МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Проектирование и разработка серверных частей интернет-ресурсов
по профилю: Разработка и дизайн компьютерных игр и мультимедиа приложений
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Абрамович Юрий Александрович
Группа: ИКБО-22-23
Срок представления к защите: 08.12.2025
Руководитель: Синицын Анатолий Васильевич, Старший преподаватель

Тема: Серверная часть веб-приложения "Онлайн-квест"

Исходные данные: Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18; Учебно-методическое пособие по выполнению курсовой работы

Перечень вопросов, подлежащих разработке, и обязательного графического материала:
1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Разработать слой клиентского представления веб-приложения 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____/Р. Г. Болбаков/, «15» сентября 2025 г.

Задание на КР выдал: _____/А.В. Синицын/, «15» сентября 2025 г.

Задание на КР получил: _____/Ю.А. Абрамович/, «15» сентября 2025 г.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

MVC — Model–View–Controller, архитектурный паттерн разделения логики приложения.

ORM — Object-Relational Mapping, технология отображения объектов на строки базы данных.

БД / DB — База данных / Database.

SQL — Structured Query Language, язык структурированных запросов.

HTML — HyperText Markup Language, язык разметки.

CSS — Cascading Style Sheets, таблицы стилей.

JS — JavaScript, язык программирования для работы с клиентской частью.

UI — User Interface, интерфейс пользователя.

HTTP — протокол передачи гипертекста.

POST / GET — методы HTTP-запросов.

ID — уникальный идентификатор объекта.

PK — Primary Key, первичный ключ.

FK — Foreign Key, внешний ключ в таблице базы данных.

ER-диаграмма — диаграмма «сущность–связь».

PDO — PHP Data Objects, расширение PHP для работы с базами данных.

Docker — платформа для контейнеризации приложений.

Dockerfile — файл, описывающий инструкцию сборки контейнера.

Volume — постоянное хранилище Docker.

Квест — интерактивный текстовый сюжет с развилками, в котором пользователь делает выбор, влияющий на развитие истории.

Шаг — отдельный фрагмент квеста, содержащий текстовое описание и набор вариантов действий.

Сессия — игровая сессия пользователя, включающая текущий шаг и идентификатор прохождения.

ВВЕДЕНИЕ

В условиях цифровизации повседневной жизни и роста интереса к интерактивным форматам контента всё большее значение приобретают нелинейные нарративы — истории, в ходе которых пользователь принимает решения, напрямую влияющие на развитие сюжета. Одной из самых ранних и устойчивых форм таких нарративов являются текстовые квесты, также известные как «Choose Your Own Adventure» (CYOA). Изначально появившись в виде печатных книг с развилками, сегодня они активно переносятся в цифровую среду, где могут быть реализованы как в виде отдельных приложений, так и через веб-интерфейсы.

Веб-платформы для прохождения текстовых квестов предлагают ряд преимуществ:

— доступность через любой браузер без установки дополнительного ПО;

— возможность централизованного хранения сюжетов;

— поддержка сохранения прогресса между сессиями;

— простота распространения и обновления контента.

Однако многие существующие решения либо представляют собой статические HTML-страницы без сохранения состояния, либо требуют регистрации, либо ориентированы исключительно на авторов (редакторы сюжетов), а не на конечных игроков. Это создаёт потребность в простой, но функциональной системе, обеспечивающей удобное прохождение квестов через браузер, сохранение прогресса без аутентификации и поддержку расширяемой библиотеки сюжетов.

В связи с этим разработка серверной части веб-приложения «Онлайн-квест» является актуальной задачей, сочетающей в себе элементы веб-разработки, проектирования баз данных и пользовательского опыта.

Целью данной курсовой работы является разработка серверной части веб-приложения «Онлайн-квест», предназначенного для интерактивного прохождения текстовых сюжетов с сохранением прогресса, поддержкой нескольких независимых квестов и интуитивно понятным веб-интерфейсом. Система должна позволять пользователю выбирать сюжет, совершать выборы, переходить между шагами и получать соответствующие концовки, при этом не требуя регистрации или установки дополнительного программного обеспечения.

Объектом исследования является процесс взаимодействия пользователя с интерактивным текстовым контентом в веб-среде.

Предметом исследования — методы и технологии реализации серверной логики веб-приложения, включая архитектурные решения, модель данных для хранения шагов и выборов, механизм сохранения игровой сессии и взаимодействие с базой данных.

В ходе выполнения работы предполагается:

— провести анализ аналогов в области текстовых квестов;

— обосновать выбор технологического стека (PHP, PostgreSQL, Docker);

— разработать архитектуру приложения по паттерну MVC;

— спроектировать модель данных, отражающую структуру квестов, шагов и выборов;

— реализовать серверную логику обработки маршрутов, сохранения прогресса и отображения шагов;

— обеспечить контейнеризацию приложения с использованием Docker;

— провести модульное тестирование ключевых компонентов и тестирование методом «чёрного ящика».

Результатом работы станет работоспособное веб-приложение, позволяющее пользователям проходить интерактивные текстовые квесты через браузер, с автоматическим сохранением прогресса и возможностью расширения библиотеки сюжетов без изменения исходного кода. Проект демонстрирует применение современных подходов к веб-разработке в образовательном и развлекательном контексте и может служить основой для более сложных систем интерактивного повествования.

1 ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

1.1 Анализ предметной области

Современные цифровые нарративы, такие как текстовые квесты и интерактивные сюжеты, находят широкое применение в игровой индустрии, образовании, маркетинге и тренингах. Подобные приложения позволяют пользователю принимать решения, формирующие ход повествования, что повышает вовлечённость и персонализацию опыта. С ростом интереса к нелинейным историям возникает потребность в веб-платформах, которые позволяют создавать, хранить и проходить такие квесты через браузер, без установки дополнительного ПО.

Для определения требований к разрабатываемой системе рассмотрим существующие решения в области текстовых квестов.

1.1.1 Choice of Games

Описание:

Choice of Games — коммерческая платформа, выпускающая интерактивные текстовые игры с глубокой системой выбора, влияющего на развитие сюжета, характер персонажа и концовку. Игры доступны в браузере, через мобильные приложения и на Steam.

Достоинства:

- Высокое качество сценариев и балансировка решений;
- Поддержка сохранения прогресса между сессиями;
- Адаптивный интерфейс для разных устройств;
- Большое разнообразие жанров (фэнтези, киберпанк, детектив и др.).

Недостатки:

- Закрытая экосистема — пользователи не могут загружать свои квесты;

- Ориентирована на коммерческое распространение, а не на образовательные или внутренние нужды.

На Рисунке 1 представлен интерфейс платформы Choice of Games.

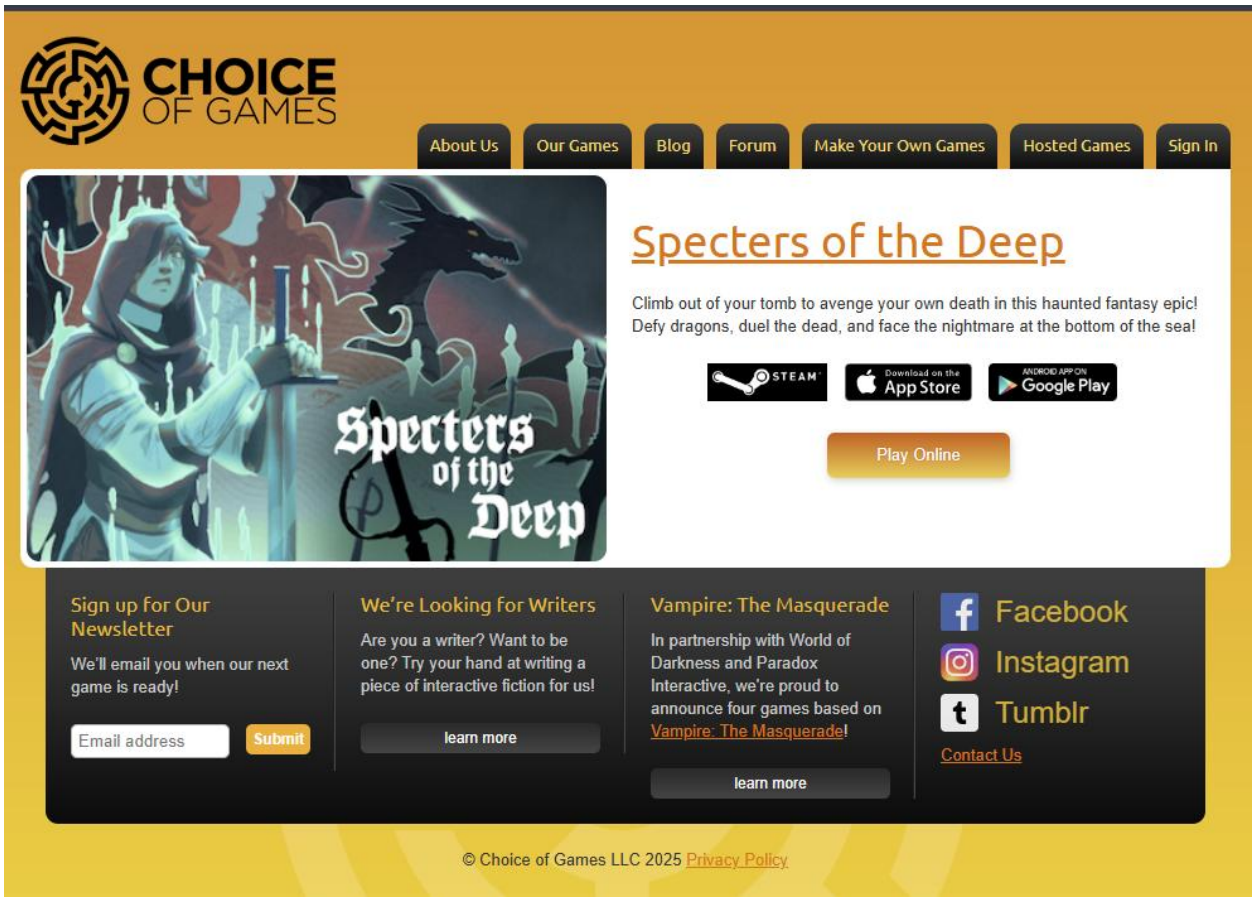

Рисунок 1 – Интерфейс платформы Choice of Games

1.1.2 TextQuest.ru

Описание:

TextQuest.ru — русскоязычная платформа, предлагающая коллекцию текстовых квестов от независимых авторов. Пользователи могут читать, проходить и комментировать квесты, а также публиковать свои работы.

Достоинства:

- Открытая площадка для авторов;
- Бесплатный доступ к большинству квестов;
- Поддержка классических механик «Choose Your Own Adventure»;
- Простой, минималистичный веб-интерфейс.

Недостатки:

- Отсутствие автоматического сохранения прогресса (в большинстве квестов);
- Нет систематизированной структуры хранения сюжетов в БД — многие квесты представлены как статические HTML-страницы;
- Ограниченные возможности по расширению функционала.

На Рисунке 2 представлен интерфейс сайта TextQuest.ru.

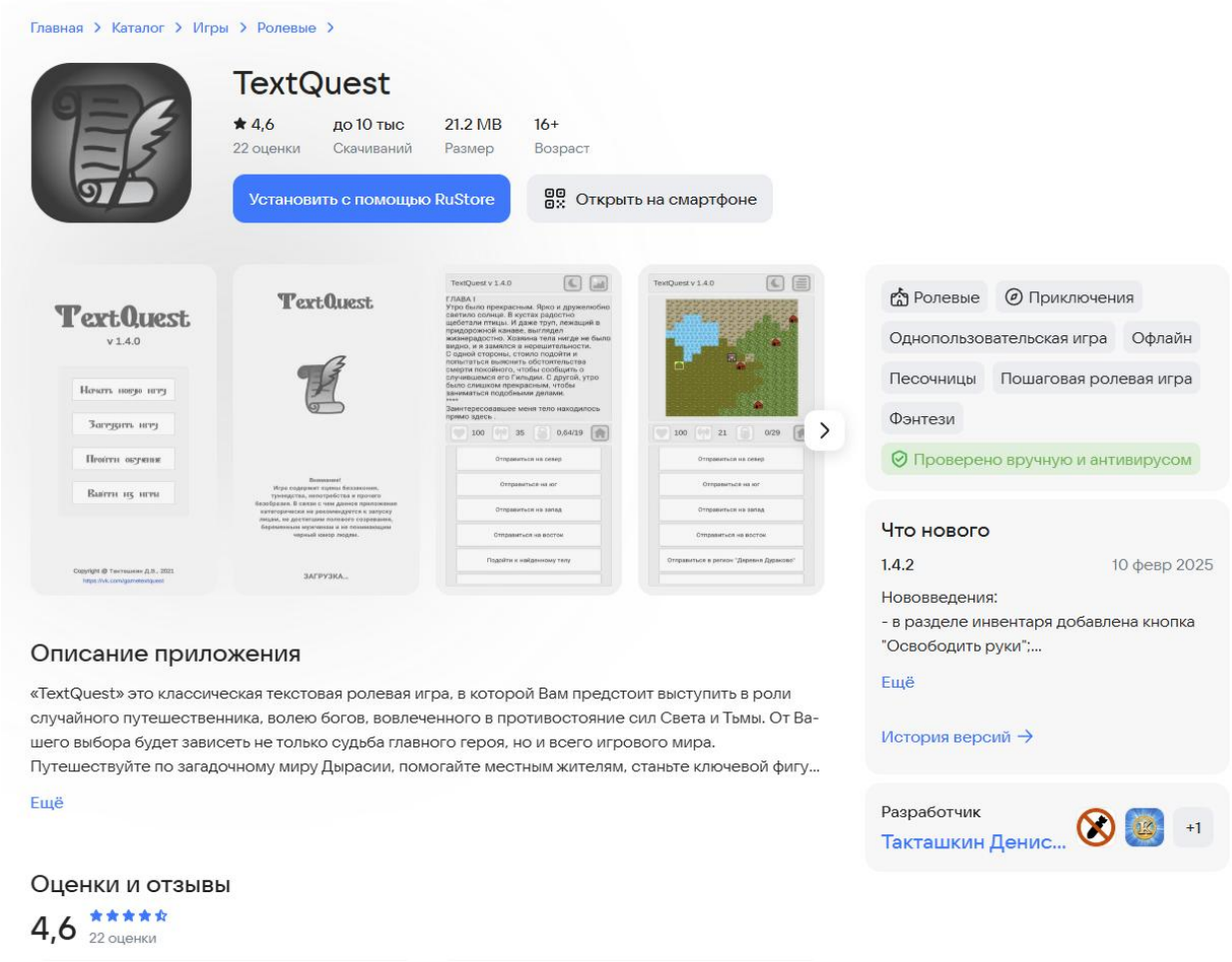

Рисунок 2 – Интерфейс сайта TextQuest.ru

1.1.3 Inklewriter

Описание:

Inklewriter — онлайн-редактор от студии Inkle, позволяющий создавать интерактивные истории с визуальным редактором сюжетных развилок. Экспорт возможен в виде HTML-файла или на мобильные устройства.

Достоинства:

- Визуальное проектирование сюжетного графа;
- Автоматическая генерация веб-версии квеста;
- Поддержка условных переходов и переменных.

Недостатки:

- Сервис больше ориентирован на авторов, а не на игроков (нет централизованной библиотеки квестов);
- Ограниченные возможности серверной части — нет аутентификации, хранения прогресса в БД, мультиплеера;
- Платформа в настоящее время не развивается активно.

На Рисунке 3 представлен интерфейс редактора Inklewriter.

Рисунок 3 – Интерфейс редактора Inklewriter

Вывод по анализу

На основе изучения аналогов можно сформулировать ключевые функциональные требования к системе «Онлайн-квест»:

- Поддержка нескольких сюжетов (расширяемая библиотека квестов);
- Интерактивность: выбор вариантов действия влияет на развитие сюжета;
- Сохранение прогресса между сессиями (даже без регистрации);
- Простой, адаптивный веб-интерфейс для прохождения квестов;

- Централизованное хранение сюжетов в базе данных (а не в статических файлах).

Нефункциональные требования:

- Быстрая загрузка шагов;
- Надёжное хранение данных;
- Простота развёртывания и поддержки;
- Возможность расширения (добавление новых квестов без изменения кода).

На основании полученных данных составим таблицу отслеживаемости.

Таблица 1 – Сравнительная таблица функциональности аналогов

Возможность / Система	Choice of Games	TextQuest.ru	Inklewriter	Онлайн-квест
Интерактивный выбор сюжета	Да	Да	Да	Да
Несколько сюжетов в библиотеке	Да	Да	Нет	Да
Сохранение прогресса между сессиями	Да	Нет	Частично	Да
Работа без регистрации	Нет	Да	Да	Да
Централизованное хранение в БД	Да	Нет	Нет	Да
Расширяемость (добавление квестов без кода)	Нет	Частично	Нет	Да

Минимальная сложность интерфейса	Средняя	Высокая	Средняя	Высокая
Поддержка мобильных устройств	Да	Нет	Да	Да

1.1 Выбор и обоснование технологий

Разработка серверной части веб-приложения «Онлайн-квест» требует выбора технологий, обеспечивающих надёжность, простоту развёртывания, удобство сопровождения и соответствие учебным целям. Исходя из функциональных требований — хранение структурированных сюжетов, обработка пользовательских выборов, сохранение прогресса, развёртывание в изолированной среде — был проведён анализ различных технологических стеков. Ниже приведено обоснование выбранных решений.

1.2.1 Выбор языка программирования и серверного фреймворка

- В качестве языка программирования серверной части выбран PHP. Этот выбор обусловлен следующими причинами:
- Широкое распространение в веб-разработке — PHP используется на значительной доле серверов в интернете, что делает его релевантным для изучения;
- Простота разработки небольших веб-приложений — не требуется сложной конфигурации или фреймворка для реализации базовой логики;
- Хорошая поддержка работы с реляционными базами данных через расширение PDO;

- Соответствие учебным задачам — позволяет наглядно продемонстрировать принципы MVC, маршрутизации и взаимодействия с БД без избыточной абстракции.

В отличие от Python (Flask) или Node.js, PHP позволяет реализовать серверную логику без использования сторонних фреймворков, что упрощает структуру проекта и делает её прозрачной для проверяющего. В данном проекте применяется чистый PHP с паттерном MVC, реализованным вручную.

1.2.2 Хранение данных

В качестве системы управления базами данных выбрана PostgreSQL. Основные причины выбора:

- Надёжность и соответствие стандартам SQL — PostgreSQL обеспечивает строгую целостность данных и поддержку транзакций;
- Поддержка сложных связей между сущностями — в квесте шаги связаны выборами, а сессии — с конкретными шагами и квестами;
- Лёгкая интеграция с Docker — официальный образ PostgreSQL позволяет автоматически инициализировать базу при первом запуске;
- Расширяемость — в будущем можно добавить поддержку JSON-полей (например, для условных переходов или переменных персонажа);
- Соответствие требованиям курсовой — в методических рекомендациях и эталонном отчёте PostgreSQL указан как предпочтительная СУБД.

1.2.3 Технологии клиентского представления

Для реализации пользовательского интерфейса используется классический стек веб-технологий:

- HTML5 — для структуры страниц;
- CSS3 — для стилизации, включая адаптивный дизайн под мобильные устройства;
- Минимальное использование JavaScript — отсутствует, так как всё взаимодействие реализовано через гиперссылки и HTTP-запросы.

Подобный подход обеспечивает максимальную простоту и совместимость с любыми браузерами, включая устаревшие, и соответствует принципу прогрессивного улучшения: даже при отключённых стилях контент остаётся читаемым.

1.2.4 Хранение игрового прогресса

Система не требует регистрации пользователя, поэтому игровая сессия сохраняется через HTTP-куки:

- При первом заходе генерируется уникальный session_id;
- Значение записывается в куку и сохраняется в таблице sessions;
- Прогресс (текущий шаг, квест) хранится в базе данных с привязкой к этому session_id.

Такой подход:

Не требует аутентификации;

- Обеспечивает персистентность между сессиями;
- Безопасен — session_id генерируется криптографически стойко (random_bytes);
- Масштабируем — легко расширить до поддержки статистики прохождений.

1.2.5 Контейнеризация

Для развёртывания приложения используется Docker. Контейнеризация решает следующие задачи:

- Изоляция зависимостей — PHP-приложение и PostgreSQL запускаются в отдельных средах;
- Единая среда разработки и развёртывания — проект работает одинаково на любом компьютере с Docker;
- Автоматическая инициализация БД — SQL-скрипт загружается при первом запуске;
- Простота запуска — достаточно выполнить команду docker-compose up.

Структура включает два сервиса:

- web — PHP-контейнер с встроенным сервером;
- db — PostgreSQL с томом для постоянного хранения данных.

Это полностью соответствует практикам, описанным в современных учебных курсах по веб-разработке.

1.2.6 Инструменты тестирования

Для автоматизированного тестирования используется PHPUnit — стандартная библиотека для модульного тестирования на PHP. PHPUnit был выбран по следующим причинам:

- Нативная поддержка PHP — не требует дополнительных сред выполнения;
- Простота написания тестов — декларативный синтаксис, понятный даже при начальном уровне владения языком;
- Интеграция с Docker — тесты запускаются внутри контейнера, что гарантирует воспроизводимость;

- Покрытие ключевых компонентов — модели данных, логика сессий, загрузка шагов.

Также проведено тестирование методом «чёрного ящика», подтверждающее корректность работы пользовательского интерфейса.

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Описание вариантов использования

В данном разделе описываются ключевые сценарии взаимодействия пользователя с системой, которые определяют основной функционал веб-приложения «Онлайн-квест». Актором выступает Пользователь — игрок, проходящий интерактивный текстовый квест через браузер.

На основе анализа предметной области и требований к системе выделены следующие варианты использования:

1. Выбор квеста — пользователь просматривает список доступных сюжетов и выбирает один для прохождения.
2. Просмотр текущего шага — система отображает текст шага квеста и набор доступных вариантов действий.
3. Совершение выбора — пользователь нажимает на одну из кнопок, и система переходит к следующему шагу сюжета.
4. Автоматическое сохранение прогресса — при каждом выборе система сохраняет текущее состояние (идентификатор квеста и шага) в базе данных, привязывая его к уникальной игровой сессии.
5. Просмотр финального шага — по достижении финального шага («КОНЕЦ») система отображает завершающий текст и предлагает начать заново.
6. Сброс игровой сессии — пользователь может в любой момент завершить текущее прохождение и вернуться к выбору квеста.

Эти сценарии охватывают полный цикл взаимодействия: от запуска до завершения и повторного запуска. При этом не требуется регистрация или авторизация — всё состояние хранится в HTTP-куках и таблице sessions.

Для визуализации внешнего поведения системы разработана диаграмма вариантов использования (см. Рисунок 4).

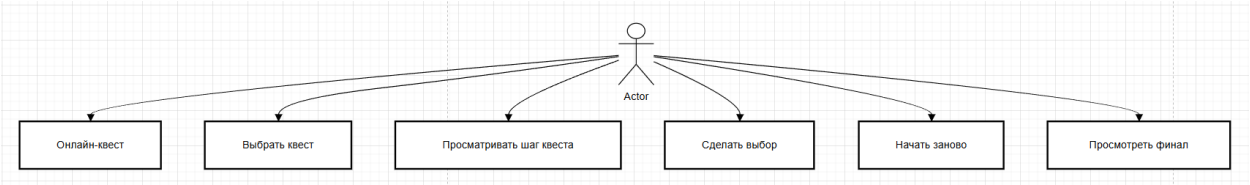

Рисунок 4 – Use-case диаграмма

Диаграмма отражает:

- одного актора — Пользователь;
- шесть основных вариантов использования;
- отсутствие сложных зависимостей (include/extend), что соответствует минимализму проекта;
- чёткое разграничение между действиями пользователя и автоматическими операциями системы (например, «Сохранение прогресса»).

2.2 Разработка архитектуры приложения

Архитектура веб-приложения «Онлайн-квест» построена на основе классического паттерна MVC (Model–View–Controller), который обеспечивает чёткое разделение ответственности между компонентами системы:

1. Модель (Model) — отвечает за работу с данными: взаимодействие с базой данных, описание сущностей (квест, шаг, выбор, сессия) и логику доступа к ним.
2. Представление (View) — формирует HTML-страницы, отображаемые пользователю. Реализовано через шаблоны на чистом PHP с подключением CSS.
3. Контроллер (Controller) — обрабатывает HTTP-запросы, координирует взаимодействие между моделью и представлением, реализует бизнес-логику (начало квеста, обработка выбора, сброс сессии).

Такой подход повышает читаемость кода, упрощает сопровождение и позволяет расширять функционал без внесения изменений в другие слои.

Описание компонентов

- Слой моделей (models/) включает классы:
  - Quest — загрузка списка квестов;
  - Step — получение текста шага и связанных с ним вариантов выбора;
  - Choice — описание перехода между шагами;
  - Session — управление игровой сессией (создание, обновление, поиск по session_id).
- Контроллер (controllers/QuestController.php) реализует:
  - обработку маршрутов через параметры ?action=...;
  - логику выбора квеста и перехода между шагами;
  - взаимодействие с моделями и передачу данных в шаблоны.
- Представления (views/) содержат:
  - шаблон выбора квеста (start.php);
  - шаблон игрового шага (step.php);
  - шаблон финального экрана (final.php);
  - общий макет страницы (layout.php).
- Точка входа (public/index.php) служит front controller’ом, маршрутизируя все запросы в контроллер.

Взаимодействие компонентов

Взаимодействие между компонентами, а также связь с внешними системами (PostgreSQL, HTTP-клиент), отражено на диаграмме компонентов (см. Рисунок 5).

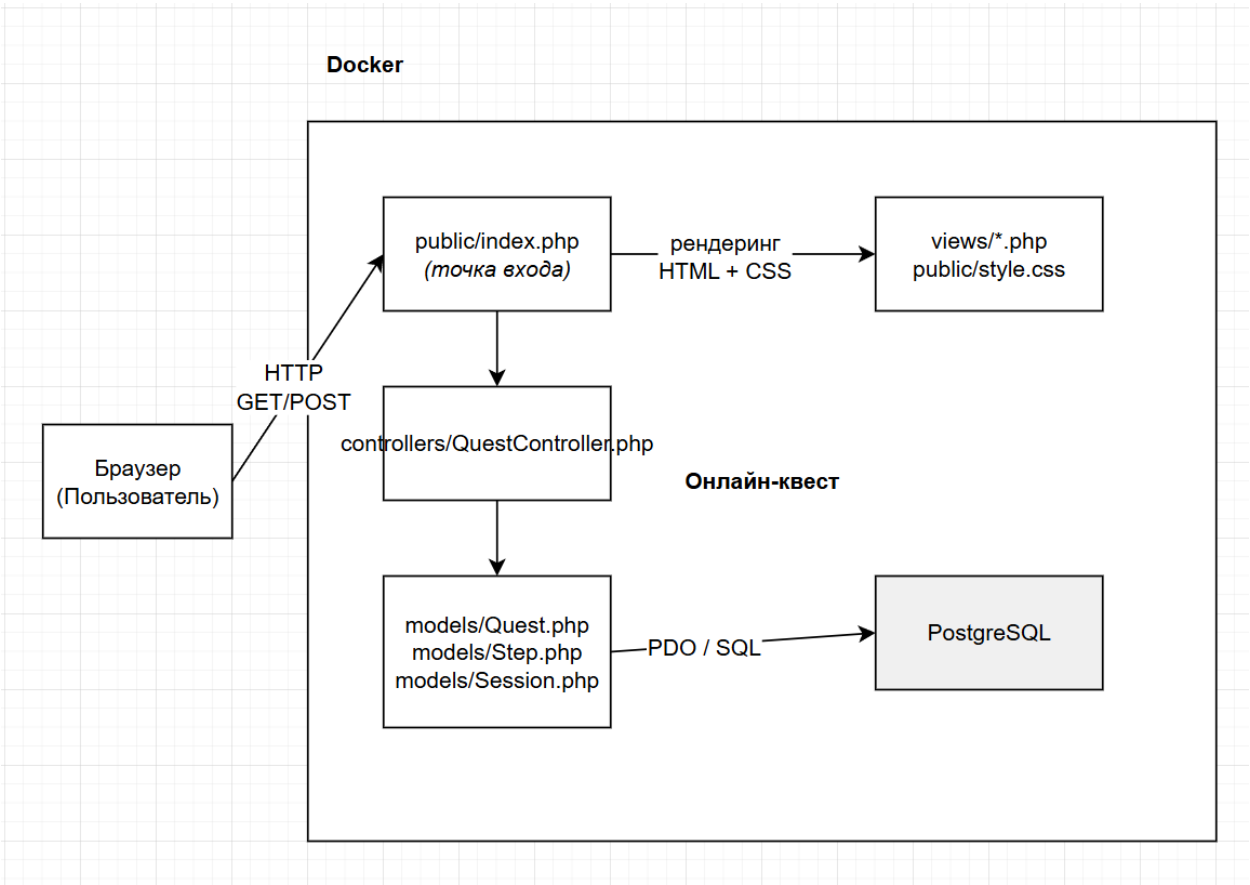

Рисунок 5 – Диаграмма компонентов

Диаграмма демонстрирует:

- запрос пользователя через браузер к index.php;
- роль контроллера как центрального узла логики;
- обращение моделей к PostgreSQL через PDO;
- генерацию HTML-ответа на основе шаблонов;
- отсутствие прямого взаимодействия между представлением и базой данных — всё проходит через контроллер и модели.

Такая архитектура обеспечивает высокую связанность внутри слоёв и низкую связность между слоями, что соответствует принципам гибкой и поддерживаемой разработки.

2.3 Модель данных системы

Для реализации функционала интерактивного прохождения текстовых квестов была разработана реляционная модель данных, отражающая структуру сюжетов, шагов, выборов и игровых сессий. Модель обеспечивает:

- хранение нескольких независимых квестов;
- описание каждого шага с текстом и флагом завершения;
- связи между шагами через варианты выбора;
- сохранение прогресса пользователя без регистрации.

В системе выделены четыре основные сущности:

1. quests — справочник квестов.

Содержит:

- id (PK) — уникальный идентификатор квеста;
- title — название квеста;
- description — краткое описание.

2. steps — шаги квеста.

Каждый шаг содержит текст и информацию о завершении сюжета. Поля:

- id (PK);
- quest_id (FK → quests.id) — привязка к квесту;
- page_number — номер шага (для удобства импорта);
- content — основной текст шага;
- is_final — флаг финального шага («КОНЕЦ»).

3. choices — варианты выбора на шаге.

Определяет переход от одного шага к другому:

- id (PK);
- from_step_id (FK → steps.id) — шаг, откуда возможен выбор;
- to_step_id (FK → steps.id) — шаг, куда ведёт выбор;
- text — текст кнопки (например, «Пойти налево»).

4. sessions — игровые сессии пользователей.

Хранит текущее состояние прохождения:

- id — уникальный идентификатор сессии (генерируется как session_id в куках);
- quest_id (FK → quests.id);
- current_step_id (FK → steps.id);
- created_at, updated_at — временные метки.

Структура сущностей и связей между ними представлена на ER-диаграмме (см. Рисунок 6).

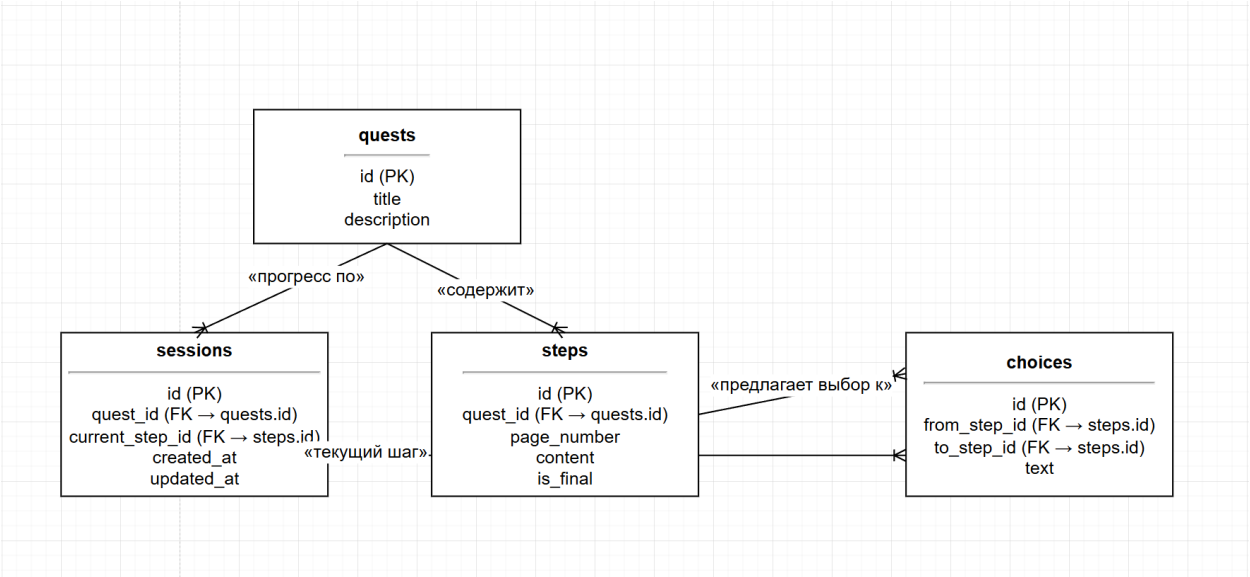

Рисунок 6 – ER-диаграмма

Модель данных обеспечивает:

- целостность сюжета — каждый квест состоит из своих шагов;
- гиперграф повествования — шаги могут иметь несколько входов и выходов;
- сохранение прогресса — даже без учётной записи;
- расширяемость — новые квесты добавляются через SQL-запросы, без изменения схемы.

Все отношения реализованы через внешние ключи, что гарантирует консистентность данных при удалении квеста (каскадное удаление шагов и выборов).

2.4 Разработка серверной части интернет-ресурса

Серверная часть веб-приложения «Онлайн-квест» реализована на языке PHP без использования фреймворков. Основная задача сервера — обрабатывать HTTP-запросы пользователя, управлять состоянием игровой сессии, загружать данные из базы PostgreSQL и формировать HTML-ответ.

Основные функции серверной логики:

- Обработка маршрутов через единый entrypoint (public/index.php) с параметрами ?action=...;
- Запуск нового квеста — при выборе сюжета создаётся новая игровая сессия, генерируется session_id, который сохраняется в HTTP-куку и в таблице sessions;
- Отображение текущего шага — сервер загружает текст шага и связанные с ним варианты выбора, формирует HTML-страницу;
- Обработка выбора пользователя — при нажатии на кнопку обновляется current_step_id в таблице sessions;
- Сброс сессии — удаление записи из БД и очистка куки;
- Работа с PostgreSQL через расширение PDO, обеспечивающее безопасность от SQL-инъекций.

Диаграммы последовательностей

Для иллюстрации ключевых сценариев взаимодействия пользователя с системой разработаны две диаграммы последовательностей.

Сценарий 1: Начало прохождения квеста

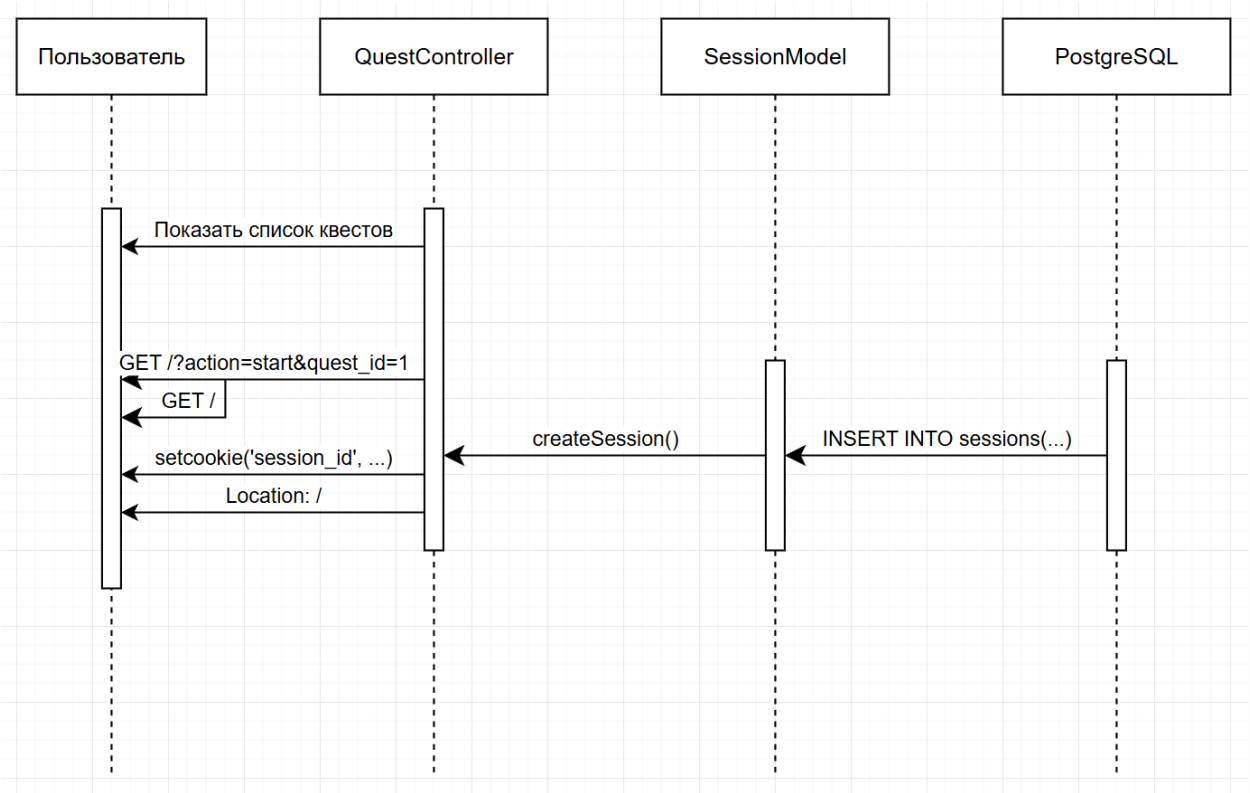

Рисунок 7 – Диаграмма последовательностей: «Начало прохождения квеста»

Диаграмма отражает следующую последовательность:

- Пользователь выбирает квест на главной странице;
- Браузер отправляет GET-запрос с параметрами ?action=start&quest_id=1;
- Контроллер ищет начальный шаг (page_number = 1);
- Генерируется уникальный session_id, записывается в куку и в таблицу sessions;
- Пользователь перенаправляется на главную страницу, где теперь отображается первый шаг квеста.

Сценарий 2: Совершение выбора и переход к следующему шагу

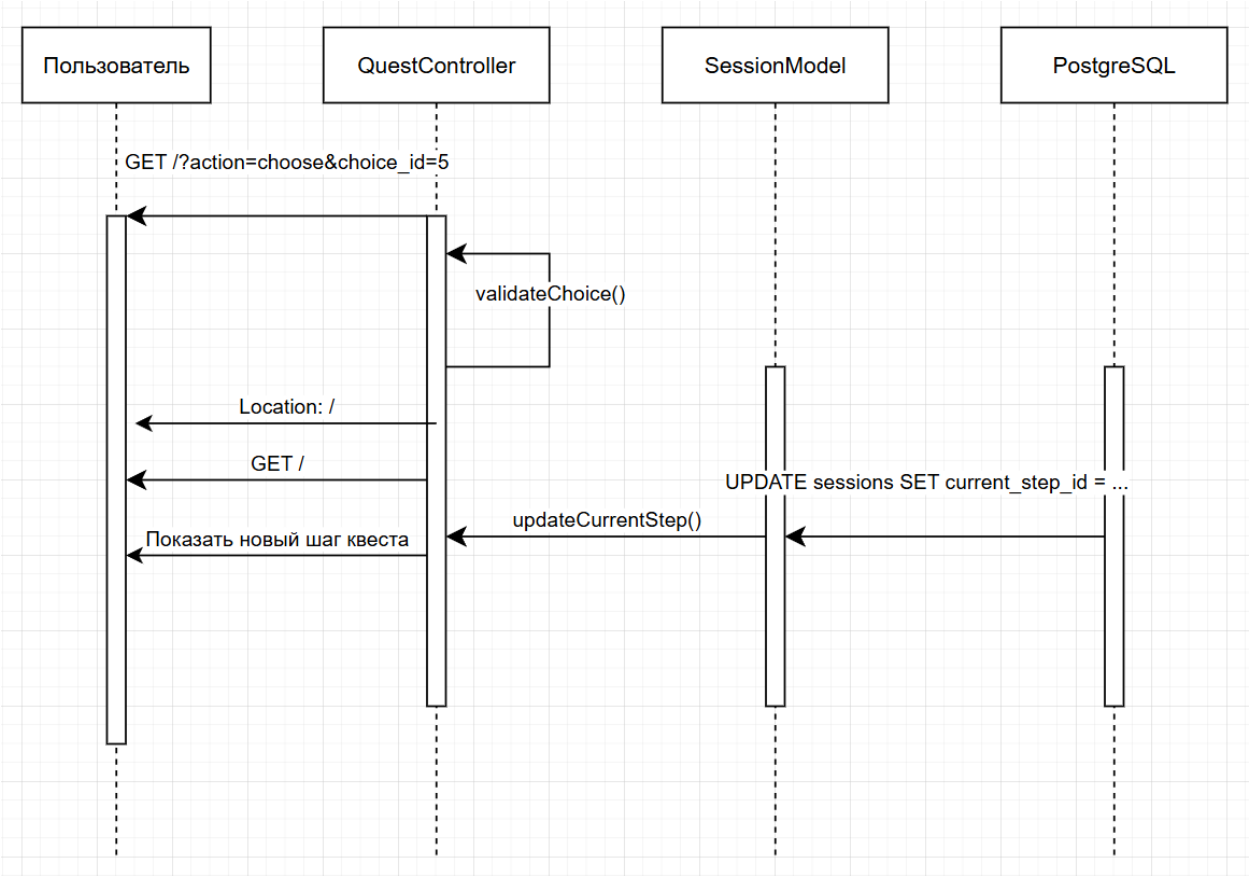

Рисунок 8 – Диаграмма последовательностей: «Совершение выбора»

Последовательность:

- Пользователь нажимает на кнопку «Пойти налево»;
- Браузер отправляет ?action=choose&choice_id=5;
- Контроллер проверяет, что выбор относится к текущему шагу (по session_id);
- Обновляется current_step_id в БД;
- Сервер отображает новый шаг.

Обе диаграммы подчёркивают отсутствие аутентификации, прозрачность работы с куками и централизованную роль контроллера в координации логики.

3 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Описание модулей и структур данных

Серверная часть веб-приложения «Онлайн-квест» реализована на языке PHP без использования сторонних фреймворков. Проект структурирован в соответствии с архитектурным паттерном MVC (Model–View–Controller), что обеспечивает чёткое разделение ответственности и упрощает сопровождение кода.

На Рисунке 9 представлена итоговая структура проекта.

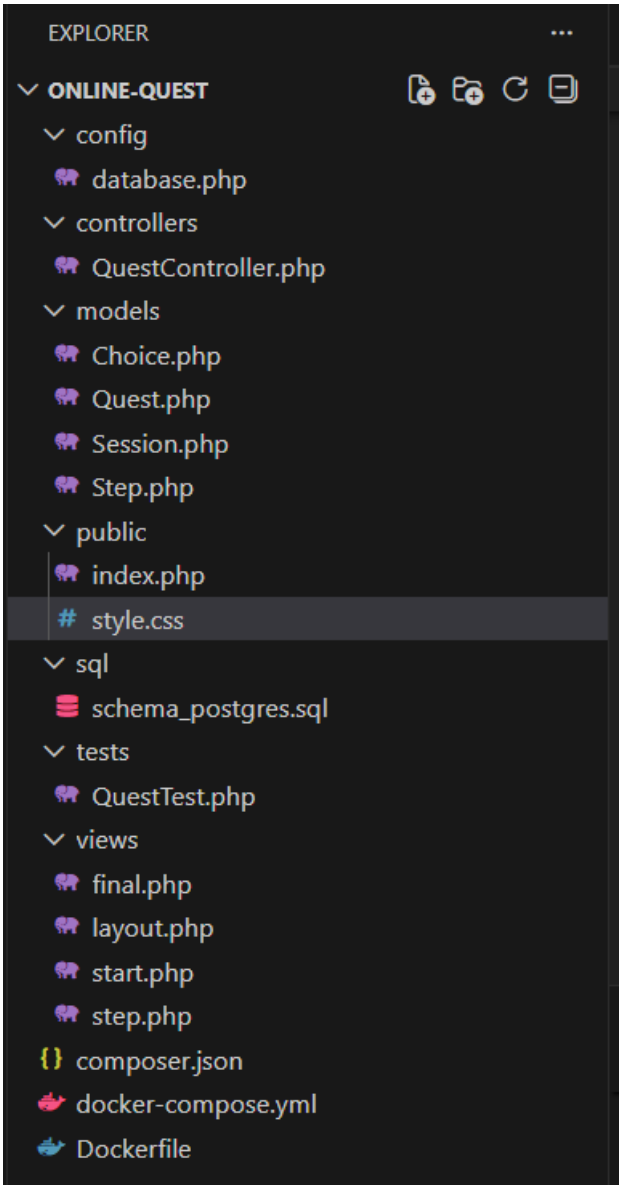

Рисунок 9 – Итоговая структура проекта

Проект состоит из следующих ключевых модулей:

1. Модуль config/database.php

Отвечает за установление соединения с базой данных PostgreSQL. Использует расширение PDO для обеспечения безопасности от SQL-инъекций и совместимости с различными СУБД. Подключение инициализируется один раз при запуске приложения и передаётся во все модели.

2. Модуль слоя моделей (models/)

Содержит классы, описывающие сущности базы данных и логику доступа к ним:

- Quest.php — загружает список доступных квестов;
- Step.php — получает текст шага и связанные с ним варианты выбора (getChoices);
- Choice.php — служебный класс (в данном проекте не содержит логики, но сохранён для структурной целостности);
- Session.php — управляет игровой сессией: создаёт, обновляет, ищет по session_id.

Все методы используют подготовленные запросы (prepare + execute), что исключает риски инъекций.

3. Модуль контроллера (controllers/QuestController.php)

Является центральным компонентом логики приложения. Обрабатывает все возможные действия пользователя:

- showStartOrStep() — отображает либо выбор квеста, либо текущий шаг;
- startQuest($quest_id) — создаёт новую сессию, генерирует session_id, сохраняет в куку;
- handleChoice($choice_id) — проверяет корректность выбора и обновляет текущий шаг;
- reset() — удаляет сессию из БД и очищает куку.

Контроллер не содержит HTML, работает только с данными и вызывает соответствующие шаблоны.

4. Модуль представлений (views/)

Реализован через простые PHP-шаблоны с подключением CSS:

- start.php — список квестов;
- step.php — игровой шаг с кнопками выбора;
- final.php — финальный экран с надписью «КОНЕЦ»;
- layout.php — общий макет страницы, определяющий структуру HTML и подключение стилей.

Шаблоны используют htmlspecialchars() для защиты от XSS.

5. Точка входа (public/index.php)

Выполняет роль front controller'а. Все запросы проходят через этот файл, который:

- определяет действие по параметру ?action=...;
- инициализирует контроллер;
- передаёт управление соответствующему методу.

Такой подход упрощает маршрутизацию и делает систему легко расширяемой.

6. Дополнительные файлы

- public/style.css — стили с адаптивным дизайном, скруглёнными кнопками и декоративным фоном;
- sql/schema_postgres.sql — SQL-скрипт для инициализации базы данных с двумя квестами;
- Dockerfile и docker-compose.yml — конфигурация контейнеризации.

3.2 Тестирование программного продукта

Для обеспечения надёжности и корректности работы серверной части веб-приложения «Онлайн-квест» было проведено два вида тестирования: модульное (автоматизированное) и тестирование методом «чёрного ящика» (ручное). Оба подхода дополняют друг друга и позволяют убедиться в работоспособности как отдельных компонентов, так и системы в целом.

3.2.1 Модульное тестирование

Для автоматизированного тестирования была использована библиотека PHPUnit — стандартный инструмент модульного тестирования на PHP. Были разработаны тесты, охватывающие ключевые компоненты системы:

- Загрузка списка квестов — проверка, что возвращаются оба сюжета: «Кольб и дракон» и «Голос из подвала»;
- Загрузка шага по идентификатору — проверка текста и флага завершения;
- Получение вариантов выбора для шага — корректность связей между шагами;
- Создание и сохранение игровой сессии — проверка записи в таблицу sessions;
- Обнаружение финального шага — проверка, что шаг с «КОНЕЦ» помечен как is_final = true.

Все тесты запускались внутри Docker-контейнера, что гарантирует изоляцию и воспроизводимость результатов. Для этого использовалась глобальная установка PHPUnit через Phar-архив.

На Рисунке 12 представлен фрагмент кода тестового класса QuestTest.php.

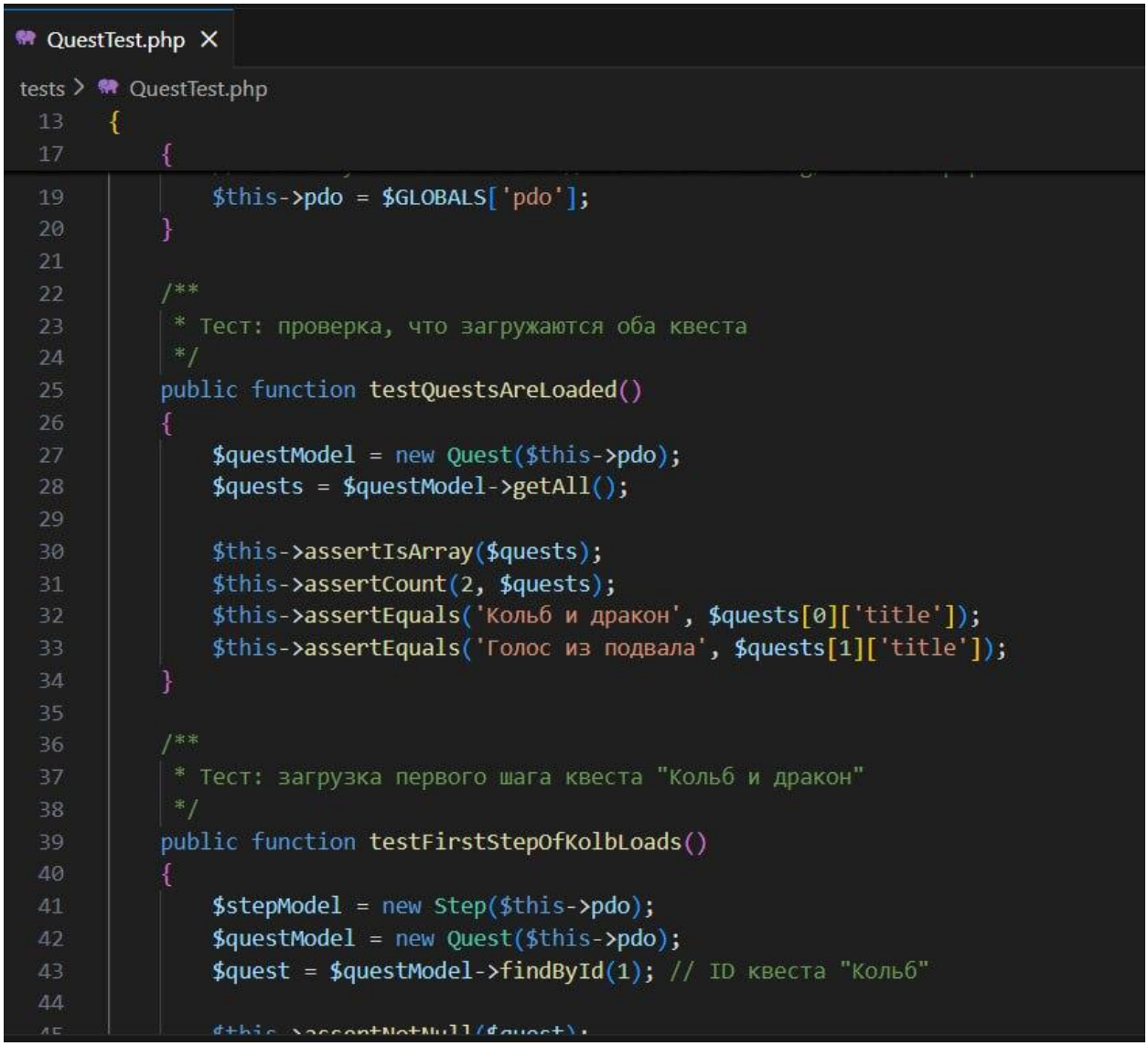

Рисунок 12 – Пример модульных тестов

На Рисунке 13 показан результат выполнения тестов — все 5 тестов успешно пройдены.

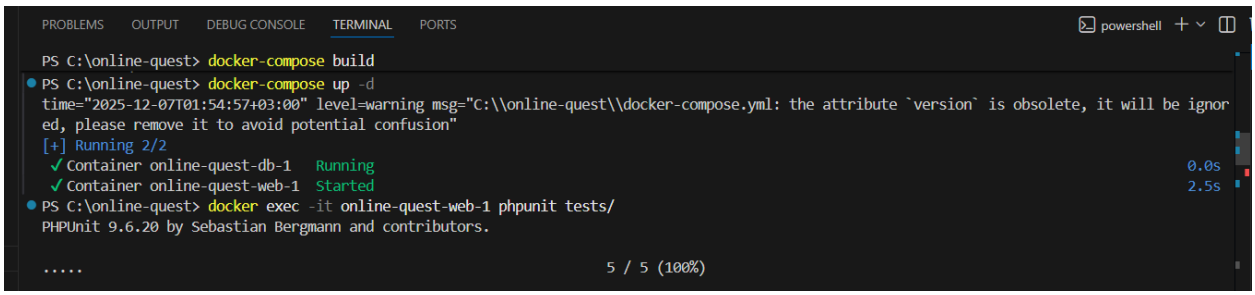

Рисунок 13 – Пройденные тесты

3.2.2 Тестирование методом «чёрного ящика»

Тестирование методом «чёрного ящика» было проведено вручную без знания внутренней структуры кода. Цель — проверить соответствие системы заявленному функционалу с точки зрения конечного пользователя.

Были протестированы следующие сценарии:

- Запуск приложения — при открытии главной страницы отображается список доступных квестов;
- Начало прохождения — при выборе квеста создаётся сессия, отображается первый шаг;
- Переход между шагами — при нажатии на кнопку выбора система корректно переходит к следующему шагу;
- Достижение финала — по завершению сюжета отображается надпись «КОНЕЦ»;
- Сброс сессии — кнопка «Начать заново» очищает прогресс и возвращает на главную страницу;
- Адаптивность интерфейса — на мобильных устройствах кнопки занимают всю ширину экрана и легко нажимаются.

Все сценарии выполнены успешно. Интерфейс интуитивно понятен, ошибок при некорректных действиях (например, повторное нажатие на устаревшую ссылку) не возникает — система корректно возвращает пользователя к актуальному шагу.

На Рисунке 14 представлена главная страница с выбором квестов.

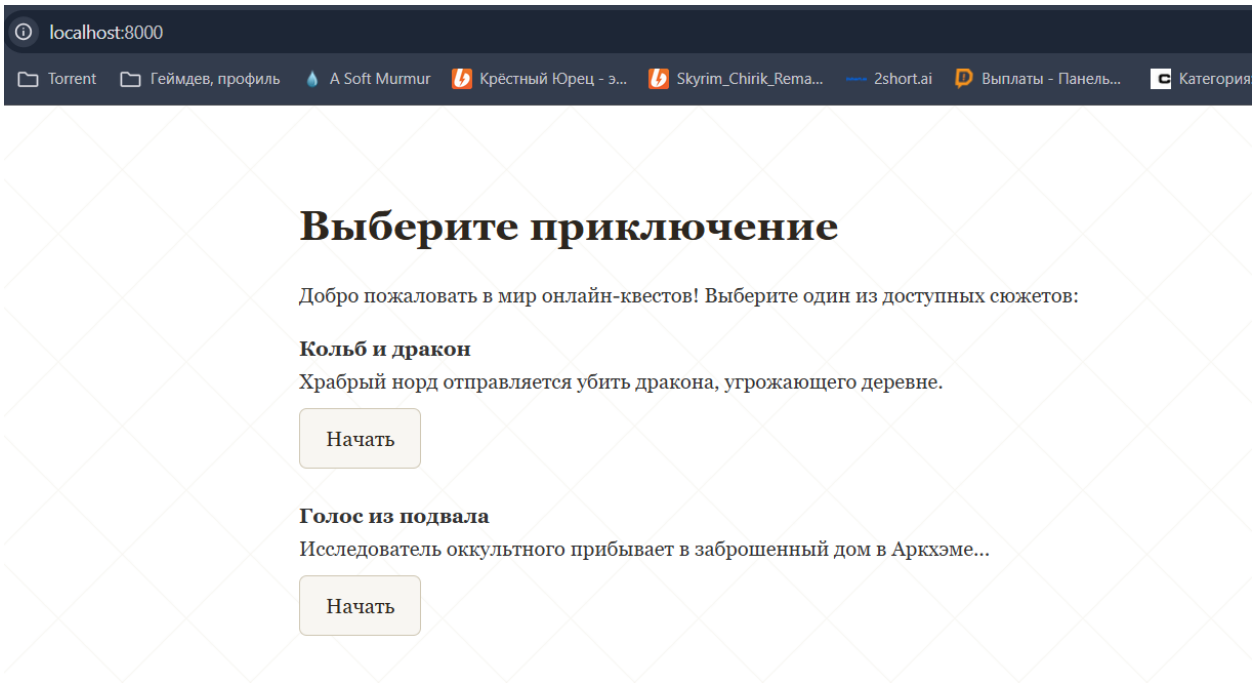

Рисунок 14 – Главная страница приложения

На Рисунке 15 показан игровой шаг с кнопками выбора.

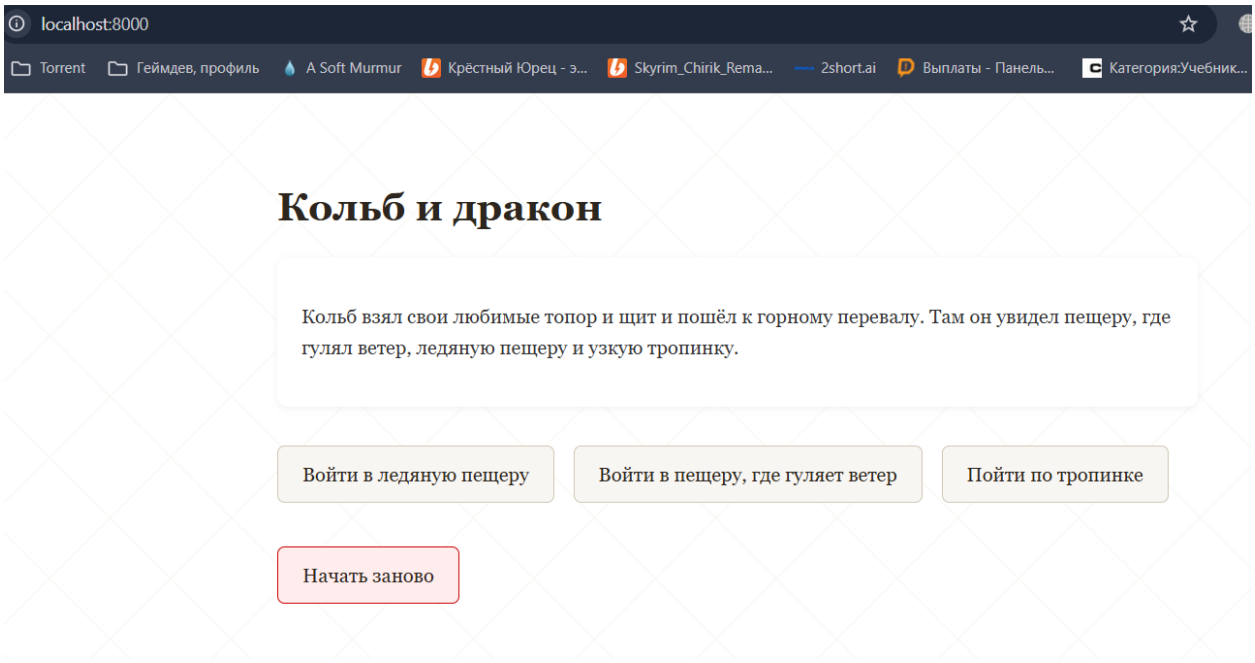

Рисунок 15 – Игровой шаг квеста

На Рисунке 16 отображён финальный экран с надписью «КОНЕЦ».

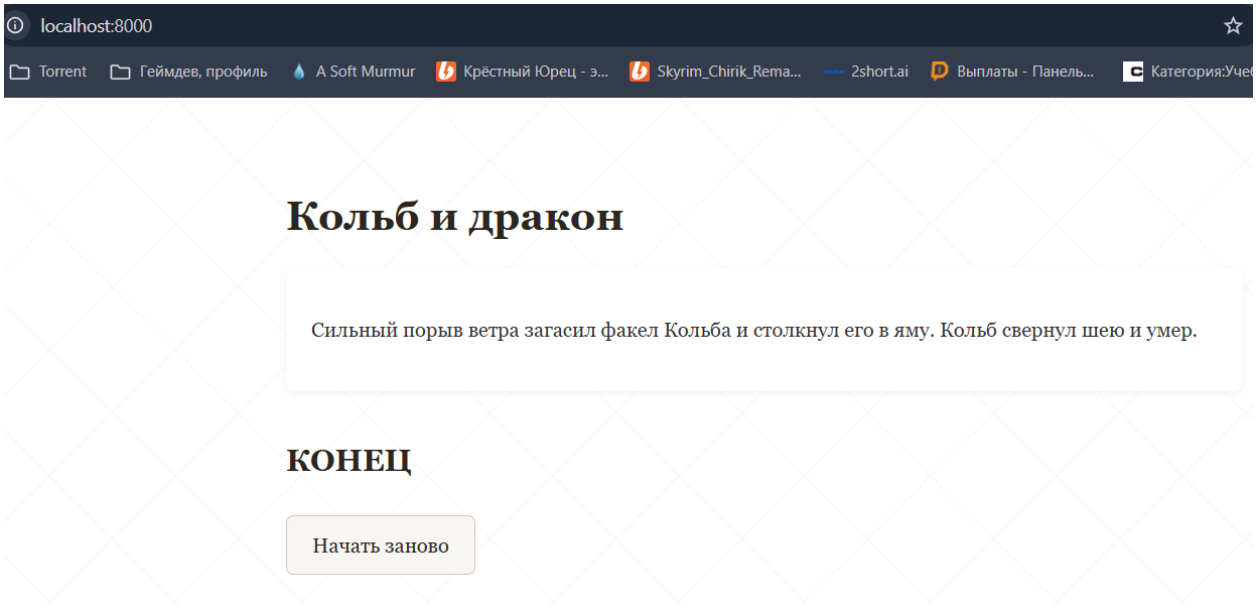

Рисунок 16 – Завершение прохождения квеста

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана серверная часть веб-приложения «Онлайн-квест», предназначенного для интерактивного прохождения текстовых сюжетов через браузер. Проведён анализ предметной области, изучены существующие решения (Choice of Games, TextQuest.ru, Inklewriter) и сформулированы требования к системе. На основе анализа была выбрана архитектурная модель MVC, а также технологический стек, включающий PHP, PostgreSQL, PDO и Docker.

Разработана архитектура приложения, включающая:

- модели данных (quests, steps, choices, sessions);
- контроллер, обрабатывающий выбор квеста, переходы между шагами и сброс сессии;
- представления, реализующие минималистичный, но адаптивный пользовательский интерфейс.

Создана реляционная модель данных, обеспечивающая хранение нескольких независимых квестов, описание шагов с текстом и флагом завершения, а также связи между шагами через варианты выбора. Реализован механизм сохранения прогресса без регистрации — через HTTP-куки и таблицу sessions.

Построены диаграммы:

- вариантов использования;
- компонентов;
- ER-диаграмма модели данных;
- две диаграммы последовательностей (начало квеста и совершение выбора).

Серверная часть была протестирована двумя методами:

- модульное тестирование с использованием PHPUnit — подтверждена корректность работы моделей и логики сессий;
- тестирование методом «чёрного ящика» — проверены все пользовательские сценарии: выбор квеста, прохождение, завершение, сброс.

Все ключевые функции подтверждены корректной работой. Приложение стабильно обрабатывает запросы, обеспечивает целостность данных и удобный интерфейс как на десктопе, так и на мобильных устройствах.

Таким образом, цель курсовой работы достигнута: разработана и протестирована серверная часть веб-приложения для прохождения интерактивных текстовых квестов. Продукт может служить основой для образовательных платформ, игровых проектов или систем интерактивного повествования, а также легко расширяется: можно добавить авторизацию, статистику прохождений, переменные персонажа или поддержку условных переходов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Куликов, А. А. Разработка интернет ресурсов : учебное пособие / А. А. Куликов, А. А. Русляков. — Москва : РТУ МИРЭА, 2023. — 306 с. — ISBN 978-5-7339-2047-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/398264 (дата обращения: 16.09.2025). — Режим доступа: для авториз. пользователей.

2. Волков, М. Ю. Разработка серверных частей интернет-ресурсов : учебное пособие / М. Ю. Волков, В. В. Литвинов, А. А. Лобанов. — Москва : РТУ МИРЭА, 2021. — 188 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/218420 (дата обращения: 16.09.2025). — Режим доступа: для авториз. пользователей.

3. Баланов, А. Н. Бэкенд-разработка веб-приложений: архитектура, проектирование и управление проектами : учебное пособие для вузов / А. Н. Баланов. — 2-е изд., стер. — Санкт-Петербург : Лань, 2025. — 312 с. — ISBN 978-5-507-52472-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/451820 (дата обращения: 16.09.2025). — Режим доступа: для авториз. пользователей.

ПРИЛОЖЕНИЕ

Репозиторий разрабатываемого прототипа веб приложения находится по адресу: https://github.com/GodfatherYurez/Kursovaja