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 частной**

**ветеринарной клиники**

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

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 309ИС-22** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **С.А. Мейроян** |
| **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 Сурен Мейроян**

**ТЕМА: «Разработка программного модуля информационной**

**системы «Ресторан**

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 П. Д. Донских

*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

Оглавление

[ВВЕДЕНИЕ 6](#_Toc185471754)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ 8](#_Toc185471755)

[1.1. Информационное обеспечение задачи 8](#_Toc185471756)

[1.2. Обзор и анализ существующих программных решений 9](#_Toc185471757)

[1.3.  Постановка задачи. Структура входной и выходной информации 10](#_Toc185471758)

[**1.3.1. Основные задачи и функции** 10](#_Toc185471759)

[**1.3.2 Структура входной и выходной информации** 10](#_Toc185471760)

[2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДУЛЯ 12](#_Toc185471761)

[2.1. Построение модели системы 12](#_Toc185471762)

[2.2. Выбор инструментов 13](#_Toc185471763)

[2.3. Логическая структура 15](#_Toc185471764)

[2.4. Реализация интерфейса приложения 15](#_Toc185471765)

[2.5. Проектирование сценария 18](#_Toc185471766)

[2.6. Диаграмма классов 19](#_Toc185471767)

[2.7. Описание главного модуля 20](#_Toc185471768)

[3. Тестирование и отладка программного модуля 21](#_Toc185471769)

[3.1. Описание тестовых наборов модулей 21](#_Toc185471770)

[3.2. Описание применения средств тестирования и отладки 24](#_Toc185471771)

[ЗАКЛЮЧЕНИЕ 25](#_Toc185471772)

[Приложение 1. 27](#_Toc185471773)

[Приложение 2. 27](#_Toc185471774)

[Приложение 3. 28](#_Toc185471775)

[Приложение 4. 28](#_Toc185471776)

[Приложение 5. 28](#_Toc185471777)

[Приложение 6. 28](#_Toc185471778)

[Приложение 7. 29](#_Toc185471779)

# **ВВЕДЕНИЕ**

В условиях современного рaзвития здравоохранения особое значение приобретает вопрос эффективного упрaвления ветеринарными клиникaми. Быстрое развитие информационных технологий открывает новые перспективы для автоматизации управленческих процессов, что способствует улучшению качества обслуживания животных и повышению эффективности работы ветеринарного персонала.

*Актуальность темы* обусловлена увеличением потребности   
в автоматизации процессов упрaвления регистрационными записями, необходимости повышения качества обслуживания пациентов и важностью обеспечения эффективного хранения и обработки ветеринарной информации.   
Помимо этого, существует потребность в оптимизации рабочего времени как пользователей, так и ветеринарных специалистов.

*Значимость работы* заключается в разработке информационной   
системы, которая позволит автоматизировать регистрационный процесс частной ветеринарной клиники и повысить эффективность её работы.

*Объектом исследования* является ветеринарная клиника «Лапка   
помощи», а предметом - автоматизация управления регистрационными записями ветеринарной клиники с использованием информационных технологий.

*Цель работы*: разработка информационной системы для автоматизации процессов управления регистрационными записями ветеринарной клиники, обеспечивающей эффективное взаимодействие между пациентами и медицинским персоналом.

Задачи проекта:

* Анализ существующих бизнес-процессов в ветеринарных медицинских учреждениях
* Определение функциональных требований к разрабатываемой информационной системе
* Проектирование архитектуры информационной системы
* Создание пользовательского интерфейса для работы с системой
* Реализация основных модулей системы
* Тестирование и отладка разработанного программного обеспечения

Круг решаемых проблем:

* Организация записи на прием к ветеринарам
* Ведение электронной медицинской документации
* Управление расписанием работы медицинского персонала

В рамках проекта предполагается использование современных технологий программирования, включая Python для разработки backend-части   
системы и PyQT6 для создания пользовательского интерфейса для удобного использования программой.

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

# 1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ**

## **1.1. Информационное обеспечение задачи**

Ветеринарная клиника – это учреждение, предоставляющее медицинские услуги для животных. В процессе регистрации на прием   
и общения с ветеринаром используются различные термины, которые важно знать.

*Основные термины:*

* Запись на приём к ветеринару — это процесс, в ходе которого владелец животного оставляет заявку на получение ветеринарной помощи для своего питомца. Это может включать в себя осмотр, диагностику, лечение или профилактические мероприятия. Для записи на приём используются различные каналы связи: телефон, электронная почта, личный кабинет на сайте или в мобильном приложении ветеринарной клиники, а также специализированные порталы и сервисы для записи на услуги.
* Интерфейс — набор элементов и функций, через которые пользователь взаимодействует с программой или устройством.

*Процедуры и услуги в ветеринарной клинике:*

* Диагностика: включает в себя различные методы исследования, такие как рентгенография, УЗИ и лабораторные анализы;
* Лечение: может быть медикаментозным или хирургическим,   
  в зависимости от заболевания;
* Стационарное лечение: размещение животного в клинике для наблюдения и лечения;

*Регистрация на прием:*

При записи на прием в ветеринарную клинику обычно требуется предоставить следующую информацию:

* ФИО владельца: для идентификации клиента;
* Контактный телефон: для связи и подтверждения записи;
* Почта: резервный вариант для связи и подтверждения записи;
* Данные о животном

*Дополнительные услуги:*

* Онлайн консультация: возможность получить консультацию ветеринара удаленно;

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

В последние годы на рынке ветеринарного программного обеспечения появились различные решения, которые помогают оптимизировать процессы записи на прием, управления медицинскими данными и взаимодействия   
с клиентами. Рассмотрим несколько популярных платформ, таких как "Лапка помощи" и "Ветменеджер", а также их недостатки.

*1. Лапка помощи*

Описание: "Лапка помощи" — это платформа, которая позволяет владельцам домашних животных записываться на прием к ветеринарам через веб-интерфейс или мобильное приложение. Она включает функции управления медицинскими записями и коммуникации между клиентами   
и ветеринарами.

*Недостатки:*

* Ограниченные функции: Некоторые пользователи отмечают, что функционал системы может быть недостаточно широким для полноценного управления записями и регистрацией, включая отсутствие талонов на прием.
* Проблемы с пользовательским интерфейсом: Интерфейс может быть не интуитивно понятным для некоторых пользователей, что затрудняет его использование;

*2. Ветменеджер*

Описание: "Ветменеджер" — это комплексная система управления ветеринарной практикой, которая предлагает функции записи на прием, ведения медицинских карт, финансового учета и отчетности.

*Недостатки:*

* Сопротивление персонала: Некоторые ветеринары могут испытывать трудности с адаптацией к новой системе, что может замедлить внедрение программного обеспечения;
* Сложность обучения: Клиенты сообщают о необходимости значительного времени на изучение для освоения всех функций программы, особенно если они не имеют опыта работы с подобными системами;
* Проблемы с онлайн-доступом: В случае нестабильного интернет-соединения работа с облачной версией программы может быть затруднена, что ограничивает доступ к данным.

## **1.3.  Постановка задачи. Структура входной и выходной информации**

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

1. Ввод информации о записи клиента (Дата, время, ФИО ветеринара, ФИО клиента, данные о животном, номер телефона и почта клиента);

2. Проверка своих записей в личном кабинете;

3. Проверка заключения ветеринара после прима.

Возникает необходимость в компьютерной обработке вводимых данных о записях с целью хранения этой информации на носителях и анализа эффективности работы ветеринарной клиники.

### 

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

На вход подаётся модуль “appointment2\_0(MAIN).py” где клиент вводит свои данные для записи на прием.

Выходной файл –“ticket” где выводятся данные о записи клиента в виде талона.

# **2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОДУЛЯ**

**2.1. Построение модели системы**

В данной системе данные записываются и хранятся в формате .docx для удобной проверки записей для ветеринаров. Так же данные сохраняются   
в формате .png для обеспечения клиентов талонами на запись.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

*Рисунок 1. Талон на запись.*

Талон на запись создается при запуске модуля “appointment2\_0(MAIN).py” после введения всех данных, которые запрашивает программа (Имя ветеринара, ФИО клиента и вид услуги). Так же с использованием входных данных создаются дополнительные файлы формата.docx где все входные данные хранятся в одну строку. Это нужно для того, чтобы   
в будущем модуль “check\_appointment” смог считывать данные для вывода.

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

Автоматически созданное описание

*Рисунок 2. Проверка записей к ветеринару.*

В модуле “check\_appointment” считываются данные, созданные   
из папки с названием “appointment\_for\_vet”. Программа запрашивает имя   
ветеринара и ищет все записи, которые предназначены для него. Программа выводит список всех записей с информацией о записях (Имя ветеринара, ФИО клиента и вид услуги) который можно сортировать по дате.

**2.2. Выбор инструментов**

При выборе инструмента и языка программирования было проведено сравнение по критериям, которые представлены в таблицах 1 и 3.

Степень важности критерия выбиралась из: 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 2 языка программирования от 0 до 10 баллов за критерий.

*Таблица 2 –Оценка языков программирования*

|  |  |  |  |
| --- | --- | --- | --- |
| *Критерий/Язык программирования* | *Python* | | *C* |
| Функционал | 8 | | 9 |
| Удобство использования | 10 | | 7 |
| Скорость разработки | 7 | | 9 |
| **Итого** | 24 | 24 | |

*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, я сравнил 2 среды разработки от 0 до 10 баллов за критерий

*Таблица 4* – Оценка сред разработки

|  |  |  |
| --- | --- | --- |
| *Критерий/среда разработки* | *PyCharm* | *Microsoft Visual Studio* |
| Простота | 7 | 5 |
| Функционал | 10 | 10 |
| Удобство использования | 8 | 6 |
| Документация на русском языке | 9 | 9 |
| **Итого** | 34 | 30 |

По результатам сравнения была выбрана среда разработки PyCharm.

**PyCharm** — это интегрированная среда разработки (IDE) для Python,   
созданная компанией **JetBrains**. Она предоставляет разработчикам широкий спектр инструментов для эффективной работы с кодом на Python и другими языками программирования, такими как JavaScript, HTML, CSS, и SQL.

## **2.3. Логическая структура**

Проект представляет собой систему управления записями на прием   
в ветеринарной клинике, включающую графический интерфейс для регистрации клиентов и их животных. Основные классы, такие как Animal, Client,   
и Appointment, моделируют сущности системы, а класс Clinic управляет записями и их сохранением в файлы. Интерфейс AppointmentRegistrationWindow позволяет пользователям вводить данные и создает соответствующие   
объекты, которые затем сохраняются и визуализируются в виде билетов.   
Взаимодействие с расписанием ветеринаров осуществляется через загрузку данных из Excel-файла.

## **2.4. Реализация интерфейса приложения**

Приложение реализует интерфейс для регистрации и входа пользователей.

Описание существующего функционала:

1. **Импорт библиотек**: используются библиотеки sys, pandas, PyQt6 для создания графического интерфейса.

2. **КлассAppointment** (Classes/appointment.py)

* Назначение: представляет запись на прием в ветеринарной клинике.
* Атрибуты:
* animal: Экземпляр класса Animal, представляющий животное, связанное с записью.
* client: Экземпляр класса Client, представляющий клиента, который владеет животным.
* date: Строка, представляющая дату приема.
* time: Строка, представляющая время приема.

3. **Класс Animal** (Classes/animal.py)

* Назначение: представляет животное, зарегистрированное на прием.
* Атрибуты:
* name: Имя животного.
* species: Вид животного.
* breed: Порода животного.
* age: Возраст животного в годах.

4. **Класс Client** (Classes/client.py)

* Назначение: представляет клиента, который владеет животным.
* Атрибуты:
* name: Имя клиента.
* phone: Телефонный номер клиента.
* email: Электронная почта клиента.

5. **Модуль расписания ветеринаров** (Classes/veterinarian\_schedule.py)

Функции:

* load\_veterinarian\_schedule (filename: str): загружает расписание ветеринаров из Excel-файла в DataFrame с помощью pandas. Проверяет наличие файла и выбрасывает FileNotFoundError, если файл не найден.
* select\_veterinarian(schedule\_df): позволяет пользователю выбрать ветеринара из загруженного расписания. Выводит список ветеринаров   
  и предлагает пользователю выбрать одного, введя номер.

6. **Окно регистрации на прием** (Classes/registration\_window.py)

* Назначение: предоставляет графический интерфейс для регистрации на прием с использованием PyQt6.

Функции:

* Поля ввода для данных клиента и животного (имя, телефон, email, вид, порода, возраст).
* Комбобоксы для выбора даты, времени приема и ветеринара.
* Кнопка для добавления записи, которая вызывает метод add\_appointment.
* Метод add\_appointment проверяет корректность ввода, создает экземпляры Client, Animal и Appointment, и добавляет запись в клинику.
* Также сохраняет детали записи в файл и генерирует изображение   
  талона.

7. **Модуль талонов** (Classes/ticket.py)

Функции:

* create\_ticket\_image(appointment: Appointment, file\_path: str):   
  Создает изображение талона с деталями записи и сохраняет его по указанному пути.
* open\_ticket(file\_path): Открывает изображение талона с помощью системной команды (для macOS).

## **2.5. Проектирование сценария**

В данном разделе приведён пример сценария использования программы клиентом.

После запуска приложения клиент может: записаться на прием   
к ветеринару вводя свои личные данные, выбрав ветеринара и день приема.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

*Рисунок 3. Сценарий использования приложением.*

**2.6. Диаграмма классов**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

*Рисунок 4. Диаграмма классов*

*Объяснение диаграммы:*

* Client: хранит информацию о клиенте (имя, телефон, email).
* Animal: хранит информацию о животном (имя, вид, порода, возраст).
* Appointment: связывает клиента и животное с датой и временем приема.
* Clinic: управляет записями на прием, позволяет добавлять записи и сохранять их в документ.
* AppointmentRegistrationWindow: GUI для регистрации записей на прием, использует класс
* Clinic для добавления записей.
* Ticket: Содержит функции для создания и открытия изображений талонов.
* VeterinarianSchedule: загружает расписание ветеринаров   
  и позволяет выбирать ветеринара.

## **2.7. Описание главного модуля**

**1. Основные функции и логика:**

Импорт необходимых модулей и классов:

* Импортируются модули из PyQt6 для создания графического интерфейса.
* Импортируются классы и функции из других файлов, такие как Clinic, AppointmentRegistrationWindow, и load\_veterinarian\_schedule.

**2. Функция main():**

Создание экземпляра приложения:

* app = QApplication ([]): создается экземпляр приложения PyQt6, который управляет основным циклом событий.

Загрузка расписания ветеринаров:

* schedule\_df=load\_veterinarian\_schedule('ToDo/Classes\_for\_veterinars/veterinarian\_schedule.xlsx'): загружается расписание ветеринаров из Excel-файла в DataFrame.
* Veterinarians = ...: извлекается список ветеринаров с именами   
  и фамилиями из загруженного расписания.
* Создание экземпляра клиники и окна регистрации:
* clinic = Clinic (): Создается экземпляр класса Clinic, который управляет записями.
* window = AppointmentRegistrationWindow (clinic, veterinarians):   
  создается окно регистрации на прием, передавая в него объект клиники   
  и список ветеринаров.

Отображение окна и запуск приложения:

* window.show(): Окно регистрации отображается на экране.
* app.exec(): запускается основной цикл приложения, который обрабатывает события и взаимодействие с пользователем.

**3. Точка входа в программу:**

* if \_\_name\_\_ == "\_\_main\_\_": main (): Проверяется, что модуль запущен как основная программа, и вызывается функция main().

# **3. Тестирование и отладка программного модуля**

## **3.1.** **Описание тестовых наборов модулей**

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 (см приложение 7).

Тест кейсы:

1: Успешная регистрация записи

*Описание*: Этот тест проверяет, что пользователь может успешно зарегистрировать запись, введя корректные данные во все поля.

*Ожидаемый* *результат*: Запись успешно добавлена, отображается сообщение об успехе.

*Фактический результат:* Запись успешно добавлена, отображается   
сообщение об успехе.

2: Успешное создание талона

*Описание*: Этот тест проверяет, что после успешной регистрации записи талон создается и открывается.

*Ожидаемый результат:* талон успешно создан и открыт.

*Фактический результат:* талон успешно создан и открыт.

3: Сохранение данных в файл

*Описание*: Этот тест проверяет, что после успешной регистрации записи данные сохраняются в файл формата DOCX в указанной директории.

*Ожидаемый результат:* Файл DOCX успешно создан и содержит   
данные о записи.

*Фактический результат:* Файл DOCX успешно создан и содержит   
данные о записи.

4: Успешное удаление записи

*Описание:* Этот тест проверяет, что пользователь может успешно   
удалить существующую запись.

*Ожидаемый результат:*Запись успешно удалена, и система подтверждает удаление.

*Фактический результат:* Запись успешно удалена, и система подтверждает удаление.

5: Пустые поля ввода

*Описание*: Этот тест проверяет, как система реагирует на попытку добавить запись без заполнения обязательных полей.

*Ожидаемый результат:* появляется сообщение об ошибке, требующее заполнить все поля.

*Фактический результат:* появляется сообщение об ошибке, требующее заполнить все поля.

6: Некорректный формат телефона

*Описание*: Этот тест проверяет, что система правильно обрабатывает   
некорректный формат номера телефона.

*Ожидаемый результат*: появляется сообщение об ошибке, указывающее на некорректный формат телефона.

*Фактический результат:*сообщение не появляется. Программа игнорирует эту ошибку.

7: Некорректный возраст животного

*Описание:* Этот тест проверяет, как система реагирует на ввод отрицательного значения в поле возраста животного.

*Ожидаемый результат:* появляется сообщение об ошибке, указывающее на некорректный возраст.

*Фактический результат:* сообщение не появляется. Программа игнорирует эту ошибку.

8: Дублирование записи

*Описание:*Этот тест проверяет, как система обрабатывает попытку   
добавить запись с данными, которые уже существуют в системе.

*Ожидаемый результат:* появляется сообщение об ошибке, указывающее на дублирование записи.

*Фактический результат:* программа сохраняет данную запись.

9: Ввод нечислового значения в поле возраста

*Описание:* Этот тест проверяет, как система реагирует на ввод нечислового значения в поле возраста животного.

*Ожидаемый результат:* появляется сообщение об ошибке, указывающее на некорректный формат возраста.

*Фактический результат:* появляется сообщение об ошибке, указывающее на некорректный формат возраста если введённое число отрицательное, если число не целое оно округляется.

10: Успешное отображение списка записей

*Описание:*Этот тест проверяет, что все существующие записи корректно отображаются в интерфейсе для просмотра.

*Ожидаемый результат:* Все записи корректно отображаются в списке.

*Фактический результат:* записи не отображаются в списке.

В результате тестирования было обнаружено, что из-за проблем с локализацией при добавлении нового талона в файл с расширением png текст   
неправильно загружается в файл.



*Рисунок 5. Проблема локализации файла с талоном*

Решение проблемы: в модуле для создания талона (Ticket) в формате для текста использовать параметр текста «encoding='UTF-8'»

## **3.2. Описание применения средств тестирования и отладки**

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.

Каждый функциональный блок тестировался отдельно, что позволило изолировать возможные ошибки.

# **ЗАКЛЮЧЕНИЕ**

В процессе создания курсового проекта была создана система классов   
для упрощения регистраций на прием к ветеринару.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки с работой в PyCharm.

Так же планируется продолжить работу над данным проектом с целью   
расширения возможностей и удобства для пользователей. Планы по доработке   
представлены ниже.

1. Улучшить визуальную часть для упрощения пользованием   
   программой
2. Доработать и оптимизировать код
3. Добавить функцию для ветеринаров которая будет требовать   
   написания заключения после приема.
4. Добавить базу данных где будут хранится данные о всех ветеринарах пациентах, их животных и записях на прием.**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

**Федеральные законы:**

1. Федеральный закон "Об информации, информационных технологиях и о защите информации" от 27.07.2006 N 149-Ф3 [https://www.consultant.ru/document/cons doc LAW\_61798/](https://www.consultant.ru/document/cons%20doc%20LAW_61798/)

2. Федеральный закон "О персональных данных" от 27.07.2006 N 152-Ф3 [https://www.consultant.ru/document/cons doc LAW 61801/](https://www.consultant.ru/document/cons%20doc%20LAW%2061801/)

3. Федеральный закон "О безопасности критической информационной инфраструктуры Российской Федерации" от 26.07.2017N 187-Ф3 <https://www.consultant.ru/document/cons_doc_LAW_220885/>

**ГОСТы:**

4. РОССТАНДАРТ https://www.gost.ru/portal/gost/home/standarts/catalognational/

5.ГОСТ PИСО/МЭК 12207<https://docs.cntd.ru/document/1200082859>

6. ЕСПДhttps://www.swrit.ru/gost-espd.html

**Источники для Python:**

1. Объектно-ориентированное программирование https://metanit.com/python/tutorial/7.1.php

2.Обработка ошибок исключений <https://metanit.com/python/tutorial/2.11.php>

3.Модули <https://metanit.com/python/tutbrial/2.10.php>

4.Работа с файлами<https://metanit.com/python/tutorial/4.1.php>

5.Руководство по PyQt6 <https://habr.com/ru/companies/skillfactory/articles/599599/> https://www.pythonguis.com/pyqt6-tutorial/

# **Приложение 1.**

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

Автоматически созданное описание

Рисунок 1. Основное меню регистрации

# **Приложение 2.**

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

Автоматически созданное описание  
Рисунок 2. Выбор дня недели для записи к ветеринару (дни берутся из таблицы excel)

# **Приложение 3.**

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

Автоматически созданное описание  
Рисунок 3. Выбор времени для записи к ветеринару (дни берутся из таблицы excel)

# **Приложение 4.**

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

Автоматически созданное описание  
Рисунок 4. Выбор ветеринара для записи (дни берутся из таблицы excel)

# **Приложение 5.**

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Автоматически созданное описание  
Рисунок 5. Уведомление об ошибке о пустых полях ввода в главном меню

# **Приложение 6.**

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Автоматически созданное описание  
Рисунок 6. Уведомление об успешном создании записи

# **Приложение 7.**

*Таблица 5* – *план тестирования*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***ID***  ***Теста*** | *Описание теста (тип)* | *Предусловия* | *Шаги для воспроизведения* | *Ожидаемый результат* | *Фактический результат* |
| 1 | Успешная Регистрация записи  (Позитивный) | Приложение запущено, все поля ввода пусты. | 1.Введите корректные данные во все поля.  2.Выберите дату, время и ветеринара.  3. Нажмите кнопку "Добавить запись". | Запись успешно добавлена, отображается сообщение об успехе | Запись успешно добавлена, отображается сообщение об успехе |
| 2 | Успешное создание талона  (Позитивный) | Запись успешно добавлена. | Проверьте, что изображение билета создано и открыто. | Талон успешно создан и открыт для просмотра. | Талон успешно создан и открыт для просмотра. |
| 3 | Сохранение данных в файл  (Позитивный) | Запись успешно добавлена. | Проверьте, что файл DOCX создан в указанной директории. | Файл DOCX успешно создан и содержит данные о записи. | Файл DOCX успешно создан и содержит данные о записи. |
| 4 | Успешное удаление записи  (Позитивный) | Запись существует в системе. | Удалите существующую запись. | Запись успешно удалена, и система подтверждает удаление. | Запись успешно удалена, и система подтверждает удаление. |
| 5 | Пустые поля ввода  (Позитивный) | Приложение запущено, все поля ввода пусты. | "Добавить запись" без ввода данных. | Появляется сообщение об ошибке, требующее заполнить все поля. | Появляется сообщение об ошибке, требующее заполнить все поля. |
| 6 | Некорректный формат телефона  (негативный) | Приложение запущено. | 1.Введите некорректный номер телефона (например, "abc123").  2 Заполните остальные поля корректно.  3 Нажмите кнопку "Добавить запись". | Появляется сообщение об ошибке, указывающее на некорректный формат телефона. | Сообщение не появляется. Программа игнорирует эту ошибку |
| 7 | Некорректный возраст животного  (негативный) | Приложение запущено. | 1. Введите отрицательное значение в поле возраста животного.  2. Заполните остальные поля корректно.  3. Нажмите кнопку "Добавить запись". | Появляется сообщение об ошибке, указывающее на некорректный возраст. | Сообщение не появляется. Программа игнорирует эту ошибку |
| 8 | Дублирование записи  (негативный) | Запись с такими же данными уже существует. | 1. Введите данные, идентичные уже существующей записи.  2. Нажмите кнопку "Добавить запись". | Появляется сообщение об ошибке, указывающее на дублирование записи. | Программа сохраняет данную запись |

Продолжение таблицы 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 9 | Ввод нечислового значения в поле возраста  (негативный) | Приложение запущено. | 1. Введите нечисловое значение в поле возраста (например, "два").  2. Заполните остальные поля корректно.  3. Нажмите кнопку "Добавить запись". | Появляется сообщение об ошибке, указывающее на некорректный формат возраста. | Появляется сообщение об ошибке, указывающее на некорректный формат возраста если введённое число отрицательное, если число не целое оно округляется |
| 10 | Успешное отображение списка записей  (негативный) | В системе есть несколько записей. | Откройте интерфейс для просмотра всех записей. | Все записи корректно отображаются в списке. | Записи не отображаются в списке |