Персональные данные отправляются на сервер  10)Приходит уведомление об спешной записи! |
| 2 | B | Окно выбора услуг | Услуга «маникюр» | Выбор услуги «Маникюр»  1)Открыть приложение  2)Нажать кнопку «Услуги»  3)Нажать кнопку «Маникюр»  4)Не позволяет выбрать услугу «Маникюр» | 1)Приложение открывается  2)Появляется кнопка «Услуги»  3)Появляется кнопка «Маникюр»  4)Появляется кнопка «Мастера»  5) Появляется кнопка «Виктория»  6)Появляется кнопка «Выбрать дату и время»  7)Персональные данные вводятся  8) Появляется кнопка «Завершить запись»  9)Персональные данные отправляются на сервер  10)Приходит уведомление об спешной записи! |
| 3 | D | Окно выбора мастера | Заголовок окна | Выбор мастера «Ирина»  1)Открыть приложение  2)Нажать кнопку «Услуги»  3)Нажать кнопку «Укладка»  4)Переход в окно выбора мастера  5)Ошибка в тексте, при записи к мастеру: вместо «выберете мастера» написано «выбирите мастира» | 1)Приложение открывается  2)Появляется кнопка «Услуги»  3)Появляется кнопка «Укладка»  4)Появляется кнопка «Мастера»  5) Появляется кнопка «Ирина»  6)Появляется кнопка «Выбрать дату и время»  7)Персональные данные вводятся  8) Появляется кнопка «Завершить запись»  9)Персональные данные отправляются на сервер  10)Приходит уведомление об спешной записи! |

* 1. **Описание программного кода**

Проект приложения BeatySalonApp (рисунок 4.1) содержит несколько подпапок ниже рассмотрим их подробнее.

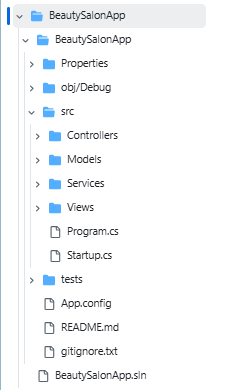


Рисунок 4.1 – проект приложения в GitHub

В основной папке проекта src содержится несколько папок с классами. Рассмотрим их ниже.

1)Models: содержит классы, представляющие данные, такие как записи, клиенты и услуги.

2) Services: содержит классы, которые обрабатывают запросы и взаимодействуют с моделями и представлениями.

3)Controllers: содержит классы, которые обрабатывают запросы и взаимодействуют с моделями и клиентами.

4)Views: содержит классы, отвечающие за отображение данных пользователю.

5)Tests: содержит тесты.

А также gitignore – игнорируемые файлы в Git.

Readme.md –документация проекта.

BeautySalonApp.sln – файл решения для VS.

В таблице 4.4 рассмотрим классы данного проекта.

Таблица 4.4 – Описание классов проекта

|  |  |
| --- | --- |
| Класс | Описание |
| AppointmentController | Контроллер для записей |
| MasterController | Контроллер для мастеров |
| Appointment | Модель для записи |
| Master | Модель мастера |
| Service | Модель услуги |
| User | Модель клиента |
| AppointmentService | Сервис для управления записями |
| MasterService | Сервис для управления мастерами |
| NotificationServise | Сервис для уведомлений |
| Program | Главный файл приложения |
| Startup | Конфигурация приложения |
| ControllerTests | Тест для контроллера для записи |
| ServicesTests | Тесты для модели клиента |

1. **Оценка качества проекта**

Таблица 5 - Оценка качества проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | Атрибуты | Оценка | Обоснование |
|  | Полнота функций | 5 | Приложение выполняет все требуемые функции. |
|  | Точность | 5 |
| Функциональность | Интероперабельность | 5 |
|  | Защищенность | 4 |
|  | Согласованность | 4 | Средняя оценка: 4.6 |
|  | Завершенность | 5 | Приложение надёжно, но не рассчитано на количество запросов, превышающее 3000 |
| Надежность | Отказоустойчивость | 4 |
|  | Восстанавливаемость | 4 |
|  | Согласованность | 5 | Средняя оценка: 4.5 |
|  | Понимаемость | 5 | Приложение удобно в использовании |
| Удобство | Обучаемость | 5 |
|  | Восстанавливаемость | 4 |
|  | Согласованность | 5 | Средняя оценка: 4.8 |
|  | Реактивность | 5 | Приложение эффективно в работе |
| Эффективность | Используемость ресурсов | 4 |
|  | Согласованность | 4 | Средняя оценка: 4.3 |
|  | Анализируемость | 5 | Продукт хорошо сопровождаем, компоненты или система могут изменяться специалистами по сопровождению |
|  | Изменяемость | 4 |
| Сопровождаемость | Стабильность | 3 |
|  | Тестируемость | 4 |
|  | Согласованность | 4 | Средняя оценка: 4 |
|  | Адаптируемость | 5 | Приложение может легко работать с разными операционными системами |
|  | Простота настройки | 5 |
| Переносимость | Совместимость | 4 |
|  | Заменяемость | 5 |
|  | Согласованность | 4 | Средняя оценка: 4.6 |

Общая оценка продукта: 4.5

# **ЗАКЛЮЧЕНИЕ**

В ходе автоматизации деятельности салона красоты, был разработан удобный и функциональный продукт, который упрощает процессы обслуживания клиентов - приложение для записи клиента в салон красоты, где клиент может выбрать: услугу, мастера, удобное время для записи; просмотреть портфолио мастера и ознакомиться с прайсом салона.   
Для создания данного продукта была собрана команда опытных специалистов и установлены требования и задачи для разработчиков. Для систематизации работы были составлены следующие таблицы и диаграммы: диаграмма Ганта, диаграмма классов, ментальная карта проекта, таблица багов, чек - лист для проверки, test case и таблица с оценками качества ПО.  
Разработанный продукт имеет множество преимуществ: клиенты получают возможность легко и быстро записываться на услуги, а также управлять своими записями в любое время. Сотрудники, в свою очередь, могут сосредоточиться на предоставлении качественных услуг, не отвлекаясь на рутинные задачи, такие как ведение записей и планирование. Кроме того, платформа включает в себя функционал для сбора и анализа отзывов клиентов, что позволяет владельцам салонов красоты получать важную обратную связь и оперативно реагировать на пожелания и замечания.   
Также, приложение позволяет салону красоты эффективно управлять ресурсами, оптимизировать рабочие процессы и повышать уровень обслуживания клиентов, что способствует увеличению клиентской базы и, как следствие, росту прибыли. Благодаря продуманной стратегии автоматизации, салоны могут планировать свое развитие на основе точных данных, минимизируя риски и увеличивая шансы на успех.  
Подведя итоги, можно сделать вывод, что с помощью данного продукта и маркетинга, становится возможным активное привлечение новых клиентов через различные рекламы, включая социальные сети и email-маркетинг. Автоматизированное приложение для салона красоты является важным шагом к повышению эффективности работы и улучшению качества обслуживания, что в конечном итоге способствует успешному развитию бизнеса в условиях современного рынка.

# **СПИСОК ЛИТЕРАТУРЫ**

1)Правила составления Software requirements specification / Хабр [Сайт]. (24.02.2009). URL: <https://habr.com/ru/articles/52681/> (дата обращения: 06.09.2024).

2) Интеллект-карты для планирования проектов / Интеллект-карты [Сайт]. [2003-2024]. URL: <https://mind-map.ru/intellekt-karty-planirovanie-proektov/> (дата обращения: 20.09.2024).

3) Проектирование диаграмм состояний UML (Statechart Diagram) / Национальная сборная Worldskills Россия [Сайт]. [2014-2024]. URL: <https://nationalteam.worldskills.ru/skills/proektirovanie-diagrammy-sostoyaniy-uml-statechart-diagram/> (дата обращения: 04.10.2024).

4) Как построить UML-диаграмму последовательности практический пример / Babokschool <https://babok-school.ru/blog/uml-sequence-diagram-example/> (дата обращения: 18.10.2024).

5)UML для самых маленьких: диаграмма классов / Хабр [Сайт]. (21.07.2020). URL: <https://habr.com/ru/articles/511798/> (дата обращения: 01.11.24).

6) Построение диаграммы классов / Flexberry PLATFORM (2024). URL: <https://flexberry.github.io/ru/gpg_class-diagram.html> (дата обращения: 15.11.24).

7) Простое руководство по использованию кассовых диаграмм UML / Creately [Сайт]. (10.05.2022). URL: <https://creately.com/blog/ru/uncategorized-ru/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B5-%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2/> (дата обращения: 29.11.24).

8)Тестирование программного обеспечения // Святослав Куликов / svyatoslav.biz [Сайт]. [2010-2024]. URL: https://svyatoslav.biz/software\_testing\_book/ (дата обращения: 03.12.2024).

9) Модели качества и надежности программного обеспечения / НОУ ИНТУИТ [Сайт]. [2003-2024]. URL: <https://intuit.ru/studies/courses/2279/237/lecture/6136> (дата обращения: 14.12.24).

# **Приложение**

**Определения, обозначения и сокращения**

|  |  |
| --- | --- |
| Сокращение | Определение |
| ПО | Программное обеспечение |
| SRS | Спецификация требований |
| UI | Пользовательский интерфейс |
| Use case diagram | Диаграмма вариантов использования |
| User story | Пользовательские истории |
| Б/д | База данных |
| Test cases | Алгоритм действий для проверки написанной программы |
| Баги | Ошибка в коде или в работе программы. |
| Бэклог | Перечень задач, расположенных в порядке важности, для команды разработчиков |
| Спринт | Короткий временной интервал, в течение которого scrum-команда выполняет заданный объем работы |
| Mindmap | Метод записи, позволяющий структурировать информацию, вычленить самое главное, визуально отобразить взаимосвязи разных фрагментов |
| Scrum | Гибкая методология (фреймворк), которая помогает автоматизировать совместную работу над проектом |