СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc184600133)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc184600134)

[1.1. Введение в предметную область 5](#_Toc184600135)

[1.2. Анализ готовых решений 8](#_Toc184600136)

[2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 11](#_Toc184600137)

[2.1. Общее назначение системы 11](#_Toc184600138)

[2.2. Требования к функциям, выполняемым системой 12](#_Toc184600139)

[2.3 План тестирования 13](#_Toc184600140)

[3. РЕАЛИЗАЦИЯ ПРОЕКТА СИСТЕМЫ 15](#_Toc184600141)

[3.1. Описание среды разработки 15](#_Toc184600142)

[3.1.1. Выбор и описание инструментария 15](#_Toc184600143)

[3.1.2. Обоснование выбора инструментария 16](#_Toc184600144)

[3.2 Разработка программного модуля 16](#_Toc184600145)

[3.2.1 Реализация пользовательского интерфейса программы 17](#_Toc184600146)

[3.2.2 Описание функциональных узлов модуля с кодом 20](#_Toc184600147)

[3.2.3. Результат работы и тестирования 22](#_Toc184600148)

[ЗАКЛЮЧЕНИЕ 25](#_Toc184600149)

[СПИСОК ЛИТЕРАТУРЫ 26](#_Toc184600150)

[Приложение 1. 27](#_Toc184600151)

[Приложение 2. 29](#_Toc184600152)

[Приложение 3. 39](#_Toc184600153)

[Приложение 4. 54](#_Toc184600154)

# ВВЕДЕНИЕ

Компьютерные игры становятся все более значимым инструментом в обучении и развитии детей. В современном обществе они не только служат источником развлечения, но и активно способствуют формированию ключевых когнитивных навыков, развитию моторики и социальным взаимодействиям. Исследования показывают, что игры могут улучшать внимание, память и способности к решению проблем, что особенно заметно в стратегических играх, требующих планирования и анализа, а также в головоломках, которые задействуют логическое мышление и визуальное восприятие.

Помимо этого, многие компьютерные игры способствуют развитию мелкой моторики и координации движений, особенно те, которые используют контроллеры или технологии виртуальной реальности. Многопользовательские игры становятся лагерем для формирования социальных навыков, таких как работа в команде и управление конфликтами. Умеренное время, проведенное за играми, может позитивно сказаться на интеллектуальном развитии, делая игры отличным дополнением к традиционным методам обучения. Однако важно выбрать соответствующие возрасту и интересам ребенка игры и контролировать время, проведенное за ними, чтобы максимально использовать их образовательный потенциал.

Целью курсового проекта является разработка программного модуля системы «Игра «Собачья академия». Проект направлен на создание детской 2D-игры, которая будет не только развлекать, но и обучать детей в возрасте от 6 до 10 лет основам ухода за собаками.

Задачи курсового проекта:

1. Проанализировать предметную область, чтобы понять роль игр в обучении детей.
2. Изучить готовые решения на рынке детских игр и выявить их недостатки.
3. Подготовить техническое задание для разработки игры.
4. Разработать план тестирования, чтобы обеспечить качество продукта.
5. Обосновать выбор инструментов и средств разработки, подходящих для реализации проекта.
6. Описать реализацию технического задания, включая ключевые этапы разработки.
7. Провести тестирование игры, чтобы убедиться в ее функциональности и образовательной ценности.

Объектом исследования является процесс обучения, а предметом — создание, хранение и обработка данных, связанных с проектированием и разработкой образовательной игры «Собачья академия».

Актуальность данного проекта заключается в необходимости разработки игр, которые эффективно объединяют развлечение и обучение. На сегодняшний день многие существующие на рынке решения не выделяют образовательный компонент, что делает «Собачью академию» уникальным предложением, способным занять свою нишу. Игра не только развлекает, но и способствует воспитанию таких важных качеств, как ответственность, любовь и забота, что, в свою очередь, содействует социальному взаимодействию и гармоничному развитию детей в обществе.

Таким образом, проект «Собачья академия» представляет собой не просто развлекательную игру, а полноценный обучающий инструмент, который поможет детям развивать навыки заботы о питомцах и формировать ответственное отношение к животным. Это делает его актуальным и востребованным в условиях современного рынка детских игр.

# 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Введение в предметную область

История образовательных игр, включая игры про уход за питомцами, началась в 1980-х с текстовых приключений. В 1990-х, благодаря развитию графики, появились более сложные обучающие игры. С 2000-х, с ростом мобильных устройств и соцсетей, такие игры, как «Собачья академия», стали популярны, используя геймификацию для обучения и развития навыков ухода за животными.

Предметная область разработки программного модуля для информационной системы «Игра «Собачья академия» охватывает создание интерактивного программного обеспечения, которое сочетает в себе элементы обучения, развлечения и взаимодействия. Основная цель данной игры — обучение пользователей основам ухода за собаками, их воспитанию и дрессировке через игровые механики. Важным аспектом является использование информационных технологий для создания увлекательного и образовательного контента, который будет доступен на ПК.

Информационные технологии играют ключевую роль в разработке таких игр, обеспечивая возможность создания графики, анимации, звукового сопровождения и взаимодействия с пользователем. Использование современных языков программирования, игровых движков и платформ для разработки позволяет создавать высококачественные и интерактивные приложения. Кроме того, технологии облачных вычислений и интернет-сервисов открывают новые горизонты для многопользовательских режимов и интеграции с социальными сетями, что значительно увеличивает вовлеченность пользователей.

Теоретические основы разработки программного модуля для игры «Собачья академия» включают в себя несколько ключевых аспектов. Во-первых, это психология обучения, которая помогает понять, как пользователи воспринимают информацию и как лучше всего организовать процесс обучения через игру. Во-вторых, это принципы геймдизайна, которые определяют, как создать увлекательный и интересный игровой процесс, способствующий обучению.

Методология разработки информационных систем (ИС) включает в себя системный анализ, проектирование, реализацию и тестирование программного обеспечения. Связь между теоретическими основами и методологией проявляется в том, что понимание психологии пользователей и принципов геймдизайна позволяет более эффективно применять методологии разработки. Например, использование Agile-методов позволяет гибко реагировать на изменения в требованиях пользователей и адаптировать игру в процессе разработки, что особенно важно в условиях быстро меняющегося рынка.

Современные требования к информационным системам в области разработки игр, таких как «Собачья академия», включают в себя несколько ключевых аспектов. Во-первых, это высокая интерактивность и увлекательность игрового процесса. Игроки ожидают, что игра будет не только развлекать, но и обучать, предоставляя полезную информацию о собаках и их воспитании.

Во-вторых, важна адаптивность контента, позволяющая подстраиваться под уровень знаний и навыков пользователя. Это может включать в себя различные уровни сложности, а также возможность выбора различных сценариев и заданий.

Безопасность данных пользователей и защита личной информации становятся все более актуальными. Игры, ориентированные на детей, должны соответствовать законодательным нормам, касающимся защиты данных и безопасности контента.

Законодательные аспекты, касающиеся информационных систем, имеют особое значение для разработки игр, ориентированных на детей, в Российской Федерации. В стране действуют несколько ключевых законов и нормативных актов, регулирующих защиту персональных данных и безопасность информации. Основным документом в этой области является Федеральный закон от 27 июля 2006 года № 152-ФЗ «О персональных данных». Этот закон устанавливает правила сбора, хранения, обработки и передачи персональных данных, требуя от разработчиков получения согласия пользователей на обработку их данных, а также обеспечения их безопасности и конфиденциальности. Особенно важно соблюдать эти требования, когда речь идет о детях, так как их данные требуют повышенной защиты.

Кроме того, в России существуют ГОСТы, касающиеся защиты информации и обеспечения безопасности данных. Например, ГОСТ Р 15249-2012 «Системы управления информационной безопасностью. Требования» определяет требования к системам управления информационной безопасностью, которые могут быть применимы к разработке программного обеспечения, включая игры.

Также следует учитывать закон «О защите детей от информации, причиняющей вред их здоровью и развитию», который регулирует контент, доступный для детей, и требует от разработчиков соблюдения определенных стандартов безопасности и этики.

Не менее важным аспектом является соблюдение авторских прав на контент, используемый в игре. Разработчики должны быть уверены, что имеют право использовать все элементы, включая графику, музыку и текст, чтобы избежать юридических проблем. Это включает в себя необходимость получения лицензий на использование защищенных авторским правом материалов и соблюдение норм, касающихся интеллектуальной собственности.

Существует несколько методологий разработки программного обеспечения, каждая из которых имеет свои преимущества и недостатки. Итеративные и инкрементальные подходы, такие как Agile, позволяют разработчикам работать в коротких циклах, называемых спринтами, что дает возможность быстро реагировать на изменения и улучшать продукт на основе обратной связи от пользователей. Это особенно важно для разработки игр, где требования могут изменяться в зависимости от отзывов игроков.

Scrum, как часть Agile, предлагает структуру для управления проектами, позволяя командам организовывать свою работу и достигать поставленных целей. Этот подход способствует более эффективному взаимодействию между членами команды и позволяет быстрее достигать результатов.

## 1.2. Анализ готовых решений

Примеры информационных систем и их характеристики

1. Nintendogs

Описание:

Nintendogs — это одна из самых известных игр, посвященных уходу за собаками, выпущенная для консоли Nintendo DS. Игра позволяет пользователям ухаживать за виртуальными щенками, обучать их командам и участвовать в различных соревнованиях.

Характеристика:

* Интерактивность: Игроки могут взаимодействовать с питомцами с помощью сенсорного экрана и микрофона, что создает ощущение реального общения.
* Обучение: Игра включает в себя элементы дрессировки, где пользователи могут обучать своих собак выполнять команды, что способствует развитию навыков и понимания ухода за животными.
* Социальные функции: Возможность обмена питомцами с другими игроками через Wi-Fi, что добавляет элемент социализации и взаимодействия.

2. Pet Society

Описание:

Pet Society — это социальная игра, доступная на платформе Facebook, где пользователи могут создавать и ухаживать за виртуальными питомцами, участвовать в различных мини-играх и взаимодействовать с друзьями.

Характеристика:

* Социальное взаимодействие: Игра позволяет пользователям общаться друг с другом, обмениваться подарками и посещать дома друзей, что создает сообщество вокруг игры.
* Кастомизация: Игроки могут настраивать внешний вид своих питомцев и домов, что способствует индивидуализации игрового процесса.
* Мини-игры: В игре присутствует множество мини-игр, которые позволяют зарабатывать виртуальную валюту для покупки предметов и улучшений.

3. Dog Simulator

Описание: Dog Simulator — это игра, в которой пользователи могут управлять виртуальной собакой, исследовать открытый мир и выполнять различные задания.

Характеристика:

* Открытый мир: Игроки могут свободно исследовать окружающую среду, что создает ощущение свободы и приключения.
* Разнообразие заданий: Игра предлагает множество различных миссий и задач, которые помогают игрокам лучше понять поведение собак и их потребности.
* Обучение через игру: Игроки могут обучать свою собаку различным трюкам и командам, что способствует развитию навыков и понимания ухода за питомцем.

4. My Dog: Pet Care Game

Описание:

My Dog: Pet Care Game — это мобильное приложение, которое позволяет пользователям ухаживать за виртуальной собакой, обучать ее и участвовать в различных активностях.

Характеристика:

* Уход за питомцем: Игроки могут кормить, мыть и играть со своей собакой, что помогает развивать навыки заботы о животных.
* Обучение: Игра включает в себя элементы дрессировки, где пользователи могут обучать свою собаку выполнять команды и участвовать в соревнованиях.
* Графика и анимация: Высококачественная графика и анимация делают игровой процесс более увлекательным и реалистичным.

Таким образом, предметная область разработки программного модуля информационной системы «Игра «Собачья академия» охватывает множество аспектов, включая теоретические основы, современные требования, законодательные нормы и методологии разработки. Понимание назначения программных продуктов и их роли в обучении пользователей основам ухода за собаками является ключевым для успешной реализации проекта.

# 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 2.1. Общее назначение системы

Система "Собачья академия" предназначена для обучения пользователей основам ухода за собаками через игровой процесс и образовательные материалы. Основная цель — повышение осведомленности и ответственности владельцев домашних животных через интерактивные и увлекательные механики.

Цели и задачи системы:

* Образовательная: Повышение знаний пользователей о породах, их потребностях и уходе. Доступна база знаний с текстами, изображениями и видео.
* Развлекательная: Создание увлекательного опыта через игровые механики, такие как сбор косточек и прохождение уровней.
* Социальная: Формирование у пользователей ответственного отношения к содержанию домашних животных.
* Эффекты от внедрения:
* Повышение знаний о правильном уходе за собаками.
* Увеличение вовлеченности пользователей благодаря интерактивным заданиям и прогрессу в игре.
* Создание устойчивой пользовательской базы благодаря образовательному и развлекательному контенту.
* Расширяемость системы для добавления новых пород, материалов и функций.
* Функциональные возможности:
* Персонализированные профили: Статистика, достижения и прогресс пользователя.
* Интерактивная база знаний: Доступ к информации о породах, характеристиках и уходе.
* Обучающие вопросы: Проверка теоретических и практических знаний.
* Прогресс и уровни: Разные уровни сложности с новыми элементами игры.
* Мультимедийная поддержка: Изображения и видеоуроки для наглядного обучения.

## 2.2. Требования к функциям, выполняемым системой

Программный модуль игры "Собачья академия" выполняет следующие ключевые задачи:

* Авторизация и регистрация
  + Проверка учетных данных пользователей (логин, пароль) для предоставления доступа к системе.
  + Создание нового профиля с сохранением данных о пользователе в базе данных.
* Управление данными
  + Получение, обработка и предоставление информации о пользователях, собаках и связанных вопросах.
  + Обновление данных об игровых достижениях и уровне пользователя.
* Игровой процесс
  + Управление игровыми сессиями: установка уровней сложности, сбор косточек, расчет очков, а также учет состояния собаки (здоровье, голод, сонливость).
  + Сохранение прогресса игры, включая временные показатели и шаги игрока.
* Администрирование
  + Администратор может управлять профилями пользователей, базой вопросов, информацией о собаках и анализировать статистику.

Основные объекты, участвующие в работе системы:

* Пользователь (User): пользователь, создающий учетную запись, авторизующийся в системе, взаимодействующий с собакой и проходящий уровни. информация о пользователе включает логин, уровень и достижения.
* Собака (Dog): каждая собака связана с конкретным пользователем. У собаки есть характеристики, поведение, порода и рекомендации по уходу.
* Игровая сессия (GameSession): сохраняет информацию о ходе игры, включая текущий уровень, время, очки и состояние собаки.
* Вопросы (Questions): содержат задания для пользователей, связанные с собаками, включая текст вопроса, изображения и справочную информацию.
* Уведомления (Notifications): отправляются пользователям для информирования о событиях игры или важной информации.
* Администратор (Admin): управляет всеми аспектами игры, включая профили пользователей, данные о собаках и базу вопросов.

Связи между объектами показаны на ER-диаграмме базы данных (Рисунок 2, Приложение 1).

## 2.3 План тестирования

Цель тестирования заключается в проверке работы игры и ее функциональных компонентов, а также в обеспечении высококачественного пользовательского опыта, производительности, безопасности и стабильности.

Основные направления тестирования:

1. Функционирование: Тестирование каждого модуля игры на корректность работы всех функций:
   * Авторизация и регистрация пользователей.
   * Работа с базой данных (пользователи, собаки, вопросы, игровые сессии).
   * Административная панель: добавление/редактирование пользователей, собак, вопросов.
   * Игровой процесс: правильность работы уровней, сбор косточек, отображение статистики.
2. Удобство и юзабельность:
   * Оценка интерфейса с точки зрения удобства взаимодействия.
   * Проверка работы всех кнопок и навигации (кнопки "Войти", "Зарегистрироваться", "Играть", "Назад").
   * Оценка простоты и понятности интерфейса на всех уровнях, включая административную панель.
3. Обеспечение безопасности данных:
   * Тестирование защиты паролей (хэширование паролей, безопасность входа).

Чтобы ознакомиться с планом тестирования изучите Таблицу 1, Приложение 2.

Таким образом, проектирование системы охватывает все ключевые элементы, начиная от определения основных объектов и их связей до проведения тестирования, что обеспечивает надежность и стабильность работы всей системы.

# 3. РЕАЛИЗАЦИЯ ПРОЕКТА СИСТЕМЫ

## 3.1. Описание среды разработки

В рамках реализации проекта игры «Собачья академия» была выбрана интегрированная среда разработки (IDE) PyCharm. Это популярная среда разработки, предназначенная для работы с языком программирования Python и поддерживающая широкий спектр инструментов для создания, отладки и тестирования программного обеспечения.

### 3.1.1. Выбор и описание инструментария

В дополнение к выбору среды разработки PyCharm, для реализации проекта игры «Собачья академия» использовались следующие инструменты:

1. Язык программирования: Python. Python был выбран из-за своей универсальности, читаемости кода и широкого набора библиотек, которые позволяют легко работать с графическими интерфейсами, базами данных и производить вычисления.
2. Библиотеки:

* tkinter — стандартный GUI-фреймворк для создания интерфейсов.
* PIL (Pillow) — работа с изображениями, включая обработку графики.
* sqlalchemy — управление базами данных и выполнение запросов.
* logging — логирование действий пользователей и ошибок.
* os — взаимодействие с файловой системой.
* csv — импорт и экспорт данных в формате CSV.
* datetime — управление временными метками и сессиями.
* matplotlib — визуализация данных, например, диаграммы результатов.
* random — генерация случайных чисел для игровых объектов.
* math — математические расчёты для игрового процесса.
* sys — управление параметрами командной строки и выходом из программы.

### 3.1.2. Обоснование выбора инструментария

Выбор PyCharm в качестве среды разработки обоснован несколькими важными факторами:

1. Поддержка Python: PyCharm является одной из лучших IDE для Python, предоставляя все необходимые инструменты для разработки на этом языке. Так как игра «Собачья академия» разрабатывается с использованием Python, выбор этой среды является естественным.
2. Удобство разработки и отладки: Возможности PyCharm для отладки и тестирования позволяют значительно упростить процесс разработки, выявлять ошибки и ускорять решение возникающих проблем.
3. Гибкость и расширяемость: PyCharm поддерживает множество плагинов и интеграций, что позволяет адаптировать среду под специфические нужды проекта.
4. Интеграция с Git: PyCharm имеет встроенную поддержку системы контроля версий Git, что облегчает работу в команде и позволяет эффективно отслеживать изменения в проекте.

Таким образом, PyCharm является оптимальной средой для разработки игры «Собачья академия», обеспечивая удобство и эффективность на всех этапах создания и тестирования программы.

## 3.2 Разработка программного модуля

Разработка программного модуля DogAcademyGame основывалась на архитектуре MVC (Model-View-Controller), что позволило разделить логику данных, пользовательский интерфейс и взаимодействие. Проект был реализован на языке Python с использованием следующих технологий:

* SQLAlchemy для работы с базой данных.
* PyQt для создания пользовательского интерфейса.
* SQLite для хранения данных.
* Логирование выполнено с использованием стандартного модуля logging.

Основная структура проекта представлена в директории DogAcademyGame/ (Рисунок 1, Приложение 1). В ней организованы модули для работы с базой данных и пользовательскими интерфейсами.

### 3.2.1 Реализация пользовательского интерфейса программы

1. Главное окно (Рисунок 1, Приложение 3)

* Главное окно предоставляет пользователю доступ к основным функциям программы: авторизация или регистрация.
* Элементы интерфейса:
  + Кнопки:
    - «Войти»
    - «Зарегистрироваться»
  + Заголовок: «Dog Academy Game»

2. Окно регистрации и авторизации (Рисунок 2.1 – 2.2, Приложение 3)

* Регистрация содержит текстовые блоки для ввода логина, пароля, никнейма, а также кнопки:
  + «Зарегистрироваться»
  + «Вернуться на главную»
* Авторизация включает текстовые блоки для ввода логина и пароля, кнопки:
  + «Показать пароль»
  + «Войти»
  + «Вернуться на главную»
* Дополнительно реализован триггер-условие для администратора, использующий данные из config.py.

3. Главное меню (Рисунок 3, Приложение 3)

Интерактивное меню с элементами:

Верхняя панель:

* Кнопки:
  + «Профиль»
  + «База знаний»
  + «Магазин» (неактивно)

Центральный экран:

* Изображения собак, расположенные по кругу
* Кнопка «Играть» в центре экрана

Нижняя панель:

* Кнопка «Выйти»

4. База знаний (Рисунок 4, Приложение 3)

Информация о породах собак:

* Заголовок с названием породы
* Изображение
* Разделы: характеристики, поведение, уход, комментарии администратора
* Навигация: кнопки «Предыдущая», «Следующая», «Назад».

5. Окно Профиль пользователя (Рисунок 5, Приложение 3)

* Заголовок с именем пользователя
* Статистика:
* Пройденные уровни
* Собранные косточки
* Кнопка «Назад» для возврата в главное меню.

6. Окна Игрового процесса (Рисунок 6.1 – 6.5, Приложение 3)

* Выбор породы собаки
  + Отображает список пород с изображениями. Пользователь выбирает собаку для прохождения уровня.
* Выбор уровня
  + Список доступных уровней с указанием их сложности. Заблокированные уровни затемнены.
* Игровой процесс
  + Карта-сетка с объектами (кости, собаки). Управление движением собаки осуществляется клавишами W, A, S, D. Прогресс отображается в панели.
* Пауза
  + Меню паузы с опциями продолжения игры или выхода.
* Экран победы
  + Сообщение о завершении уровня с результатами и кнопками для повторного прохождения, перехода на следующий уровень или выхода в меню.

7. Панель администратора (Рисунок 7.1 – 7.14, Приложение 3)

Доступно только администратору. Интерфейс поддерживает:

Навигация по модулям:

* Логирование
* Статистика
* Уведомления (в разработке)
* Безопасность (в разработке)
* Авторизация под данными пользователя
* Бургер-меню:
  + Работа с базой данных
    - Редактирование пользователей
    - Управление вопросами
    - Управление собаками
    - Просмотр таблиц
  + Управление игровым контентом
    - Создание и настройка уровней (в разработке)
    - Настройка параметров собаки (в разработке)
  + Управление интерфейсом пользователя
    - Добавление подсказок в интерфейс (в разработке)

### 3.2.2 Описание функциональных узлов модуля с кодом

1. Инициализация базы данных

Функциональный узел отвечает за создание сессий базы данных, необходимой для взаимодействия с таблицами системы.

* Модуль: db\_session.py
* Основные задачи:
  + Установка соединения с базой данных.
  + Обеспечение потока безопасной работы сессий.
  + Закрытие соединений после завершения операций.
* Связанные элементы интерфейса: все формы и операции, которые обращаются к базе данных (регистрация, авторизация, админ-панель).
* Пример кода: см. Рисунок 1, Приложение 4.

2. Регистрация пользователя

Функционал позволяет пользователю создать учетную запись.

* Модуль: методы create\_user() в db\_events.py, register\_user() в auth.py
* Основные задачи:
  + Обработка данных, введенных в форме регистрации.
  + Проверка корректности логина и пароля.
  + Запись данных в таблицы Auth (логин/пароль) и Users (никнейм, статистика).
* Связанные элементы интерфейса: окно регистрации.
* Пример кода: см. Рисунок 2.1-2.2, Приложение 4.

3. Авторизация

Процедура проверки учетных данных для предоставления доступа к системе.

* Модуль: метод login\_user() в auth.py, метод login\_user() в auth\_ui.py.
* Основные задачи:
  + Ввод и проверка логина и пароля.
  + Обработка статуса пользователя (администратор или обычный пользователь).
  + Перенаправление пользователя к соответствующему интерфейсу (главное меню или админ-панель).
* Связанные элементы интерфейса: окно авторизации.
* Пример кода: см. Рисунок 3.1 – 3.2, Приложение 4.

4. Генерация игровой карты

Процесс создания уникальных уровней для игрового процесса.

* Модуль: методы draw\_grid(), generate\_bones(), update\_map() в game\_ui.py.
* Основные задачи:
  + Генерация случайного расположения объектов (кости, собаки).
  + Построение карты на основе заданного шаблона уровня.
  + Обеспечение баланса между сложностью и игровыми возможностями.
* Связанные элементы интерфейса: окно игрового процесса.
* Пример кода: см. Рисунок 4.1 – 4.3, Приложение 4.

5. Обновление базы данных

Функция добавления новой информации о породах собак, о вопросах и пользователях.

* Модуль: методы manage\_users(), manage\_questions(), manage\_dogs() в admin\_ui.py
* Основные задачи:
  + Ввод данных
  + Обновление таблицы Dogs, Auth, Users, Questions.
  + Связанные элементы интерфейса: окно «База знаний» и админ-панель.
* Пример кода: см. Таблица 1.1 – 1.3, Приложение 4.

Связь функциональных узлов с системой

Узлы тесно связаны через взаимодействие с базой данных.

Диаграмма прецедентов (Рис. 3, Приложение 1) показывает, как пользователи и администратор работают с каждым узлом через интерфейсы.

### 3.2.3. Результат работы и тестирования

1. Тестирование функциональности

Тест-кейс 1: Регистрация пользователя

Цель: проверить корректность регистрации нового пользователя.

Ожидаемый результат:

* + Пользователь успешно зарегистрирован.
  + Данные записаны в таблицы Auth и Users.
  + Событие регистрации зафиксировано в логах.

Тест-кейс 2: Авторизация пользователя

Цель: проверить вход зарегистрированного пользователя.

Ожидаемый результат:

* + Пользователь переходит в главное меню.
  + В случае некорректных данных выводится сообщение об ошибке.

Тест-кейс 3: Начало игровой сессии

Цель: убедиться в корректности запуска игрового процесса.

Ожидаемый результат:

* + Загружается игровая карта.
  + Игрок может управлять персонажем.

Тест-кейс 4: Завершение уровня

Цель: проверить функциональность завершения уровня.

Ожидаемый результат:

* + Показывается экран победы.
  + Прогресс сохраняется в базе данных.

Тест-кейс 5: Сохранение прогресса

Цель: убедиться, что прогресс игры сохраняется корректно.

Ожидаемый результат:

* + Прогресс сохранён.
  + Статистика обновлена.

Тест-кейс 6: Проверка базы знаний

Цель: убедиться, что база знаний отображается корректно.

Ожидаемый результат:

* + Корректное отображение информации и навигация.

Тест-кейс 7: Работа с профилем пользователя

Цель: проверить корректность отображения профиля.

Ожидаемый результат:

* + Профиль отображает актуальные данные (пройденные уровни, косточки, никнейм).

2. Тестирование взаимодействия администратора

Тест-кейс 8: Редактирование данных: добавление записи

Цель: проверить возможность редактирования данных пользователей, вопросов, пород собак.

Ожидаемый результат:

* + Изменения сохраняются в базе данных.

Тест-кейс 9: Удаление данных

Цель: проверить функциональность удаления записей из базы.

Ожидаемый результат:

* + Запись удалена из базы данных.

Тест-кейс 10: Обработка некорректных данных при регистрации

Цель: проверить, как система реагирует на ввод некорректных данных.

Ожидаемый результат:

* + Появляется сообщение об ошибке.
  + Регистрация не завершена.

Тест-кейс 11: Обработка ошибок при авторизации

Цель: проверить, как система обрабатывает неверные данные при входе.

Ожидаемый результат:

* + Отображается сообщение о неверных данных.

Тест-кейс 12: Мониторинг статистики пользователей

Цель: проверить корректность отображения статистики.

Ожидаемый результат:

* + Статистика отображается корректно.
  + Данные актуальны.

# ЗАКЛЮЧЕНИЕ

В рамках курсового проекта была выполнена разработка программного модуля для информационной системы «Игра «Собачья академия». Проект доказал свою актуальность в контексте современных требований к образовательным компьютерным играм. Проведён анализ предметной области, выявлены существующие решения и их недостатки, что позволило обосновать разработку оригинального продукта.

В ходе выполнения проекта были достигнуты следующие результаты:

1. Разработан программный модуль, включающий функционал авторизации пользователей, управления базами данных, игрового процесса и административного контроля.
2. Реализованы интерфейсы для пользователей и администраторов с применением современных библиотек Python.
3. Проведено тестирование ключевых компонентов модуля, что подтвердило его работоспособность и соответствие требованиям технического задания.
4. Создана архитектура системы с возможностью её дальнейшего расширения, что обеспечивает устойчивость проекта к изменениям.

Результаты разработки позволяют утверждать, что система «Собачья академия» способна обеспечить не только образовательный, но и развлекательный эффект. Проект имеет потенциал для внедрения на рынок образовательных игр и может быть дополнительно улучшен с учётом обратной связи пользователей.

Таким образом, курсовая работа достигла своей цели, продемонстрировав применение полученных знаний в области проектирования и программирования информационных систем.

Ссылка на репозиторий Github: <https://github.com/Anymorexxx/DogAcademyGame>.

# СПИСОК ЛИТЕРАТУРЫ

1. Федеральный закон от 27 июля 2006 года №152-ФЗ «О персональных данных» (ред. 2019).
2. Федеральный закон от 29 декабря 2010 года №436-ФЗ «О защите детей от информации, причиняющей вред их здоровью и развитию» (ред. 2020).
3. ГОСТ Р 15249-2012. Системы управления информационной безопасностью. Требования. – М.: Стандартинформ, 2019.
4. Поляков И.А. Программирование на Python: от основ до профессионального уровня. – СПб.: Питер, 2020.
5. Иванов А.В. Геймдизайн. Практическое руководство. – М.: Альпина Паблишер, 2021.
6. Краснов М.А. Современные технологии разработки ПО. – СПб.: БХВ-Петербург, 2021.
7. Смирнов А.Н. Методология Agile в разработке ПО. – М.: Экономика, 2021.
8. Романов Е.В. Основы программирования: учебник. – М.: Академия, 2022.
9. Петров В.В., Сидоров И.И. Базы данных: проектирование и оптимизация. – СПб.: Питер, 2022.
10. Суриков Д.Н. Python для начинающих: учебник. – М.: Инфра-М, 2022.
11. Смирнова Т.П. Основы графического дизайна в ИТ. – СПб.: БХВ-Петербург, 2023.
12. Алексеев К.С. Технологии программирования в разработке игр. – М.: Альпина Диджитал, 2023.
13. Карпов О.В. Защита информации в компьютерных системах. – М.: Юрайт, 2023.
14. Захаров И.И. Интерактивные технологии в образовании. – СПб.: Лань, 2023.
15. Морозов Н.Н. Игровое проектирование и разработка: учебное пособие. – М.: Физматлит, 2023.

# Приложение 1.

  
Рисунок 1 – Структура программного модуля

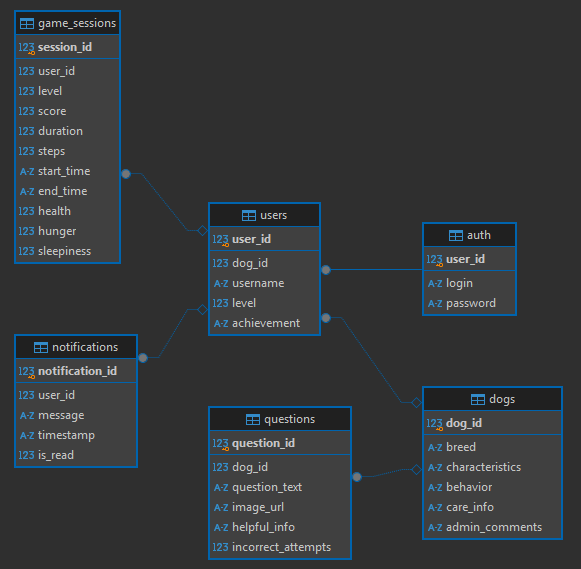
  
Рисунок 2 – ER-диаграмма базы данных

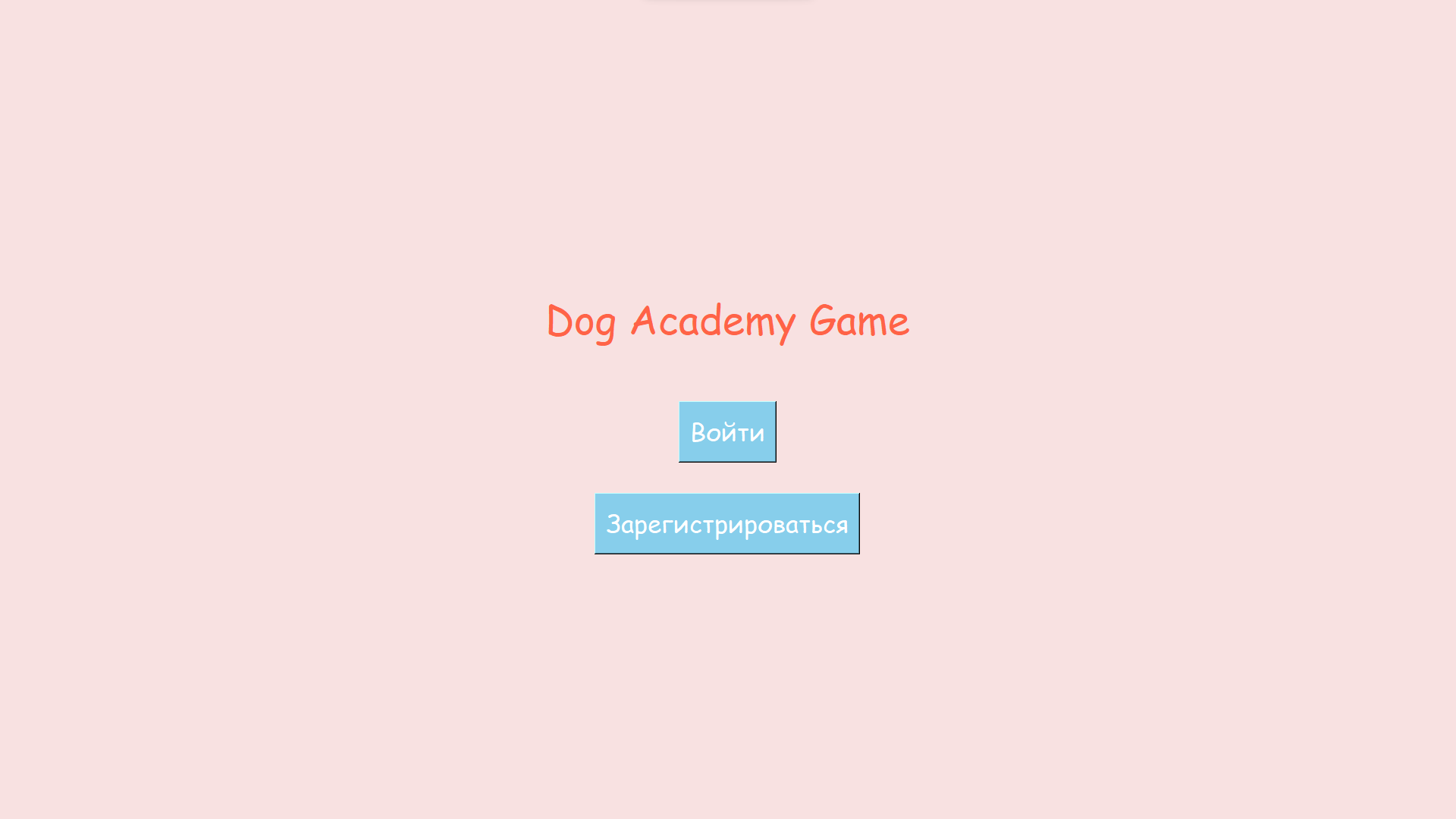
  
Рисунок 3 – Диаграмма прецендентов

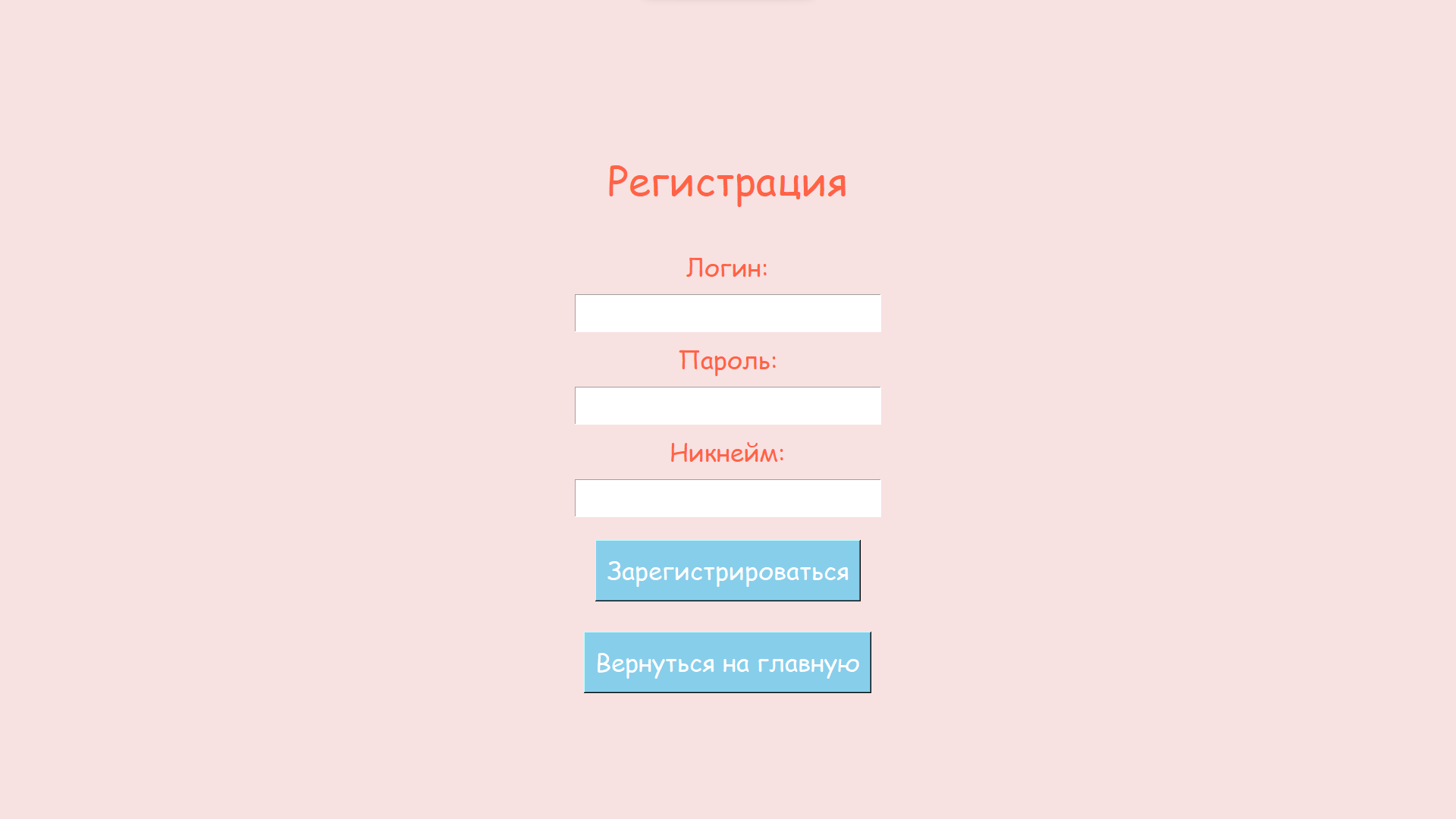
# Приложение 2.

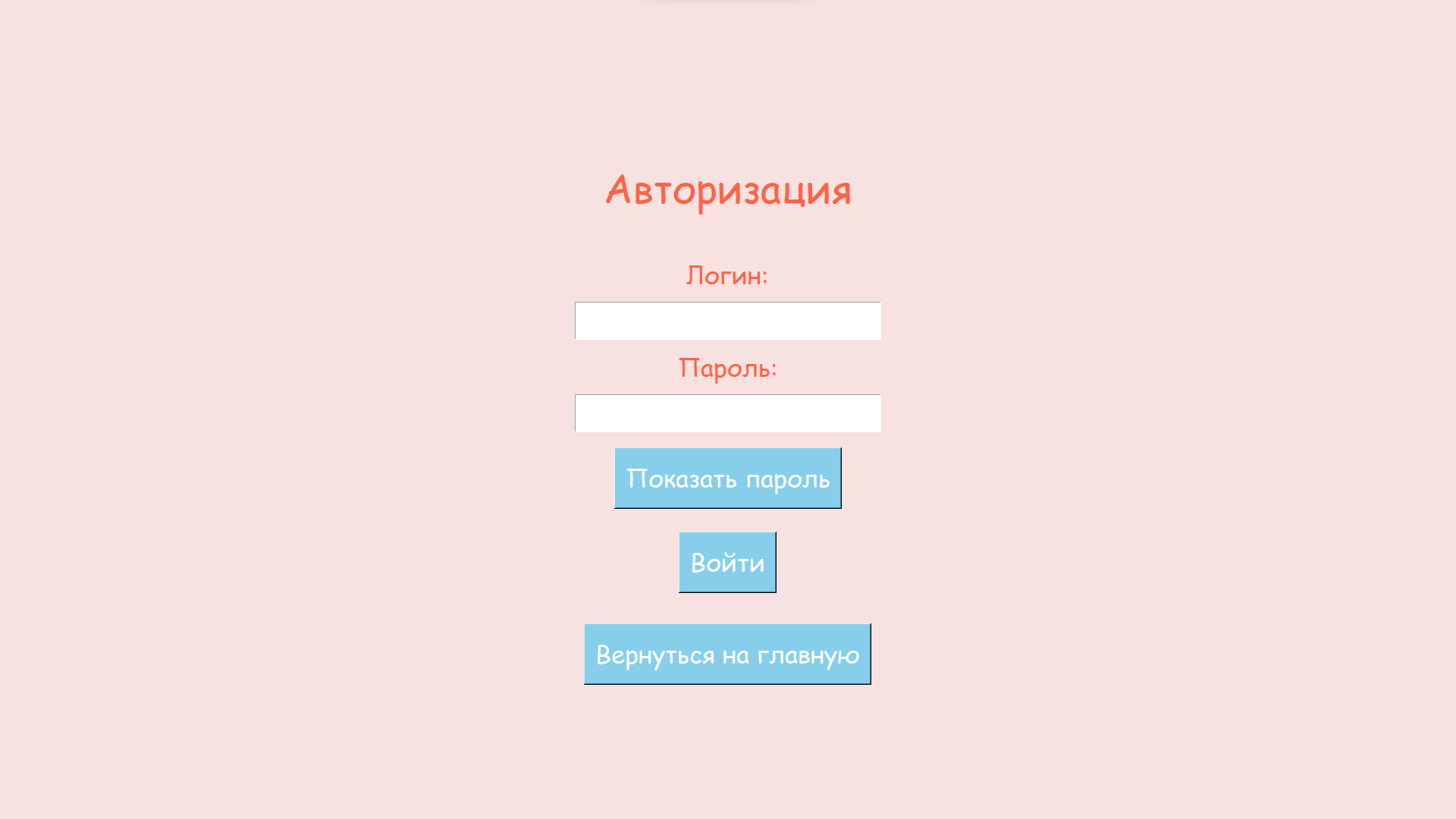
Таблица 1 – План тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование функциональности | Наименование поля | Тестовый набор | Ожидаемый результат |
| 1 | Регистрация пользователя | Логин | Anymorexxx | Регистрация успешна. |
| Пароль | meow123 |
| Никнейм | test |
| Фактический результат: Тест-кейс 1 – Регистрация пользователя. | | | | |
| 2 | Авторизация пользователя | Логин | Anymorexxx | Пользователь переходит в главное меню. |
| Пароль | meow123 |
| Фактический результат: Тест-кейс 2 – Авторизация пользователя. | | | | |
| 3 | Начало игровой сессии | Выберите собаку | Chihuahua | Загружается игровая карта. Игрок может управлять персонажем. |
| Выберите уровень | 1 |
| Фактический результат: Тест-кейс 3 – Начало игровой сессии | | | | |
| 4 | Завершение уровня | - | - | Показывается экран победы. Прогресс сохраняется в базе данных. |
| Фактический результат: Тест-кейс 4 – Завершение уровня      - на скриншоте показана запись о том, что игровой процесс сохранился в базу данных в таблицу game\_sessions | | | | |
| 5 | Сохранение прогресса | - | - | Прогресс сохранён. Статистика обновлена. |
| Фактический результат: Тест-кейс 5 – Сохранение прогресса | | | | |
| 6 | Проверка базы знаний | - | - | Корректное отображение информации и навигация. |
| Фактический результат: Тест-кейс 6 – Проверка базы знаний | | | | |
| 7 | Работа с профилем пользователя | - | - | Профиль отображает актуальные данные (пройденные уровни, косточки, никнейм). |
| Фактический результат: Тест-кейс 7 – Работа с профилем пользователя | | | | |
| 8 | Редактирование данных | | | |
| Добавить пользователя | Логин | ahaha | Изменения сохраняются в базе данных. |
| Пароль | m1m1m1 |
| Никнейм | test1 |
| Добавить породу | Порода | test |
| Характеристики | test |
| Поведение | test |
| Уход | test |
| Комментарий администратора | test |
| Добавить вопрос | ID собаки | 1 |
| Вопрос | Как часто нужно выгуливать Чихуахуа? |
| Изображение URL | None |
| Полезная информация | test |
| Фактический результат: Тест-кейс 8 – Редактирование данных  Добавить пользователя:      Добавить породу:    Добавление вопроса: | | | | |
| 9 | Удаление данных | - | - | Запись удалена из базы данных |
| Фактический результат: Тест-кейс 9 – Удаление данных | | | | |
| 10 | Обработка некорректных данных при регистрации | Логин | admin | Появляется сообщение об ошибке |
| Пароль | 111 |
| Никнейм | admin |
| Фактический результат: Тест-кейс 11 – Обработка некорректных данных при регистрации | | | | |
| 11 | Обработка ошибок при авторизации | Логин | anymorexxx | Отображается сообщение о неверных данных |
| Пароль | meow123 |
| Фактический результат: Тест-кейс 11 – Обработка ошибок при авторизации  Введён логин – все буквы строчные | | | | |
| 12 | Мониторинг статистики пользователей | - | - | Статистика отображается корректно. Данные актуальны |
| Фактический результат: Тест-кейс 12 – Мониторинг статистики пользователей | | | | |

# Приложение 3.

  
Рисунок 1 – Главное окно

  
Рисунок 2.1 – Регистрация

  
Рисунок 2.2 – Авторизация

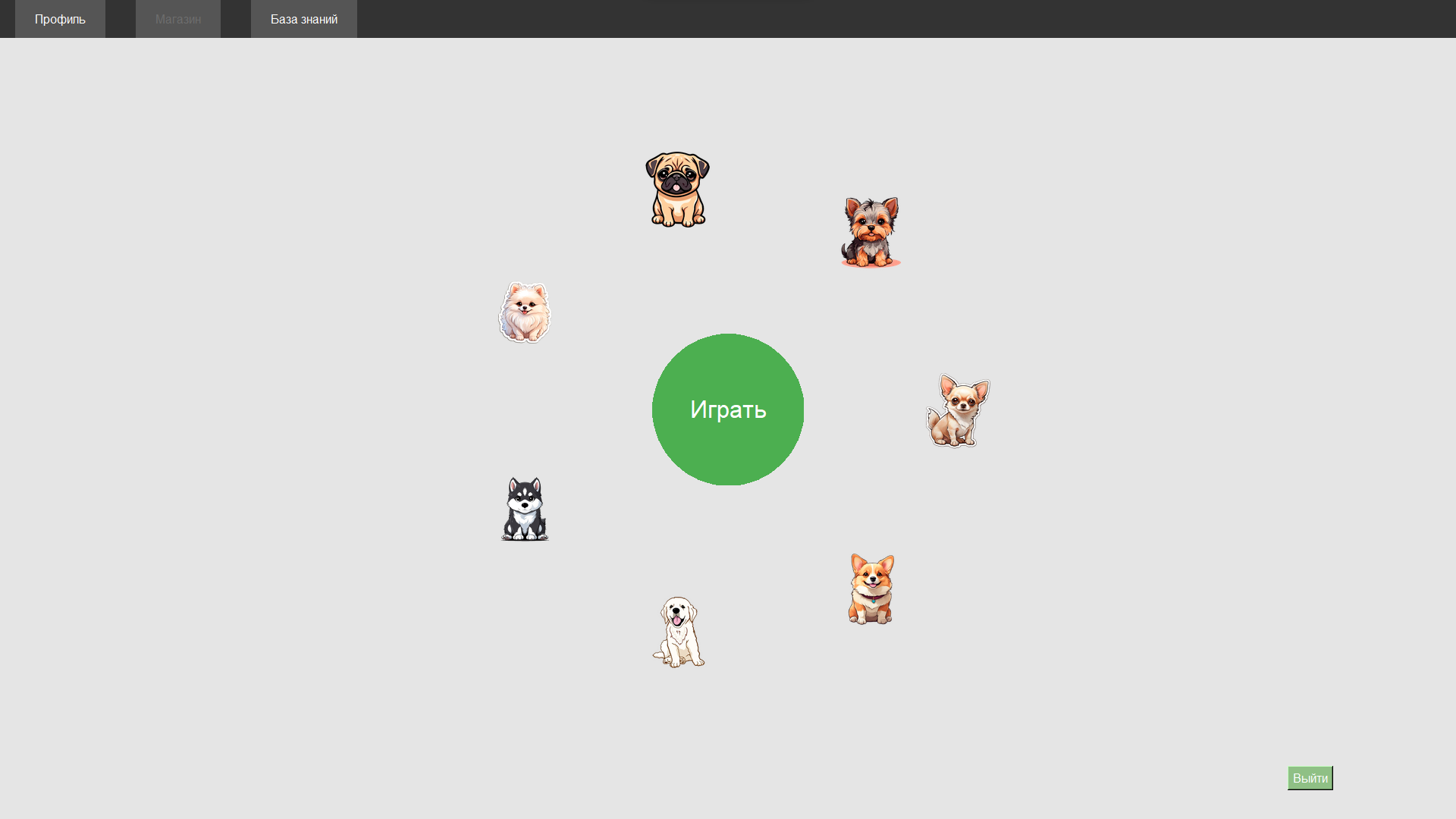


Рисунок 3 – Главное меню

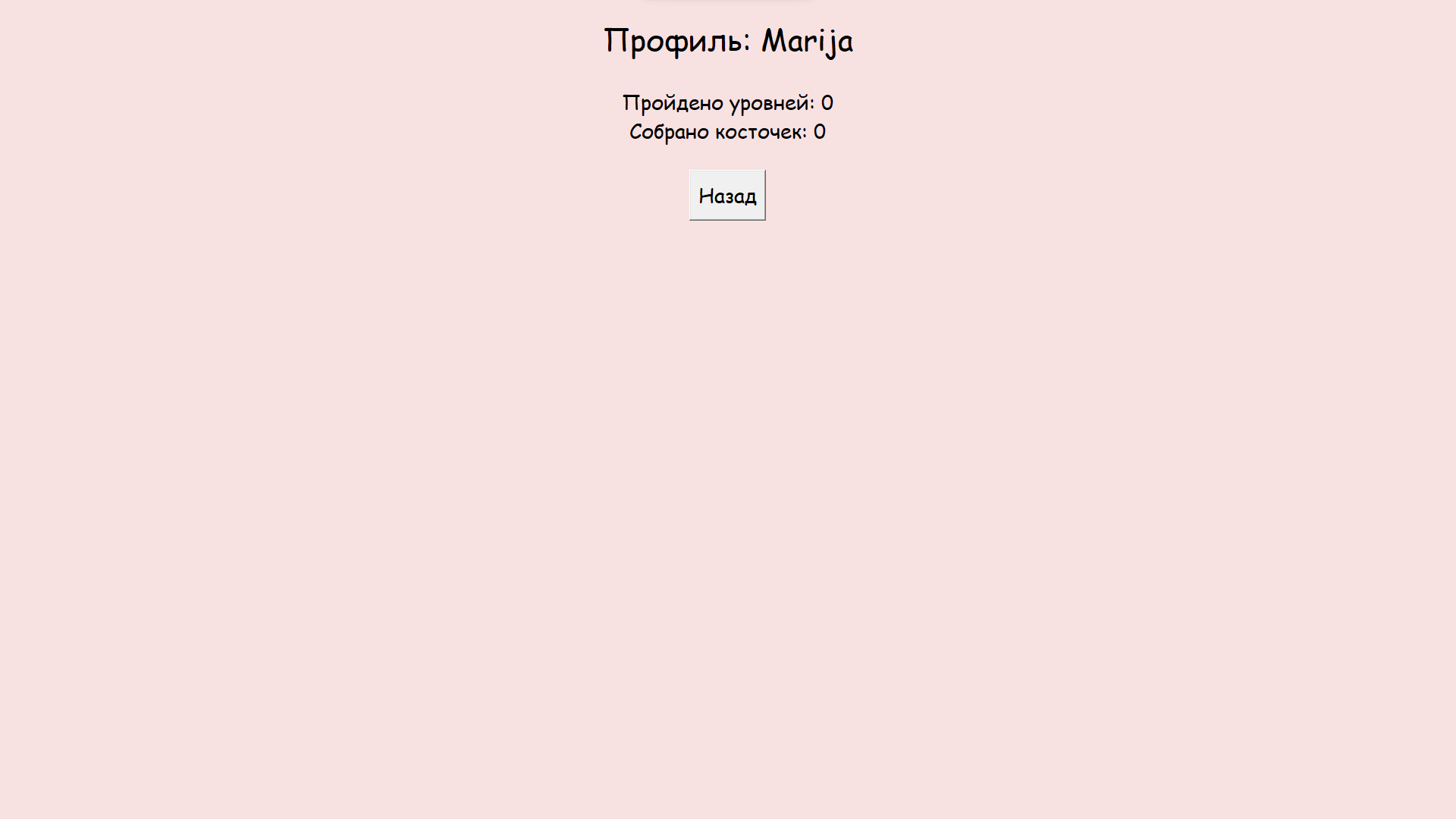


Рисунок 4– Профиль пользователя

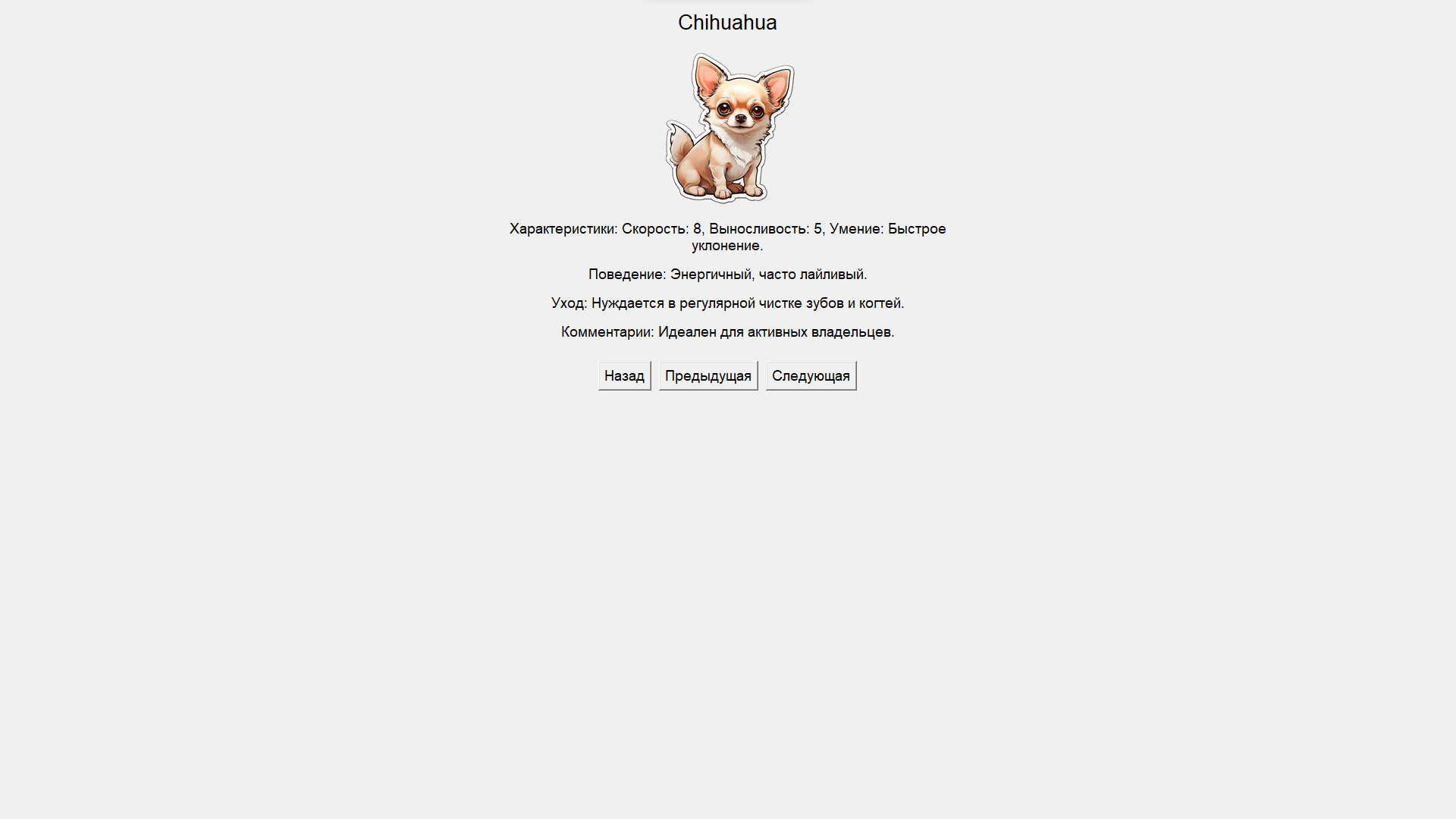


Рисунок 5 – База знаний



Рисунок 6.1 – Игровой процесс: Выбор породы собаки



Рисунок 6.2 – Игровой процесс: Выбор уровня

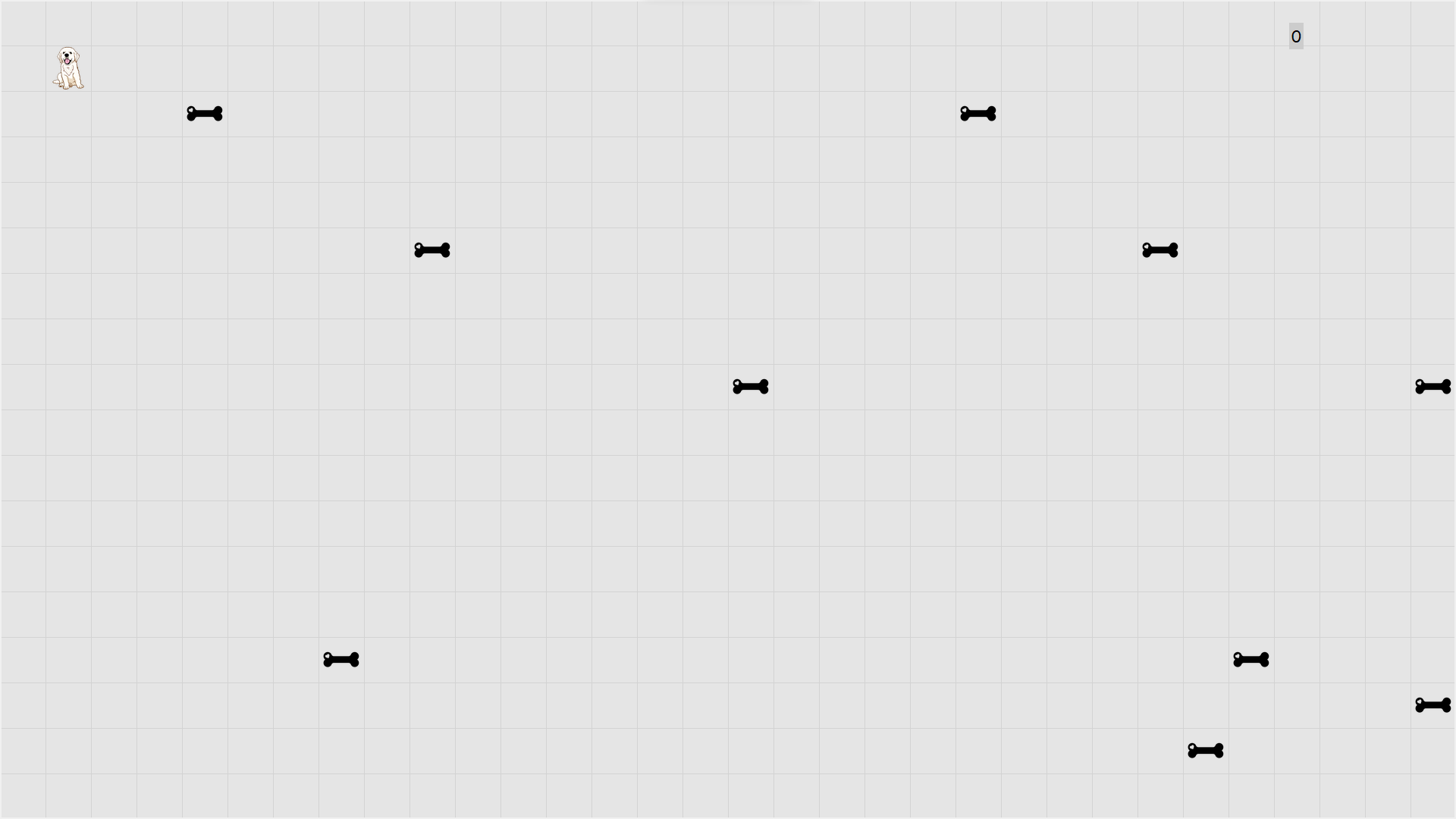


Рисунок 6.3 – Игровой процесс: Игровой процесс

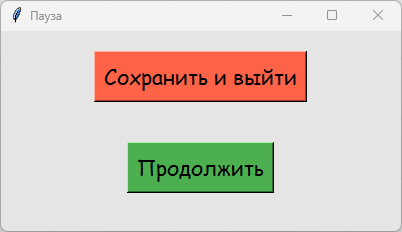


Рисунок 6.4 – Игровой процесс: Пауза



Рисунок 6.5 – Игровой процесс: Экран победы

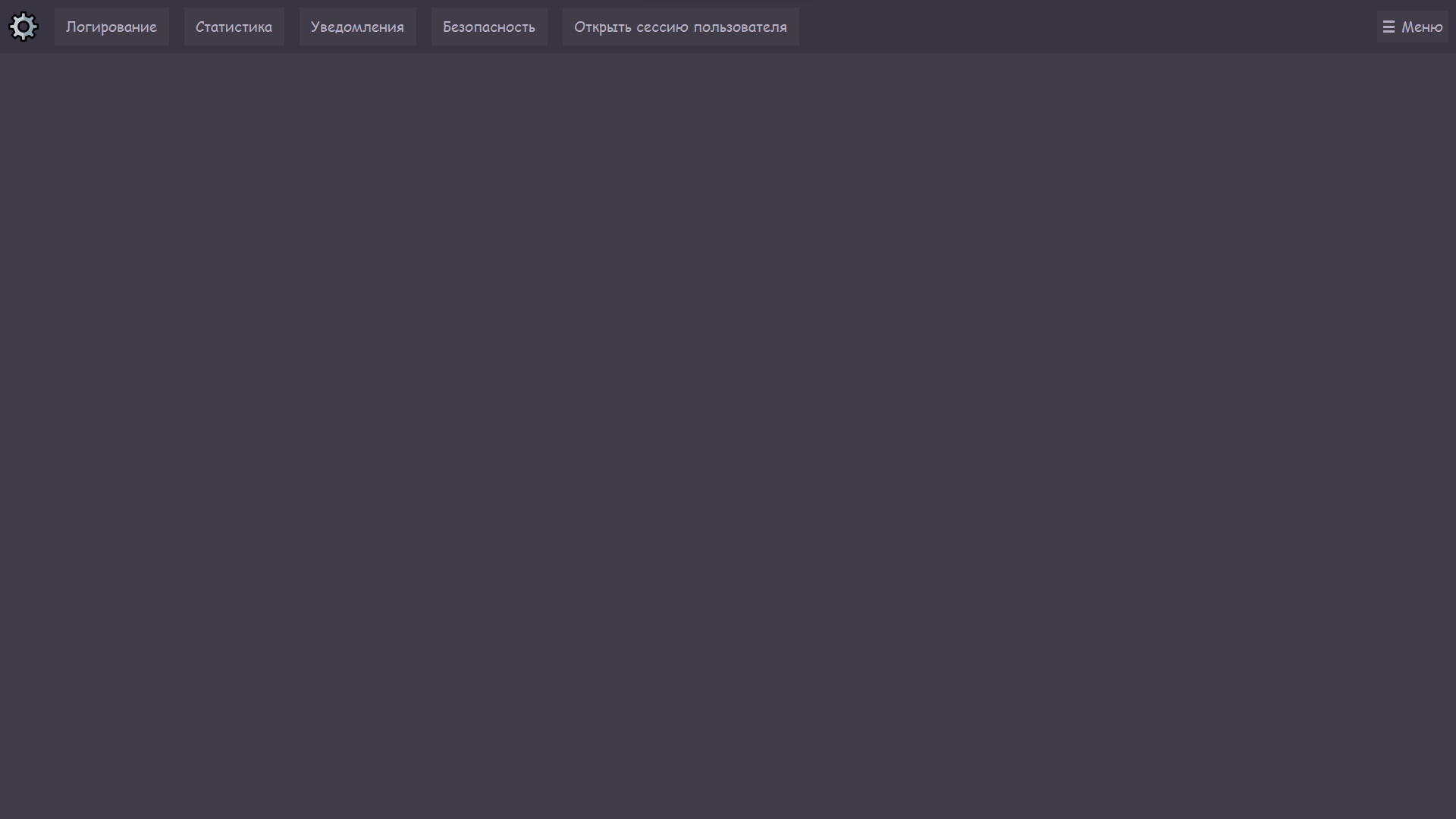


Рисунок 7.1 – Главное меню администратора

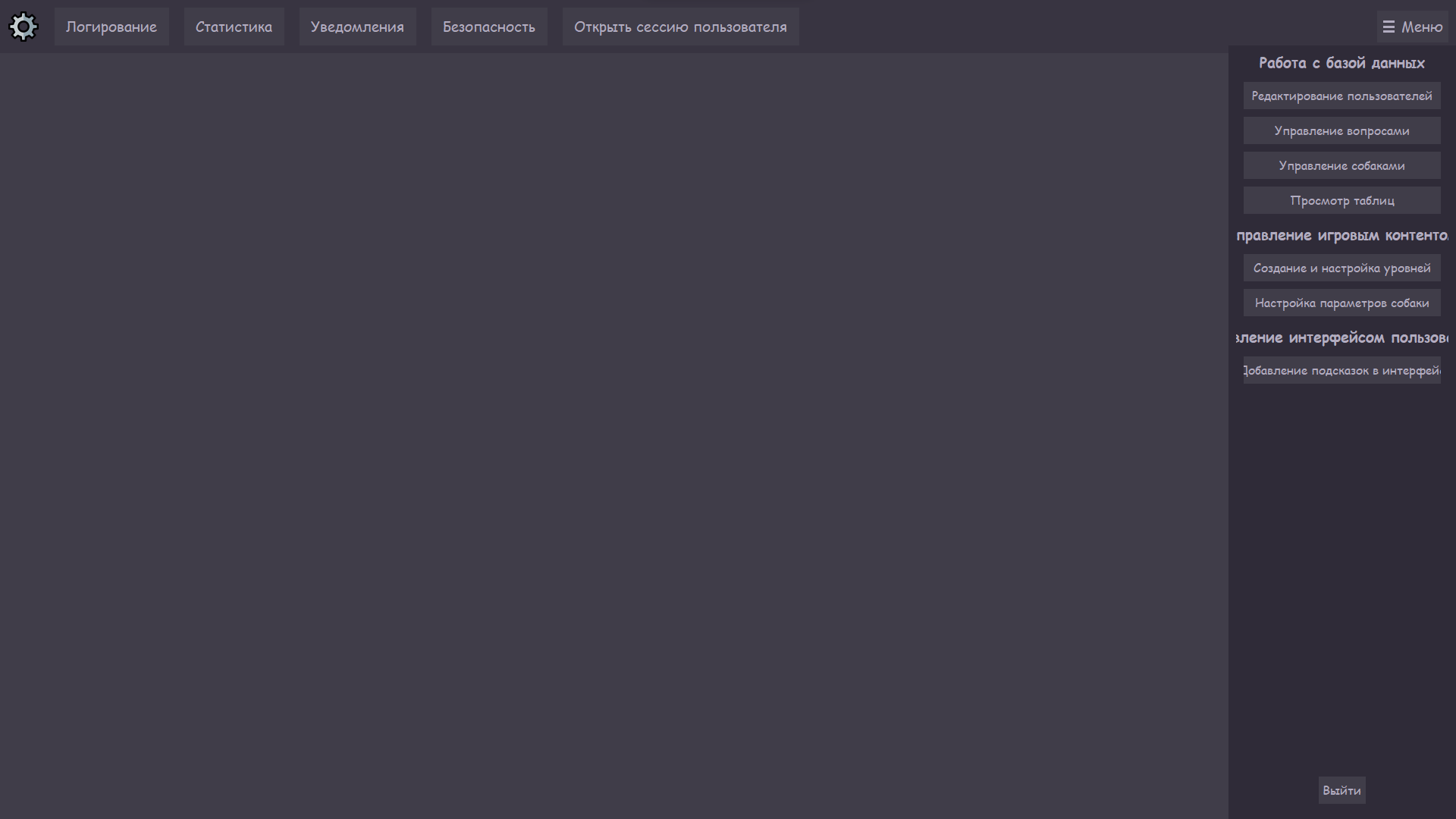


Рисунок 7.2 – Бургер-меню

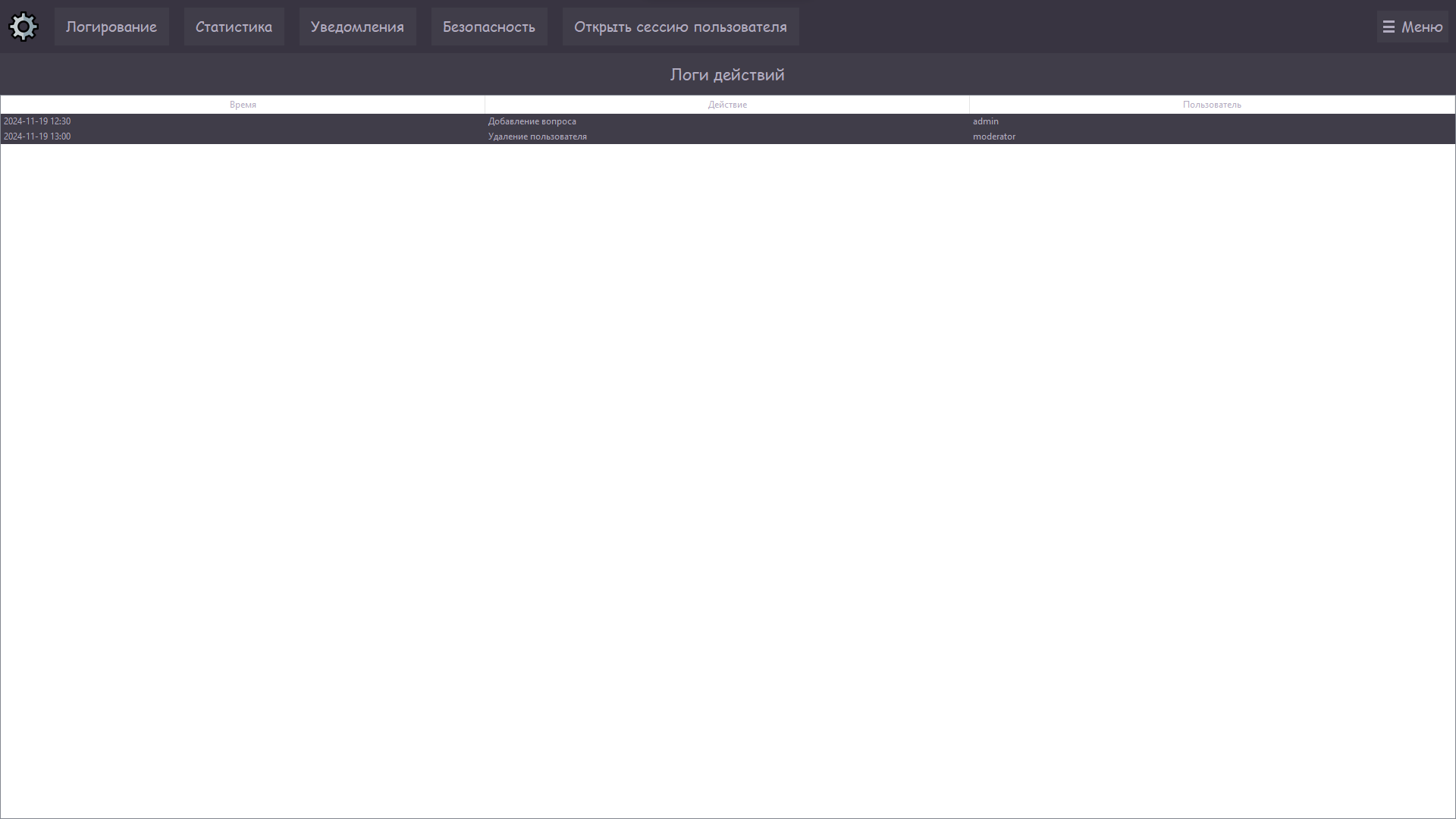


Рисунок 7.3 – Логирование

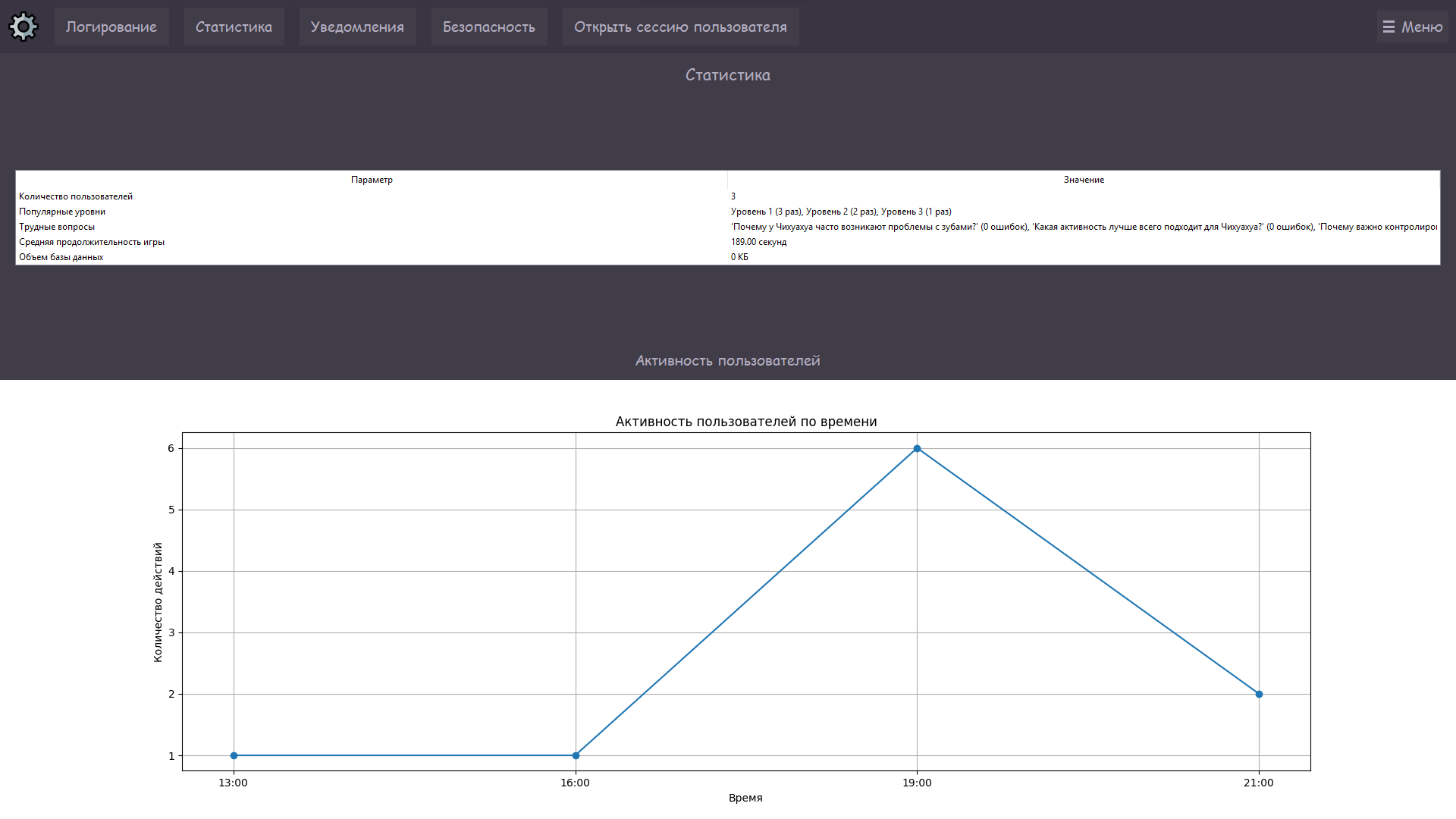


Рисунок 7.4 – Статистика

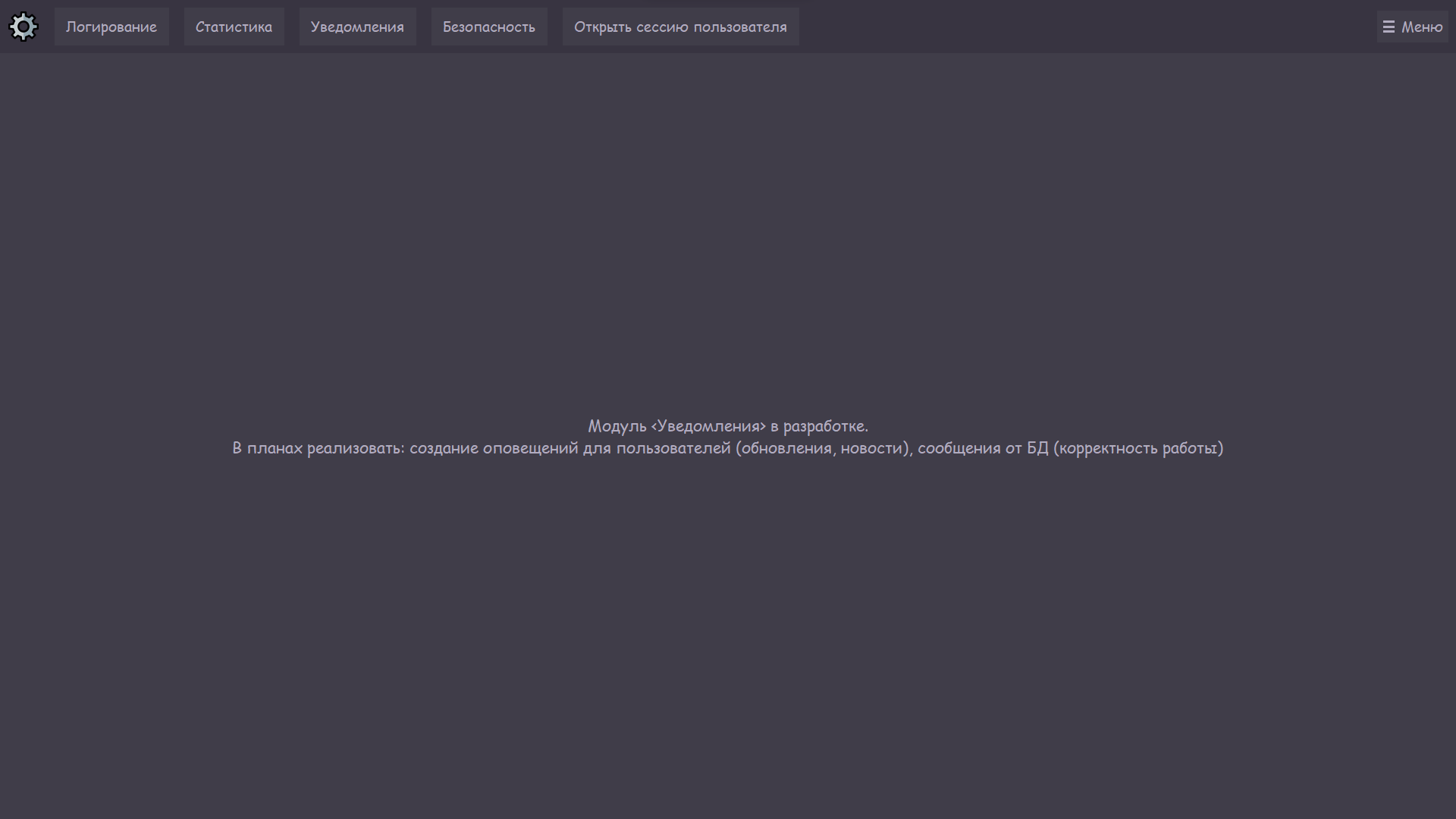


Рисунок 7.5 – Уведомления

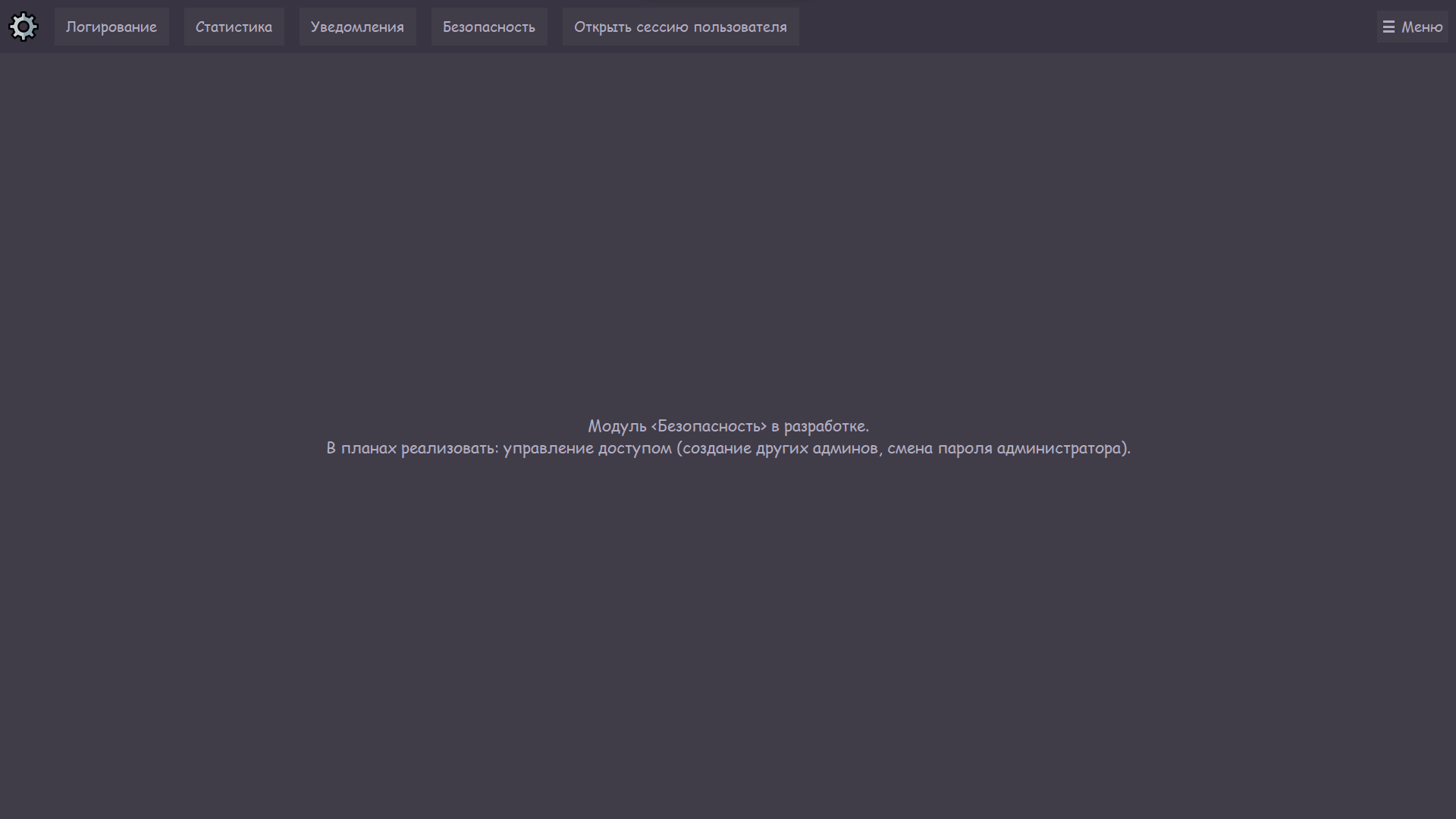


Рисунок 7.6 – Безопасность

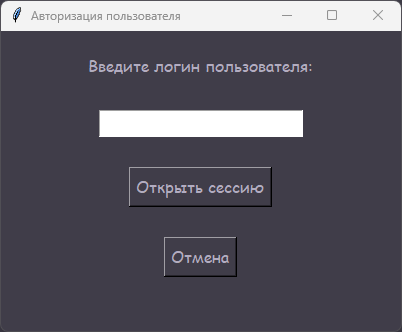


Рисунок 7.7 – Авторизация пользователя

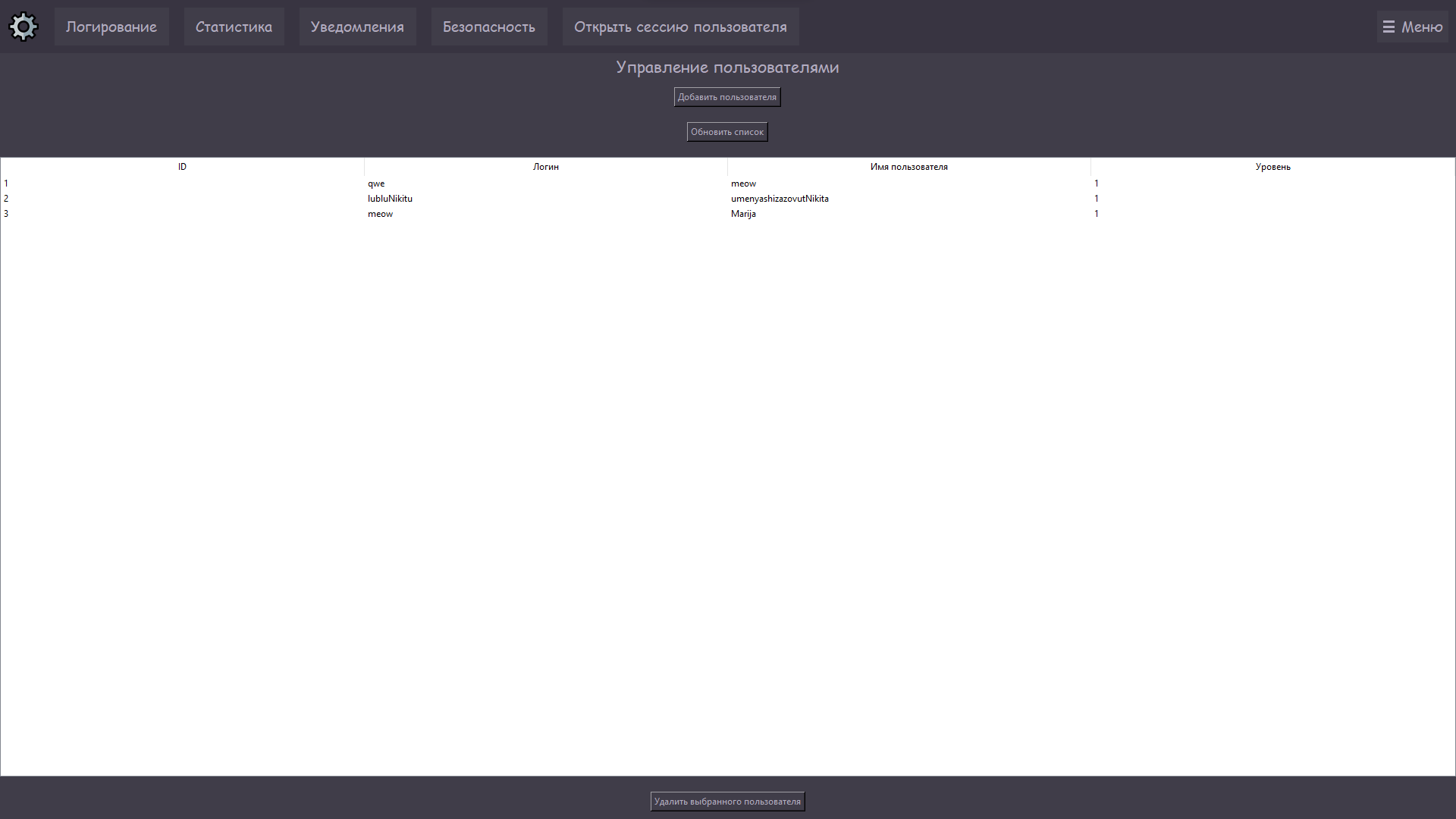


Рисунок 7.8 – Работа с базой данных: Редактирование пользователей

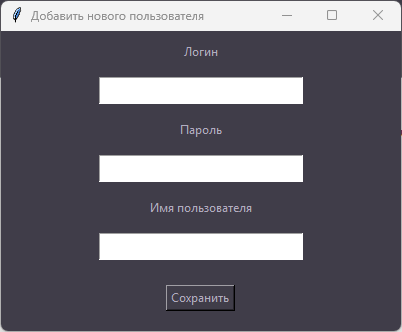


Рисунок 7.8.1 – Добавить нового пользователя



Рисунок 7.9 – Работа с базой данных: Управление вопросами

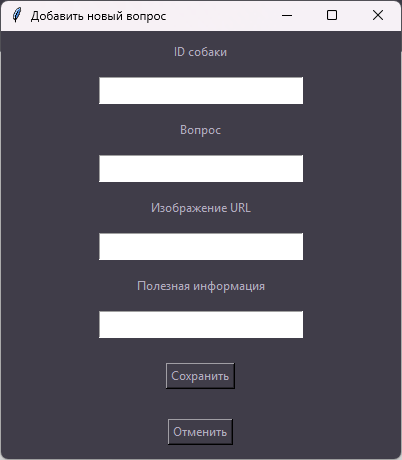


Рисунок 7.9.1 – Добавить новый вопрос

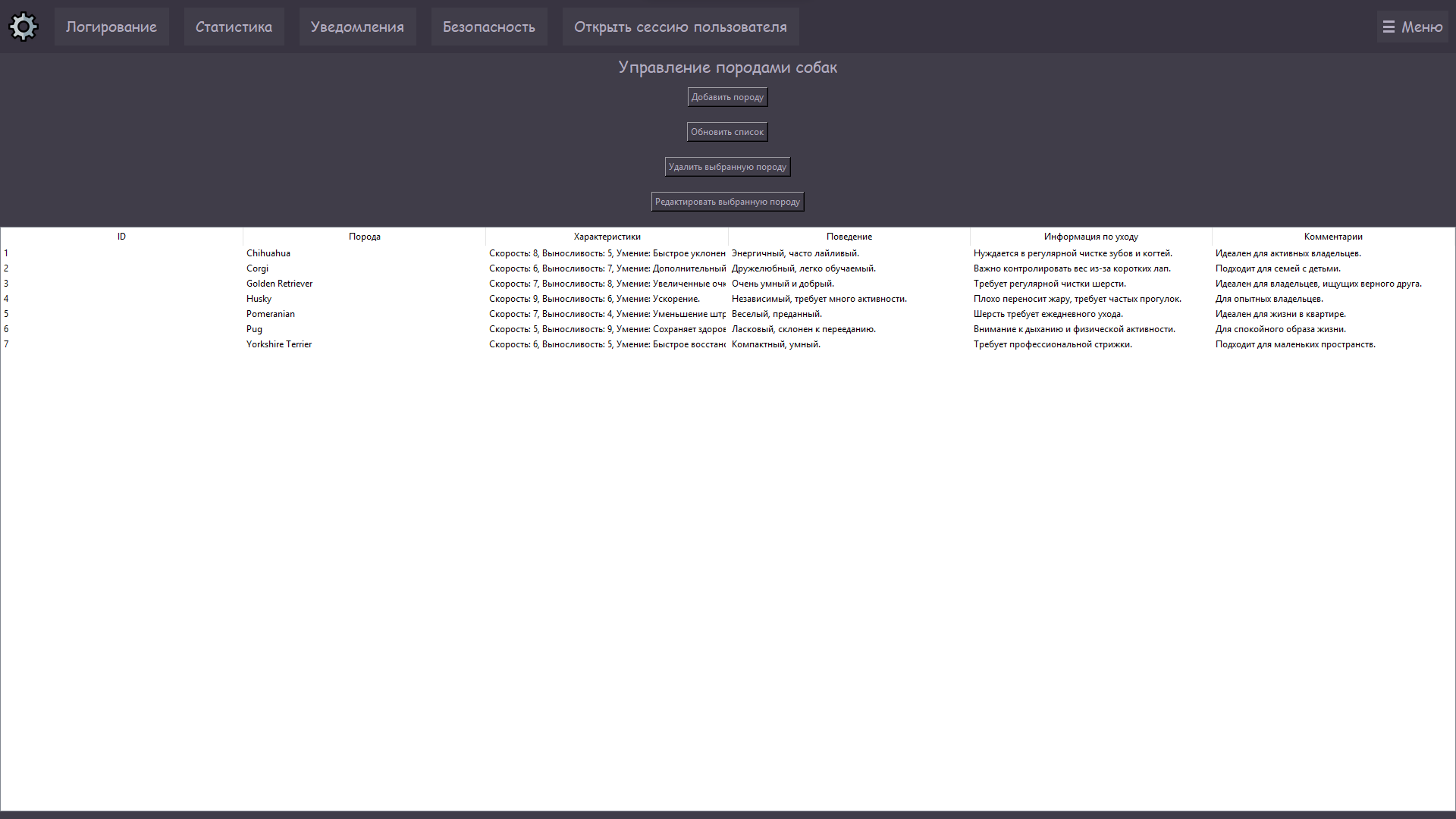


Рисунок 7.10 – Работа с базой данных: Управление собаками

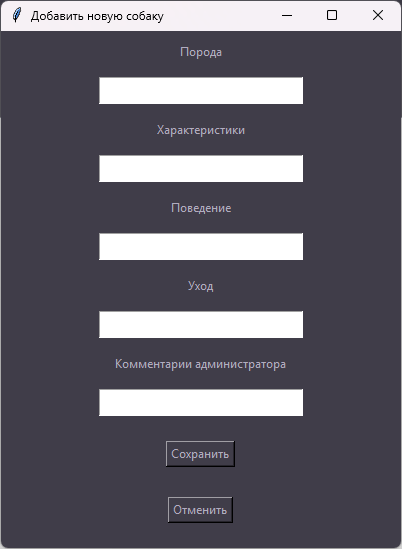
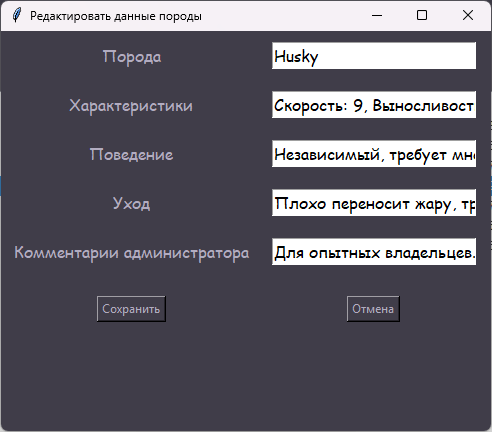


Рисунок 7.10.1 – Добавить новую собаку

  
Рисунок 7.10.2 – Редактировать данные породы

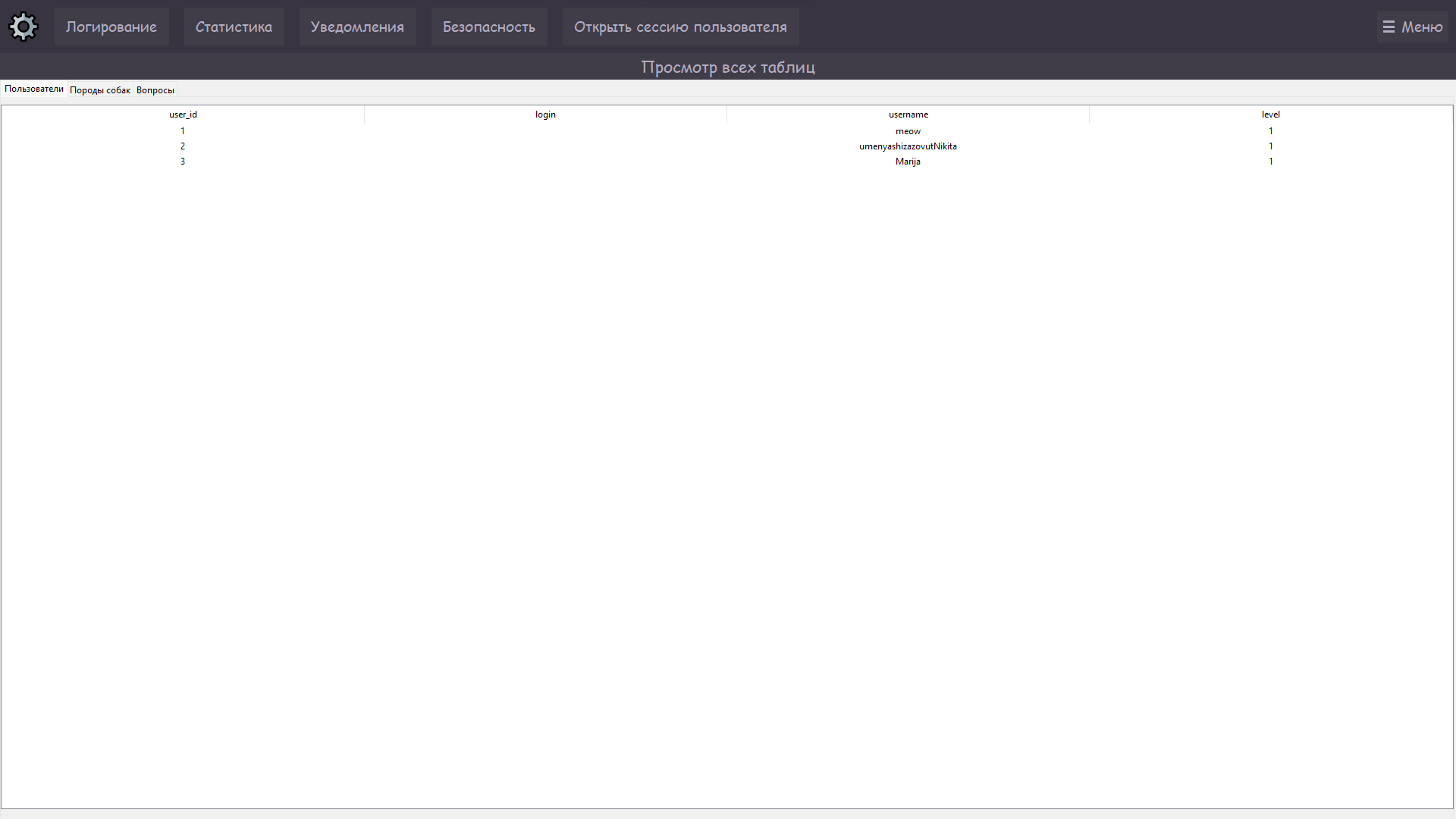


Рисунок 7.11 – Работа с базой данных: Просмотр таблиц

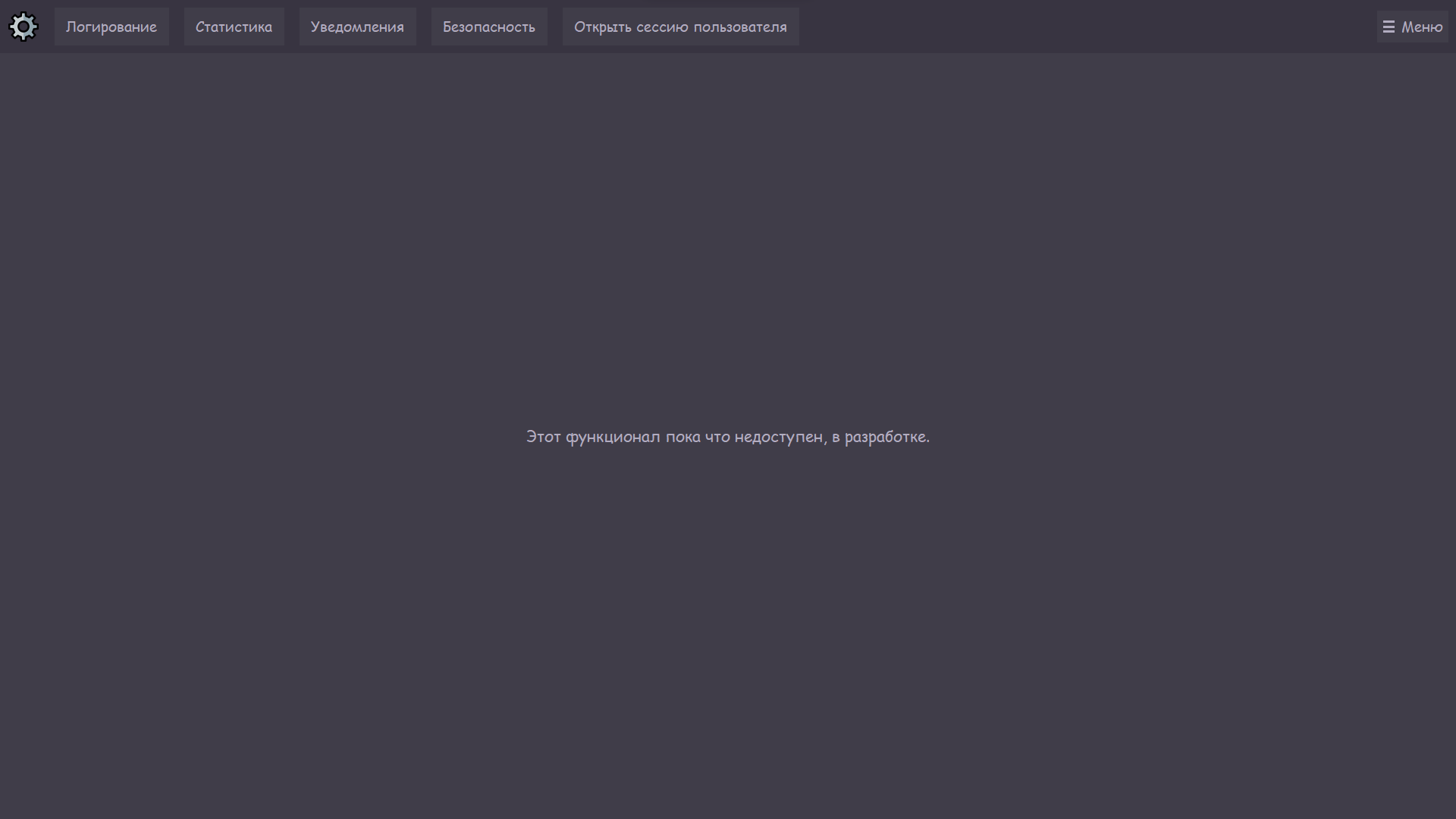


Рисунок 7.12 – Управление игровым контентом: Создание и настройка уровней

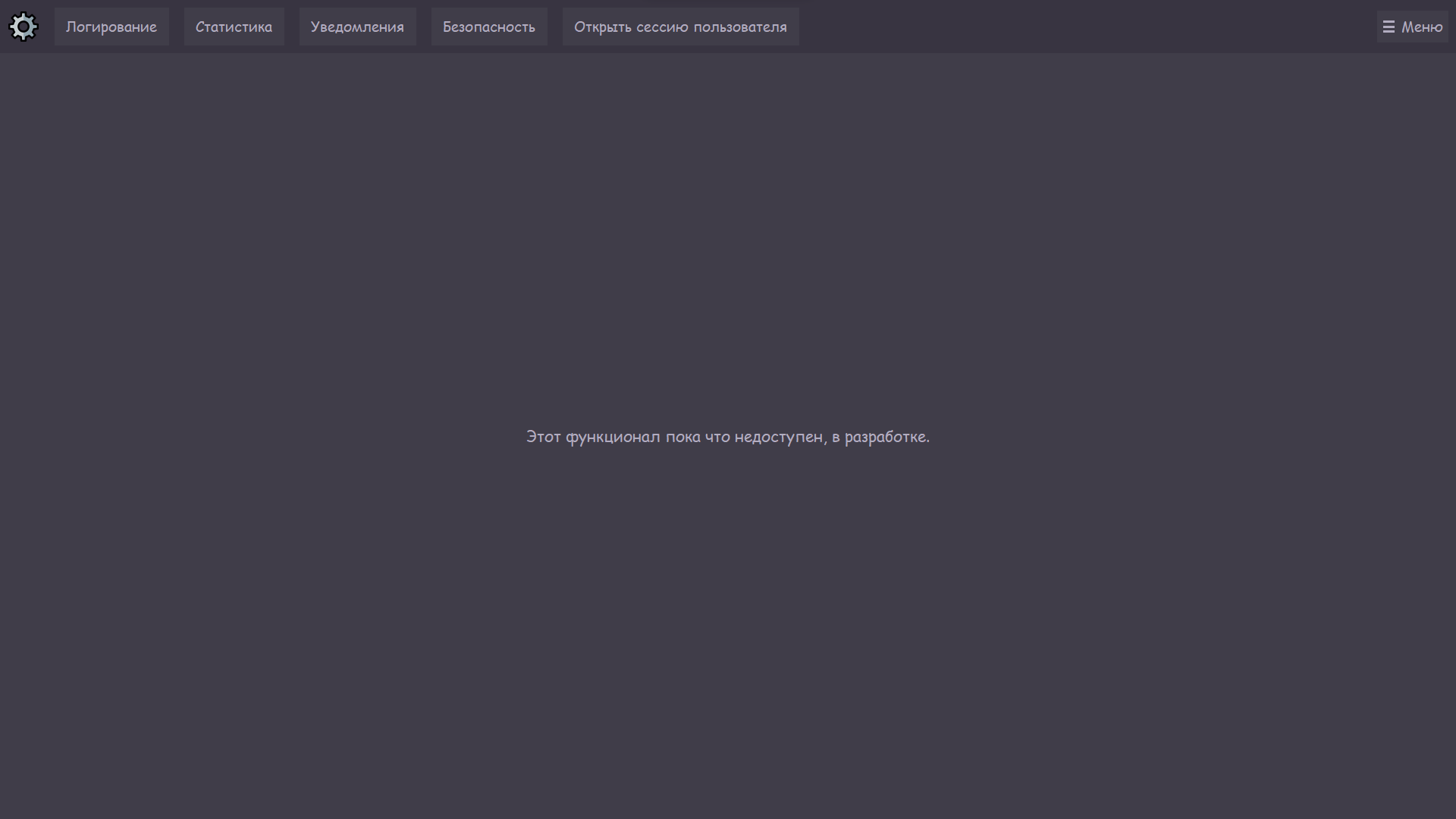


Рисунок 7.13 – Управление игровым контентом: Настройка параметров собаки

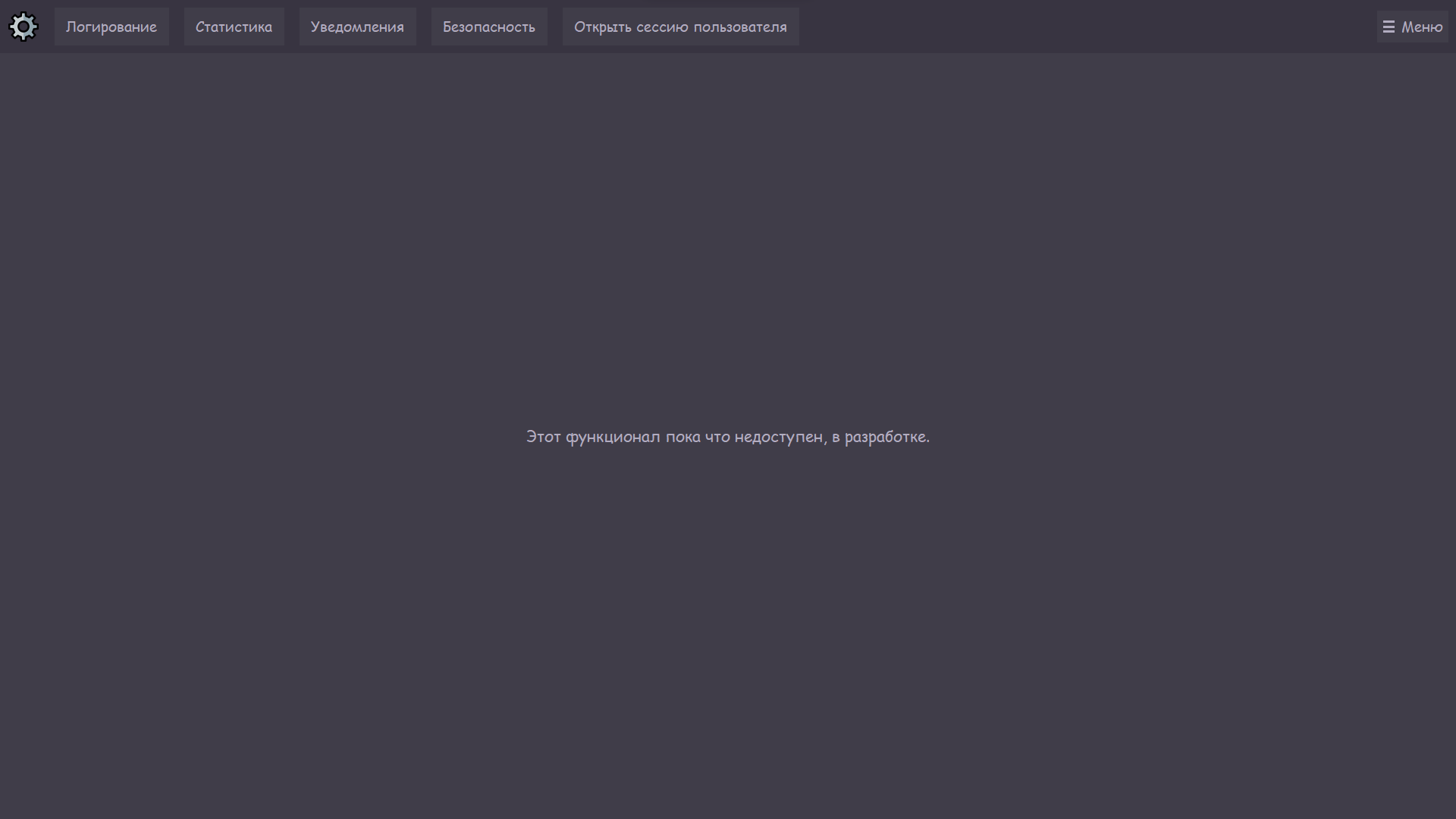
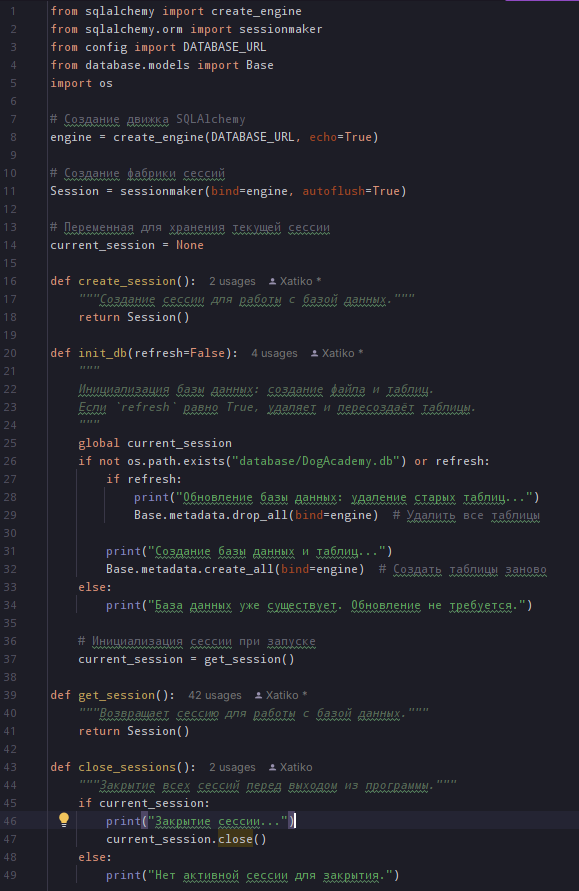


Рисунок 7.14 – Управление интерфейсом пользователя: Добавление подсказок в интерфейс

# Приложение 4.

  
Рисунок 1 – db\_session.py

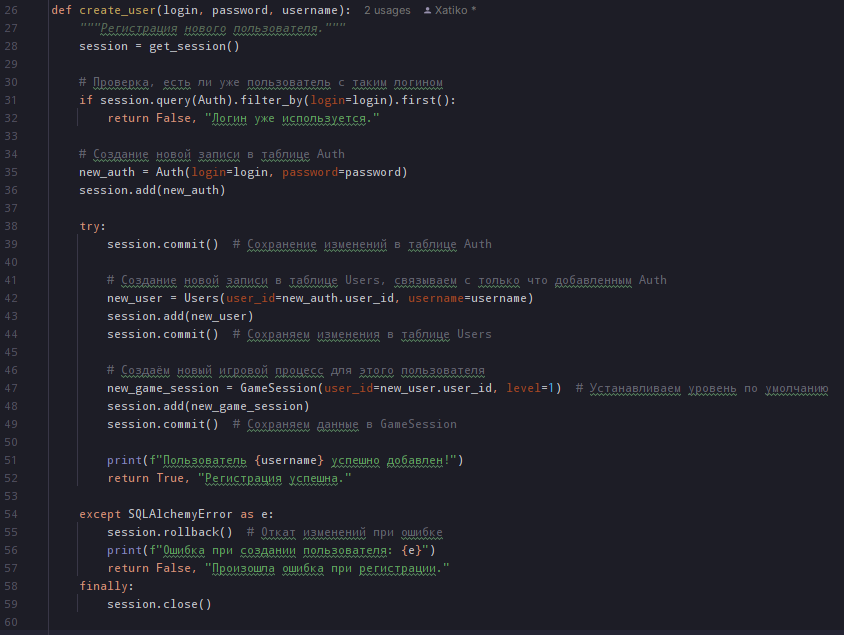
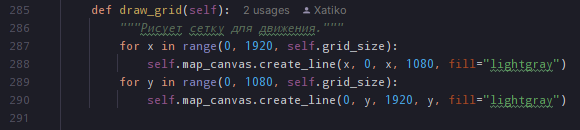
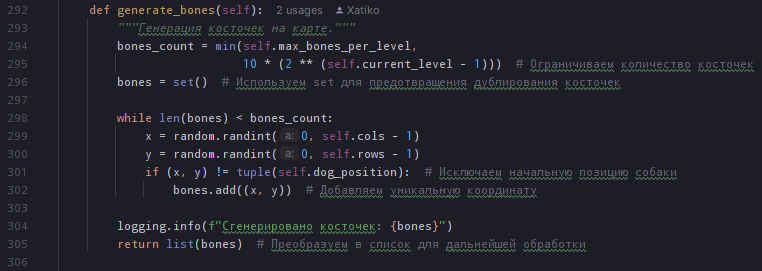
  
Рисунок 2.1 – метод create\_user() в db\_events.py

  
Рисунок 2.2 – метод register\_user() в auth.py

  
Рисунок 3.1 – метод login\_user() в auth.py

  
Рисунок 3.2 – метод login\_user() в auth\_ui.py.

  
Рисунок 4.1 – метод draw\_grid() в game\_ui.py

  
Рисунок 4.2 – метод generate\_bones() в game\_ui.py

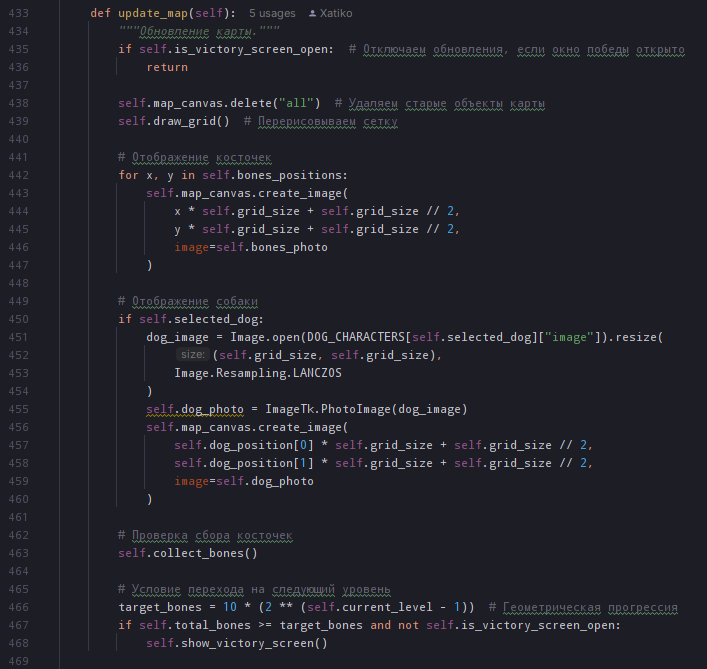
 Рисунок 4.3 – метод update\_map() в game\_ui.py

Таблица 1.1 – метод manage\_users() в admin\_ui.py

|  |
| --- |
| def manage\_users(self):  *"""Управление пользователями."""* clear\_frame(self.main\_frame)   tk.Label(self.main\_frame**,** text="Управление пользователями"**,** font=("Comic Sans MS"**,** 16)**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()   # Кнопка добавления нового пользователя  tk.Button(  self.main\_frame**,** text="Добавить пользователя"**,** command=self.open\_add\_user\_window**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Кнопка обновления списка  tk.Button(  self.main\_frame**,** text="Обновить список"**,** command=self.manage\_users**,** # Перезагрузка данных  bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   users = get\_all\_users() # Получение списка пользователей   if not users:  tk.Label(self.main\_frame**,** text="Нет пользователей в базе данных."**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()  return   # Отображение данных в таблице  table = ttk.Treeview(self.main\_frame**,** columns=("ID"**,** "Логин"**,** "Имя пользователя"**,** "Уровень")**,** show="headings")  table.heading("ID"**,** text="ID")  table.heading("Логин"**,** text="Логин")  table.heading("Имя пользователя"**,** text="Имя пользователя")  table.heading("Уровень"**,** text="Уровень")  table.pack(fill="both"**,** expand=True**,** pady=10)   # Очистка старых записей из таблицы  for row in table.get\_children():  table.delete(row)   # Добавление данных из базы  for user in users:  table.insert(""**,** "end"**,** values=(user.user\_id**,** user.auth.login**,** user.username**,** user.level))   def delete\_selected():  selected\_item = table.selection()  if not selected\_item:  messagebox.showwarning("Удаление"**,** "Выберите пользователя для удаления.")  return  user\_id = table.item(selected\_item**,** "values")[0]  success**,** message = delete\_user(user\_id)  if success:  messagebox.showinfo("Успех"**,** message)  self.manage\_users() # Обновление списка  else:  messagebox.showerror("Ошибка"**,** message)   # Кнопка удаления  tk.Button(  self.main\_frame**,** text="Удалить выбранного пользователя"**,** command=delete\_selected**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10) |

Таблица 1.2 – метод manage\_questions() в admin\_ui.py

|  |
| --- |
| def manage\_questions(self):  *"""Управление вопросами."""* clear\_frame(self.main\_frame)   tk.Label(self.main\_frame**,** text="Управление вопросами"**,** font=("Comic Sans MS"**,** 16)**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()   # Кнопка добавления нового вопроса  tk.Button(  self.main\_frame**,** text="Добавить вопрос"**,** command=self.open\_add\_question\_window**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Кнопка обновления списка  tk.Button(  self.main\_frame**,** text="Обновить список"**,** command=self.manage\_questions**,** # Перезагрузка данных  bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   questions = get\_all\_questions() # Получение списка вопросов   if not questions:  tk.Label(self.main\_frame**,** text="Нет вопросов в базе данных."**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()  return   # Отображение данных в таблице  table = ttk.Treeview(self.main\_frame**,** columns=("ID"**,** "Вопрос"**,** "Полезная информация")**,** show="headings")  table.heading("ID"**,** text="ID")  table.heading("Вопрос"**,** text="Вопрос")  table.heading("Полезная информация"**,** text="Полезная информация")  table.pack(fill="both"**,** expand=True**,** pady=10)   # Очистка таблицы перед заполнением новыми данными  for row in table.get\_children():  table.delete(row)   for question in questions:  table.insert(""**,** "end"**,** values=(question.question\_id**,** question.question\_text**,** question.helpful\_info))   def delete\_selected():  selected\_item = table.selection()  if not selected\_item:  messagebox.showwarning("Удаление"**,** "Выберите вопрос для удаления.")  return   question\_id = table.item(selected\_item**,** "values")[0] # Получение ID вопроса  success**,** message = delete\_question(question\_id) # Вызов метода для удаления вопроса  if success:  messagebox.showinfo("Успех"**,** message)  self.manage\_questions() # Обновление списка вопросов  else:  messagebox.showerror("Ошибка"**,** message)   # Кнопка удаления  tk.Button(  self.main\_frame**,** text="Удалить выбранный вопрос"**,** command=delete\_selected**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10) |

Таблица 1.3 – метод manage\_dogs() в admin\_ui.py

|  |
| --- |
| def manage\_dogs(self):  *"""Управление породами собак."""* clear\_frame(self.main\_frame)   # Заголовок  tk.Label(self.main\_frame**,** text="Управление породами собак"**,** font=("Comic Sans MS"**,** 16)**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()   # Функция удаления выбранной породы  def delete\_selected():  selected\_item = table.selection()  if not selected\_item:  messagebox.showwarning("Удаление"**,** "Выберите породу для удаления.")  return  dog\_id = table.item(selected\_item**,** "values")[0] # ID породы  success**,** message = delete\_dog(dog\_id)  if success:  messagebox.showinfo("Успех"**,** message)  self.manage\_dogs() # Обновление списка  else:  messagebox.showerror("Ошибка"**,** message)   # Функция редактирования выбранной породы  def edit\_selected():  selected\_item = table.selection()  if not selected\_item:  messagebox.showwarning("Редактирование"**,** "Выберите породу для редактирования.")  return   # Получение данных выбранной породы  dog\_data = table.item(selected\_item**,** "values")   # Пример dog\_data: ('dog\_id', 'breed', 'characteristics', 'behavior', 'care\_info', 'admin\_comments')   if len(dog\_data) < 6:  messagebox.showwarning("Ошибка"**,** "Недостаточно данных для редактирования.")  return   dog\_id = dog\_data[0]  breed = dog\_data[1]  characteristics = dog\_data[2]  behavior = dog\_data[3]  care\_info = dog\_data[4] # Дополнительная информация о породе  admin\_comments = dog\_data[5] # Комментарии администратора   # Вызов функции открытия окна редактирования породы, передавая все необходимые данные  self.open\_edit\_dog\_window(dog\_id**,** breed**,** characteristics**,** behavior**,** care\_info**,** admin\_comments)   # Кнопка добавления новой породы  tk.Button(  self.main\_frame**,** text="Добавить породу"**,** command=self.open\_add\_dog\_window**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Кнопка обновления списка  tk.Button(  self.main\_frame**,** text="Обновить список"**,** command=self.manage\_dogs**,** # Повторный вызов для обновления данных  bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Кнопка удаления  tk.Button(  self.main\_frame**,** text="Удалить выбранную породу"**,** command=delete\_selected**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Кнопка редактирования  tk.Button(  self.main\_frame**,** text="Редактировать выбранную породу"**,** command=edit\_selected**,** bg=BUTTON\_COLOR**,** fg=TEXT\_COLOR  ).pack(pady=10)   # Получение данных из базы  dogs = get\_all\_dogs()   if not dogs:  tk.Label(self.main\_frame**,** text="Нет записей о породах в базе данных."**,** bg=BACKGROUND\_COLOR**,** fg=TEXT\_COLOR).pack()  return   # Создание таблицы  table = ttk.Treeview(self.main\_frame**,** columns=(  "ID"**,** "Порода"**,** "Характеристики"**,** "Поведение"**,** "Информация по уходу"**,** "Комментарии")**,** show="headings")  table.heading("ID"**,** text="ID")  table.heading("Порода"**,** text="Порода")  table.heading("Характеристики"**,** text="Характеристики")  table.heading("Поведение"**,** text="Поведение")  table.heading("Информация по уходу"**,** text="Информация по уходу")  table.heading("Комментарии"**,** text="Комментарии")  table.pack(fill="both"**,** expand=True**,** pady=10)   # Заполнение таблицы данными о породах  for dog in dogs:  table.insert(""**,** "end"**,** values=(  dog.dog\_id**,** dog.breed**,** dog.characteristics**,** dog.behavior**,** dog.care\_info**,** dog.admin\_comments)) |