МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | Т.И.Белая |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ |
| Разработка системы обучения на основе игр |
| по дисциплине: ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СИСТЕМ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4232 |  |  |  | Е.А.Уткина |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc196849859)

[1. Анализ предметной области 5](#_Toc196849860)

[1.1. Актуальность разработки системы 5](#_Toc196849861)

[1.2. Анализ рынка в области обучения с применением геймификации 5](#_Toc196849862)

[2. Формулировка требований к системе 7](#_Toc196849863)

[2.1. Описание функциональных требований 7](#_Toc196849864)

[2.1.1. Функции подсистемы управления аккаунтами 7](#_Toc196849865)

[2.1.2. Функции подсистемы управления учебными модулями 10](#_Toc196849866)

[2.1.2.1. Создание учебных материалов 10](#_Toc196849867)

[2.1.2.2. Редактирование и обновление материалов 14](#_Toc196849868)

[2.1.3. Функции подсистемы прохождения обучения 18](#_Toc196849869)

[2.1.3.1. Прохождение теоретических модулей 18](#_Toc196849870)

[2.1.3.2. Прохождение тестирований 19](#_Toc196849871)

[2.1.4. Функции подсистемы управления персонажем 20](#_Toc196849872)

[2.1.4.1. Создание игрового персонажа 20](#_Toc196849873)

[2.1.4.2. Развитие игрового персонажа 21](#_Toc196849874)

[2.2. Описание нефункциональных требований 22](#_Toc196849875)

[2.2.1. Требования к программному обеспечению 22](#_Toc196849876)

[2.2.2. Требования к организационному обеспечению 23](#_Toc196849877)

[2.2.3. Требования к методическому обеспечению 23](#_Toc196849878)

[2.2.4. Требования к надежности 23](#_Toc196849879)

[2.2.5. Требования к защите информации от несанкционированного доступа 23](#_Toc196849880)

[2.2.6. Требования к стандартизации и унификации 23](#_Toc196849881)

[2.3. Исходные данные и ограничения 24](#_Toc196849882)

[3. Моделирование системы 25](#_Toc196849883)

[3.1. Моделирование системы в методологии UML 25](#_Toc196849884)

[3.2. Разработка структуры базы данных 26](#_Toc196849885)

[3.3. Проектирование классов и моделей 26](#_Toc196849886)

[3.4. Использование паттернов проектирования 27](#_Toc196849887)

[4. Реализация системы 28](#_Toc196849888)

[4.1. Настройка окружения и зависимостей 30](#_Toc196849889)

[4.2. Интеграция с базой данных 30](#_Toc196849890)

[4.3. Реализация функционала 30](#_Toc196849891)

[ЗАКЛЮЧЕНИЕ 32](#_Toc196849892)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc196849893)

[ПРИЛОЖЕНИЕ А. Руководство программиста 34](#_Toc196849894)

[ПРИЛОЖЕНИЕ Б. Руководство пользователя 38](#_Toc196849895)

[ПРИЛОЖЕНИЕ В. Листинг программного кода 47](#_Toc196849896)

**ВВЕДЕНИЕ**

Разработка программного обеспечения является одной из ключевых задач в области компьютерных технологий. Данный курсовой проект посвящен созданию информационной системы для обучения на основе игровых механик, что позволяет повысить вовлеченность студентов в образовательный процесс. Объектом исследования является процесс организации интерактивного обучения, а предметом — программная система, реализующая игровые элементы для управления учебными модулями и персонажами.

Целью работы является проектирование и реализация приложения, которое обеспечивает управление учебными модулями, создание и развитие персонажей, а также учет прогресса студентов.

Проект направлен на получение практических навыков решения реальных задач, а также на изучение подходов к созданию масштабируемых и гибких программных систем.

1. **Анализ предметной области**
   1. **Актуальность разработки системы**

Современные образовательные системы все чаще используют игровые механики (геймификацию) для повышения мотивации студентов и улучшения усвоения материала. В условиях цифровизации образования такие системы становятся особенно актуальными, так как позволяют сделать обучение более интерактивным и персонализированным.

Разработка системы обучения на основе игр решает несколько ключевых задач:

* **повышение мотивации:** игровые элементы, такие как баллы, персонажи и их развитие, стимулируют студентов к активному участию в учебном процессе.
* **индивидуализация обучения:** возможность настройки персонажей и выбора модулей позволяет адаптировать процесс под потребности каждого студента.
* **управление прогрессом:** учет баллов и прогресса прохождения модулей помогает преподавателям отслеживать успеваемость студентов.

Важность таких систем заключается в их способности сочетать образовательные цели с развлекательными элементами, что делает обучение более эффективным и увлекательным.

* 1. **Анализ рынка в области обучения с применением геймификации**

На рынке существует несколько систем, использующих геймификацию в образовании. Рассмотрим основных конкурентов, их преимущества и недостатки.

1. **Duolingo**

* **Преимущества:**
  + использование баллов, уровней и ежедневных заданий для мотивации;
  + поддержка множества языков;
* **Недостатки:**
  + ограниченный функционал для управления учебными модулями преподавателями;
  + отсутствие возможности создания и настройки персонажей;
  + фокус только на изучении языков, что ограничивает применение в других дисциплинах.

1. **Kahoot**

* **Преимущества:**
  + поддержка интерактивных викторин, которые вовлекают студентов;
  + возможность для преподавателей создавать собственные задания;
  + простота интеграции в учебный процесс.
* **Недостатки:**
  + отсутствие системы персонажей и их развития;
  + ограниченные возможности для индивидуализации обучения;
  + фокус на групповых активностях, а не на индивидуальном прогрессе.

Разрабатываемая система направлена на устранение некоторых недостатков конкурентов. Она предоставляет бесплатный доступ к функционалу, позволяет создавать и настраивать учебные модули (теория и тесты), а также включает систему персонажей с возможностью их развития за баллы, что делает обучение более персонализированным.

1. **Формулировка требований к системе**

Система должна обеспечивать авторизацию пользователей с разделением ролей (студент и преподаватель), управление учебными модулями (теория и тесты), настройку персонажей и учёт прогресса. Пользовательский интерфейс реализуется как веб-приложение, доступное через браузер, с поддержкой управления данными через базу PostgreSQL. Система должна быть безопасной (с хранением паролей в зашифрованном виде), масштабируемой для добавления новых функций и интуитивно понятной для пользователей. Доступность системы должна обеспечиваться круглосуточно, с учётом стандартных условий эксплуатации (персональные компьютеры, стабильное интернет-соединение).

* 1. **Описание функциональных требований**

Система реализует следующий функционал: регистрация и авторизация пользователей через логин и пароль с хешированием паролей; создание, редактирование и удаление учебных модулей преподавателями с поддержкой текстового контента, вопросов с вариантами ответов и изображений; прохождение модулей студентами с начислением баллов; настройка персонажа (имя, внешний вид, навыки) для студентов; отображение профиля с информацией о баллах и прогрессе. Доступ к функциям ограничен ролями: преподаватели управляют модулями, студенты проходят их и развивают персонажа.

* + 1. **Функции подсистемы управления аккаунтами**
* **Регистрация** – пользователи могут регистрироваться через специальную форму, указав уникальный логин, адрес электронной почты и пароль;
* **Авторизация** – пользователи и преподаватели могут войти в систему, используя зарегистрированные логин и пароль;

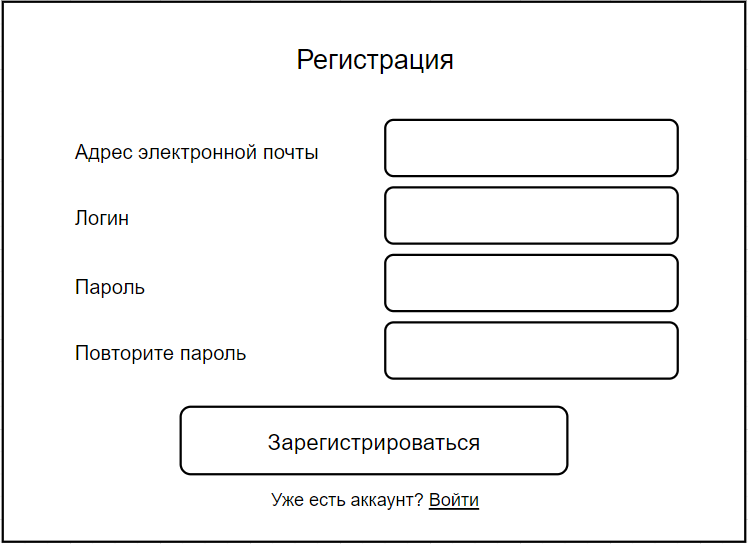


Рис. 1. Регистрация пользователя

В поле «Адрес электронной почты» должен быть введен корректный существующий адрес электронной почты в формате username@domain.com.

В поле «Логин» должен быть введен уникальный логин пользователя, содержащий только английские буквы и цифры. Минимальная длина логина – 5, максимальная – 20 символов.

В поле «Пароль» должен быть введен пароль длиной от 8 до 20 символов, содержащий как минимум одну строчную букву и одну цифру.

В поле «Повторите пароль» должен быть введен пароль, идентичный значению из поля «Пароль».

При нажатии на кнопку «Зарегистрироваться» происходит валидация введенных данных на соответствие требованиям. В случае успешной проверки система создаёт новый аккаунт и выполняется автоматическая авторизация пользователя, после чего происходит переход в личный кабинет. В случае ошибки в заполнении данных система выдаёт соответствующее сообщение об ошибке, указывая на конкретные поля, требующие исправления.

Под кнопкой «Зарегистрироваться» расположена ссылка «Войти». При нажатии на неё система перенаправляет пользователя на страницу авторизации для входа в уже существующий аккаунт.

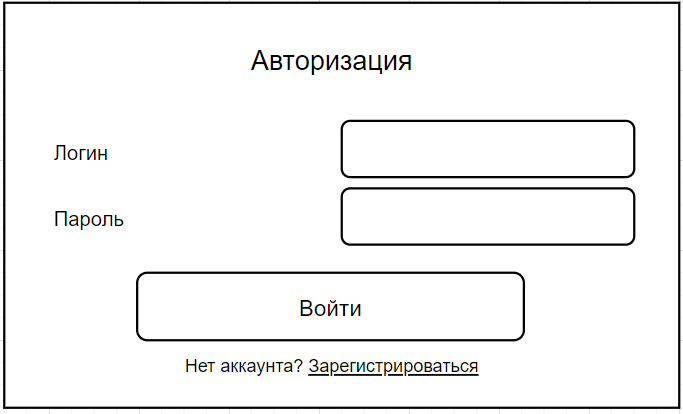


Рис. 2. Авторизация пользователя

При вводе данных в форму авторизации пользователь должен указать корректный логин и пароль. Логин должен совпадать с тем, который был ранее зарегистрирован в системе. Пароль, введенный в соответствующее поле, должен соответствовать паролю, установленному при регистрации.

При нажатии на кнопку «Войти», система проверяет введенные логин и пароль. В случае успешной проверки осуществляется авторизация, и пользователь попадает в свой личный кабинет. Если данные неверны, система выводит сообщение об ошибке, указывая на неправильный пользователь или пароль.

Под кнопкой «Войти» расположена ссылка «Зарегистрироваться». При нажатии на эту ссылку пользователь переходит на страницу регистрации для создания нового аккаунта.

Так выглядит окно личного кабинета:

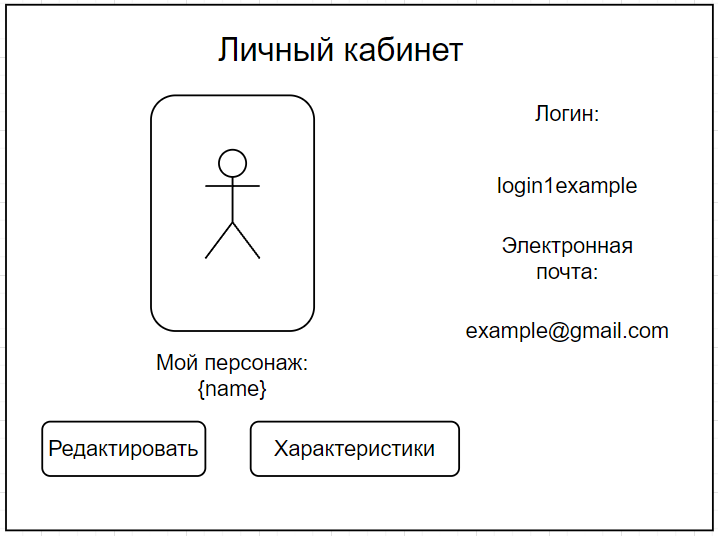


Рис. 3. Личный кабинет пользователя

Слева в личном кабинете находится изображение персонажа авторизованного пользователя. При нажатии на кнопку «Редактировать» происходит переход на окно редактирования персонажа. При нажатии на кнопку «Характеристики» происходит переход на окно управления характеристиками персонажа.

Справа в личном кабинете указаны логин и электронная почта авторизованного пользователя.

* + 1. **Функции подсистемы управления учебными модулями**
       1. **Создание учебных материалов**

Если пользователь имеет роль преподавателя, то ему доступны окна выбора, наполнения и публикации обучающего модуля:

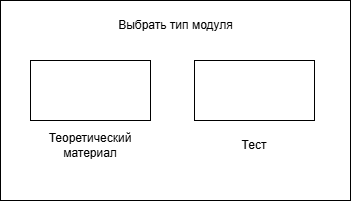


Рис. 4. Выбор типа обучающего модуля

При нажатии на один из вариантов происходит переход на страницу наполнения выбранного модуля.

Так выглядит незаполненный обучающий модуль теоретического материала:

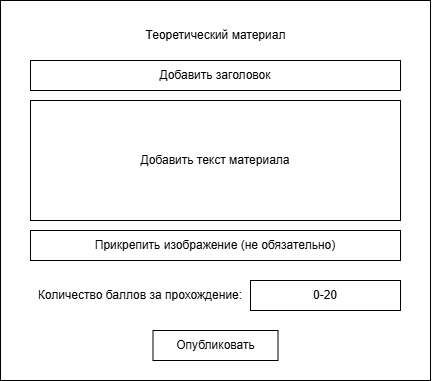


Рис. 5. Наполнение модуля «Теоретический материал»

В поле «Добавить заголовок» необходимо ввести строку, содержащую не менее 5 и не более 100 символов, используя буквы латинского или кириллического алфавита, цифры и знаки препинания (пробел, точка, запятая, дефис, тире, восклицательный и вопросительный знаки).

В поле «Добавить текст материала» вводится основной контент длиной не менее 50 символов. Текст может содержать буквы латинского или кириллического алфавита, цифры и специальные символы.

Кнопка «Прикрепить изображение» открывает возможность загрузки изображения. Изображение не является обязательным пунктом модуля. Допустимые форматы изображений — PNG, JPEG, JPG. Размер файла изображения не должен превышать 5 МБ, а разрешение — не менее 800x600 пикселей для обеспечения достаточного качества отображения в учебных модулях. Если изображение не соответствует требованиям, система выведет предупреждение и предложит выбрать другой файл.

В поле «Количество баллов за прохождение» необходимо ввести целое число от 0 до 20 включительно, обозначающее количество баллов, которое будет присваиваться пользователю при прохождении теоретического модуля.

При нажатии на кнопку «Опубликовать» система завершает процесс создания теоретического модуля. Система выполнит проверку введённых данных. В случае успешной проверки всех данных, модуль будет сохранён в системе и опубликован для пользователей. Если какие-либо поля заполнены некорректно, система выведет сообщение об ошибках и укажет, в каких полях они возникли, предложив их исправить перед публикацией.

Так выглядит шаблон теста, позволяющий создавать тесты с выбором одного правильного варианта ответа:

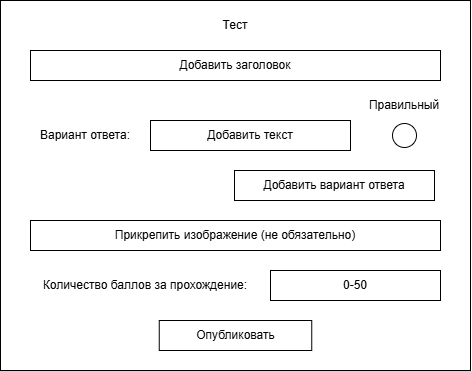


Рис. 6. Наполнение шаблона «Тест»

В поле «Добавить заголовок» необходимо ввести строку, содержащую не менее 5 и не более 100 символов, используя буквы латинского или кириллического алфавита, цифры и знаки препинания (пробел, точка, запятая, дефис, тире, восклицательный и вопросительный знаки).

В поле «Добавить вопрос» необходимо ввести строку, содержащую не менее 5 символов, используя буквы латинского или кириллического алфавита, цифры и специальные символы.

В поле «Добавить текст» вводится основной контент длиной не менее 50 символов. Текст может содержать буквы латинского или кириллического алфавита, цифры и специальные символы.

Кнопка «Добавить вариант ответа» добавляет еще один вариант ответа для заполнения.

Справа от каждого варианта ответов есть поле, при нажатии на которое вариант ответа отмечается как правильный. На каждый вопрос должен быть ровно один вариант ответа.

Кнопка «Прикрепить изображение» открывает возможность загрузки изображения. Допустимые форматы изображений — PNG, JPEG, JPG. Размер файла изображения не должен превышать 5 МБ, а разрешение — не менее 800x600 пикселей для обеспечения достаточного качества отображения в учебных модулях. Если изображение не соответствует требованиям, система выведет предупреждение и предложит выбрать другой файл.

В поле «Количество баллов за прохождение» необходимо ввести целое число от 0 до 50 включительно, обозначающее количество баллов, которое будет присваиваться пользователю при прохождении тестирования.

При нажатии на кнопку «Опубликовать» система завершает процесс создания тестирования. Система выполнит проверку введённых данных. В случае успешной проверки всех данных, модуль будет сохранён в системе и опубликован для пользователей. Если какие-либо поля заполнены некорректно, система выведет сообщение об ошибках и укажет, в каких полях они возникли, предложив их исправить перед публикацией.

* + - 1. **Редактирование и обновление материалов**

Если пользователь имеет роль преподавателя, то ему доступны окна выбора, редактирования и сохранения материала:

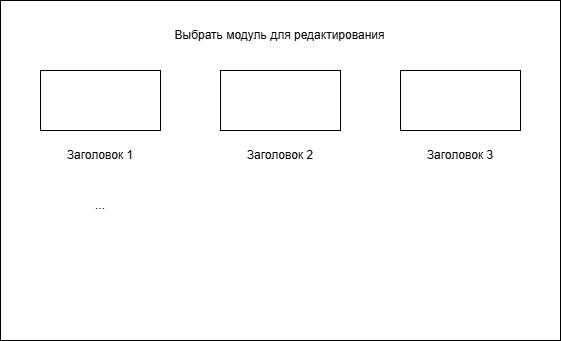


Рис. 7. Выбор модуля для редактирования

При нажатии на один из модулей происходит переход на окно редактирования в зависимости от его типа.

Так выглядит окно редактирования теоретического модуля:



Рис. 8. Окно редактирования теоретического модуля

В поле «Заголовок» будет находиться текущая версия заголовка теоретического материала. В него можно ввести измененную строку, содержащую не менее 5 и не более 100 символов, используя буквы латинского или кириллического алфавита, цифры и знаки препинания (пробел, точка, запятая, дефис, тире, восклицательный и вопросительный знаки).

В поле «Текст материала» будет находиться текущая версия текста учебного материала. В него можно ввести измененный контент длиной не менее 50 символов. Текст может содержать буквы латинского или кириллического алфавита, цифры и специальные символы.

Кнопка «Прикрепить изображение» открывает возможность изменения текущего изображения. Допустимые форматы изображений — PNG, JPEG, JPG. Размер файла изображения не должен превышать 5 МБ, а разрешение — не менее 800x600 пикселей для обеспечения достаточного качества отображения в учебных модулях. Если изображение не соответствует требованиям, система выведет предупреждение и предложит выбрать другой файл.

В поле «Количество баллов за прохождение» будет находиться текущее количество баллов, которое будет присваиваться пользователю при прохождении теоретического модуля. В него можно внести измененное количество баллов за прохождение: целое число от 0 до 20.

При нажатии на кнопку «Удалить модуль» будет выведено окно подтверждения удаления выбранного обучающего модуля и его удаление при подтверждении пользователем.

При нажатии на кнопку «Сохранить изменения» система завершает процесс редактирования теоретического модуля. Система выполнит проверку введённых данных. В случае успешной проверки всех данных, модуль будет сохранён в системе и опубликован для пользователей. Если какие-либо поля заполнены некорректно, система выведет сообщение об ошибках и укажет, в каких полях они возникли, предложив их исправить перед публикацией.

Так выглядит окно редактирования теста:

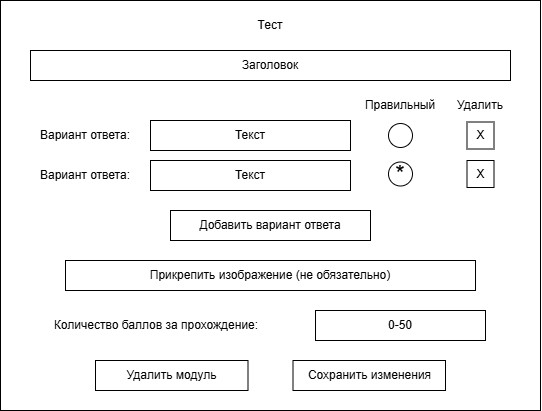


Рис. 9. Окно редактирования теста

В поле «Заголовок» будет находиться текущая версия вопроса теста. В него можно ввести измененную строку, содержащую не менее 5 и не более 100 символов, используя буквы латинского или кириллического алфавита, цифры и знаки препинания (пробел, точка, запятая, дефис, тире, восклицательный и вопросительный знаки).

Кнопка «Добавить вариант ответа» добавляет еще один вариант ответа для заполнения.

В полях «Текст» можно изменить тексты текущих вариантов ответа. Текст может содержать буквы, цифры, пробелы, а также базовые пунктуационные символы (точки, запятые, тире, двоеточия). Текст варианта ответа должен быть длиной не более 200 символов.

Справа от каждого варианта ответов есть поле, при нажатии на которое вариант ответа отмечается как правильный. На каждый вопрос должен быть ровно один вариант ответа.

При нажатии на кнопку удаления выбранный вариант ответа на вопрос удаляется.

Кнопка «Прикрепить изображение» открывает возможность изменения текущего изображения. Допустимые форматы изображений — PNG, JPEG, JPG. Размер файла изображения не должен превышать 5 МБ, а разрешение — не менее 800x600 пикселей для обеспечения достаточного качества отображения в учебных модулях. Если изображение не соответствует требованиям, система выведет предупреждение и предложит выбрать другой файл.

В поле «Количество баллов за прохождение» будет находиться текущее количество баллов, которое будет присваиваться пользователю при прохождении тестирования. В него можно внести измененное количество баллов за прохождение: целое число от 0 до 50.

При нажатии на кнопку «Удалить модуль» будет выведено окно подтверждения удаления выбранного обучающего модуля и его удаление при подтверждении пользователем.

При нажатии на кнопку «Сохранить изменения» система завершает процесс редактирования тестирования. Система выполнит проверку введённых данных. В случае успешной проверки всех данных, модуль будет сохранён в системе и опубликован для пользователей. Если какие-либо поля заполнены некорректно, система выведет сообщение об ошибках и укажет, в каких полях они возникли, предложив их исправить перед публикацией.

* + 1. **Функции подсистемы прохождения обучения**
       1. **Прохождение теоретических модулей**

Так выглядит окно с доступными материалами для прохождения. В нем будут отображаться теоретические модули и тестирования:

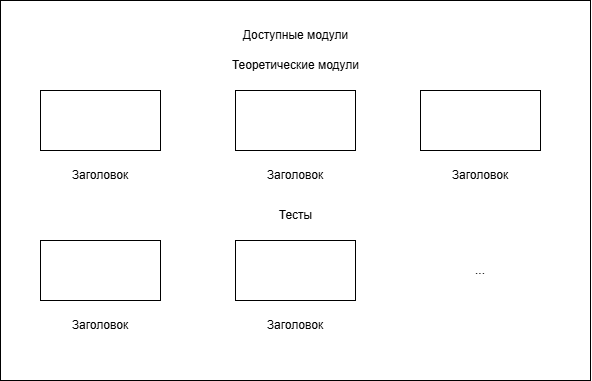


Рис. 10. Окно с материалами для обучения

При нажатии на изображение или заголовок обучающего модуля происходит переход на окно прохождения этого модуля. Галочкой в правом верхнем углу изображения модуля помечен материал, баллы за который уже получены пользователем. Проходить модули пользователь может неограниченное количество раз, но получать баллы он может только за первое прохождение.

Так выглядит окно прохождения теоретического материала:

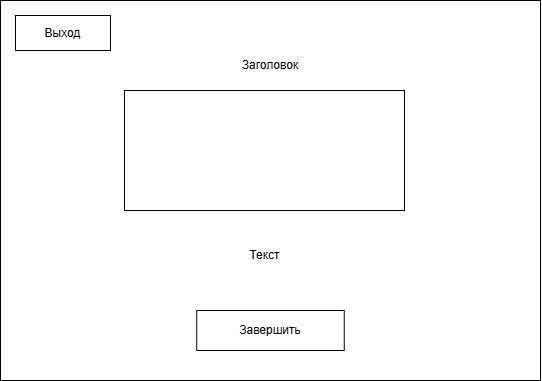


Рис. 11. Прохождение теоретического модуля

Сверху расположено изображение, прикрепленное при создании модуля. Ниже изображения – текст.

Слева от заголовка модуля расположена кнопка «Выход», при нажатии на которую происходит переход на страницу с материалами для обучения.

При нажатии на кнопку «Завершить» пользователю выводится сообщение об успешном начислении баллов за прохождение, если он проходит модуль впервые, либо переход на страницу с материалами, если он открывает теоретический модуль повторно.

* + - 1. **Прохождение тестирований**

Так выглядит окно прохождения тестирования:

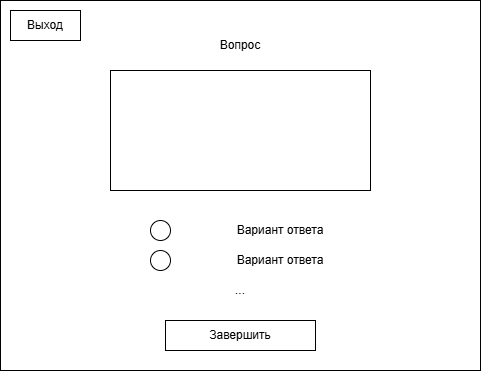


Рис. 12. Прохождение тестирования

Сверху расположен вопрос, ответ на который необходимо выбрать пользователю. Ниже расположено изображение, прикрепленное при создании модуля. Ниже изображения – варианты ответа с возможностью выбора одного из них.

Слева от вопроса теста расположена кнопка «Выход», при нажатии на которую происходит переход на страницу с материалами для обучения.

При нажатии на кнопку «Завершить» пользователю выводится сообщение об успешном начислении баллов за прохождение, если он выбрал правильный вариант ответа и проходит модуль впервые, вывод сообщения о неправильно выбранном варианте ответа и возможность повторного выбора, если пользователь ошибся, либо вывод правильного ответа и переход на страницу с материалами, если он открывает теоретический модуль повторно.

* + 1. **Функции подсистемы управления персонажем**
       1. **Создание игрового персонажа**

Так выглядит окно визуальной настройки персонажа:

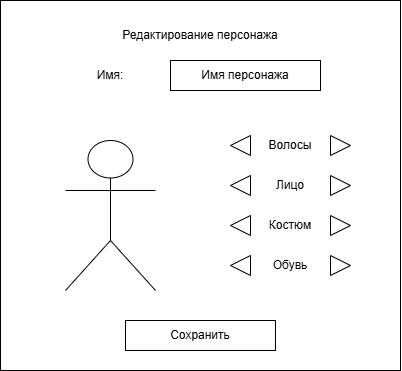


Рис. 13. Окно визуальной настройки персонажа

В поле «Имя» необходимо указать имя персонажа. Имя может содержать буквы как русского, так и английского алфавита любого регистра, а также цифры. Длина имени должна быть не менее 5 и не более 20 символов.

Ниже располагается редактор внешности персонажа. Пользователь может настроить такие параметры, как волосы, форма лица, костюм и обувь. Для изменения внешних характеристик необходимо использовать стрелки вправо и влево рядом с соответствующими элементами интерфейса, которые позволяют просматривать и выбирать различные варианты.

При нажатии кнопки «Сохранить» все внесённые изменения будут зафиксированы, и система выполнит переход к личному кабинету пользователя.

* + - 1. **Развитие игрового персонажа**

Так выглядит окно развития характеристик персонажа:

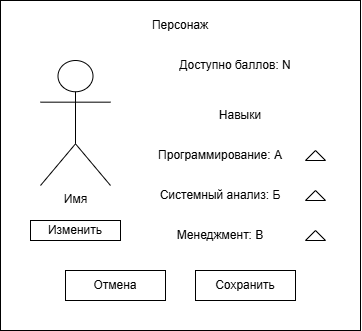


Рис. 14. Окно развития характеристик персонажа

Слева отображается изображение персонажа пользователя, под которым указано имя персонажа.

Справа расположено поле, показывающее количество доступных баллов для распределения.

Ниже представлен список навыков персонажа, которые доступны для развития. Справа от каждого навыка находится текущий уровень навыка и кнопка, нажатие на которую позволяет увеличить уровень навыка на 1. Стоимость повышения навыка на 1 уровень – 10 баллов.

При нажатии кнопки «Отмена» происходит переход в окно личного кабинета пользователя без сохранения изменений.

После нажатия кнопки «Сохранить» система сохраняет все внесённые изменения и выполняет переход в окно личного кабинета пользователя.

* 1. **Описание нефункциональных требований**
     1. **Требования к программному обеспечению**

Требования установки для клиентской части:

* Программное обеспечение: Windows 10 или Linux Astra;
* Microsoft Word 2019 и выше.

Требования для установки серверной части:

* Программное обеспечение: Ubuntu 10+;
  + 1. **Требования к организационному обеспечению**

Система должна иметь следующих пользователей:

* Студент– проходит обучающие модули, развивает игрового персонажа;
* Преподаватель– создает, редактирует и удаляет обучающие модули.
  + 1. **Требования к методическому обеспечению**

Необходимо составить следующие документы:

* Руководство пользователя;
* Руководство преподавателя;
* Руководство программиста.
  + 1. **Требования к надежности**

Серверная часть системы должна быть подключена к сети с использованием источника бесперебойного питания (ИБП) класса Online (двойного преобразования), что обеспечит стабильное питание серверов при перепадах напряжения или отключениях электроэнергии. В случае сбоя электроснабжения ИБП автоматически переключается на встроенные аккумуляторы, что позволяет системе продолжать работу без перебоев. Продолжительность автономной работы должна составлять не менее 15 минут для завершения текущих операций или перехода на резервные источники питания.

* + 1. **Требования к защите информации от несанкционированного доступа**

Система для обучения на основе игр должна предусматривать аутентификацию пользователей с использованием уникального логина и пароля. Хранение паролей происходит в зашифрованном виде с использованием устойчивого хэширования, чтобы предотвратить утечку и расшифровку паролей.

* + 1. **Требования к стандартизации и унификации**

Интерфейсы системы должны соответствовать стандарту ISO 9241-210 и включать следующие элементы:

* Шрифт Sans Serif Collection;
* Размер шрифта — 14 для основного текста, не менее 12 для дополнительного текста и 16 для названий процессов и ключевой информации;
* Цвет шрифта — темно-серый (#2C2C2C);
* Все кнопки должны иметь скругленные углы с радиусом 4 мм;
* Основной цвет интерфейса — пастельный бежевый (#FAF3E0).

Документы, указанные в Приложении А, должны соответствовать ГОСТ 7.0.97-2016 и содержать:

* Шрифт Times New Roman;
* Размер шрифта — 14;
* Абзацный отступ — 1.25 см;
* Межстрочный интервал — 1.5 строки;
* Выровненный текст по ширине с интервалом между словами в один пробел;
* Поля документа: левое — 3 см, правое — 1.5 см, верхнее и нижнее — по 2 см.
  1. **Исходные данные и ограничения**

Учебные модули включают заголовок (5–100 символов), тип (теория или тест), содержимое (текст для теории или вопрос с вариантами ответа для теста), изображение (PNG/JPEG, до 5 МБ) и баллы (0–20 для теории, 0–50 для теста). Персонажи содержат имя, параметры внешности (волосы, лицо, одежда, обувь) и навыки в формате JSON. Пользователи вводят email, логин (4–32 символа), пароль (6–32 символа) и роль. Ограничения: изображения не должны превышать 800x600 пикселей, а база данных должна поддерживать до 1000 модулей без замедления.

1. **Моделирование системы**
   1. **Моделирование системы в методологии UML**

Система смоделирована с использованием методологии UML. Диаграмма вариантов использования описывает основные сценарии взаимодействия: регистрация, авторизация, создание и прохождение модулей, настройка персонажа. Диаграмма классов отражает структуру данных: классы User, Module, Character и их связи. Диаграмма компонентов описывает компоненты системы и их связи.

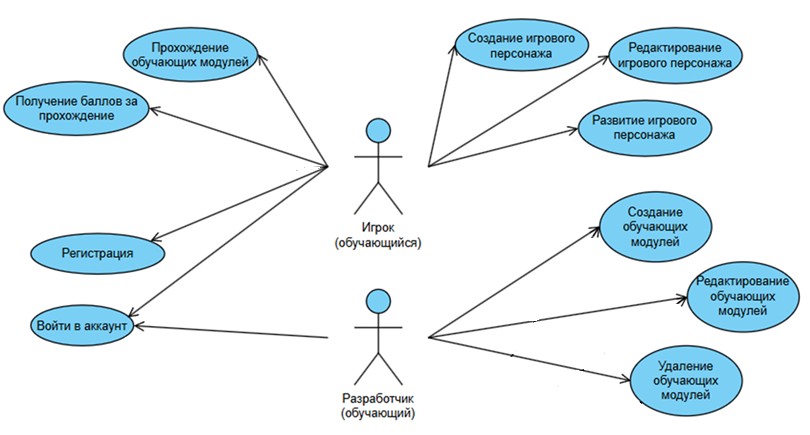


Рис. 15. Диаграмма вариантов использования

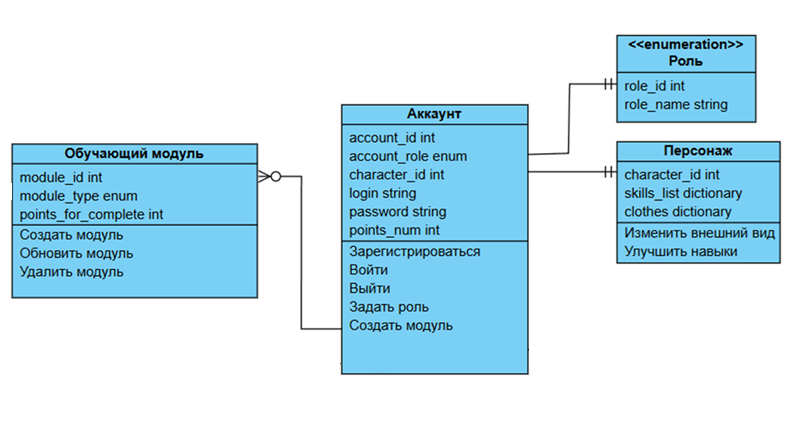


Рис. 16. Диаграмма классов

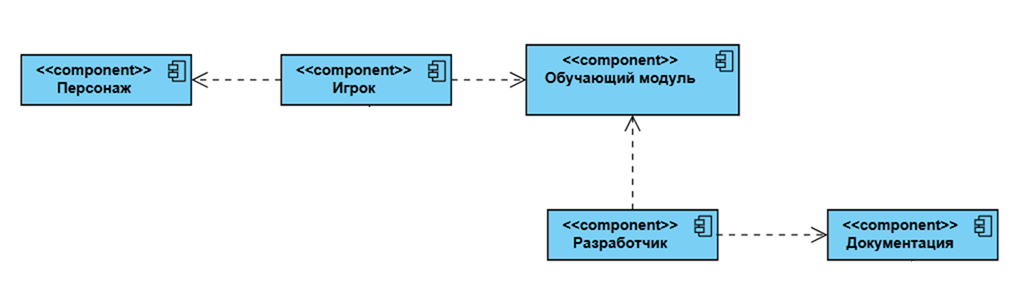


Рис. 17. Диаграмма компонентов

* 1. **Разработка структуры базы данных**

База данных PostgreSQL содержит четыре таблицы. Таблица users хранит идентификатор, email, логин, хешированный пароль, роль и баллы. Таблица modules включает идентификатор, заголовок, тип, содержимое (JSON), путь к изображению, баллы и идентификатор автора. Таблица characters содержит идентификатор, имя, параметры внешности, навыки (JSON) и идентификатор пользователя. Таблица progress связывает пользователей и модули, фиксируя статус прохождения. Внешние ключи обеспечивают целостность данных.

* 1. **Проектирование классов и моделей**

Модели реализованы с использованием SQLAlchemy. Класс User определяет поля для email, логина, пароля, роли и баллов, с отношениями к модулям, персонажу и прогрессу. Класс Module включает заголовок, тип, содержимое (JSON), изображение, баллы и связь с автором. Класс Character хранит имя, внешний вид и навыки (JSON), привязанные к пользователю. Класс Progress фиксирует завершение модуля. На рисунке 18 представлен пример модели пользователя системы со всеми необходимыми полями.

class User(Base):

\_\_tablename\_\_ = "users"

id = Column(Integer, primary\_key=True, index=True)

email = Column(String, unique=True, index=True)

login = Column(String, unique=True, index=True)

password = Column(String)

role = Column(String, default="student")

points = Column(Integer, default=0)

modules = relationship("Module", back\_populates="user")

character = relationship("Character", back\_populates="user", uselist=False)

progress = relationship("Progress", back\_populates="user")

Рисунок 18. Пример модели системы.

* 1. **Использование паттернов проектирования**

В системе применен паттерн Dependency Injection через FastAPI для управления зависимостями, такими как сессия базы данных. Паттерн Repository реализован неявно через SQLAlchemy для абстракции работы с базой. Шаблон MVC частично использован: модели (models.py) представляют данные, контроллеры (routes/\*.py) обрабатывают запросы, а представления (templates/\*.html) отображают интерфейс. Это упрощает тестирование и расширение системы.

1. **Реализация системы**

Система реализована как веб-приложение на FastAPI с использованием Python 3.9. Фронтенд построен на Jinja2-шаблонах с Bootstrap для адаптивного дизайна. Бэкенд обрабатывает запросы через маршруты, разделенные по модулям (auth, modules, characters, profile). Данные хранятся в PostgreSQL, а изображения — в папке static/uploads. Сессии пользователей управляются через middleware с секретным ключом. Код документирован и протестирован на локальном сервере. На рисунке 19 представлен результат разработки главной страницы системы.

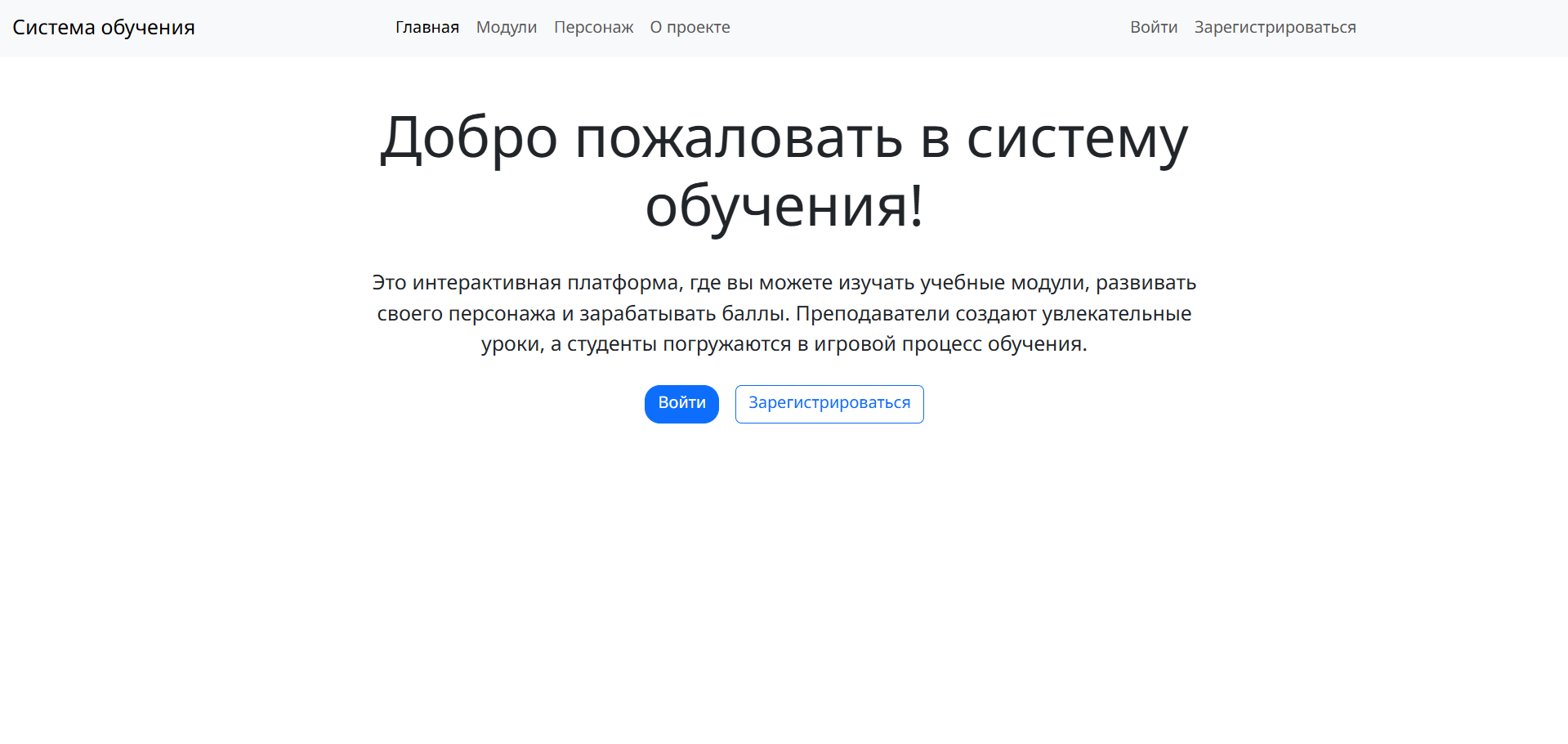


Рисунок 19. Главная страница системы

На рисунке 20 представлен пример маршрута для авторизации.

@router.get("/login", response\_class=HTMLResponse)

def get\_login(request: Request):

return templates.TemplateResponse("login.html", {"request": request})

@router.post("/login", response\_class=HTMLResponse)

def login(

login: str = Form(...),

password: str = Form(...),

db: Session = Depends(get\_db),

request: Request = None

):

user = db.query(User).filter(User.login == login).first()

if not user or not pwd\_context.verify(password, user.password):

return templates.TemplateResponse("login.html", {"request": request, "error": "Неверный логин или пароль"})

request.session["user\_id"] = user.id

return templates.TemplateResponse("login.html",

{"request": request, "success": "Вход успешен, user\_id: " + str(user.id)})

Рисунок 20. Маршруты для авторизации

На рисунке 21 представлен пример шаблона страницы авторизации.

{% extends "base.html" %}

{% block content %}

<div class="row justify-content-center">

<div class="col-md-6">

<h2 class="text-center">Вход</h2>

<form method="post" action="/auth/login" class="mt-4">

<div class="mb-3">

<label for="login" class="form-label">Логин</label>

<input type="text" class="form-control" id="login" name="login" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Пароль</label>

<input type="password" class="form-control" id="password" name="password" required>

</div>

<div class="text-center">

<button type="submit" class="btn btn-primary">Войти</button>

<a href="/auth/register" class="btn btn-link">Нет аккаунта? Зарегистрироваться</a>

</div>

</form>

</div>

</div>

{% endblock %}

Рисунок 21. Шаблон страницы авторизации

* 1. **Настройка окружения и зависимостей**

Для запуска системы создана виртуальная среда с Python 3.13. Установлены библиотеки. Файл requirements.txt содержит полный список зависимостей. Команда pip install -r requirements.txt устанавливает окружение. Uvicorn запускает сервер с параметром --reload для разработки.

* 1. **Интеграция с базой данных**

Подключение к PostgreSQL настроено в database.py с использованием строки postgresql://user:password@localhost:5432/game\_learning\_db. SQLAlchemy создает сессии через SessionLocal для транзакций. Таблицы формируются автоматически при запуске через Base.metadata.create\_all(bind=engine). Для миграций рекомендуется использовать Alembic, хотя в проекте применен прямой подход для простоты.

* 1. **Реализация функционала**

Маршрут /auth реализует регистрацию и авторизацию с хешированием паролей через bcrypt. Маршрут /modules позволяет преподавателям создавать модули (теория или тест) с валидацией данных (например, заголовок 5–100 символов, изображение до 5 МБ). Студенты проходят модули, получая баллы, которые сохраняются в progress. Маршрут /characters поддерживает настройку персонажа с сохранением внешности и навыков в JSON. Профиль (/profile) отображает баллы и роль. Проблема с полем correct при создании теоретических модулей решена изменением типа на str и ручной валидацией.

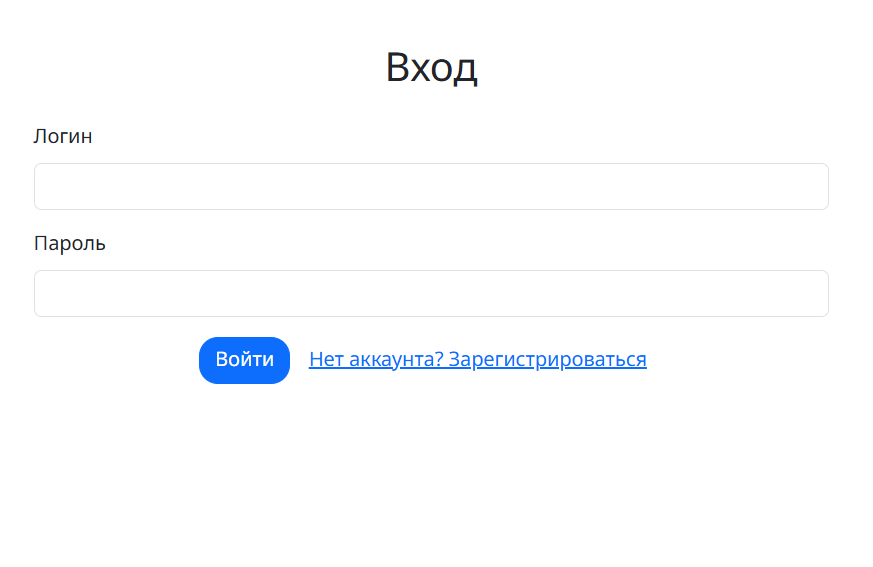


Рисунок 20. Форма авторизации с полями для логина и пароля.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана система обучения на основе игровых механик, предназначенная для повышения вовлечённости студентов и упрощения управления учебным процессом. Система обеспечивает создание и прохождение учебных модулей, настройку персонажей и учёт прогресса, ориентирована на повышение мотивации студентов и эффективности работы преподавателей.

На основании анализа предметной области были выявлены недостатки существующих решений, такие как ограниченная индивидуализация обучения и отсутствие системы персонажей. Предложенное решение устраняет эти недостатки, предоставляя интуитивно понятный и безопасный инструмент для образовательного процесса.

В рамках проекта выполнена постановка задачи, моделирование системы в методологии UML, разработка архитектуры и реализация программного обеспечения с использованием FastAPI, PostgreSQL и Bootstrap. Проведено тестирование, в результате которого подтверждена корректность работы основных функций, стабильность и соответствие заявленным требованиям.

Разработанная система может быть рекомендована к внедрению в образовательных учреждениях, нуждающихся в интерактивных инструментах обучения. Проект имеет потенциал к дальнейшему развитию: возможна реализация рейтингов, интеграция с внешними образовательными платформами и расширение функционала персонажей.

Поставленные цели курсовой работы достигнуты, и созданный программный продукт соответствует требованиям современной практики проектирования программных систем.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 ГОСТ 7.32-2017. Отчет о научно-исследовательской работе. Структура и правила оформления. – Введ. 2017-07-01. – Москва: Стандартинформ, 2017. – 37 с.

2 ГОСТ 34.602-2020. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 2020-07-01. – Москва: Стандартинформ, 2020. – 24 с.

3 FastAPI: официальная документация [Электронный ресурс] / FastAPI. – Режим доступа: <https://fastapi.tiangolo.com/> (дата обращения: 15.03.2025).

4 Duolingo [Электронный ресурс] / Duolingo Inc. – Режим доступа: <https://ru.duolingo.com/> (дата обращения: 03.04.2025).

5 Kahoot [Электронный ресурс] / Kahoot!. – Режим доступа: <https://kahoot.com/blog/2022/03/18/kahoot-stands-with-ukraine/> (дата обращения: 03.04.2025).

6 SQLAlchemy: официальная документация [Электронный ресурс] / SQLAlchemy. – Режим доступа: <https://docs.sqlalchemy.org/en/20/> (дата обращения: 20.04.2025).

7 Bootstrap: официальная документация [Электронный ресурс] / Bootstrap. – Режим доступа: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> (дата обращения: 20.04.2025).

# ПРИЛОЖЕНИЕ А. Руководство программиста

**1. Назначение и условия применения программы**

**1.1 Назначение программы**

Система обучения на основе игр (далее — Система) предназначена для организации образовательного процесса с использованием игровых механик. Она позволяет студентам изучать учебные модули и развивать персонажа, а преподавателям — создавать и управлять учебным контентом. Система реализована как веб-приложение с централизованной базой данных.

**1.2 Условия, необходимые для выполнения программы**

Для работы Системы необходимы:

* **Сервер:**
  + ОС: Linux/Windows/MacOS.
  + Python 3.9+.
  + PostgreSQL 13+.
  + Оперативная память: не менее 2 ГБ.
  + Свободное место на диске: не менее 1 ГБ.
* **Клиент:**
  + Современный веб-браузер (Google Chrome, Firefox, Edge).
  + Разрешение экрана: не менее 1024x768.
* **Зависимости:**
  + fastapi==0.103.2
  + uvicorn==0.23.2
  + sqlalchemy==2.0.21
  + psycopg2-binary==2.9.7
  + passlib[bcrypt]==1.7.4
  + jinja2==3.1.2
  + python-multipart==0.0.6
  + pillow==10.0.1

**2. Характеристики программы**

**2.1 Режим работы программы**

Система работает как веб-приложение, доступное через HTTP/HTTPS. Сервер обрабатывает запросы пользователей, взаимодействуя с базой данных PostgreSQL. Клиентская часть реализована с использованием HTML-шаблонов (Jinja2) и CSS.

**2.2 Средства контроля**

Корректность работы программы обеспечивается:

* Валидацией входных данных на стороне сервера.
* Логированием запросов и ошибок через Uvicorn (--log-level debug).
* Обработкой исключений в маршрутах FastAPI.

**3. Обращение к программе**

**3.1 Установка и запуск**

1. Клонируйте репозиторий:

git clone <repository\_url>

cd game-learning

1. Создайте виртуальную среду Python и установите зависимости:

python -m venv .venv

source .venv/bin/activate # или .venv\Scripts\activate на Windows

pip install -r requirements.txt

1. Настройте базу данных PostgreSQL:
   1. Создайте базу данных: createdb game\_learning\_db.
2. Запустите приложение:

uvicorn main:app --reload --log-level debug

1. Откройте в браузере: http://127.0.0.1:8000.

**3.2. Основные маршруты**

Таблица 1. Описание основных маршрутов Системы

|  |  |  |  |
| --- | --- | --- | --- |
| Путь | Метод | Описание | Доступ |
| / | GET | Главная страница | Все |
| /auth/login | GET | Форма входа | Все |
| /auth/login | POST | Обработка входа | Все |
| /auth/register | GET | Форма регистрации | Все |
| /auth/register | POST | Обработка регистрации | Все |
| /auth/logout | GET | Выход из системы | Авторизованные |
| /profile | GET | Просмотр профиля | Авторизованные |
| /modules | GET | Список модулей | Авторизованные |
| /modules/create | GET | Форма создания модуля | Преподаватели |
| /modules/create | POST | Обработка создания модуля | Преподаватели |

**3.4. Работа с базой данных**

База данных PostgreSQL содержит таблицы:

* **users**: пользователи (id, email, login, hashed\_password, role, points).
* **modules**: учебные модули (id, title, type, content, image, points, user\_id).
* **characters**: персонажи (id, user\_id, name, skills).

**4. Ввод и обработка данных**

**4.1. Входные данные**

Входные данные представляют собой формы HTML (email, login, password, role, title, content, image, points).

**4.2. Валидация**

Валидация входных данных осуществляется через проверку уникальности email, login, а также с помощью проверки изображение на соответствие размеру 800х600 с помощью библиотеки Pillow.

**4.2 Выходные данные**

Выходные данные представляют собой HTML-страницы, перенаправления, JSON (для ошибок).

**5. Сообщения**

**5.1 Сообщения об ошибке**

При возникновении ошибки пользователь получает отображение ошибки на странице HTML с поясняющим текстом. Пример: при вводе неверного логина вместо авторизации выводится сообщение «Неверный логин или пароль».

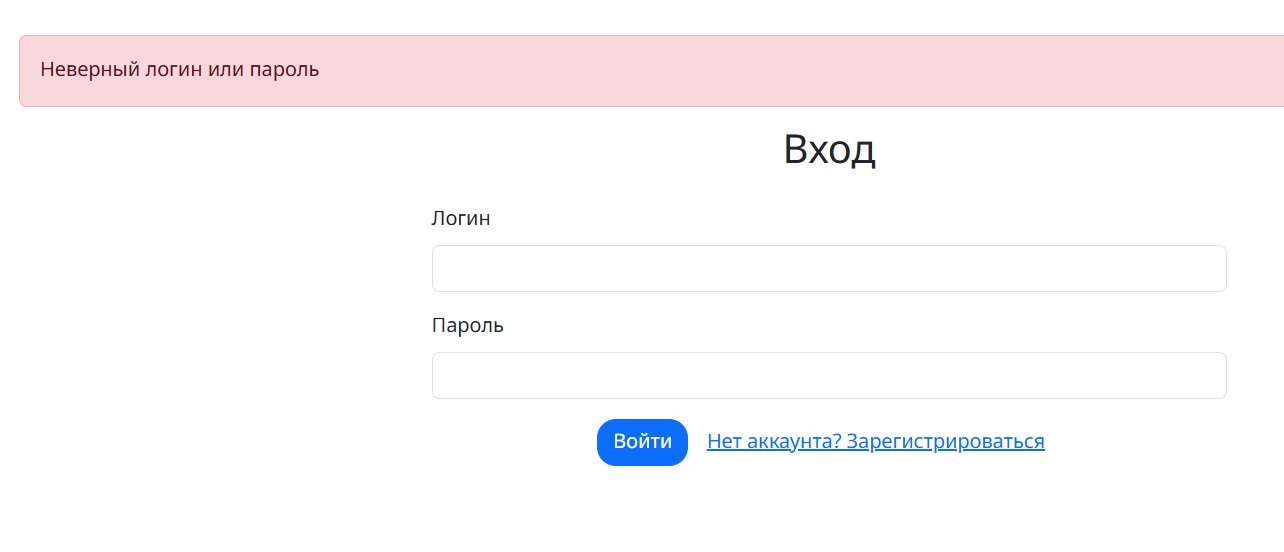
****

Рисунок 1. Сообщение об ошибке ввода.

**5.2 Сообщения об успехе**

После успешного выполнения операции пользователь получает соответствующее отображение на странице HTML. Пример: при вводе правильного логина и пароля происходит авторизация и выводится сообщение «Вход успешен, user\_id: <user\_id>».

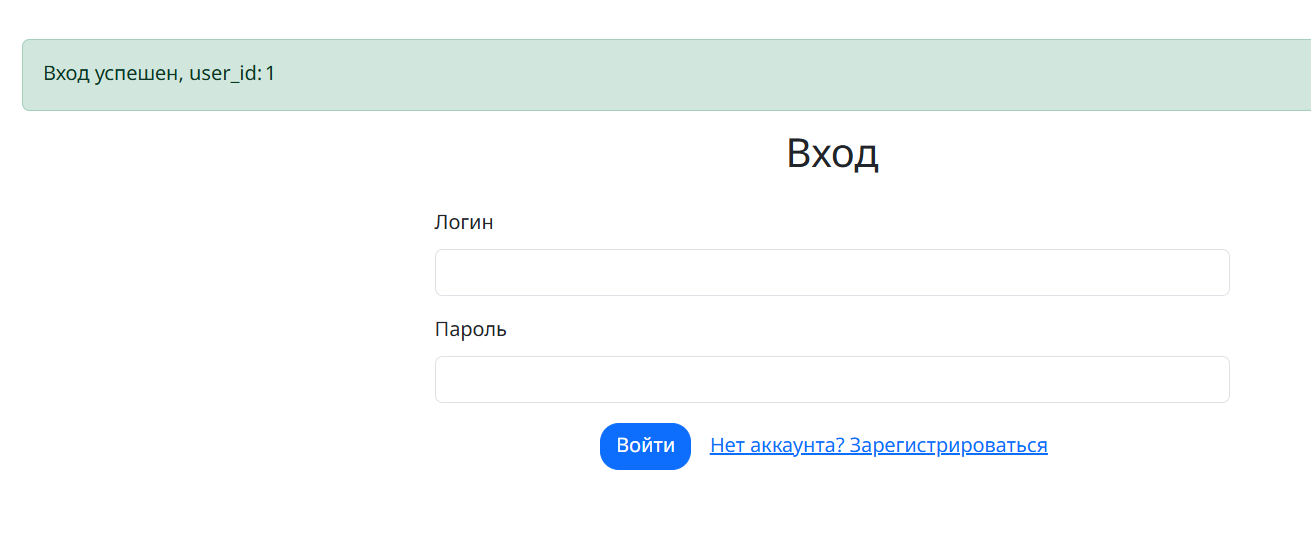


Рисунок 2. Сообщение об успехе.

# ПРИЛОЖЕНИЕ Б. Руководство пользователя

**Аннотация**

Настоящий документ представляет собой руководство пользователя для системы обучения на основе игр (далее — Система). Руководство определяет порядок работы с Системой, включая авторизацию, регистрацию, просмотр и создание учебных модулей, управление персонажем и работу с учетными записями. Перед началом работы рекомендуется внимательно ознакомиться с настоящим руководством.

Документ подготовлен в соответствии с РД 50-34.698-90.

**1. Введение**

Данное руководство предназначено для пользователей Системы, включая студентов и преподавателей. Система разработана для поддержки образовательного процесса с использованием игровых элементов, таких как учебные модули и персонализированные персонажи.

**2. Область применения**

Пользовательский интерфейс Системы обеспечивает:

* Авторизацию и аутентификацию пользователей с разделением по ролям (студент, преподаватель).
* Просмотр и создание учебных модулей (теория, тесты).
* Управление персонажем (настройка имени, навыков).
* Просмотр профиля пользователя.

**3. Краткое описание возможностей**

Система обеспечивает выполнение следующих основных функций:

* Регистрация новых пользователей (студентов или преподавателей).
* Авторизация через логин и пароль.
* Просмотр списка доступных модулей.
* Создание и редактирование модулей (доступно преподавателям).
* Настройка персонажа (доступно студентам).
* Просмотр профиля с информацией о баллах и роли.

**4. Уровень подготовки пользователей**

Для работы с Системой определены две роли: студент и преподаватель.

**Студент должен:**

* Знать основы работы с веб-приложениями (браузер, ввод данных).
* Уметь регистрироваться, входить в систему, просматривать модули и управлять персонажем.

**Преподаватель должен:**

* Знать основы работы с веб-приложениями.
* Уметь создавать и редактировать учебные модули, загружать изображения (800x600).
* Просматривать список модулей и информацию о студентах.

**5. Перечень эксплуатационной документации**

* Руководство пользователя (настоящее руководство);
* Руководство программиста.

**6. Список обозначений и сокращений**

Система – система обучения на основе игр.

GUI – графический пользовательский интерфейс.

**7. Назначение и условия применения**

**7.1 Назначение**

Система предназначена для организации образовательного процесса с использованием игровых механик. Она позволяет студентам изучать материалы через модули и развивать персонажа, а преподавателям — создавать учебный контент.

**7.2 Условия применения**

Для работы с Системой необходимо:

* Современный веб-браузер (Google Chrome, Firefox, Edge).
* Доступ к интернету.
* Учетная запись (создается через регистрацию или предоставляется администратором).
* Для преподавателей: изображения модулей размером 800x600 пикселей.

**8. Подготовка к работе**

**8.1 Порядок загрузки**

Система доступна через веб-интерфейс по адресу http://127.0.0.1:8000 (или другой URL, предоставленный администратором). Установка дополнительного ПО не требуется.

**8.2 Проверка работоспособности**

1. Откройте браузер и перейдите на главную страницу Системы.
2. Убедитесь, что страница входа загружается.
3. Выполните вход с тестовыми учетными данными (если предоставлены).

**9. Описание операций**

В данном разделе описаны операции, доступные в Системе, и их распределение по ролям.

Таблица 1. Описание операций

|  |  |  |
| --- | --- | --- |
| Роль | Доступные разделы | Доступные действия |
| Студент | Главная страница | Переход входу, регистрации, выход из системы |
| Профиль | Просмотр информации о баллах и роли |
| Модули | Просмотр списка модулей (теория, тесты) |
| Персонаж | Настройка имени и навыков персонажа |
| Преподаватель | Главная страница | Переход входу, регистрации, выход из системы |
| Профиль | Просмотр информации о баллах и роли |
| Модули | Просмотр, создание, редактирование и удаление модулей |

**10. Вход в систему и выход из неё**

1. Откройте страницу http://127.0.0.1:8000/auth/login.
2. Введите логин и пароль.
   * Логин: латинские буквы и цифры, 4–32 символа.
   * Пароль: латинские буквы и цифры, 6–32 символа.
3. Нажмите «Войти».

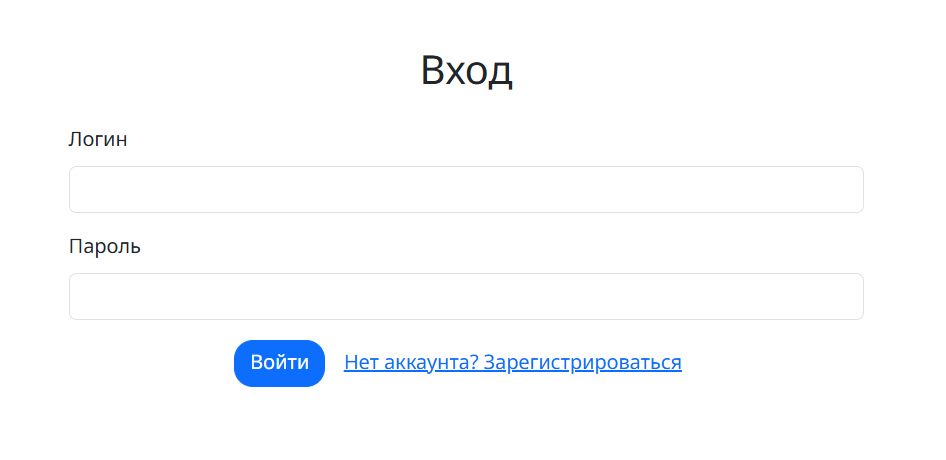


Рисунок 1. Вход в систему.

1. После входа вы будете перенаправлены в профиль.
2. Для выхода нажмите «Выйти» в профиле — вы вернетесь на страницу входа.

**11. Регистрация**

1. Перейдите на http://127.0.0.1:8000/auth/register.
2. Заполните поля: email, логин, пароль, повтор пароля.
3. Нажмите «Зарегистрироваться».

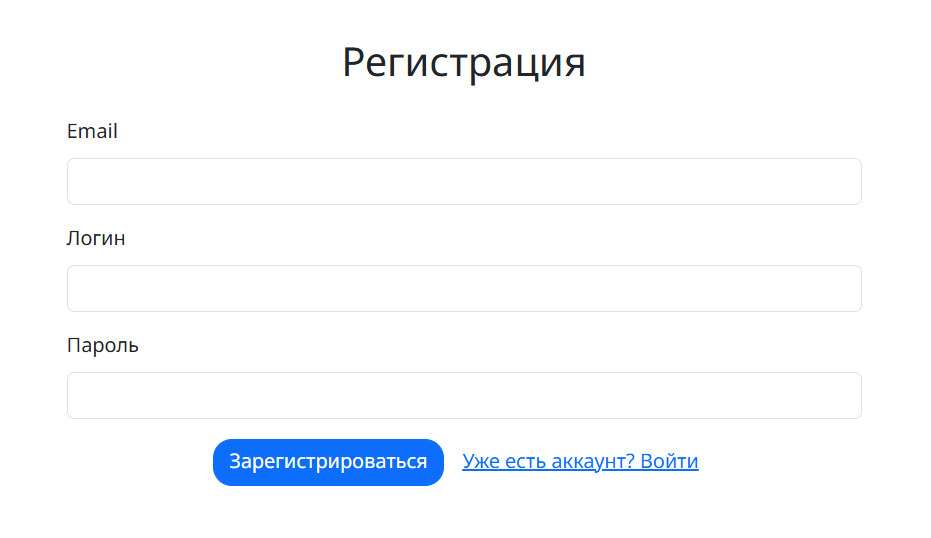


Рисунок 2. Регистрация в системе.

1. После успешной регистрации вы будете перенаправлены на страницу входа.

**12. Работа с модулями**

**Студент:**

1. Перейдите в раздел «Модули» (/modules).
2. Просмотрите список доступных модулей (теория или тесты).
3. Выберите модуль для изучения.

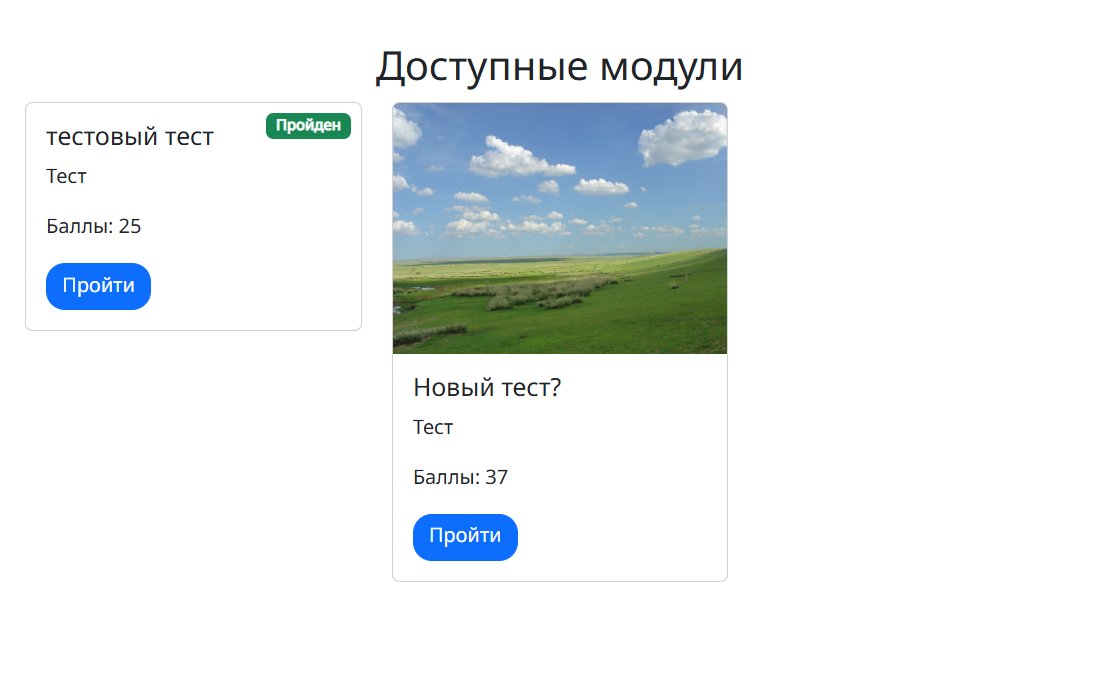


Рисунок 3. Работа с модулями.

**Преподаватель:**

1. Перейдите в раздел «Модули» (/modules).
2. Нажмите «Создать модуль» (/modules/create).
3. Заполните форму:
   * Название модуля.
   * Тип (теория/тест).
   * Содержание.
   * Изображение (800x600 пикселей).
   * Баллы за выполнение.
4. Нажмите «Создать».

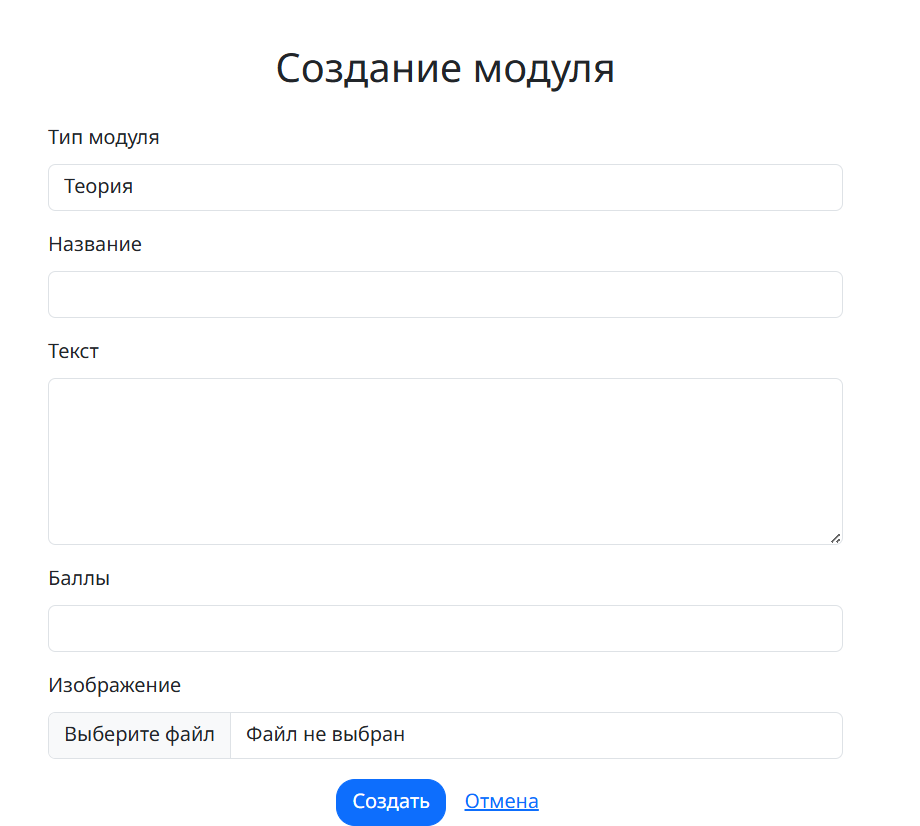


Рисунок 4. Создание модуля.

1. Для редактирования или удаления выберите модуль из списка.

**Ошибки:**

* + «Изображение должно быть 800x600» — загрузите изображение нужного размера.
  + «Поля не должны быть пустыми» — заполните все обязательные поля.

**13. Управление персонажем**

1. Перейдите в раздел «Персонаж» (/character).
2. Укажите имя персонажа и настройте навыки.
3. Нажмите «Сохранить».

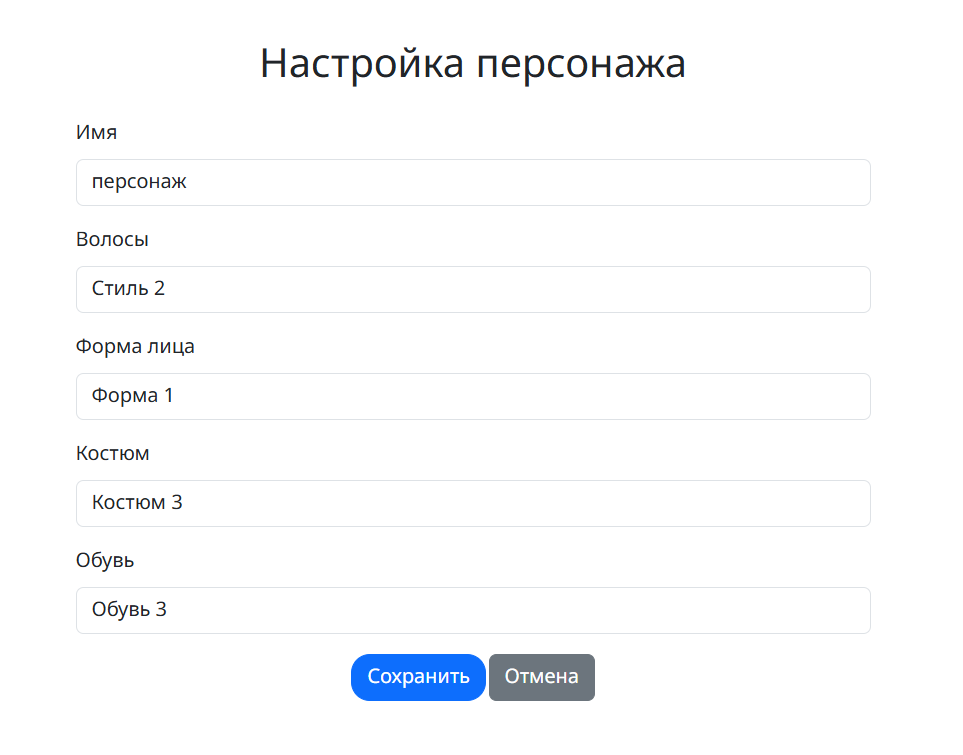


Рисунок 5. Настройка персонажа.

**Ошибки:**

* + «Имя не должно быть пустым» — укажите имя персонажа.

**14. Возможные ошибки и рекомендации по устранению**

Таблица 2. Описание ошибок

|  |  |
| --- | --- |
| Описание ошибки | Рекомендации по устранению |
| Неверный логин или пароль | Проверьте данные и повторите попытку |
| Логин уже занят | Выберите другой логин |
| Email уже зарегистрирован | Используйте другой email |
| Изображение должно быть 800х600 | Загрузите изображение нужного размера |
| Поля не должны быть пустыми | Заполните все обязательные поля |

# ПРИЛОЖЕНИЕ В. Листинг программного кода

**models.py:**

from sqlalchemy import Column, Integer, String, ForeignKey, Boolean, JSON

from sqlalchemy.orm import relationship

from database import Base

class User(Base):

\_\_tablename\_\_ = "users"

id = Column(Integer, primary\_key=True, index=True)

email = Column(String, unique=True, index=True)

login = Column(String, unique=True, index=True)

password = Column(String)

role = Column(String, default="student")

points = Column(Integer, default=0)

modules = relationship("Module", back\_populates="user")

character = relationship("Character", back\_populates="user", uselist=False)

progress = relationship("Progress", back\_populates="user")

class Module(Base):

\_\_tablename\_\_ = "modules"

id = Column(Integer, primary\_key=True, index=True)

title = Column(String, index=True)

type = Column(String)

content = Column(JSON)

image = Column(String, nullable=True)

points = Column(Integer, default=0)

user\_id = Column(Integer, ForeignKey("users.id"))

user = relationship("User", back\_populates="modules")

progress = relationship("Progress", back\_populates="module")

class Character(Base):

\_\_tablename\_\_ = "characters"

id = Column(Integer, primary\_key=True, index=True)

name = Column(String, index=True)

hair = Column(String, default="hair\_1")

face = Column(String, default="face\_1")

costume = Column(String, default="costume\_1")

shoes = Column(String, default="shoes\_1")

skills = Column(JSON, default={"strength": 1, "intelligence": 1, "agility": 1})

user\_id = Column(Integer, ForeignKey("users.id"), unique=True)

user = relationship("User", back\_populates="character")

class Progress(Base):

\_\_tablename\_\_ = "progress"

id = Column(Integer, primary\_key=True, index=True)

user\_id = Column(Integer, ForeignKey("users.id"))

module\_id = Column(Integer, ForeignKey("modules.id"))

completed = Column(Boolean, default=False)

user = relationship("User", back\_populates="progress")

module = relationship("Module", back\_populates="progress")

**auth.py:**

from fastapi import APIRouter, Depends, HTTPException, Form, Request, Response

from fastapi.responses import HTMLResponse, RedirectResponse

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from database import get\_db

from models.models import User

from passlib.context import CryptContext

router = APIRouter(prefix="/auth", tags=["auth"])

templates = Jinja2Templates(directory="templates")

pwd\_context = CryptContext(schemes=["bcrypt"], deprecated="auto")

@router.get("/register", response\_class=HTMLResponse)

def get\_register(request: Request):

return templates.TemplateResponse("register.html", {"request": request})

@router.post("/register", response\_class=HTMLResponse)

def register(

email: str = Form(...),

login: str = Form(...),

password: str = Form(...),

db: Session = Depends(get\_db),

request: Request = None

):

if db.query(User).filter(User.login == login).first():

return templates.TemplateResponse("register.html", {"request": request, "error": "Логин уже занят"})

if db.query(User).filter(User.email == email).first():

return templates.TemplateResponse("register.html", {"request": request, "error": "Email уже зарегистрирован"})

hashed\_password = pwd\_context.hash(password)

new\_user = User(email=email, login=login, password=hashed\_password, role="student")

db.add(new\_user)

db.commit()

db.refresh(new\_user)

return templates.TemplateResponse("register.html", {"request": request, "success": "Регистрация успешна"})

@router.get("/login", response\_class=HTMLResponse)

def get\_login(request: Request):

return templates.TemplateResponse("login.html", {"request": request})

@router.post("/login", response\_class=HTMLResponse)

def login(

login: str = Form(...),

password: str = Form(...),

db: Session = Depends(get\_db),

request: Request = None

):

user = db.query(User).filter(User.login == login).first()

if not user or not pwd\_context.verify(password, user.password):

return templates.TemplateResponse("login.html", {"request": request, "error": "Неверный логин или пароль"})

request.session["user\_id"] = user.id

return templates.TemplateResponse("login.html",

{"request": request, "success": "Вход успешен, user\_id: " + str(user.id)})

@router.get("/logout", response\_class=RedirectResponse)

def logout(request: Request):

request.session.clear()

return RedirectResponse(url="/", status\_code=302)

**characters.py:**

from fastapi import APIRouter, Depends, Form, Request

from fastapi.responses import HTMLResponse, RedirectResponse

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from database import get\_db

from models.models import Character, User

router = APIRouter(prefix="/character", tags=["characters"])

templates = Jinja2Templates(directory="templates")

@router.get("/edit", response\_class=HTMLResponse)

def get\_edit\_character(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

character = db.query(Character).filter(Character.user\_id == user\_id).first()

return templates.TemplateResponse("character\_edit.html", {"request": request, "character": character})

@router.post("/edit", response\_class=RedirectResponse)

def edit\_character(

name: str = Form(...),

hair: str = Form(...),

face: str = Form(...),

costume: str = Form(...),

shoes: str = Form(...),

request: Request = None,

db: Session = Depends(get\_db)

):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

character = db.query(Character).filter(Character.user\_id == user\_id).first()

if not character:

character = Character(

name=name,

hair=hair,

face=face,

costume=costume,

shoes=shoes,

user\_id=user\_id

)

db.add(character)

else:

character.name = name

character.hair = hair

character.face = face

character.costume = costume

character.shoes = shoes

db.commit()

return RedirectResponse(url="/profile", status\_code=302)

@router.get("/skills", response\_class=HTMLResponse)

def get\_skills(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

character = db.query(Character).filter(Character.user\_id == user\_id).first()

if not character:

return RedirectResponse(url="/character/edit", status\_code=302)

return templates.TemplateResponse("character\_skills.html",

{"request": request, "user": user, "character": character})

@router.post("/skills", response\_class=RedirectResponse)

def update\_skills(

strength: int = Form(...),

intelligence: int = Form(...),

agility: int = Form(...),

request: Request = None,

db: Session = Depends(get\_db)

):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

character = db.query(Character).filter(Character.user\_id == user\_id).first()

if not character:

return RedirectResponse(url="/character/edit", status\_code=302)

current\_skills = character.skills

new\_skills = {"strength": strength, "intelligence": intelligence, "agility": agility}

# Подсчитываем стоимость изменений

total\_cost = 0

for skill, new\_value in new\_skills.items():

current\_value = current\_skills.get(skill, 1)

if new\_value > current\_value:

total\_cost += (new\_value - current\_value) \* 10 # 10 баллов за уровень

if total\_cost > user.points:

return templates.TemplateResponse("character\_skills.html", {

"request": request,

"user": user,

"character": character,

"error": "Недостаточно баллов для повышения навыков"

})

# Обновляем навыки и баллы

user.points -= total\_cost

character.skills = new\_skills

db.commit()

return RedirectResponse(url="/profile", status\_code=302)

**modules.py:**

from fastapi import APIRouter, Depends, Form, Request, UploadFile, File

from fastapi.responses import HTMLResponse, RedirectResponse

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from database import get\_db

from models.models import Module, User, Progress

from typing import Optional # Добавляем для Optional[int]

router = APIRouter(prefix="/modules", tags=["modules"])

templates = Jinja2Templates(directory="templates")

@router.get("/", response\_class=HTMLResponse)

def get\_modules(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role == "teacher":

modules = db.query(Module).filter(Module.user\_id == user\_id).all()

return templates.TemplateResponse("modules\_teacher.html", {"request": request, "modules": modules})

else:

modules = db.query(Module).all()

progress = db.query(Progress).filter(Progress.user\_id == user\_id).all()

completed\_modules = {p.module\_id for p in progress if p.completed}

return templates.TemplateResponse("modules\_student.html", {"request": request, "modules": modules,

"completed\_modules": completed\_modules})

@router.get("/create", response\_class=HTMLResponse)

def get\_create\_module(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role != "teacher":

return RedirectResponse(url="/modules", status\_code=302)

return templates.TemplateResponse("module\_create.html", {"request": request})

@router.post("/create", response\_class=RedirectResponse)

async def create\_module(

type: str = Form(...),

title: str = Form(...),

text: str = Form(None),

question: str = Form(None),

options: str = Form(None),

correct: Optional[int] = Form(None), # Изменяем на Optional[int]

points: int = Form(...),

image: UploadFile = File(None),

request: Request = None,

db: Session = Depends(get\_db)

):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role != "teacher":

return RedirectResponse(url="/modules", status\_code=302)

# Валидация

if len(title) < 5 or len(title) > 100:

return templates.TemplateResponse("module\_create.html",

{"request": request, "error": "Заголовок должен быть от 5 до 100 символов"})

if type == "theory" and (not text or len(text) < 50):

return templates.TemplateResponse("module\_create.html",

{"request": request, "error": "Текст должен быть не менее 50 символов"})

if type == "test" and (not question or not options or correct is None):

return templates.TemplateResponse("module\_create.html", {"request": request,

"error": "Заполните вопрос, варианты ответа и правильный ответ"})

if type == "theory" and (points < 0 or points > 20):

return templates.TemplateResponse("module\_create.html", {"request": request, "error": "Баллы для теории: 0-20"})

if type == "test" and (points < 0 or points > 50):

return templates.TemplateResponse("module\_create.html", {"request": request, "error": "Баллы для теста: 0-50"})

# Сохранение изображения

image\_path = None

if image and image.filename:

if image.content\_type not in ["image/png", "image/jpeg", "image/jpg"]:

return templates.TemplateResponse("module\_create.html", {"request": request,

"error": "Формат изображения должен быть PNG, JPEG или JPG"})

if image.size > 5 \* 1024 \* 1024:

return templates.TemplateResponse("module\_create.html", {"request": request,

"error": "Размер изображения не должен превышать 5 МБ"})

image\_path = f"static/uploads/{image.filename}"

with open(image\_path, "wb") as f:

f.write(await image.read())

# Формирование содержимого

content = {}

if type == "theory":

content = {"text": text}

else:

options\_list = options.split(",")

content = {"question": question, "options": options\_list, "correct": correct}

new\_module = Module(

title=title,

type=type,

content=content,

image=image\_path,

points=points,

user\_id=user\_id

)

db.add(new\_module)

db.commit()

return RedirectResponse(url="/modules", status\_code=302)

# Остальные маршруты без изменений

@router.get("/edit/{module\_id}", response\_class=HTMLResponse)

def get\_edit\_module(module\_id: int, request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role != "teacher":

return RedirectResponse(url="/modules", status\_code=302)

module = db.query(Module).filter(Module.id == module\_id, Module.user\_id == user\_id).first()

if not module:

return RedirectResponse(url="/modules", status\_code=302)

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module})

@router.post("/edit/{module\_id}", response\_class=RedirectResponse)

async def edit\_module(

module\_id: int,

title: str = Form(...),

text: str = Form(None),

question: str = Form(None),

options: str = Form(None),

correct: Optional[int] = Form(None), # Аналогично для edit

points: int = Form(...),

image: UploadFile = File(None),

request: Request = None,

db: Session = Depends(get\_db)

):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role != "teacher":

return RedirectResponse(url="/modules", status\_code=302)

module = db.query(Module).filter(Module.id == module\_id, Module.user\_id == user\_id).first()

if not module:

return RedirectResponse(url="/modules", status\_code=302)

# Валидация

if len(title) < 5 or len(title) > 100:

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module,

"error": "Заголовок должен быть от 5 до 100 символов"})

if module.type == "theory" and (not text or len(text) < 50):

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module,

"error": "Текст должен быть не менее 50 символов"})

if module.type == "test" and (not question or not options or correct is None):

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module,

"error": "Заполните вопрос, варианты ответа и правильный ответ"})

if module.type == "theory" and (points < 0 or points > 20):

return templates.TemplateResponse("module\_edit.html",

{"request": request, "module": module, "error": "Баллы для теории: 0-20"})

if module.type == "test" and (points < 0 or points > 50):

return templates.TemplateResponse("module\_edit.html",

{"request": request, "module": module, "error": "Баллы для теста: 0-50"})

# Обновление изображения

if image and image.filename:

if image.content\_type not in ["image/png", "image/jpeg", "image/jpg"]:

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module,

"error": "Формат изображения должен быть PNG, JPEG или JPG"})

if image.size > 5 \* 1024 \* 1024:

return templates.TemplateResponse("module\_edit.html", {"request": request, "module": module,

"error": "Размер изображения не должен превышать 5 МБ"})

if module.image and os.path.exists(module.image):

os.remove(module.image)

image\_path = f"static/uploads/{image.filename}"

with open(image\_path, "wb") as f:

f.write(await image.read())

module.image = image\_path

# Обновление содержимого

if module.type == "theory":

module.content = {"text": text}

else:

options\_list = options.split(",")

module.content = {"question": question, "options": options\_list, "correct": correct}

module.title = title

module.points = points

db.commit()

return RedirectResponse(url="/modules", status\_code=302)

@router.get("/delete/{module\_id}", response\_class=RedirectResponse)

def delete\_module(module\_id: int, request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role != "teacher":

return RedirectResponse(url="/modules", status\_code=302)

module = db.query(Module).filter(Module.id == module\_id, Module.user\_id == user\_id).first()

if module:

if module.image and os.path.exists(module.image):

os.remove(module.image)

db.delete(module)

db.commit()

return RedirectResponse(url="/modules", status\_code=302)

@router.get("/view/{module\_id}", response\_class=HTMLResponse)

def view\_module(module\_id: int, request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role == "teacher":

return RedirectResponse(url="/modules", status\_code=302)

module = db.query(Module).filter(Module.id == module\_id).first()

if not module:

return RedirectResponse(url="/modules", status\_code=302)

progress = db.query(Progress).filter(Progress.user\_id == user\_id, Progress.module\_id == module\_id).first()

completed = progress.completed if progress else False

return templates.TemplateResponse("module\_view.html",

{"request": request, "module": module, "completed": completed})

@router.post("/complete/{module\_id}", response\_class=RedirectResponse)

def complete\_module(

module\_id: int,

answer: int = Form(None),

request: Request = None,

db: Session = Depends(get\_db)

):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

if user.role == "teacher":

return RedirectResponse(url="/modules", status\_code=302)

module = db.query(Module).filter(Module.id == module\_id).first()

if not module:

return RedirectResponse(url="/modules", status\_code=302)

progress = db.query(Progress).filter(Progress.user\_id == user\_id, Progress.module\_id == module\_id).first()

if progress and progress.completed:

return RedirectResponse(url="/modules", status\_code=302)

if module.type == "test":

if answer is None:

return templates.TemplateResponse("module\_view.html",

{"request": request, "module": module, "error": "Выберите ответ"})

correct\_answer = module.content.get("correct")

if answer != correct\_answer:

return templates.TemplateResponse("module\_view.html", {"request": request, "module": module,

"error": "Неправильный ответ. Попробуйте снова"})

if not progress:

progress = Progress(user\_id=user\_id, module\_id=module\_id, completed=True)

db.add(progress)

else:

progress.completed = True

user.points += module.points

db.commit()

return RedirectResponse(url="/modules?success=Модуль успешно пройден! Баллы начислены: " + str(module.points),

status\_code=302)

**profile.py:**

from fastapi import APIRouter, Depends, Request

from fastapi.responses import HTMLResponse, RedirectResponse

from fastapi.templating import Jinja2Templates

from sqlalchemy.orm import Session

from database import get\_db

from models.models import User

router = APIRouter(prefix="/profile", tags=["profile"])

templates = Jinja2Templates(directory="templates")

@router.get("/", response\_class=HTMLResponse)

def get\_profile(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return RedirectResponse(url="/auth/login", status\_code=302)

user = db.query(User).filter(User.id == user\_id).first()

return templates.TemplateResponse("profile.html", {"request": request, "user": user})

**database.py:**

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

# строка подключения

DATABASE\_URL = "postgresql://gl\_user:pass@localhost/game\_learning"

# движок

engine = create\_engine(DATABASE\_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

# базовый класс для моделей

Base = declarative\_base()

def get\_db():

db = SessionLocal()

try:

yield db

finally:

db.close()

**main.py:**

from fastapi import FastAPI, Depends, Request

from fastapi.templating import Jinja2Templates

from fastapi.responses import HTMLResponse

from fastapi.staticfiles import StaticFiles

from routes import auth, modules, characters, profile

from database import engine, Base, SessionLocal

from sqlalchemy.orm import Session

from models.models import User

from starlette.middleware.sessions import SessionMiddleware

app = FastAPI()

app.mount("/static", StaticFiles(directory="static"), name="static")

templates = Jinja2Templates(directory="templates")

app.add\_middleware(SessionMiddleware, secret\_key="your-secret-key")

app.include\_router(auth.router)

app.include\_router(modules.router)

app.include\_router(characters.router)

app.include\_router(profile.router)

Base.metadata.create\_all(bind=engine)

def get\_db():

db = SessionLocal()

try:

yield db

finally:

db.close()

def get\_current\_user(request: Request, db: Session = Depends(get\_db)):

user\_id = request.session.get("user\_id")

if not user\_id:

return None

return db.query(User).filter(User.id == user\_id).first()

@app.get("/", response\_class=HTMLResponse)

async def index(request: Request, user=Depends(get\_current\_user)):

response = templates.TemplateResponse("index.html", {"request": request, "user": user})

response.headers["Cache-Control"] = "no-store" # Отключаем кэширование

return response

@app.get("/about", response\_class=HTMLResponse)

async def about(request: Request):

return templates.TemplateResponse("about.html", {"request": request})