Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Тема: Разработка системы классов для приложения «Частная клиника»**

**Специальность 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент группы 309ИС-22** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Я. С. Петухов** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Л. Б. Гусятинер** |

**Москва 2024**

Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**УТВЕРЖДАЮ**

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

**По дисциплине: МДК 01.01 Разработка программных модулей**

**Специальность 09.02.07 «Информационные системы и программирование»**

**Студент группы 309ИС-22 Петухов Ярослав**

**ТЕМА: «Разработка системы классов для приложения «Частная клиника»»**

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Срок сдачи проекта «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

**Москва 2024**

**Перечень вопросов, подлежащих разработке:**

ВВЕДЕНИЕ

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Информационное обеспечение задачи

1.2. Обзор и анализ существующих программных решений

1.3. Постановка задачи. Структура входной и выходной информации

2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ КЛАССОВ

2.1. Построение модели системы

2.2. Описание главного модуля

2.3. Описание спецификаций к модулям

2.4. Описание модулей

2.5. Расчет сложности алгоритма

3. ОТЛАДКА И ТЕСТИРОВАНИЕ СИСТЕМЫ КЛАССОВ

3.1. Описание тестовых наборов модулей

3.4. Описание применения средств отладки

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЯ

**Исходные данные:** 27.02.2006 №152-ФЗ, 27.02.2006 №149-ФЗ, 26.07.2017 №187-ФЗ, ГОСТ Р ИСО/МЭК 12207, ГОСТ 19.005–85, ГОСТ 19.105–78

**Перечень разрабатываемых материалов для визуализации:** диаграмма классов, сценарий работы приложения, блок-схемы (не менее двух), дизайн приложения.

Задание выдал:

Руководитель курсового проекта Л. Б. Гусятинер

Задание принял к исполнению Я. С. Петухов

*Рассмотрено*

на заседании предметно-цикловой комиссии

информационных технологий и системного

администрирования

Протокол № \_\_\_от «\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Председатель ПЦК \_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЗЫВ**

**научного руководителя на курсовой проект**

**Студента(ки**)

(фамилия, имя и отчество)

**группа**  **курс**  3

**Дисциплина**  МДК 01.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

**Тема работы**

Отмечаются следующие моменты: актуальность темы исследования; степень разработанности проблемы, наиболее интересно исследованные вопросы. Оценивается степень самостоятельности и творчества студента; уровень его теоретической подготовки; умение анализировать научные материалы, делать практические выводы. Знание основных концепций, научной и специальной литературы по избранной теме.

**Научный руководитель**

(Уч. степень, уч. звание, должность, ФИО)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г

**Содержание**

[**ВВЕДЕНИЕ** 6](#_Toc185423838)

[**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 8](#_Toc185423839)

[**1.1 Информационное обеспечение задачи** 8](#_Toc185423840)

[**1.2 Обзор и анализ существующих программных решений** 10](#_Toc185423841)

[**1.3 Постановка задачи** 13](#_Toc185423842)

[**1.3.1 Основные задачи и функции** 13](#_Toc185423843)

[**1.3.2 Структура входной и выходной информации** 13](#_Toc185423844)

[**2. ПРОЕКТНАЯ ЧАСТЬ** 15](#_Toc185423845)

[**2.1 Выбор инструментов** 15](#_Toc185423846)

[**2.2 Диаграмма классов** 17](#_Toc185423847)

[**2.3 Проектирование сценария** 18](#_Toc185423848)

[**2.4 Построение модели системы** 19](#_Toc185423849)

[**2.5 Реализация интерфейса приложения** 21](#_Toc185423850)

[**2.6 Модули системы** 22](#_Toc185423851)

[**2.7 Сложность системы** 27](#_Toc185423852)

[**3. ТЕСТИРОВАНИЕ** 29](#_Toc185423853)

[**ЗАКЛЮЧЕНИЕ** 33](#_Toc185423854)

[**СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ** 34](#_Toc185423855)

[**Приложение 1 Диаграмма прецедентов** 35](#_Toc185423856)

[**Приложение 2 Экран авторизации пациентов** 36](#_Toc185423857)

[**Приложение 3 Экран авторизации для врачей** 37](#_Toc185423858)

[**Приложение 4 Регистрация для пациентов** 38](#_Toc185423859)

[**Приложение 5 Экран регистрации для врачей** 39](#_Toc185423860)

[**Приложение 6 Экран личного кабинета пациента** 40](#_Toc185423861)

[**Приложение 7 Экран личного кабинета для врача** 41](#_Toc185423862)

[**Приложение 8 Экран записи ко врачу** 42](#_Toc185423863)

[**Приложение 9 Экран записи в медицинскую карту** 43](#_Toc185423864)

[**Приложение 10 Блок-схема приложения** 44](#_Toc185423865)

[**Приложение 11 Таблица план тестирования** 45](#_Toc185423866)

# **ВВЕДЕНИЕ**

В современных условиях развития здравоохранения особую актуальность приобретает вопрос эффективного управления медицинскими учреждениями, в частности, частными клиниками. Стремительное развитие информационных технологий открывает новые возможности для автоматизации процессов управления медицинскими учреждениями, что способствует повышению качества обслуживания пациентов и оптимизации работы медицинского персонала.

Актуальность темы обусловлена тем, что растет потребность в автоматизации процессов управления медицинскими учреждениями, необходимо повысить качество обслуживания пациентов, важность обеспечения эффективного хранения и обработки медицинской информации, потребность в оптимизации рабочего времени медицинского персонала.

Практическая значимость работы заключается в разработке информационной системы, которая позволит автоматизировать основные бизнес-процессы частной клиники и повысить эффективность её работы.

Объектом исследования является частная медицинская клиника «CMD», а предметом - автоматизация управления частной клиникой с использованием информационных технологий.

Цель работы: разработка информационной системы для автоматизации процессов управления частной клиникой, обеспечивающей эффективное взаимодействие между пациентами и медицинским персоналом.

Задачи проекта:

* Анализ существующих бизнес-процессов в частных медицинских учреждениях
* Определение функциональных требований к разрабатываемой информационной системе
* Проектирование архитектуры информационной системы
* Разработка базы данных для хранения информации о пациентах, врачах и медицинских услугах
* Создание пользовательского интерфейса для работы с системой
* Реализация основных модулей системы
* Тестирование и отладка разработанного программного обеспечения

Круг решаемых проблем:

* Автоматизация процесса регистрации и учета пациентов
* Организация записи на прием к врачам
* Ведение электронной медицинской документации
* Управление расписанием работы медицинского персонала

В рамках проекта предполагается использование современных технологий программирования, включая Python для разработки backend-части системы, реляционную базу данных для хранения информации и современные подходы к обеспечению информационной безопасности.

Реализация данного проекта позволит значительно повысить эффективность работы частной клиники, улучшить качество обслуживания пациентов и обеспечить надежное хранение медицинской информации в соответствии с требованиями законодательства о защите персональных данных.

# **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Информационное обеспечение задачи**

Частная клиника – это медицинское учреждение, которое предоставляет услуги по лечению, диагностики и профилактики заболеваний на основе частной формы собственности. Основные потребители частной клиники – это врачи и пациенты

Организационная структура частной клиники представлена ниже на рисунке 1.

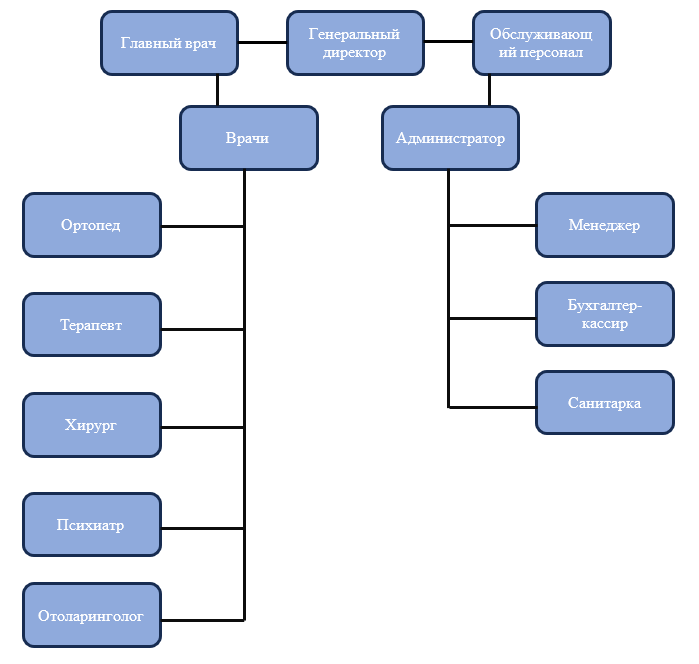


Рисунок 1 – Организационная структура частной клиники

Частная клиника (ЧК) предлагает обширный спектр услуг, такие как:

* Терапевтические услуги – диагностика, профилактика, лечение заболеваний. Услуга включает проведение осмотра, постановка предварительного диагноза.
* Ортопедические услуги – диагностика, лечение и профилактика заболеваний и травм опорно-двигательного аппарата.
* Психиатрические услуги – диагностика и лечение психиатрических заболеваний, а также психологическую поддержку и социальную реабилитацию.
* Отоларингологические услуги – процедуры и манипуляции, которые проводятся врачом-отоларингологом при заболеваниях уха, носа, горла, гортани и придаточных пазух.
* Хирургические услуги – диагностические и лечебные процедуры, которые проводятся врачом-хирургом, в том числе и хирургические операции.

Частная клиника в своей деятельности опирается следующие нормативные документы:

* Закон РФ от 07.02.1992 N 2300–1 «О защите прав потребителей" — регулирует права пациентов как потребителей медицинских услуг.
* Федеральный закон от 21.11.2011 N 323-ФЗ "Об основах охраны здоровья граждан в Российской Федерации" — определяет основы организации здравоохранения и права граждан на медицинскую помощь.

Автоматизация деятельности ЧК включает в себя следующие ключевые аспекты:

* электронное хранение информации пациентов, справок и данных, связанных с их лечением;
* применение информационных технологий для автоматизации и поддержки бизнес-процессов управления ЧК;

Применение определенных ИТ-решений для автоматизации деятельности ЧК зависит от стратегии автоматизации и финансовых возможностей ЧК.

## **1.2 Обзор и анализ существующих программных решений**

Медицинская система «YCLIENTS» предназначена для автоматизации процессов в медицинских учреждениях, включая частные клиники и медицинские центры.

Основные функции «YCLIENTS»:

* онлайн-запись;
* управление расписанием;
* финансовый учет;
* аналитика и статистика;

Преимущества «YCLIENTS»:

* доступность: система доступна на различных устройствах и поддерживает работу в облаке;
* удобный интерфейс: система сделана в понятном и простом дизайне, что облегчает работу врачей и использование системы клиентами;
* автоматизация процессов: система автоматизирует множество бизнес-процессов
* круглосуточная техническая поддержка: дает возможность пользователям быстро решать возникшие проблемы.

Недостатки «YCLIENTS»:

* зависимость от интернета: поскольку это облачная система, то проблемы с интернет-соединением могут привести к сбою в работе системы;
* сложность при внедрении: требуются высококвалифицированные сотрудники для внедрения системы;
* высокая стоимость внедрения: не выгодно для малых клиник.

Медицинская система «Medesk» предназначена для автоматизации процессов в медицинских учреждениях, включая частные клиники и медицинские центры.

Основные функции «Medesk»:

* онлайн запись;
* ведение электронных медицинских карт;
* управление расписанием
* финансовый учет
* аналитика и отчеты

Преимущества «Medesk»:

* удобный интерфейс: система сделана в понятном и простом дизайне, что облегчает работу врачей и использование системы клиентами;
* автоматизация процессов: система автоматизирует множество бизнес-процессов;
* доступность: облачное решение позволяет обеспечить доступ к системе с любых устройств и работать из любого места;
* гибкость и модульность: возможность подключения различных модулей в зависимости от потребностей клиники;
* круглосуточная техническая поддержка: дает возможность пользователям быстро решать возникшие проблемы.

Недостатки «Medesk»:

* зависимость от интернета: поскольку это облачная система, то проблемы с интернет-соединение могут привести к сбою в работе системы;
* высокая стоимость внедрения: не выгодно для малых клиник.
* сложность при внедрении: требуются высококвалифицированные сотрудники для внедрения системы;
* отсутствие личного кабинета пациента: на данный момент система не предоставляет возможности для пациентов самостоятельно управлять своей медицинской информацией.

Медицинская система «МедАнгел» предназначена для автоматизации процессов в медицинских учреждениях, включая частные клиники и медицинские центры.

Основные функции «МедАнгел»:

* онлайн запись;
* ведение электронных медицинских карт;
* управление расписанием врачей;
* финансовый учет;
* аналитика и отчетность;
* интеграция с другими системам;
* учет медикаментов и расходных материалов.

Преимущества «МедАнгел»:

* модульная структура: позволяет выбрать и внедрять только необходимые функции, что способствует оптимизации затрат;
* открытый код: возможность доработать код под индивидуальные требования клиники;
* гибкость в настройках: гибкость настройки интерфейса и функционала под различные требования медучреждений;
* поддержка офлайн-режима: возможность работы без постоянного интернет-соединения, что помогает в работе удаленным и нестабильным регионам;
* круглосуточная техническая поддержка: дает возможность пользователям быстро решать возникшие проблемы.

Недостатки «МедАнгел»:

* сложность с оборудованием: при использовании локальной версии системы может быть затратно необходимость закупки серверного оборудования;
* необходимость высокой квалификации персонала: требуется определенный уровень подготовки персонала для эффективного использования системы;
* необходимость доработок: некоторые пользователи сообщают о необходимости дополнительных доработок для достижения желаемого функционала.

## **1.3 Постановка задачи**

Разработка программного модуля информационной системы «Частная клиника» направлена на автоматизацию ключевых процессов, такие как: запись на прием, удаление записи, заполнение медицинской карты пациента, улучшения качества обслуживания клиентов и оптимизацию работы медицинского персонала.

### **1.3.1 Основные задачи и функции**

Основные задачи, которые должен решать разрабатываемый модуль:

* хранение данных: программа использует базу данных для хранения информации о пользователях, врачах, записях на прием и медицинских картах. Это позволяет эффективно управлять данными и обеспечивать их целостность;
* управление расписанием врачей;
* управление записями:

1. автоматизация записи пациентов к врачу (клиент может записаться и удалить запись на прием);
2. врачи могут добавлять записи в медицинские карты пациентов, а также просматривать свои записи и данные о пациентах.

### **1.3.2 Структура входной и выходной информации**

Структура входной информации:

1. данные о пациентах: ФИО, логин, пароль;
2. данные о врачах: ФИО, логин, пароль;
3. данные на запись на прием: дата и время приема, выбор врача;
4. ввод данных в медицинскую карту клиента врачом.

Структура выходной информации:

1. подтверждение записи на прием: уведомление о подтверждении записи;
2. вывод в личном кабинете клиента его данных: ФИО, медицинская карта, записи к врачам;
3. вывод в личном кабинете врача: личных данных, записей к этому врачу, просмотр медицинской книжки записанного клиента;
4. готовый документ медицинской карты клиента;
5. готовый документ на запись: ФИО пациента, врач, дата и время приема;

# **2. ПРОЕКТНАЯ ЧАСТЬ**

## **2.1 Выбор инструментов**

Для выбора языка программирования были проведены сравнения по критериям, которые представлены в таблицах 1 и 3.

Степень важности критерия: низкая, ниже средней, средняя, ниже высокой, высокая.

|  |  |
| --- | --- |
| Таблица 1 – Критерии выбора инструмента | |
| Критерий | Важность критерия |
| Функционал | Ниже высокой |
| Удобство использования | Высокая |
| Скорость разработки | Ниже высокой |

Исходя из данных критериев, я сравнил 3 языка программирования цветами: белый – 1 балл, светло-серый – 0,5 балла, темно-серый - 0 баллов за критерий (см. таблицу 2).

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 2 – Оценка языков программирования | | | |
| Критерий/ Язык программирования | С | Python | JavaScript |
| Функционал | Низкоуровневый язык, подходящий для системного программирования, разработки операционных систем и встраиваемых систем. | Высокоуровневый язык, широко используемый в веб-разработке, анализе данных, машинном обучении и автоматизации. | Язык для веб-разработки, используемый как на стороне клиента (frontend), так и на стороне сервера (backend) с помощью Node.js. |
| Удобство использования | Сложный синтаксис, требует глубокого понимания управления памятью и указателей. | Простой и читаемый синтаксис, легко осваивается новичками. | Умеренная сложность; синтаксис может быть запутанным из-за асинхронности и особенностей работы с DOM. |
|  | | | |
| Продолжение таблицы 2 | | | |
| Скорость разработки | Долгий процесс разработки из-за необходимости управления памятью и сложного синтаксиса. | Быстрая разработка благодаря простоте синтаксиса и большому количеству библиотек. | Быстрая разработка для веб-приложений, особенно с использованием фреймворков (например, React, Angular). |
| Итого | 0 | 3 | 2,5 |

Исходя из сравнения, был выбран Python как язык программирования.

|  |  |
| --- | --- |
| Таблица 3 – Критерии выбора среды разработки | |
| Критерий | Важность критерия |
| Простота | Средняя |
| Функционал | Высокая |
| Удобство использования | Высокая |
| Документация на русском языке | Ниже средней |

Исходя из данных критериев и выбора Python в качестве языка программирования, я сравнил 2 среды разработки цветами: белый – 1 балл, светло-серый – 0,5 балла, темно-серый – 0 баллов за критерий (см. таблицу 4).

|  |  |  |
| --- | --- | --- |
| Таблица 4 – Оценка сред разработки | | |
| Критерий/среда разработки | PyCharm | Microsoft Visual Studio |
| Простота | Удобный интерфейс, специально разработанный для Python-разработчиков. | Сложный интерфейс с множеством функций; может быть перегруженным для новичков. |
| Функционал | Специализированные инструменты для Python: авто дополнение, анализ кода, интеграция с фреймворками. | Поддержка множества языков (C#, C++, JavaScript и др.), мощные инструменты для отладки и тестирования. |
| Удобство использования | Удобное управление проектами и виртуальными окружениями; специализированные функции для Python делают работу более эффективной. | Высокая степень настройки интерфейса; может быть сложным для пользователей, работающих только с Python. |
| Документация на русском языке | Хорошая документация на русском языке; активное сообщество и поддержка пользователей. | Обширная документация на английском языке; наличие русскоязычных ресурсов ограничено. |
| Итого | 4 | 1 |

По результатам сравнения была выбрана среда разработки PyCharm.

PyCharm — это интегрированная среда разработки (IDE) для языка программирования Python, разработанная компанией JetBrains. Она предоставляет разработчикам все необходимые инструменты для написания, запуска и отладки кода, а также для работы с различными фреймворками и библиотеками.

## **2.2 Диаграмма классов**

В диаграмме классов представлены классы: clients, doctors, DoctorCabinet, UserCabinet, check\_user, autorisation, DB\_appointment, ClinicApp, autorisation\_doctor.

В классах используется спецификатор доступа public и private. Public говорит нам о том, что доступ открыт всем, кто видит определение данных классов, а private изменяет имена атрибутов и методов, что делает их менее доступными снаружи класса.

Блок, стоящий после названия класса, показывает нам объявленные переменные, а знаки квадрат и круг указывает на спецификацию доступа.

После блока с атрибутами, расположен блок с операциями.

Диаграмма классов представлена на рисунке 2.

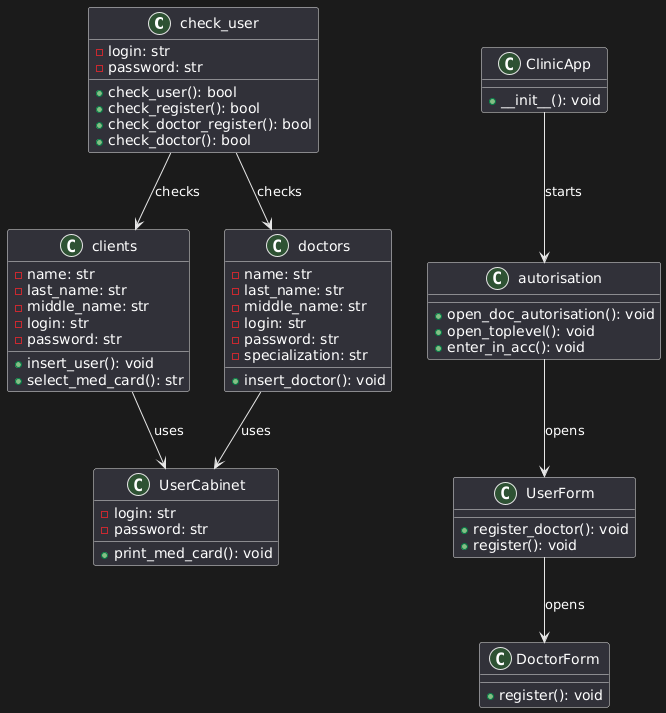


Рисунок 2. Диаграмма классов

## **2.3 Проектирование сценария**

Сценарий должен отражать основные бизнес-процессы клиники и потребности пользователей. Диаграмма прецедентов представляет собой схему пользования программой как врачом, так и клиентом. (Приложение 1)

**Пациент:**

Сценарий 1: Регистрация пациента

1. Пациент открывает окно регистрации.
2. Вводит свои данные (имя, фамилия, отчество, логин, пароль).
3. Нажимает кнопку "Регистрация".
4. Система проверяет уникальность логина и создает запись в базе данных.

Сценарий 2: Авторизация пользователя

1. Пользователь открывает окно авторизации.
2. Вводит логин и пароль.
3. Нажимает кнопку "Войти".
4. Система проверяет данные и открывает личный кабинет пользователя.

Сценарий 3: Запись на прием

1. Пациент входит в личный кабинет.
2. Выбирает врача и время приема.
3. Подтверждает запись.
4. Система сохраняет запись в базе данных.

Сценарий 4: Просмотр медицинской карты

1. Пациент входит в личный кабинет.
2. Нажимает кнопку "Просмотреть медицинскую карту".
3. Система отображает данные медицинской карты.

**Врач:**

Сценарий 1: Регистрация врача

1. Врач открывает окно регистрации.
2. Вводит свои данные (имя, фамилия, отчество, логин, пароль, специальность).
3. Нажимает кнопку "Регистрация".
4. Система проверяет уникальность логина и создает запись в базе данных.
5. Врач получает уведомление о успешной регистрации.

Сценарий 2: Авторизация врача

1. Врач открывает окно авторизации.
2. Вводит логин и пароль.
3. Нажимает кнопку "Войти".
4. Система проверяет данные и открывает личный кабинет врача.
5. Врач видит свои записи на прием и может управлять ими.

Сценарий 3: Добавление или редактирование медицинской карты

1. Врач открывает медицинскую карту пациента.

1. Нажимает кнопку "Добавить в мед. карту".
2. Вносит изменения в данные медицинской карты (например, добавляет назначения или комментарии).
3. Нажимает кнопку "Добавить".
4. Система обновляет данные в базе и уведомляет врача об успешном сохранении.

Сценарий 4: Просмотр записей на прием

1. Врач входит в личный кабинет.
2. Нажимает на раздел "Записи на прием".
3. Система отображает список всех записей на прием к врачу, включая дату, время и данные пациентов.

## **2.4 Построение модели системы**

Для проектирования системы необходимо определить основные компоненты и их взаимодействие со структурой. В интерфейсе используется библиотека customtkinter и некоторые элементы tkinter. Для базы данных используется sqlite3, так как эта библиотека входит в стандартную библиотеку Python (начиная с версии 2.5), поэтому её не нужно устанавливать отдельно.

База данных состоит из трех таблиц.

Сущности базы данных:

* пациенты ‘clients’: хранит информацию о каждом пациенте частной клиники;
* доктора ‘doctors': хранит информацию о докторах, которые работают в частной клинике;
* записи к врачу ‘appointments’: хранит информацию о записи пациента
* медицинские карты ‘medical\_card’: хранит информацию медицинских карт каждого пациента.

Атрибуты сущностей:

* ‘clients’: id\_client – первичный ключ, name - текстовый, last\_name - текстовый, phone – текстовый, login – текстовый, password - текстовый.
* ‘doctors’: id\_doctor – первичный ключ, full\_name текстовый.
* ‘appointments’: id\_appointment – первичный ключ, id\_doctor – вторичный ключ, id\_client – вторичный ключ, now\_time – текстовый, date - текстовый.
* ‘medical\_card’: id\_medical\_card – первичный ключ, id\_client – вторичный ключ, med\_card – текстовый.
* `Specialization`: id\_specialization – первичный ключ, specialization – текстовый.
* `SpecDoc`: id\_spec\_doc- первичный ключ, id\_doctor – вторичный ключ, id\_ specialization – вторичный ключ.

Связи между сущностями показаны в er-диаграмме (Рисунок 3):

* У доктора может быть много пациентов и у пациента может быть много врачей, поэтому связь - многие ко многим (N:N). Для третьей нормальной формы была создана таблица `Appointment` для создания связи один ко многим (1:N);
* У врача может быть много специальностей и у специальности может быть много докторов, поэтому связь многие ко многим (N:N). Для третьей нормальной формы была создана таблица `SpecDoc` для создания связи один ко многим (1:N)

Изображение выглядит как текст, диаграмма, число, Шрифт

Автоматически созданное описание

Рисунок 3. ER-диаграмма

## **2.5 Реализация интерфейса приложения**

Благодаря интерфейсу, приложением можно управлять быстрее и проще.

Компоненты интерфейса:

* Экран авторизации для пациентов: экран с полями ввода (логин, пароль), а также с кнопками «Войти», «Войти как врач» и «Регистрация». (Приложение 2)
* Экран авторизации для врача: экран с полями ввода (логин, пароль), а также кнопкой «Войти». (Приложение 3)
* Экран регистрации для пациентов: экран с полями ввода (логин, пароль, ФИО) и кнопкой «Регистрация». (Приложение 4)
* Экран регистрации для врачей: экран с полями ввода (логина, пароль, специальность, ФИО) и кнопкой «Регистрация». (Приложение 5)
* Экран личного кабинета пациента: экран с выводом информации пациента (id, ФИО), его ближайшие записи, а также кнопки «Запись на прием», «Распечатать талон», «Медицинская карта». (Приложение 6)
* Экран личного кабинета врача: экран с выводом информации врача (id, ФИО, специальность), ближайшие записи к данному доктору, кнопка «Добавить в мед. карту». (Приложение 7)
* Экран записи к врачу: экран со списком выбора врача и полями ввода даты и времени. (Приложение 8)
* Экран записи в медицинскую карту: экран с полем ввода для добавления записи в медицинскую карту и кнопкой «Добавить». (Приложение 9)

Технологический стек:

* Бэкенд: использовал библиотеку sqlite3 для хранения данных в базе данных; добавление и получение данных из базы данных.
* Фронтенд: использовал библиотеки tkinter и customtkinter; создаются окна, виджеты, формы ввода.

## **2.6 Модули системы**

1) Модуль «main»

Главный модуль «main» является точкой входа в приложение. Он отвечает за инициализацию и запуск всей системы. Основные функции этого модуля включают:

* Импорт необходимых библиотек и модулей: модуль импортирует другие модули, такие как «DB\_appointment», «cabinet», «autorisation», «check», «clients», «register\_clients», которые обеспечивают функциональность системы.
* Инициализация базы данных: в этом модуле вызывается функция «init\_database()» из модуля «DB\_appointment», которая создает необходимые таблицы в базе данных, если они еще не существуют.
* Запуск окна авторизации: модуль создает экземпляр класса «autorisation», который отвечает за процесс входа пользователей в систему. После успешной авторизации пользователи перенаправляются в соответствующие кабинеты (для клиентов или врачей).

2) Модуль «DB\_appointment»

Модуль «DB\_appointment» отвечает за взаимодействие с базой данных. Основные функции этого модуля включают:

* Создание и управление базой данных: модуль устанавливает соединение с базой данных SQLite и создает необходимые таблицы для хранения информации о клиентах, врачах, записях на прием и медицинских картах.
* Выполнение SQL-запросов: модуль предоставляет функции для выполнения различных SQL-запросов, таких как добавление, обновление и выборка данных из таблиц.

3) Модуль «clients»

Модуль clients отвечает за управление клиентами в базе данных. Он включает в себя следующие функции:

* Инициализация клиента: конструктор класса clients принимает параметры, такие как имя, фамилия, отчество, логин и пароль, и сохраняет их как атрибуты объекта.
* Вставка пользователя: метод insert\_user добавляет нового клиента в таблицу clients базы данных «clinics.db», используя переданные атрибуты объекта.
* Выбор медицинской карты: метод «select\_med\_card» извлекает медицинскую карту клиента из таблицы medical\_card, основываясь на логине клиента. Он возвращает результат запроса.

4) Модуль «cabinet»

Модуль «cabinet» реализует функциональность пользовательского интерфейса для клиентов. Основные функции этого модуля включают:

* Отображение информации о клиенте: модуль позволяет клиентам просматривать свои данные, такие как имя, фамилия и медицинская карта.
* Управление записями на прием: клиенты могут записываться на прием к врачам, просматривать свои записи и удалять их при необходимости.
* Функция печати талонов: модуль предоставляет возможность распечатки талонов на прием.

5) Модуль «cabinet\_doctor»

Модуль «cabinet\_doctor» предназначен для врачей и предоставляет интерфейс для работы с их записями. Основные функции этого модуля включают:

* Отображение информации о враче: врачи могут просматривать свои данные, такие как имя, фамилия и специализация.
* Добавление медицинских записей: врачи могут добавлять записи в медицинские карты пациентов, а также просматривать свои записи на прием.
* Управление записями пациентов: модуль позволяет врачам видеть записи пациентов и взаимодействовать с ними.

6) Модуль «autorisation»

Модуль «autorisation» отвечает за процесс авторизации пользователей. Основные функции этого модуля включают:

* Форма входа: модуль предоставляет графический интерфейс для ввода логина и пароля.
* Проверка учетных данных: модуль использует класс check\_user для проверки правильности введенных логина и пароля.
* Перенаправление пользователей: после успешной авторизации пользователи перенаправляются в соответствующий кабинет (клиента или врача).

7) Модуль «autorisation\_doctor»

Модуль «autorisation\_doctor» аналогичен модулю «autorisation», но предназначен для врачей. Основные функции этого модуля включают:

* Форма входа для врачей: модуль предоставляет интерфейс для авторизации врачей.
* Проверка учетных данных врачей: модуль использует класс «check\_user» для проверки логина и пароля врачей.

8) Модуль «doctors»

Модуль doctors управляет информацией о врачах в базе данных. Он включает в себя следующие функции:

* Инициализация врача: конструктор класса «doctors» принимает параметры, такие как имя, фамилия, отчество, логин, пароль и специализация, и сохраняет их как атрибуты объекта.
* Вставка врача: метод «insert\_doctor» добавляет нового врача в таблицу doctors базы данных «clinics.db», используя переданные атрибуты объекта.
* Выборка клиентов: метод «select» извлекает информацию о клиентах из таблицы «clients», включая идентификатор клиента, имя, фамилию и телефон. Он выводит информацию о последнем клиенте в списке.

9) Модуль «register\_clients»

Модуль «register\_clients» отвечает за регистрацию новых клиентов. Основные функции этого модуля включают:

* Форма регистрации: модуль предоставляет интерфейс для ввода данных новых клиентов.
* Проверка уникальности логина: модуль проверяет, существует ли уже пользователь с таким логином, и, если нет, добавляет нового клиента в базу данных.

10) Модуль «register\_doctors»

Модуль «register\_doctors» аналогичен модулю «register\_clients», но предназначен для регистрации врачей. Основные функции этого модуля включают:

* Форма регистрации для врачей: модуль предоставляет интерфейс для ввода данных новых врачей.
* Проверка уникальности логина: модуль проверяет, существует ли уже врач с таким логином, и, если нет, добавляет нового врача в базу данных.

11) Модуль «check»

Модуль check отвечает за проверку учетных данных пользователей и врачей. Основные функции этого модуля включают:

* Проверка логина и пароля: модуль содержит методы для проверки правильности введенных логина и пароля как для клиентов, так и для врачей.
* Проверка уникальности логина при регистрации: модуль проверяет, существует ли уже пользователь или врач с таким логином.

Все эти модули работают вместе, создавая функциональное приложение для управления клиникой. Пользователи могут:

* Регистрироваться и авторизовываться через интерфейс.
* Записывать и управлять записями на прием к врачам.

Также, врачи могут просматривать информацию о своих пациентах.

Таким образом, приложение обеспечивает полный цикл управления клиникой, начиная от регистрации клиентов и врачей до управления записями на прием и получения медицинской информации. Блок схема – схема, которая описывает алгоритмы и процессы программы. Блок-схема программы. (Приложение 10)

## **2.7 Сложность системы**

Алгоритмическая сложность программы зависит от различных факторов, включая количество операций с базой данных, количество элементов в списках и количество вызовов функций.

1. Операции с базой данных:

Запросы к базе данных, такие как SELECT, INSERT, UPDATE, DELETE, имеют сложность, зависящую от структуры таблиц и индексов. В общем случае, если таблицы не индексированы, сложность может быть O(n), где n — количество записей в таблице. Если используются индексы, сложность может быть O(log n) для поиска.

2. Регистрация пользователей:

Проверка существования пользователя при регистрации (метод check\_register) выполняет запрос к базе данных, который может иметь сложность O(n) в худшем случае.

Выборка данных:

Методы, которые выбирают данные (например, select\_med\_card, select\_specialization), также имеют сложность, зависящую от количества записей в таблицах.

4. Графический интерфейс:

Создание и управление элементами интерфейса (например, кнопками, метками) имеет постоянную сложность O(1) для каждого элемента, но общее время зависит от количества элементов, которые вы создаете.

5. Обработка событий:

Вся логика, связанная с обработкой событий (например, нажатие кнопок), также имеет постоянную сложность O(1) для каждого события, но может вызывать функции с различной сложностью.

6. Общая сложность:

Если рассматривать всю программу в целом, то сложность будет зависеть от того, как часто выполняются операции с базой данных и сколько данных обрабатывается. В общем случае, можно сказать, что сложность программы может варьироваться от O(1) до O(n) в зависимости от конкретных операций и их частоты.

Таким образом, алгоритмическая сложность программы в целом может быть охарактеризована как O(n) в худшем случае, где n — это количество записей в базе данных, с учетом того, что многие операции выполняются с постоянной сложностью O(1).

# **3. ТЕСТИРОВАНИЕ**

Во время разработки информационной системы проводились тестирование и отладка. Тестирование позволяет выявить ошибки и несоответствия в работе системы на ранних стадиях разработки, что значительно снижает риски и затраты на исправление проблем в будущем, Отладка же, в твою очередь, включает в себя анализ и исправление обнаруженных ошибок.

Для тестирования использовались тест-кейсы (позитивные и негативные), которые проводились по тест плану (Приложение 11).

Позитивные тест-кейсы:

1) Тест успешной авторизации клиента:

* Описание: ввод логина «yara» и пароля «yara».
* Ожидаемый результат: клиент успешно авторизуется и попадает в личный кабинет.
* Фактический результат: клиент успешно авторизуется и попадает в личный кабинет.

2) Тест успешной авторизации врача:

* Описание: ввод логина «yapetukhov-22» и пароля «yapetukhov22» врача.
* Ожидаемый результат: врач успешно авторизуется и попадает в личный кабинет врача.
* Фактический результат: врач успешно авторизуется и попадает в личный кабинет врача.

3) Тест успешной регистрации клиента:

* Описание: ввод уникального логина «Donskih06» и пароля «d\_06pasha» для нового клиента, фамилии «Донских», имени «Павел», отчества «Иванович».
* Ожидаемый результат: клиент успешно зарегистрирован и может войти в систему.
* Фактический результат: клиент успешно зарегистрирован и может войти в систему.

4) Тест успешной регистрации врача:

* Описание: ввод уникального логина «Mikufg» и пароля «Matvey2006\_doc» для нового врача, фамилии «Бородулин», имени «Матвей», отчества «Сергеевич» и выбор специальности «Хирург».
* Ожидаемый результат: врач успешно зарегистрирован и может войти в систему.
* Фактический результат: врач успешно зарегистрирован и может войти в систему.

5) Тест успешного добавления медицинской записи:

* Описание: врач добавляет в медицинскую книжку запись для заранее записанного пациента: «Прием от 11.12.2024. Врач Бородулин Матвей Сергеевич. Диагноз: осложнения описторхоза и амебиаза».
* Ожидаемый результат: запись успешно добавлена в базу данных.
* Фактический результат: запись успешно добавлена в базу данных.

6) Тест успешного получения медицинской карты:

* Описание: клиент в личном кабинете нажимает на кнопку «Просмотреть медицинскую карту».
* Ожидаемый результат: медицинская карта успешно выводится в папку с файлом программы.
* Фактический результат: медицинская карта успешно выводится в папку с файлом программы.

7) Тест успешного создания записи на прием:

* Описание: клиент в личном кабинете нажимает на кнопку записи на прием, в открывшемся окне вводит дату «12.12.2024», время «14:30» и врача «Бородулин Матвей Сергеевич - Хирург», нажимает на кнопку «Подтвердить запись».
* Ожидаемый результат: запись успешно добавлена в базу данных.
* Фактический результат: запись успешно добавлена в базу данных.

Негативные тест-кейсы:

1) Тест неуспешной авторизации клиента с неправильным паролем:

* Описание: ввод правильного логина «yara» и неправильного пароля «tara».
* Ожидаемый результат: появляется сообщение об ошибке "Неверный логин или пароль".
* Фактический результат: появляется сообщение об ошибке "Неверный логин или пароль".

2) Тест неуспешной авторизации врача с неправильным логином:

* Описание: ввод неправильного логина «yapetukhov22» и правильного пароля «yapetukhov22».
* Ожидаемый результат: появляется сообщение об ошибке "Неверный логин или пароль".
* Фактический результат: появляется сообщение об ошибке "Неверный логин или пароль".

3) Тест неуспешной регистрации клиента с существующим логином:

* Описание: ввод логина «yara», который уже существует в базе данных и ввод остальных данных: фамилии «Петухов», имени «Ярослав», отчества «Сергеевич», пароль «yara».
* Ожидаемый результат: появляется сообщение об ошибке "Пользователь с таким логином уже существует".
* Фактический результат: появляется сообщение об ошибке "Пользователь с таким логином уже существует".

4) Тест неуспешной регистрации врача с существующим логином:

* Описание: ввод логина «Mikufg», который уже существует в базе данных и ввод остальных данных: фамилии «Забанов», имени «Алексей», отчества «Дмитриевич», пароль «Zabanov06».
* Ожидаемый результат: появляется сообщение об ошибке "Пользователь с таким логином уже существует".
* Фактический результат: появляется сообщение об ошибке "Пользователь с таким логином уже существует".

5) Тест неуспешного создания записи на прием в уже занятое время:

* Описание: клиент пытается создать запись на прием в уже занятое время. В личном кабинете клиент нажимает кнопку «Запись на прием» и вводит дату «12.12.2024» и время «14:30» к доктору «Бородулин Матвей Сергеевич - Хирург», нажимает на кнопку «Подтвердить запись».
* Ожидаемый результат: появляется сообщение об ошибке "На это время уже есть запись к данному врачу".
* Фактический результат: появляется сообщение об ошибке "На это время уже есть запись к данному врачу".

Для тестирования программного обеспечения был применен метод белого ящика. Он позволил проверить поведение программы при различных сценариях ввода, охватить ключевые функциональные блоки приложения, и обеспечить качество их реализации.

# **ЗАКЛЮЧЕНИЕ**

В заключении данной работы была создана информационная система, которая включает в себя несколько компонентов: авторизация и регистрация пользователей и врачей, управлениями записями на прием, ведение медицинский карт.

По мере выполнения курсового проекта, был проведен анализ нескольких программ, связанных с данной темой. А также была достигнута основная цель - разработка информационной системы для автоматизации процессов управления частной клиникой, обеспечивающей эффективное взаимодействие между пациентами и медицинским персоналом.

Работа над данным проектом будет продолжаться:

* Будут учтены все ошибки, которые встречались при написании программы;
* Будет добавлен вывод медицинской карты клиента для докторов.

# **СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ**

**Федеральные законы:**

1. Федеральный закон "Об информации, информационных технологиях и о защите информации" от 27.07.2006 N 149-ФЗ - <https://www.consultant.ru/document/cons_doc_LAW_61798/>
2. Федеральный закон "О персональных данных" от 27.07.2006 N 152-ФЗ - <https://www.consultant.ru/document/cons_doc_LAW_61801/>
3. Федеральный закон "О безопасности критической информационной инфраструктуры Российской Федерации" от 26.07.2017 N 187-ФЗ - <https://www.consultant.ru/document/cons_doc_LAW_220885/>
4. РОССТАНДАРТ - <https://www.gost.ru/portal/gost/home/standarts/catalognational>
5. ГОСТ Р ИСО/МЭК 12207 - <https://docs.cntd.ru/document/1200082859>
6. ЕСПД - <https://www.swrit.ru/gost-espd.html>

**Интернет-источники:**

1. Объектно-ориентированное программирование - <https://metanit.com/python/tutorial/7.1.php>
2. Обработка ошибок и исключений - <https://metanit.com/python/tutorial/2.11.php>
3. Модули - <https://metanit.com/python/tutorial/2.10.php>
4. Работа с файлами - <https://metanit.com/python/tutorial/4.1.php>
5. Руководство по Tkinter - <https://metanit.com/python/tkinter/>
6. Руководство по CustomTkinter - https://customtkinter.tomschimansky.com/

# **Приложение 1 Диаграмма прецедентов**

**Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание**

# **Приложение 2 Экран авторизации пациентов**

**Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание**

# **Приложение 3 Экран авторизации для врачей**

**Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание**

# **Приложение 4 Регистрация для пациентов**

**Изображение выглядит как снимок экрана, текст, программное обеспечение, дизайн

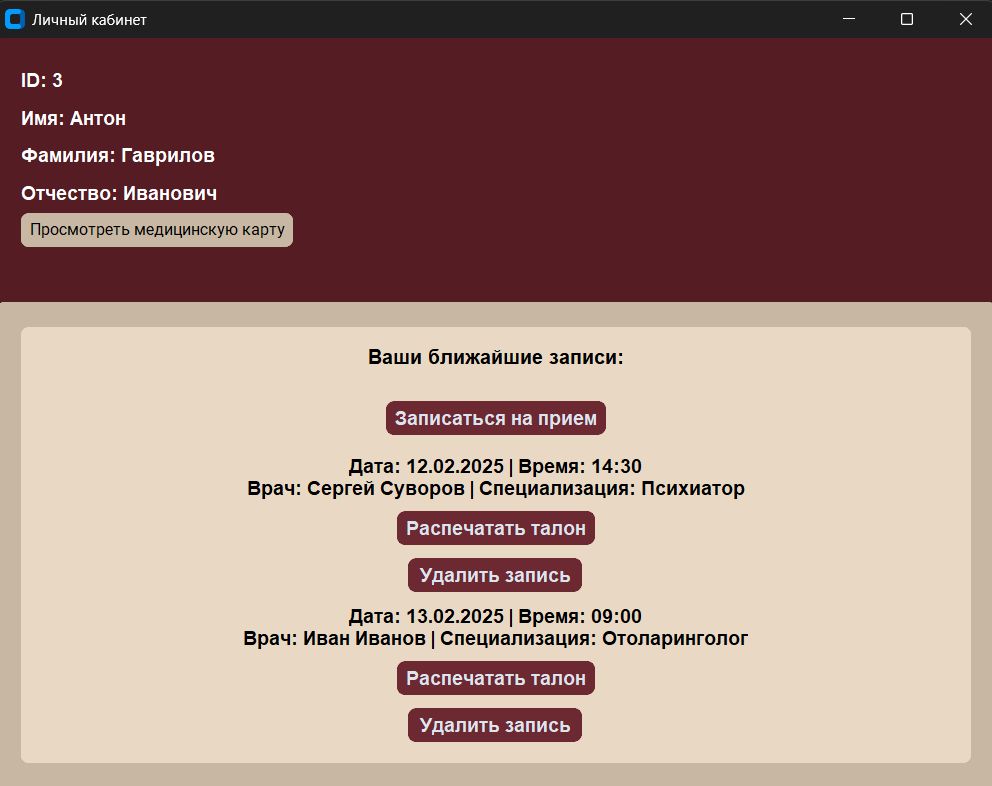
Автоматически созданное описание**

# **Приложение 5 Экран регистрации для врачей**

**Изображение выглядит как снимок экрана, текст, программное обеспечение, дизайн

Автоматически созданное описание**

# **Приложение 6 Экран личного кабинета пациента**

****

# **Приложение 7 Экран личного кабинета для врача**

**Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание**

# **Приложение 8 Экран записи ко врачу**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание**

# **Приложение 9 Экран записи в медицинскую карту**

**Изображение выглядит как текст, снимок экрана, мультимедиа, гаджет

Автоматически созданное описание**

# **Приложение 10 Блок-схема приложения**

**Изображение выглядит как текст, снимок экрана, диаграмма

Автоматически созданное описание**

# **Приложение 11 Таблица план тестирования**

Таблица 5

План тестирования для программного модуля частная клиника

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID теста** | **Описание теста (тип)** | **Предусловия** | **Шаги воспроизведения** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Авторизация клиента (позитивный) | 1.Пользователь находится в окне авторизации  2.Пользователь был ранее зарегистрирован в системе | 1. Ввести правильные данные в поля  «Логин», «Пароль».  2. Нажать кнопку «Войти». | Клиент успешно авторизуется и попадает в личный кабинет. | Клиент успешно авторизуется и попадает в личный кабинет. |
| 2 | Авторизация врача (позитивный) | 1.Клиент находится в окне авторизации.  2.Клиент был ранее зарегистрирован в системе. | 1. Ввести правильные данные в поля «Логин», «Пароль».  2. Нажать кнопку «Войти». | Врач успешно авторизуется и попадает в личный кабинет врача. | Врач успешно авторизуется и попадает в личный кабинет врача. |
| 3 | Регистрация клиента (позитивный) | 1. Клиент находится в окне регистрации  2. Клиент не был зарегистрирован в системе. | . Ввести уникальные значения в поля «Логин» и «Пароль» для нового клиента, а также «Фамилия», «Имя»,  «Отчество».  2. Нажать кнопку «Регистрация». | Клиент получает уведомление об успешной регистрации и может войти в систему. | Клиент получает уведомление об успешной регистрации и может войти в систему. |
| 4 | Регистрация врача (позитивный) | 1. Врач находится в окне регистрации.  2. Врач не был зарегистрирован в системе. | 1. Ввести уникальные значения в поля «Логин» и «Пароль» для нового врача, а также «Фамилия», «Имя»,  «Отчество», «Специальность».  2. Нажать кнопку «Регистрация». | Врач получает уведомление об успешной регистрации и может войти в систему. | Врач получает уведомление об успешной регистрации и может войти в систему. |

**Продолжение таблицы 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | Добавление медицинской записи врачом (позитивный) | 1. Врач зарегистрирован в системе.  2. Врач успешно прошел авторизацию.  3. К врачу записан минимум 1 клиент. | 1. В окне личного кабинета врача нажать на кнопку добавления мед записи конкретного клиента.  2. Ввести в отрывшемся поле новую медицинскую запись. | В базу данных сохраняется данные медицинской карты, введенные врачом. | В базу данных сохраняется данные медицинской карты, введенные врачом. |
| 6 | Получение медицинской карты клиентом (позитивная) | 1. Клиент зарегистрирован в системе.  2. Клиент успешно прошел авторизацию.  3. Медицинская карта клиента имеет минимум 1 записью | 1. В окне личного кабинета клиента нажать на кнопку «Просмотреть медицинскую карту». | В папке программы появится текстовый документ с данными медицинской карты. | В папке программы появится текстовый документ с данными медицинской карты. |
| 7 | Создание записи на прием (позитивный) | 1. Клиент зарегистрирован в системе.  2. Клиент успешно прошел авторизацию. | 1. В окне личного кабинета нажать на кнопку «Записаться на прием».  2. Выбрать врача, в поля «Дата» и «Время» ввести корректные данные даты и времени записи. | В личном кабинете клиента появится запись с выводом информации (врач, дата, время). | В личном кабинете клиента появится запись с выводом информации (врач, дата, время). |
| 8 | Просмотр записей на прием (позитивный) | 1. Клиент зарегистрирован в системе.  2. Клиент успешно прошел авторизацию.  3. Клиент записан минимум к 1 врачу. | Внизу в окне личного кабинета отображаются записи ко врачам. | Клиент в личном кабинете может просмотреть свои записи ко врачам. | Клиент в личном кабинете может просмотреть свои записи ко врачам. |

**Продолжение таблицы 5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | Авторизация клиента (негативный) | 1. Врач находится в окне авторизации.  2. Врач был ранее зарегистрирован в системе. | 1. В окне авторизации ввести неверное поле «Логин» и верное поле «Пароль».  2. Нажать кнопку «Вход» | Врач в окне авторизации получает сообщение «Пользователь с таким логином уже существует». | Врач в окне авторизации получает сообщение «Пользователь с таким логином уже существует». |
| 10 | Авторизация врача (негативный) | 1. Врач находится в окне авторизации.  2. Врач был ранее зарегистрирован в системе. | 1. В окне авторизации ввести неверное поле «Логин» и верное поле «Пароль».  2. Нажать кнопку «Вход» | Врач в окне авторизации получает сообщение «Пользователь с таким логином уже существует». | Врач в окне авторизации получает сообщение «Пользователь с таким логином уже существует». |
| 11 | Регистрация клиента (негативный) | 1. Клиент находится в окне регистрации.  2. Клиент не был зарегистрирован ранее в системе. | 1. В окне регистрации ввести в поле «Логин» логин, который уже существует, затем заполнить поля «Пароль», «Фамилия», «Имя», «Отчество» корректными данными.  2.Нажать кнопку «Регистрация». | Появляется сообщение об ошибке "Пользователь с таким логином уже существует". | Появляется сообщение об ошибке "Пользователь с таким логином уже существует". |
| 12 | Создание записи на прием (негативный) | 1. Клиент зарегистрирован в системе.  2. Клиент успешно прошел авторизацию. | 1. В окне личного кабинета нажать кнопку «Записаться на прием».  2. В окне записи ввести данные в поля «Дата», «Время» на уже занятые время и дату к определенному доктору. | Появляется сообщение об ошибке "На это время уже есть запись к данному врачу". | Появляется сообщение об ошибке "На это время уже есть запись к данному врачу". |