МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МО и ПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Интерактивный лист персонажа для настольной игры

Студент _____ Ендуткин Вячеслав Алексеевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Кольчугина Е.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«__» _____2024 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Ендуткин Вячеслав Алексеевич гр. 21ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Интерактивный лист персонажа для настольной игры

Тема утверждена приказом ПГУ №_____от "____"_____2024 г.

3. Руководитель работы профессор кафедры «МО и ПЭВМ» Кольчугина Е.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать приложение создания персонажа для настольных ролевых игр с возможностью хранения, редактирования и переноса.

2) Среда и средства разработки:

Языковые средства разработки: C#, JavaScript, UML.

Инструментальные средства: Visual Studio 2022, Visual Paradigm,

Технологии разработки: ООП, шаблоны проектирования, Web API сервисно-ориентированный подход.

3) Среда функционирования: Браузеры

4) Основные функции приложения:

а) создать персонажа;

б) просмотреть информацию о персонаже;

в) редактировать информацию о персонаже;

г) сохранять информацию о персонаже отдельный в файл;

д) добавлять персонажа из существующего файла.

5) Особенности: возможность локального хранения данных о персонаже, адаптация под мобильные устройства, локализация

6) Требования к СУБД и модели данных: гибкость, масштабируемость, быстрота чтения/записи

7) Провести тестирование с использованием браузеров: Google Chrome, Opera, Microsoft Edge

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и постановка задачи;*
2) *Проектирование;*
3) *Реализация;*
4) *Контроль качества программного обеспечения;*
5) *План разработки бюджета;*
6) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	13.04.2025 – 27.04.2025	
2	Проектирование	25%	27.04.2025 – 10.05.2025	
3	Реализация	35%	10.05.2025 – 30.06.2025	
4	Контроль качества программного обеспечения	10%	31.05.2025 – 02.06.2025	
5	План разработки бюджета	5%	02.06.2025 – 03.06.2025	
6	Оформление пояснительной записки	15%	03.06.2025 – 14.06.2025	

Дата выдачи задания «22» ____декабря____2024 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 142 листа, 50 рисунков, 12 таблиц, 20 использованных источников, 4 приложения.

ВЕБ-ПРИЛОЖЕНИЕ, ЛИСТ ПЕРСОНАЖА, НАСТОЛЬНАЯ ИГРА, C#, ASP-NET, UML, БАЗА ДАННЫХ, MICROSOFT VISUAL STUDIO, MONGODB

Объектом разработки является средство хранения редактирования и переноса между устройствами листа персонажа для настольной игры.

Цель выпускной квалификационной работы – разработка приложения листа персонажа для настольной игры, которая позволить улучшить игровой опыт.

Технология разработки – разработка веб-приложения на языке программирования C# с использованием кроссплатформенного фреймворка ASP.NET Core, клиентская часть была разработана на языке JavaScript, база данных была создана в MongoDB. Среда разработки – Microsoft Visual Studio 2022.

В результате выпускной квалификационно работы были разработаны клиентская и серверная часть приложения интерактивного листа персонажа для настольной игры.

Содержание

Введение 7

1 Анализ альтернативных реализаций цифровых листов персонажей для настольных игр 9

1.1 Анализ предметной области и постановка задачи 9

1.2 Анализ аналогов 10

1.2.1 Анализ программы Aurora Builder 10

1.2.2 Анализ интерактивного листа персонажа Roll20 11

1.2.3 Анализ конструктора персонажей D&D Beyond 12

1.2.4 Сравнительный анализ аналогичных систем 13

2 Выбор методов и средств разработки 15

3. Проектирование веб-приложения интерактивного листа персонажа для настольной игры 19

3.1 Архитектура приложения 19

3.2 Проектирование структуры данных 20

3.2.1 Концептуальная модель базы данных 20

3.2.2 Логическая модель базы данных 21

3.3 Проектирование пользовательского интерфейса 22

3.4 Проектирование программных средств 26

3.4.1 Проектирование классов серверной части приложения 26

3.4.2 Проектирование алгоритма создания листа персонажа 28

3.4.3 Проектирование алгоритма добавления заклинания 29

3.4.4 Проектирование алгоритма импорта листа персонажа 30

4 Реализация веб-приложения интерактивного листа персонажа для настольной игры 32

4.1 Реализация базы данных 32
4.2 Реализация серверной части приложения 34
4.3 Реализация клиентской части приложения 37
5 Отладка и тестирование 41
5.1 Выбор режима тестирования 41
5.2 Функциональное тестирование 42
5.3 Тестирование серверной части приложения 46
5.4 Тестирование клиентской части приложения 47
6 Контроль качества 49
6.1 Метрики Холстеда 49
6.2 Метрики Чепина 51
6.3 Метрики SLOC 52
7 Расчет бюджета и разработки 53
Заключение 56
Список использованных источников 57
Приложение А Листинг документа MongoDB 59
Приложение Б Листинг серверной части приложения 62
Приложение В Листинг клиентской части приложения 77
Приложение Г Результаты тестирования 134

Введение

В последние годы большую популярность среди молодого поколения набирают настольные ролевые игры (НРИ) наподобие Dungeons & Dragons, Parminder’s и другие. Подобные жанр настольных игр отличается от большинства других довольно весомым объемом правил, помещающимся в большие книги, а также разнообразными ситуациями и приключениями, в которые могут погружаться игроки.

Неотъемлемой частью НРИ является такой элемент, как лист персонажа, в котором содержится всеобъемлющая информация об персонаже игрока, необходимая для проведения игры. Как правило это один или несколько листов А4, с распечатанными на них шаблонными листами, которые предоставляются разработчиками игр. Подобный способ хранения прост, но сопровождается с различными рисками и неудобствами свойственные всем физическим носителям данных.

Все выше сказанное дает понять, что существует необходимость в реализации цифрового варианта листа персонажа, который позволит нивелировать недостатки бумажного варианта, позволяя хранить, редактировать и переносить персонажа на разные устройства.

Целью выпускной квалификационной работы является разработка приложения интерактивного листа персонажа для настольной игры. Разработка включает в себя анализ требований, проектирование, реализацию программного обеспечения и его тестирование.

Для выполнения выпускной квалификационной работы были поставлены следующие цели:

– изучить аналогичные программные продукты и предоставить сравнительный анализ;

– выполнить проектирование архитектуры приложения, пользовательского интерфейса и структуры данных разрабатываемого программного обеспечения;

– выполнить разработку программного обеспечения;

– провести тестирование разработанного приложения для оценки качества полученного программного продукта;

– оформить пояснительную записку в соответствии с требованиями;

– выполнить планирование разработки и оценку бюджета.

1 Анализ альтернативных реализаций цифровых листов персонажей для настольных игр

1.1 Анализ предметной области и постановка задачи

Лист персонажа – это документ, используемы в настольных ролевых играх, который содержит всю ключевую информацию о персонаже игрока: его характеристики, навыки, инвентарь, способности, историю и другие параметры. Он служит инструментом для отслеживания прогресса героя, упрощения игрового процесса и взаимодействия с правилами игры.

Традиционные бумажные листы персонажей имеют ряд недостатков. Во-первых, они подвержены износу, что приводит к их потере, повреждению и утрате игрового прогресса. Во-вторых, ручное заполнение и исправление занимает много времени и повышает риск ошибок, что особенно актуально для неопытных игроков. В-третьих, они ограничены в объеме информации, который помещается на лист, что может стать неудобным в некоторых случаях.

В зависимости от того, к какой ролевой системе принадлежит отдельный лист персонажа, меняется и его содержание в виду индивидуальных особенностей правил в различных играх подобного жанра. Наибольшую популярность среди всех НРИ имеет Dungeons & Dragons(D&D) от компании Wizard of the Coast, чей формат листа персонажа ляжет в основу разрабатываемого приложения [1].

Категорией пользователей интерактивного листа персонажей являются игроки в настольные ролевые игры, ведущие данных игр, а также разработчики игровых правил и систем.

Приложение должно решать ключевые проблемы бумажных аналогов:

– обеспечивать удобное создание и редактирование персонажа;

– автоматизировать расчёты (броски, модификаторы, уровень здоровья);

– предоставлять удобное хранение, импорт и экспорт данных о персонаже;

Таким образом, требуется разработать кроссплатформенное приложение, которое имеет серверную и клиентскую части, а также базу данных.

1.2 Анализ аналогов

1.2.1 Анализ программы Aurora Builder

Первым на рассмотрение среди аналогов стал Aurora Builder. Данное решение является десктопным, то есть поддерживается на персональных компьютерах и ноутбуках [2]. На рисунке 1 приведено окно интерфейса данного программного обеспечения.

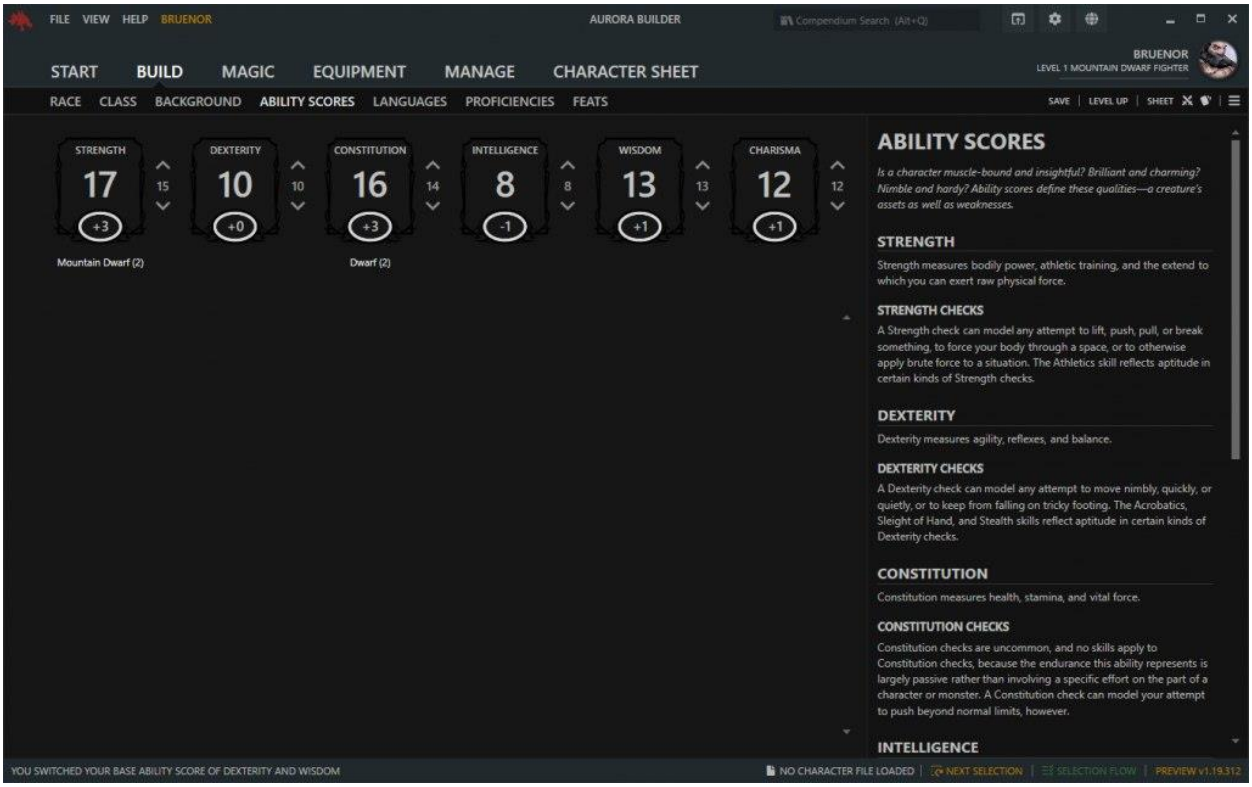

Рисунок 1 – Окно программы Aurora Builder

Основным достоинствами данной программы являются: поддержка множества НРИ систем, таких как (D&D, Pathfinder, Savage Worlds). Программа динамически пересчитывает характеристики, модификаторы,

уровни и другие игровые показатели. Предоставляет гибкий инструмент по созданию персонажа. Предоставляет возможность экспорта персонажей в различные форматы.

Недостатком данной программы является отсутствие русской локализации что может стать барьером для множества пользователей. Помимо это программа не поддерживается на мобильных устройствах.

1.2.2 Анализ интерактивного листа персонажа Roll20

Roll20 – это полноценный сервис, предоставляющий проводить игровые сессии в онлайн формате. В свою очередь интерактивный лист персонажа интегрирован в самом сайте [3]. Интерфейс листа персонажа представлен на рисунке 2.

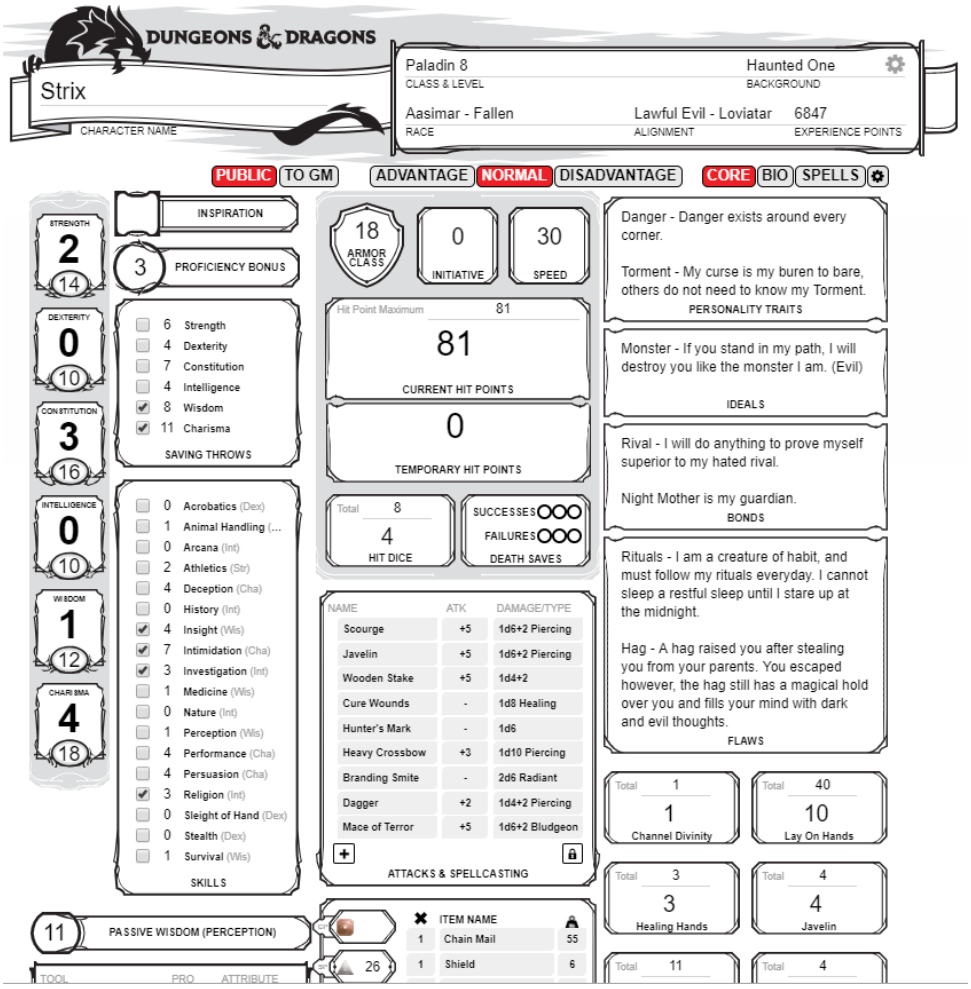

Рисунок 2 – Интерфейс интерактивного листа персонажа Roll20

Главное преимущество данного интерактивного листа персонажа является онлайн формат, что позволяет иметь доступ к нему в любом месте с доступом к интернету. Автоматизация расчетов навыков, модификаторов, бросков, что упрощает игровой процесс. Возможность печати листа персонажа и экспорта в формат PDF.

Недостатком является обязательно подключение к интернету, без которого доступ к листу персонажа получить невозможно. Еще к проблемам данного аналога можно отнести ограничения функционала в бесплатной версии, так как сервис работает по подписке. Также сам сайт данного сервиса потребляет много ресурсов во время работы, что может быть проблемой на слабых устройствах.

1.2.3 Анализ конструктора персонажей D&D Beyond

D&D Beyond – это официальная цифровая платформа, разработанная для игры Dungeons & Dragons. Она предоставляет множество инструментов для создания и управления персонажами, а также для проведения игровых сессий [4]. На рисунке 3 приведен интерфейс листа персонажа D&D Beyond.

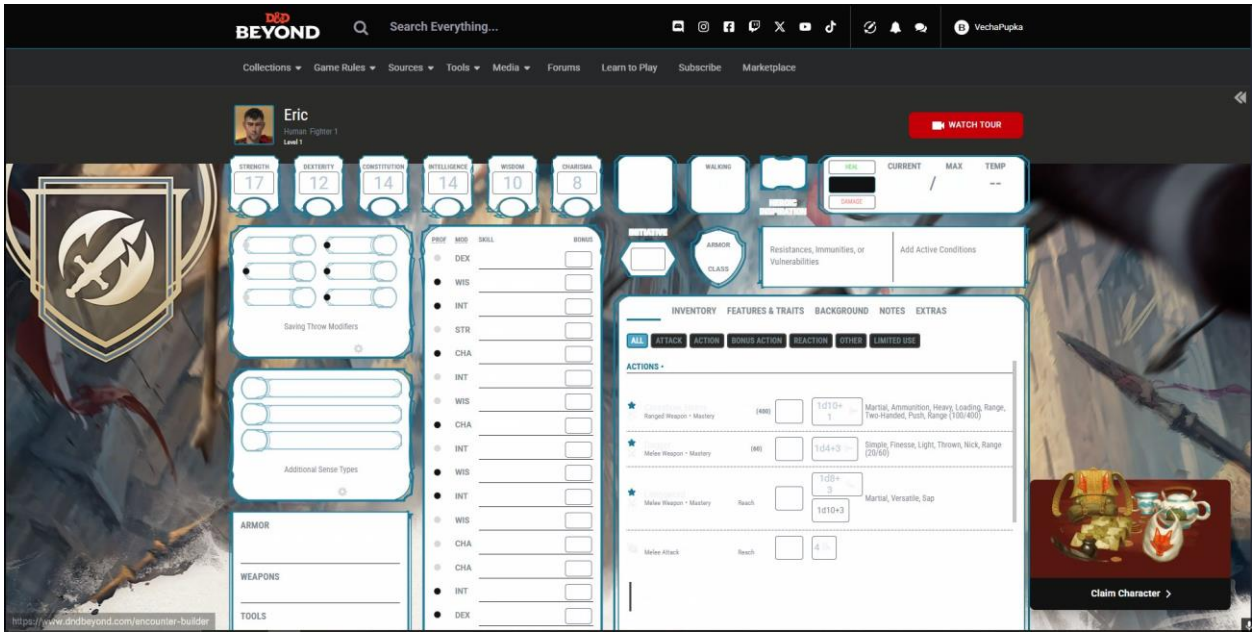

Рисунок 3 – Интерфейс интерактивного листа персонажа D&D Beyond

Основные достоинства данного листа персонажей удобство и возможность доступа из любых устройств с подключенным интернетом. Данный лист персонажа интегрирован с самой системой D&D Beyond что делает игровой процесс более плавным и удобным. Также есть возможность создания шаблонных персонажей, что особенно полезно для новых игроков.

К недостаткам стоит отнести необходимость подключения к интернету, что может помешать игровому процессу в местах с плохим соединением. На сайте отсутствует русская локализация, что также может негативно сказаться на игривом опыте. Наличие платной подписки, без которой пользователь ограничен в числе персонажей, которых он может создать.

1.2.4 Сравнительный анализ аналогичных систем

Был выполнен качественное сравнение рассмотренных ранее аналогичных программных средств.

Auroral Builder – отличное автономное решение. В отличии от браузерных аналогов позволяет работать оффлайн. Данное программное обеспечение также поддерживает домашние правила и модификации механик, что особенно удобно для опытных игроков и мастеров разрабатывающих уникальные сеттинги. Детализированные автоматические расчеты упрощают взаимодействие с механикой игры для новых игроков, которые могут упустить различные мелкие тонкости в правилах игры. Возможность экспорта листа в формат позволяет использовать его для чтения на других устройствах или для печати, но в данном случае внесение изменений в лист невозможно без наличия самой программы под рукой. К неудобствам следует отнести отсутствие русского языка, что для не владеющих английским может привести к трате времени на перевод вместо проведения непосредственно игры. Главный и весомый недостаток, это наличие лишь десктопной версии программы, что делает её недоступной для пользователей без персональных компьютеров и ноутбуков.

Roll20 – онлайн платформа для проведения игр по D&D, чей лист персонажа напрямую интегрирован в функционал онлайн стола. Данное решение отлично подходит для пользователей, предпочитающих онлайн формат для проведения игр. Возможности данной платформы позволяют легко автоматизировать игровой процесс, вроде броска кубов проверки характеристик или применения заклинаний. Дизайн листа персонажа повторяет оригинальный лист, предоставляемый для печати компанией Wizard of the Coast. Однако данный архаичный дизайн может быть сложен и неудобен в цифровом формате, особенно для новых игроков. Низка производительность сайта на слабом железе с медленным интернетом может стать проблемой для комфортной игры.

D&D Beyond – лицензионная цифровая платформа для D&D. Будучи разработанной при поддержке самой Wizard of the Coast, предоставляет весь официальный контент, дополнения и обновления внутри сайта. Наличие удобного интерфейса является несомненным преимуществом, вся необходимая информация умещается на экране, а сам сайт имеет версию для мобильных устройств. Но отсутствие русского языка на сайте будет неудобством для некоторых пользователей. На сайте можно создавать шаблонных персонажей, что экономит время пользователя. D&D Beyond предоставляет платный контент, без которого некоторые возможности листа персонажа становятся ограниченными. Например, для создания персонажа с расой, описанной в книге с дополнительными правилами, требуется покупка цифровой версии данной книги. При этом не важно имеется ли уже у вас физическая копия книги или нет. И конечно сама возможность оформления покупок невозможна в виду ограничений на иностранные платежные системы.

Наиболее удачное решение интерактивного листа персонажа, из вышеописанных, стоит отнести к D&D Beyond. Удобный интерфейс, поддержка различных устройств и хорошая производительность делают данную систему наиболее выигрышной на фоне других.

1.3 Выбор методов и средств разработки

Изучив аналогичные программные решения, были сформированные следующие функциональные требования к разрабатываемому приложение:

– обеспечения создания и хранения персонажей;

– просмотр списка персонажей;

– просмотр и редактирование листа персонажа;

– предоставление автоматических расчетов в соответствии с правилами и механиками игры;

– возможность экспорта листа персонажа из приложения в отдельный файл;

– возможность импорта персонажа из файла в приложение;

Исходя из описанных выше функциональных требований была разработана UML диаграмма вариантов использования, которая представлена на рисунке 4.

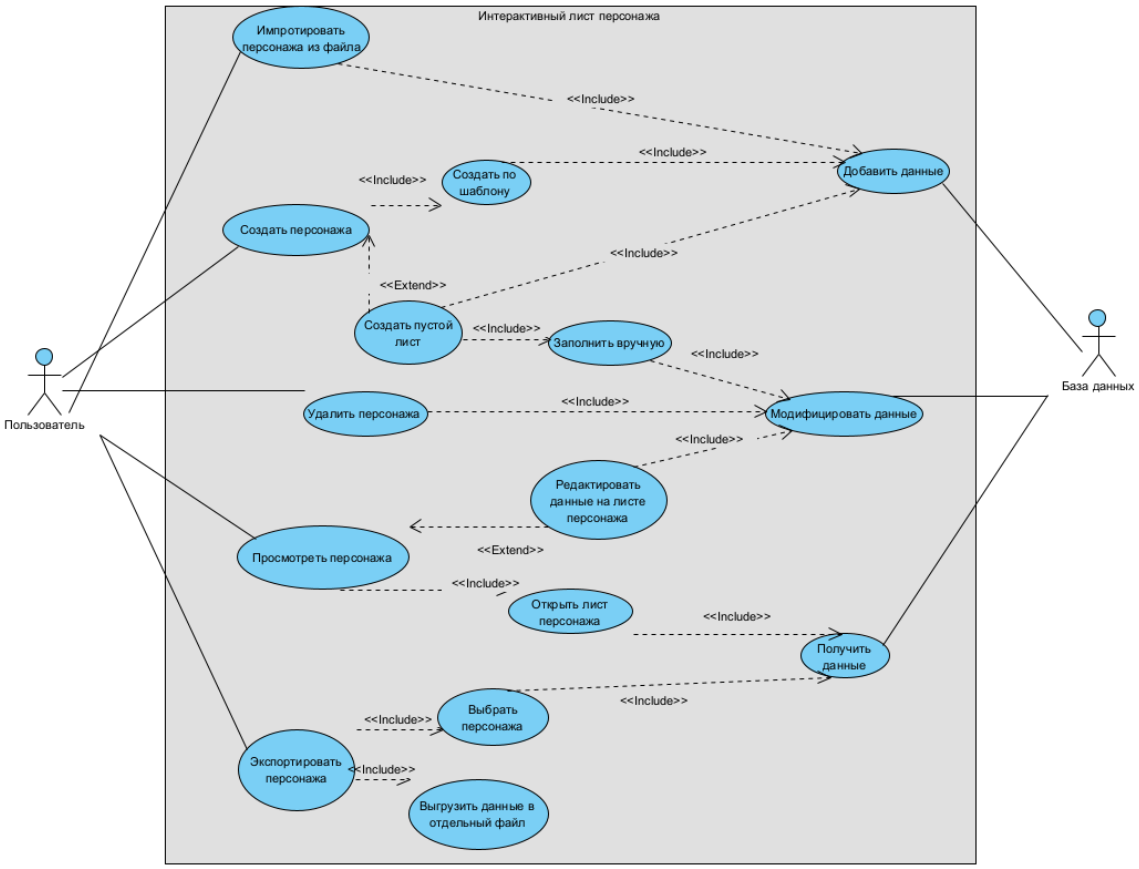

Рисунок 4 – Диаграмма вариантов использования

Для более подробной документации и визуализации, а также для более четкого понимания функциональных требований разрабатываемого приложения, были описаны сценарии основных вариантов использования, изображенных на данной диаграмме [5].

На рисунке 5 представлен сценарий для варианта использования «Создать персонажа».

Прецедент: Создать персонажа
ID: 2
Краткое описание: Пользователь создает новый лист персонажа
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение.
Основной поток: 1. Если пользователь создает пустой лист персонажа 1.1 Открывается пустой лист с полями для заполнения 2. Иначе, если пользователь создает персонажа по шаблону 2.1 Приложение предлагает выбрать шаблонные данные о персонаже. 2.2 Открывается лист персонажа с уже заполненными полями
Постусловия: Новый лист персонажа сохраняется в базе данных
Альтернативные потоки: Нет

Рисунок 5 – Сценарий варианта использования «Создать персонажа»

На рисунке 6 представлен сценарий варианта использования «Редактировать данные на листе персонажа».

Прецедент: Редактировать данные на листе персонажа
ID: 11
Краткое описание: Пользователь редактирует открытый лист персонажа
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловие: 1. Пользователь открыл приложение. 2. Пользователь открыл нужный лист персонажа
Основной поток: 1. Пользователь внес изменения на листе персонажа. 2. Пользователь произвел нажатие по кнопке сохранить.
Постусловия: Внесенные изменения сохраняются в базе данных
Альтернативные потоки: Нет

Рисунок 6 – Сценарий варианта использования «Редактировать данные на листе персонажа»

На рисунке 7 представлен сценарий варианта использования «Экспортировать персонажа».

Прецедент: Экспортировать персонажа
ID: 5
Краткое описание: Пользователь экспортирует данные о персонаже в отдельный файл
Главные актеры: Пользователь
Второстепенные актеры:
Предусловие: 1. Пользователь открыл приложение. 2. Пользователь открыл лист персонажа.
Основной поток: 1. Пользователь нажал на кнопку экспортировать персонажа
Постусловия: Данные о персонаже сохраняются в отдельном файле
Альтернативные потоки: Нет

Рисунок 7 – Сценарий варианта использования «Экспортировать персонажа»

В ходе анализа функциональных требований к разрабатываемому приложению была создана диаграмма вариантов использования, а также выделены основные её элементы и их спецификации, которые отражают ход взаимодействия пользователя с приложением.

Нефункциональные требования описывают такие характеристики системы, которые не связаны с её функционалом. В ходе разработки интерактивного листа персонажа были определены следующие нефункциональные требования:

– доступность – система должна быть доступна с минимальным временем простоя;

– производительность – система должна обеспечивать быстрое время отклика;

– масштабируемость – система должна поддерживать увеличение количества данных без значительной потери производительности;

– расширяемость – архитектура системы должна быть гибкой, чтобы позволять добавление новых функций;

– удобство и простота использования – интерфейс системы должен быть интуитивно понятен и прост в использовании.

Исходя из выявленных требований приложение будет представлять собой клиент-серверное приложение с использованием кроссплатформенного фреймворка ASP.NET Core. Данный фреймворк является платформой для разработки веб-приложений с открытым кодом разработанной компанией Microsoft. Таким образом серверная часть приложения будет написана на языке C#. Клиентская часть приложения будет написана на HTML, CSS и JavaScript [6].

В качестве базы данных будет выбрана MongoDB. Данная NoSQL база данных предоставляет множество преимуществ, таких как: гибкость системы, легкая масштабируемость, поддержка больших объемов данных и производительность.

В качестве среды разработки всего проекта будет использоваться Microsoft Visual Studio 2022. Клиентское приложение должно успешность функционировать в браузерах: Google Chrome, Opera, Firefox.

2. Проектирование веб-приложения интерактивного листа персонажа для настольной игры

2.1 Архитектура приложения

Разрабатываемый интерактивный лист персонажа имеет многослойную клиент-серверную архитектуру, которая отображена на рисунке 8.

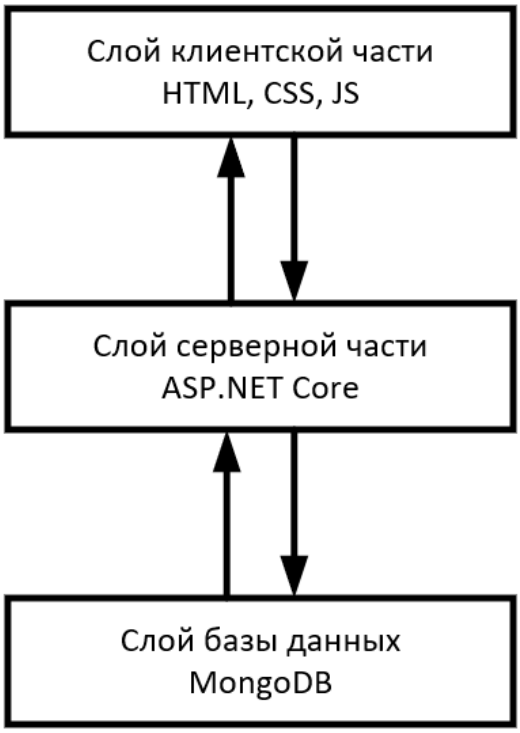

Рисунок 8 – Архитектура приложения

В данной многослойно архитектуре имеются следующие слои:

– слой клиентской части;
– слой серверной части;
– слой базы данных.

Слой клиентской части содержит все с чем будет взаимодействовать пользователь, обрабатывая его действия и отображая данные. Клиентская часть взаимодействует с серверной выполняя отправку API запросов, затем получает обработанный сервером ответ. Серверная часть, реализованная на RESTful API, в зависимости от полученного клиентской частью запроса,

контроллеры обрабатывают их и следом взаимодействуют с базой данных сохраняя в неё данные, обновляя их или получая данные из БД. Данные между клиентом и сервером передаются в формате JSON [7].

Слой базы данных использует MongoDB, которая является документно-ориентированной NoSQL базой данных. Доступ к данный осуществляется через официальный .NET драйвер MongoDB, который предоставляет LINQ-подобный API для выполнения запросов к базе данных.

2.2 Проектирование структуры данных

2.2.1 Концептуальная модель базы данных

В результате анализа предметной области была разработана концептуальная модель базы данных, которая описывает основные сущности я связи между ними без привязки к конкретной СУБД. На данной диаграмме описывается лишь то, какие данные будут храниться в базе данных, без определения технических деталей, таких как типы полей, индексы и прочее.

На рисунке 9 представлена концептуальная модель базы данных для интерактивного листа персонажа.

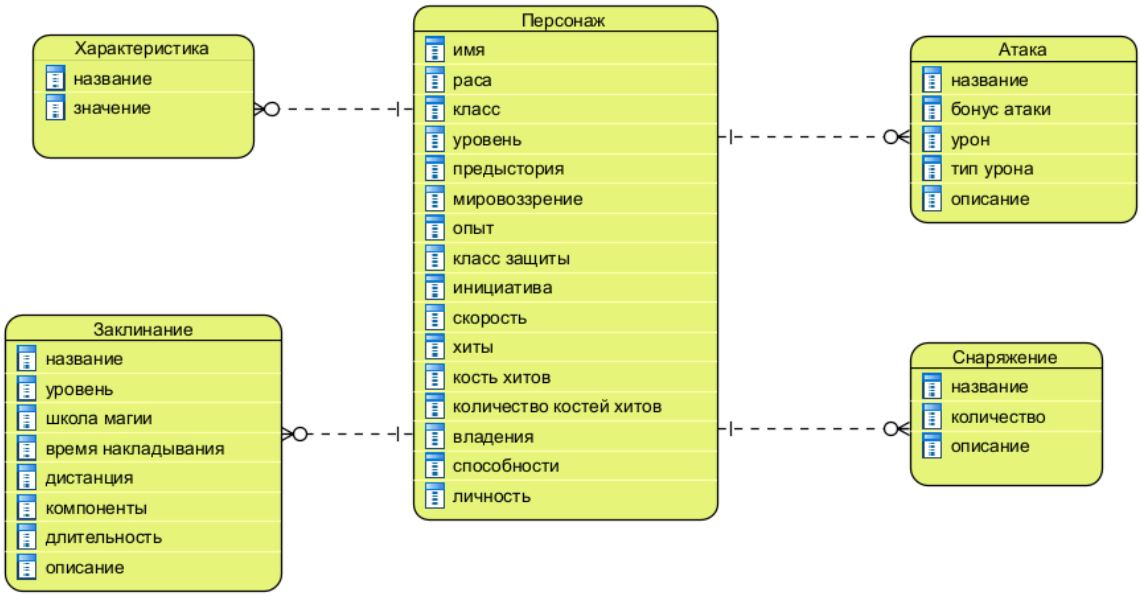

Рисунок 9 – Концептуальная модель базы данных

Описание сущностей, полученных в результате проектирования данной диаграммы, представлено в таблице 1.

Таблица 1 – Описание сущностей

Название сущности	Идентификатор сущности	Назначение сущности
Персонаж	Character	Содержит основную информацию о персонаже
Характеристика	Abilities	Содержит значение одной из шести основных характеристик персонажа
Атака	Attacks	Содержит информацию о доступных персонажу атаках
Заклинание	Spells	Содержит сведения об известных персонажу заклинаниях
Снаряжение	Equipment	Содержит информацию об имеющемся у персонажа снаряжении

Концептуальная модель была построена в среде Visual Paradigm в нотации ERD (Entity Relationship Diagram) [8].

2.2.2 Логическая модель базы данных

На основе полученной ранее диаграммы была разработана логическая модель данных, которая дополняет концептуальную модель детализируя описание структуры данных. Диаграмма логической модели базы данных для интерактивного листа персонажа представлена на рисунке 10.

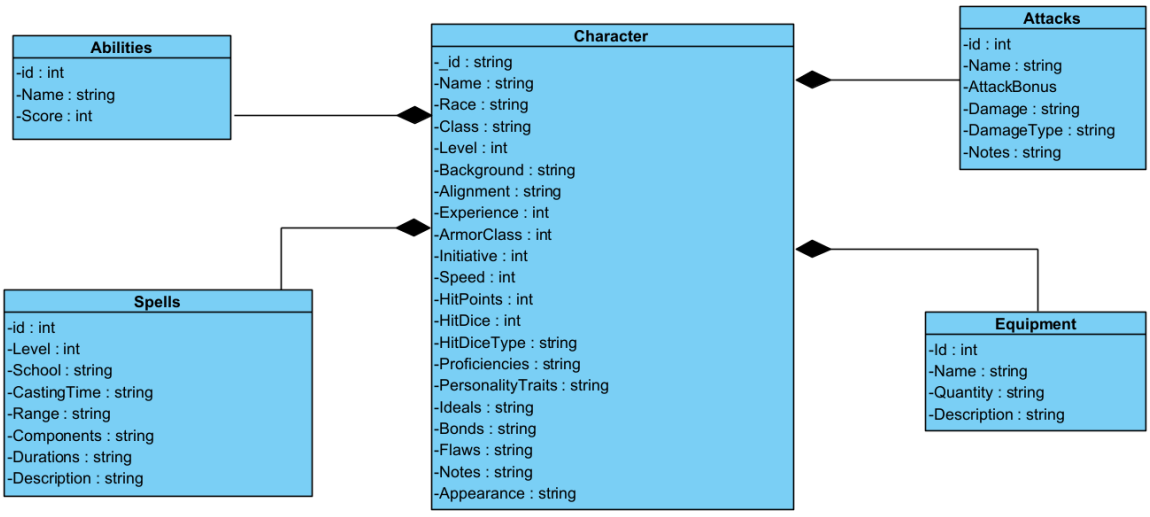

Рисунок 10 – Логическая модель базы данных

В виду того, что при разработке применятся не реляционная база данных, для разработки диаграммы концептуальной модели была использована среда Visual Paradigm в нотации UML Class Diagram [9].

2.3 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это визуальная и интерактивная часть приложения, которая позволяет пользователю взаимодействовать с программным обеспечением. Основные требования, предъявляемые к графическому интерфейсу это простота, удобство и единство стиля всех компонентов [10].

При проектирование пользовательского интерфейса были применены функциональные требования, которые были выявлены на этапе разработки диаграммы вариантов использования.

На рисунке 11 представлен макет стартовой страницы со списком персонажей.

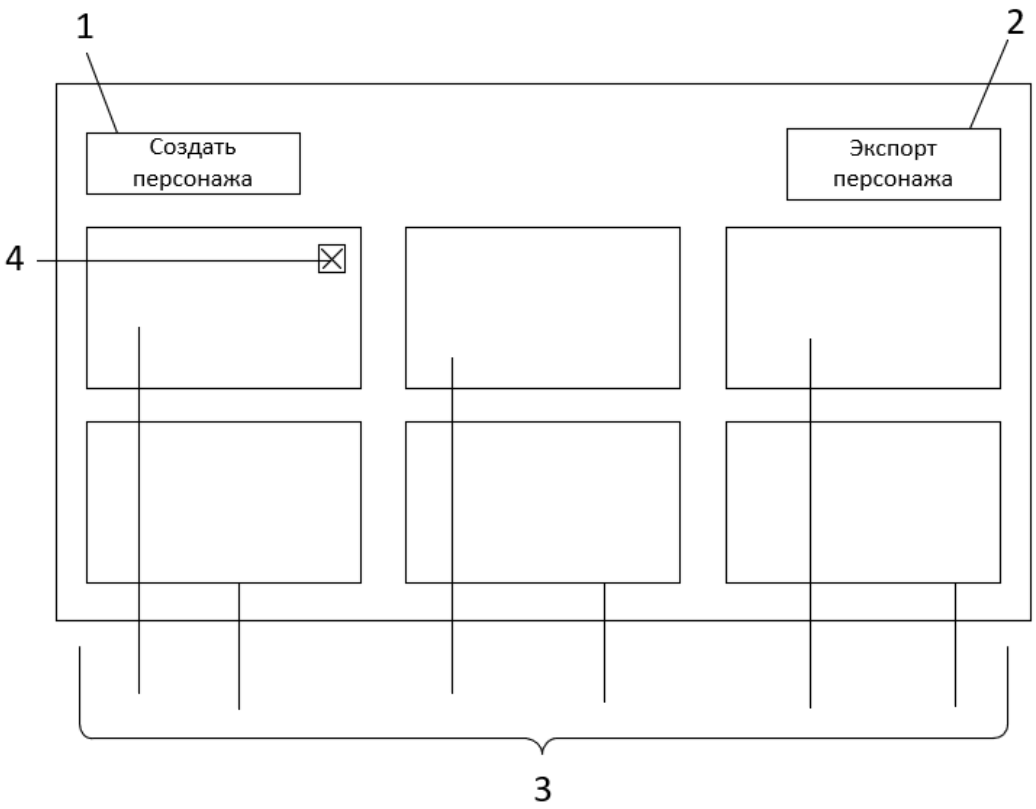

Рисунок 11 – Макет страницы со списком персонажей

На данной странице присутствуют следующие элементы:

1 – кнопка открытия модального окна для создания нового персонажа;

2 – кнопка импорта нового персонажа из файла;

3 – блок карточки существующего карточки персонажа;

4 – кнопка удаления персонажа.

При открытии кнопки «Создать персонажа» будет открываться модальное окно, в котором пользователю будет предложено создать пустой лист персонажа или лист, заполненный по шаблону с выбором пользователем расы и класса.

При нажатии кнопки «Импорт персонажа» будет открыт файловый менеджер, в котором можно открыть соответствующий JSON файл.

При нажатии по блоку карточки персонажа, будет открыта страница листа того персонажа, по которому было выполнено нажатие.

На рисунке 12 представлен макет страницы листа персонажа.

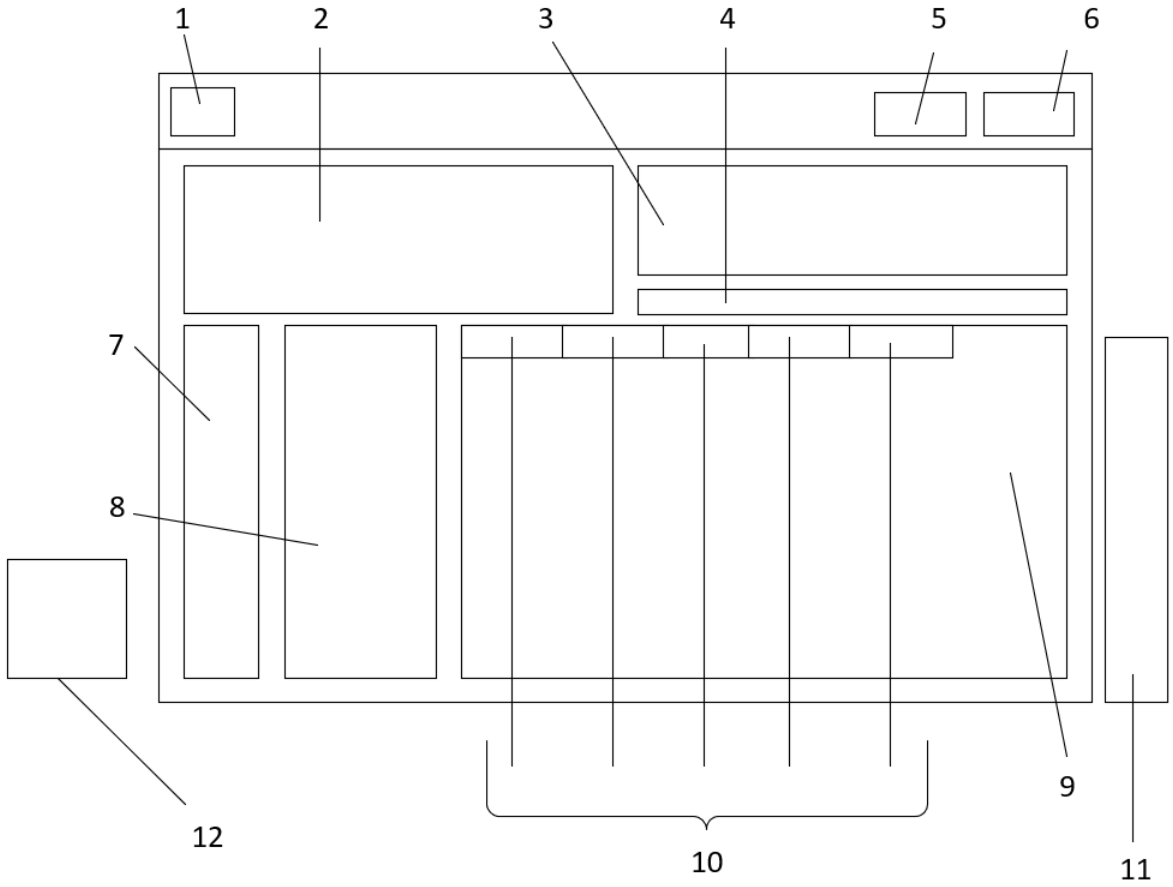

Рисунок 12 – Макет страницы листа персонажа

На странице листа персонажа присутствуют следующие элементы:

1 – кнопка «назад», которая возвращает на страницу списка персонажей;

2 – блок с основной информацией о персонаже;

3 – блок с боевыми параметрами персонажа;

4 – шкала прогресса персонажа;

5 – кнопка для экспорта персонажа в JSON файл;

6 – кнопка для сохранения внесенных изменений;

7 – блок с основными характеристиками персонажа;

8 – блок с навыками персонажа;

9 – блок вкладок;

10 – кнопки для переключения вкладок;

11 – блок кнопок для броска костей;

12 – блок, отображающий историю результатов бросков.

Блок с основной информацией содержит следующие поля: имя, класс, раса, происхождение, мировоззрение, опыт и уровень. Все поля редактируемые, кроме поля «Уровень», его значение будет высчитываться автоматически от количества опыта.

Шкала прогресса будет обновляться при изменении значения поля «Опыт» и отображать необходимое количество опыта до следующего уровня.

В блоке вкладок можно будет переключаться между вкладками: атаки, заклинания, снаряжение, персональная информация и заметки.

Блок кнопок будет состоять из шести кнопок, которые позволят бросать четырёхгранную, шестигранную, восьмигранную, десятигранную, двенадцатигранную и двадцатигранную кости.

В блоке с историей бросков будет располагаться информация о том какая кость была брошена, результат и время броска.

Для отображения навигации между элементами интерфейса была разработана диаграмма состояний, которая представлена на рисунке 13.

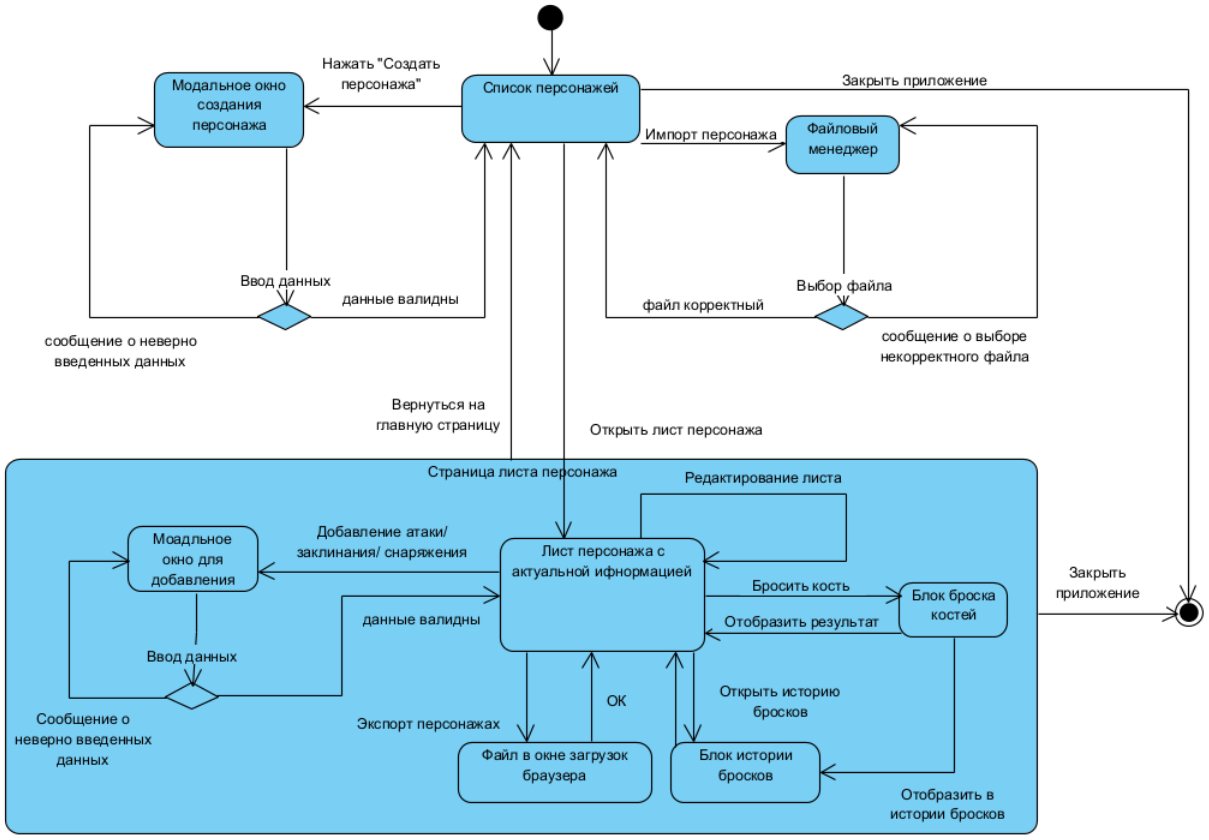

Рисунок 13 – Диаграмма состояний

Основными состояниями интерфейса являются: список персонажей, лист персонажа, а также модальные окна, которые будет всплывать при действиях пользователя.

К пользовательским действиям относятся: создание персонажа, импорт персонажа, открытие листа, редактирование персонажа, добавление атаки, заклинания или снаряжения, а также бросок кости и экспорт персонажа в файл.

На диаграмме также отражены системные ветвления, к которым относится: валидация данных, которые пользователь вводит в модальных окнах и обработка корректности файла при импорте персонажа.

2.4 Проектирование программных средств

2.4.1 Проектирование классов серверной части приложения

На основе полученной ранее логической модели базы данных, была разработана диаграмма классов для серверной части приложения интерактивного листа персонажа. Данная диаграмма отражает структуру, атрибуты, методы и взаимосвязи между классами программы, помогает проектировать систему перед её реализацией, документировать архитектуру и визуально анализировать разрабатываемое программное обеспечение [11].

На рисунке 14 представлена диаграмма классов серверной части приложения.

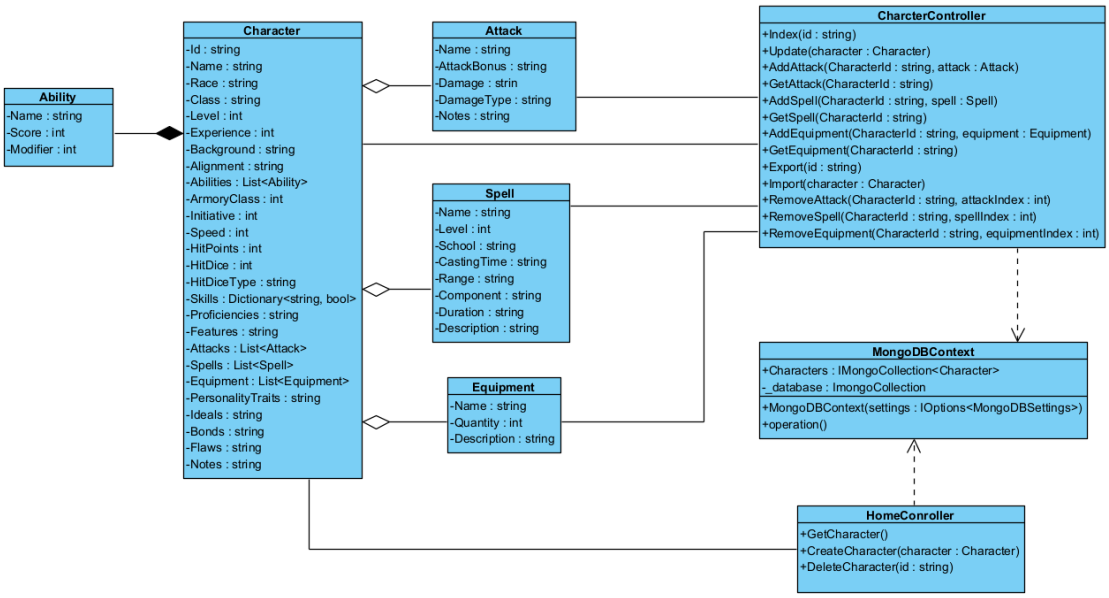

Рисунок 14 – Диаграмма классов серверной части приложения

Краткое описание и функциональное назначения каждого из классов на данной диаграмме приведено в таблице 2.

Таблица 2 – Описание классов

Класс	Описание
Character	Класс, содержащий все данные о персонаже, которые будут отображаться на листе персонажа
Ability	Класс, хранящий информацию об одной из шести основных характеристик персонажа (сила, ловкость, выносливость, интеллект, мудрость, харизма)
Attack	Класс, который хранит информацию об атаке персонажа

Продолжение таблицы 2

Класс	Описание
Spell	Класса, хранящий информацию об одном заклинании
Equipment	Класс, хранящий информацию об одном элементе снаряжения
CharacterController	Обрабатывает HTTP-запросы, связанные с листом персонажа (редактирование, добавление/удаление атак, заклинаний, снаряжения и экспорт персонажа)
HomeController	Обрабатывает HTTP-запросы от страницы списка персонажей (создание персонажа, удаление персонажа, открытие листа персонажа)
MongoDBContext	Обеспечивает взаимодействие контроллеров с базой данных MongoDB, управляет коллекцией персонажей

Класс Ability связан с классом Character через композицию, так как не может существовать вне его контекста. Классы Attack, Spell, Equipment связаны с классом Character через агрегацию, они представляют составные части информации о персонаже и могут быть добавлены/удалены через методы контроллера.

Для визуализации взаимодействия между объектами в системе была создана диаграмма коммуникаций, которая отражает структурные связи между частями программы и порядок обмена сообщениями. Данная диаграмма представлена на рисунке 15

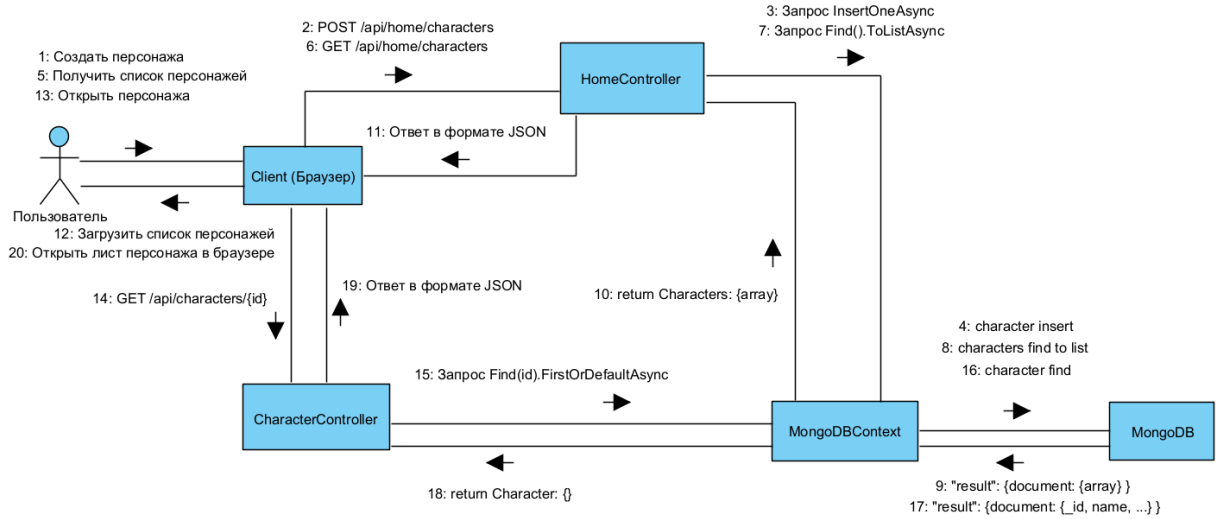

Рисунок 15 – Диаграмма коммуникации

В данном случает отражен порядок обмена сообщениями между элементами приложения, когда пользователь создает новый лист персонажа и открывает его.

2.4.2 Проектирование алгоритма создания листа персонажа

Наиболее важным алгоритмом в разрабатываемом приложении интерактивного листа персонажа является непосредственно создание нового листа персонажа.

Для отображения алгоритма создания персонажа была создана диаграмма деятельности, которая отражает как пользователь и система выполняют действия, какие операции при это выполняются.

Диаграмма деятельности создания листа персонажа представлена на рисунке 16.

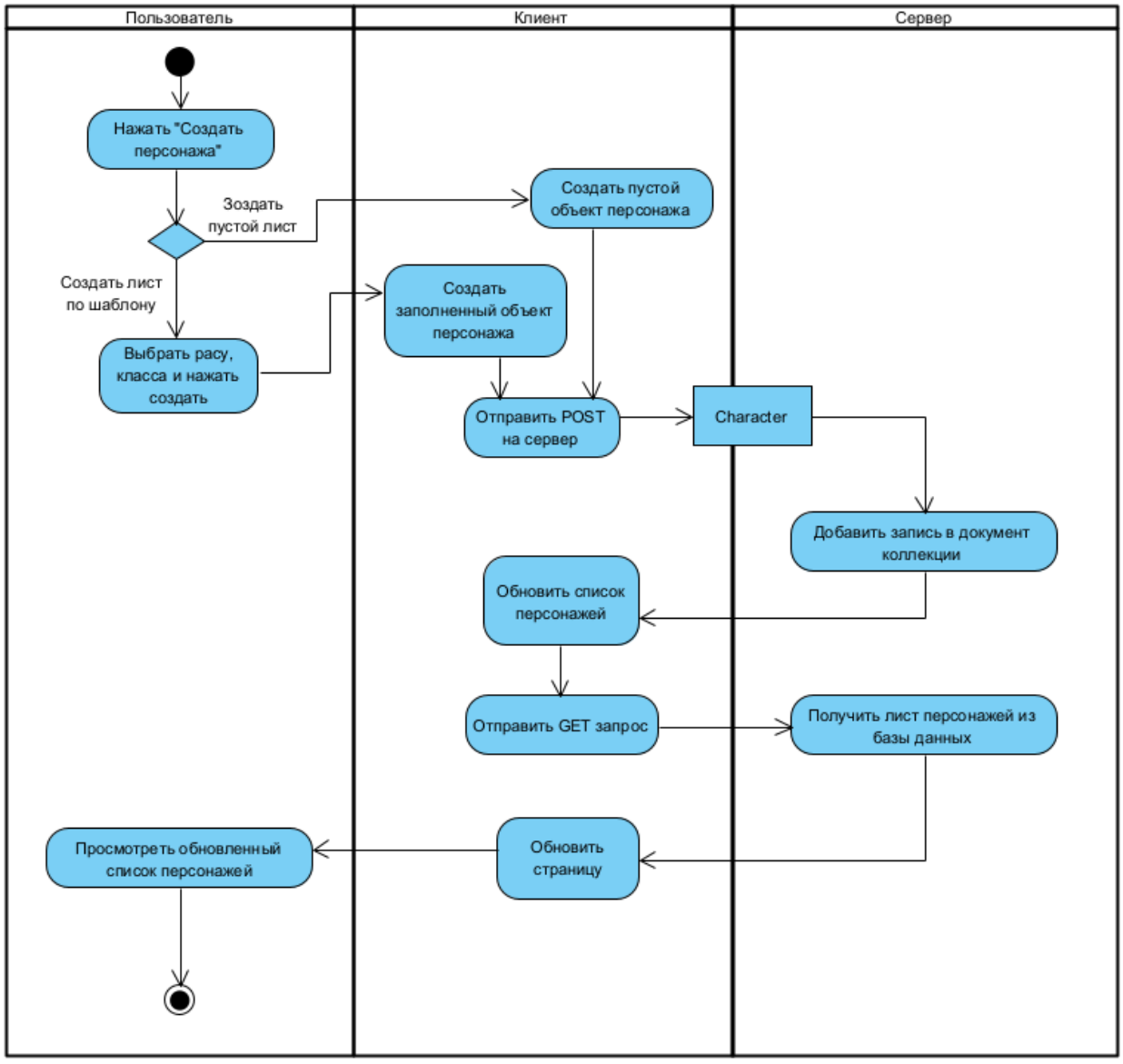

Рисунок 16 – Диаграмма деятельности для создания листа персонажа

При создании персонажа пользователь решает, будет ли создан пустой лист, либо же шаблонно заполненный. Если пользователь решил создать пустой лист персонажа, на стороне клиента объект с незаполненными полями, который передается POST запросом на сервер. При создании шаблонного листа пользователь выбирает базовую информацию о персонаже, которая записывается в объект на стороне клиента и также передается POST запросом.

После успешного добавления нового персонажа в коллекцию базы данных страница в браузере обновиться, и пользователь увидит в списке добавленного персонажа.

2.4.3 Проектирование алгоритма добавления заклинания

Алгоритм добавления нового заклинания был выполнен в виде диаграммы последовательности, которая описывает динамику взаимодействия отдельных компонентов системы на едином временном отрезке. Диаграмма последовательности алгоритма добавления заклинания представлена на рисунке 17.

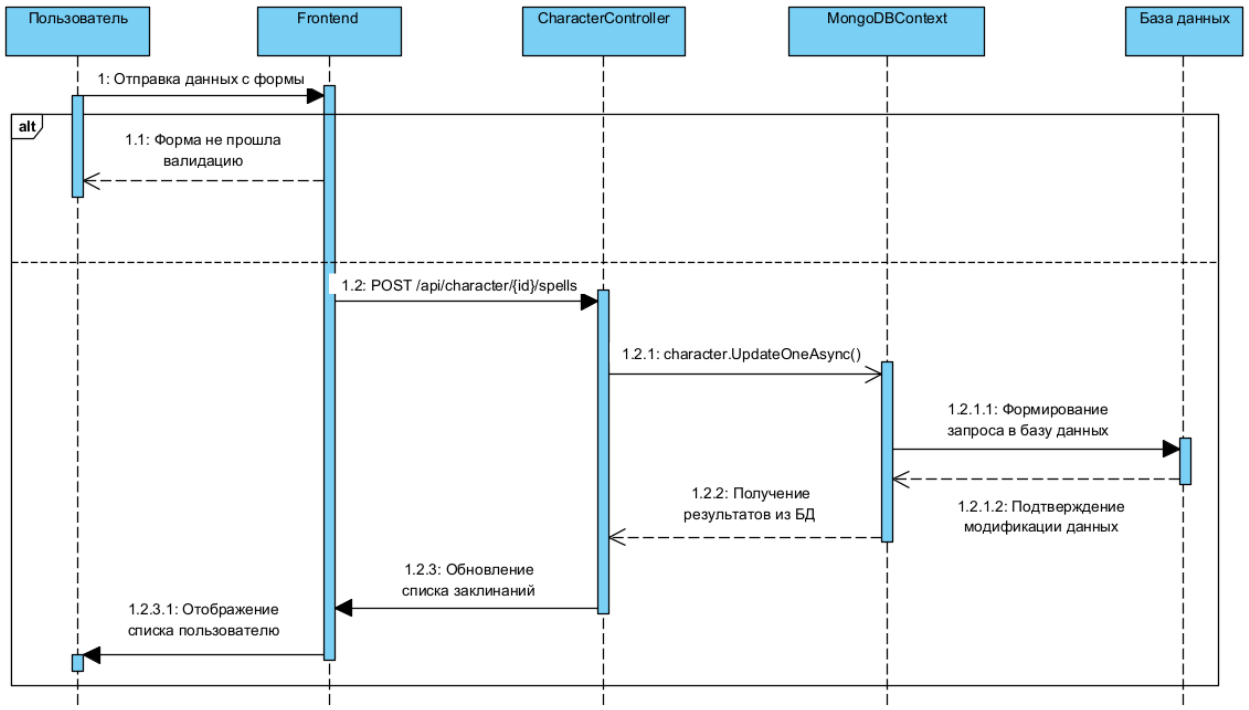

Рисунок 17 – Диаграмма последовательности процесса добавления заклинания

Пользователь инициирует добавление заклинания вводя данные в форму и отправляя их, после чего клиентское приложение проводит валидацию данных. При корректности введенных полей приложение отправляет POST запрос серверу. Сервер в свою очередь выполняет обновление документа в коллекции базы данных через запрос Update. После успешного добавления заклинания страница с листом персонажа обновиться и пользователю отобразится добавленное заклинание.

Алгоритмы добавления атаки и снаряжения в своей сути являются аналогичными данному алгоритму и отличаются лишь набором передаваемых данных с формы.

2.4.4 Проектирование алгоритма импорта листа персонажа

Процесс импорта персонажа из файла в базу данных приведен на рисунке 18.

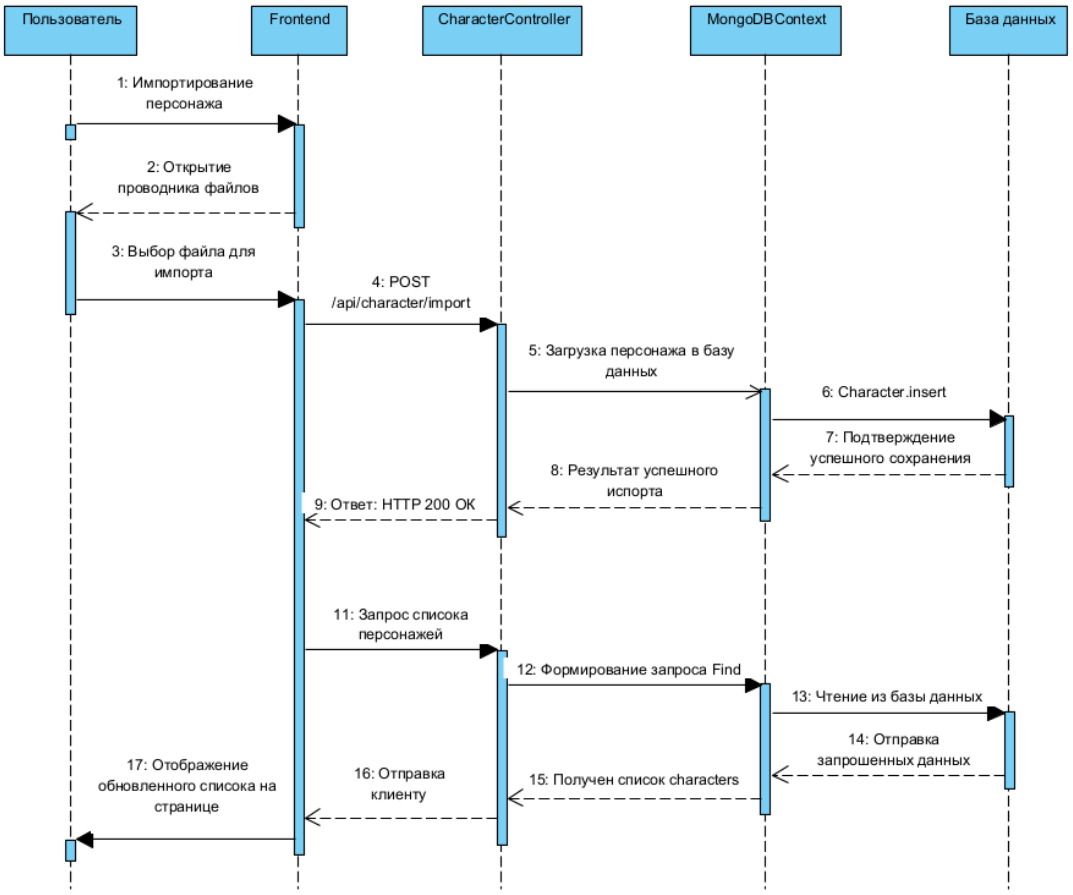

Рисунок 18 – Диаграмма последовательности процесс импорта персонажа

При начале импорта персонажа у пользователя открывается диалоговое окно, в котором он выбирает соответствующий JSON файл, который ранее был экспортирован из приложения.

Полученный файл преобразуется в объект, который передается в месте с POST запросом от клиента к серверу. После получения файла сервером система проверяет его и сравнивает id из полученного файл с id из базы данных для избегания ошибок. В случае если id уже существует, программа сгенерирует новый идентификатор для персонажа.

После успешного добавления в базу данных клиент отправит запрос GET, чтобы получить из базы данных обновленный список персонажей, который затем отобразит на странице.

3 Реализация веб-приложения интерактивного листа персонажа для настольной игры

3.1 Реализация базы данных

В отличие от реляционных баз данных, в NoSQL MongoDB не состоит из таблиц с фиксированным числом строк и столбцов и является документно-ориентированной системой. Физическая модель данных в MongoDB представлена в виде коллекций, которые в свою очередь содержат документы. Сам документ же хранит информацию как пару ключ-значение [12].

На основе разработанной ранее логической модели, был получен макет документа в коллекции Characters, отражающий физическую модель данных. Код документа представлен на листинге А.1 в приложении А.

В таблице 3 представлена спецификация полученного документа в коллекции Character.

Таблица 3 – Спецификация документа

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
_id	Идентификатор персонажа	ObjectId	Нет	0 – 9, a – f	Случайно генерируемое	PK
Name	Имя персонажа	String	Да	Любое	–	
Race	Раса персонажа	String	Да	Любое	–	
Class	Класс персонажа	String	Да	Любое	–	
Level	Уровень персонажа	Int	Нет	1 – 20	1	
Background	Предыстория Персонажа	String	Да	Любое	–	
Alignment	Мировоззрение персонажа	String	Да	Любое	–	
Experience	Очки опыта персонажа	Int	Нет	>=0	0	
Abilities	Массив характеристик	List <Ability>	Нет	–	–	
Abilities.Name	Наименование характеристики	String	Нет	«Strength», «Dexterity», «Constitution», «Intelligence», «Wisdom», «Charisma»	–	

Продолжение таблицы 3

Наименование	Описание	Тип данных	Возможность значения NULL	ОДЗ	Значение по умолчанию	Роль
Abilities.Score	Числовое значение характеристики	String	Нет	0 – 30	10	
ArmorClass	Класс доспеха персонажа	String	Нет	0 – 40	10	
Initiative	Бонус к инициативе	Int	Да	0 – 10	–	
Speed	Скорость персонажа	Int	Нет	25 – 40	30	
HitPoints	Очки здоровья	Int	Да	-1000 – 1000	0	
HitDice	Кол-во костей здоровья	Int	Нет	1 – 20	1	
HitDiceType	Тип кости	String	Нет	«1d6», «1d8», «1d10», «1d12»	«1d6»	
Skills	Словарь навыков	Dictionary <string, bool>	Нет	–	–	
Proficiencies	Владения персонажа	String	Да	Любое	–	
Features	Способности расы и класса	String	Да	Любое	–	
Attacks	Массив атак	List<Attack>	Нет	–	–	
Attacks.Name	Название атаки	String	Да	Любое	–	
Attacks.Bonus	Бонус атаки	String	Да	Любое	–	
Attacks.Damage	Урон атаки	String	Да	Любое	–	
Attacks. DamageType	Тип урона	String	Да	Любое	–	
Attacks.Notes	Описание атаки	String	Да	Любое	–	
Equipment	Массив снаряжения	List<Equipment>	Нет	–	–	
Equipment.Name	Наименование	String	Да	Любое	–	
Equipment. Quantity	Количество	Int	Да	>0	1	
Equipmnet. Description	Описание снаряжения	String	Да	Любое	–	
Spells	Массив заклинаний	List<Spell>	Нет	–	–	
Spells.Name	Наименование заклинания	String	Да	Любое	–	
Spells.Level	Уровень заклинания	String	Да	0 – 9	–	
Spells.School	Школа заклинания	String	Да	Любое	–	
Spells.Casting Time	Время сотворения	String	Да	Любое	–	
Spells.Range	Дистанция	String	Да	Любое	–	
Spells.Component	Компоненты	String	Да	Любое	–	
Spells.Duration	Длительность	String	Да	Любое	–	
Spells.Description	Описание заклинания	String	Да	Любое	–	
PersonalityTraits	Черты персонажа	String	Да	Любое	–	
Ideals	Идеалы	String	Да	Любое	–	
Bonds	Связи и прошлого	String	Да	Любое	–	
Flaws	Слабости	String	Да	Любое	–	
Notes	Прочие заметки	String	Да	Любое	–	
Appearance	Описание внешности	String	Да	Любое	–	

Ранее созданные сущности Attack, Ability, Spell и Equipment в физической реализации представляют собой массив объектов внутри документа.

3.2 Реализация программных средств

Для реализации серверной части приложения была использована среда программирования Visual Studio 2022. В разрабатываемы проект был добавлен NuGet пакет MongoDB.Drive для взаимодействия программы с базой данных.

Серверная часть приложения состоит из классов моделей, которые предназначены для работы с сущностями базы данных. Также в серверной части присутствуют контроллеры, которые получают HTTP-запросы от клиента и выполняют сохранение, модификацию или получение данных из БД.

Код серверной части приложения представлен на листингах Б.1 – Б.9 в приложении Б [13].

Взаимодействие с клиентом осуществляет при помощи RESTful API запросов, спецификации которых представлены в таблице 4.

Таблица 4 – Спецификация API запросов

№	Назначение запроса	HTTP метод	API маршрут метода	Класс контроллера
1	Получение списка персонажей	GET	/api/home/characters	HomeController
2	Создание нового персонажа	POST	/api/home/characters	HomeController
3	Удаление персонажа по ID	DELETE	/api/home/characters/{id}	HomeController
4	Получить персонажа по ID	GET	/api/characters/{id}	CharacterController
5	Обновить данные о персонаже	POST	/api/character/update	CharacterController
6	Добавить новую атаку	POST	/api/character/{characterId}/attacks	CharacterController
7	Удалить атаку	DELETE	/api/character/{characterId}/attacks/{attackIndex}	CharacterController
8	Получить список атак	GET	/api/character/{characterId}/attacks	CharacterController
9	Добавить новое заклинание	POST	/api/character/{characterId}/spells	CharacterController
10	Получить список заклинаний	GET	/api/character/{characterId}/spells	CharacterController
11	Удалить заклинание	DELETE	/api/character/{characterId}/spells/{spellIndex}	CharacterController

Продолжение таблицы 4

№	Назначение запроса	HTTP метод	API маршрут запроса	Класс контроллера
12	Добавить экипировку	POST	/api/character/{characterId}/equipment	CharacterController
13	Получить список экипировки	GET	/api/character/{characterId}/equipment	CharacterController
14	Удалить экипировку	DELETE	/api/character/{characterId}/equipment/{equipmentIndex}	CharacterController
15	Экспортировать персонажа в JSON	GET	/api/character/export/{id}	CharacterController
16	Импортировать персонажа из JSON	POST	/api/character/import	CharacterController

Для отображения физического расположения компонентов серверной части приложения была создана диаграмма развертывания, которая представлена на рисунке 19.

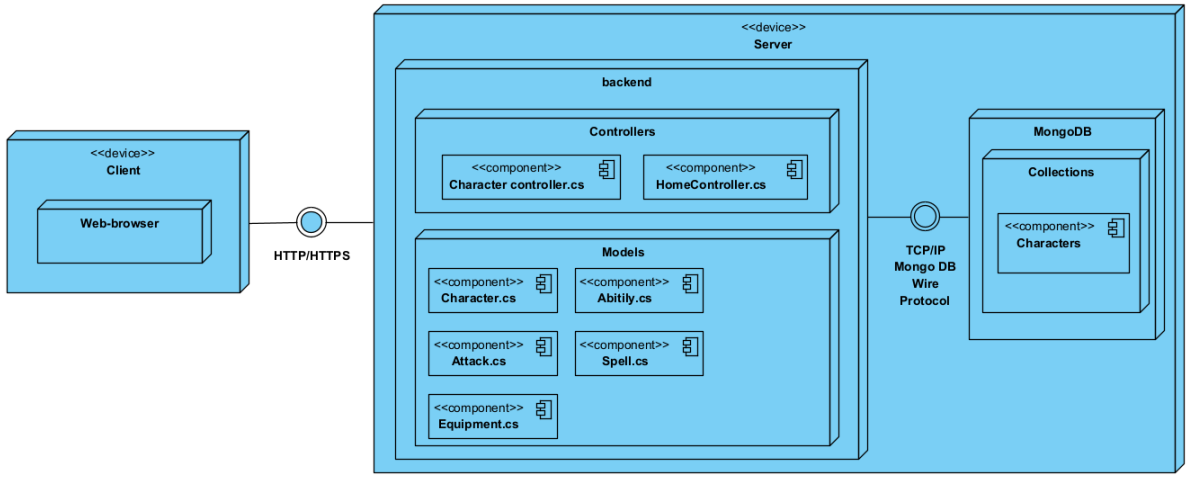

Рисунок 19 – Диаграмма развертывания

Как видно и данной диаграммы, полученное веб-приложение соответствует многослойно архитектуре. Код серверной части приложения и база данных располагаются на стороне сервера общаясь по средством TCP/IP протокола с использованием MongoDB Wire Protocol – внутреннего бинарного протокола, основанного на формате BSON (Binary JSON), который обеспечивает высокую производительность при работе с базой данных.

Пользователь запускает своё приложение в браузере, где при работе клиентская часть выполняет отправку на сервер API-запросов по протоколу HTTP/HTTPS, что обеспечивает совместимость со всеми современными браузерам и пользовать программой на любом устройстве, имеющем браузер.

Для более полной визуализации архитектуры полученного приложения и взаимодействия отдельных её компонентов между собой была создана диаграмма компонентов (Рис. 20).

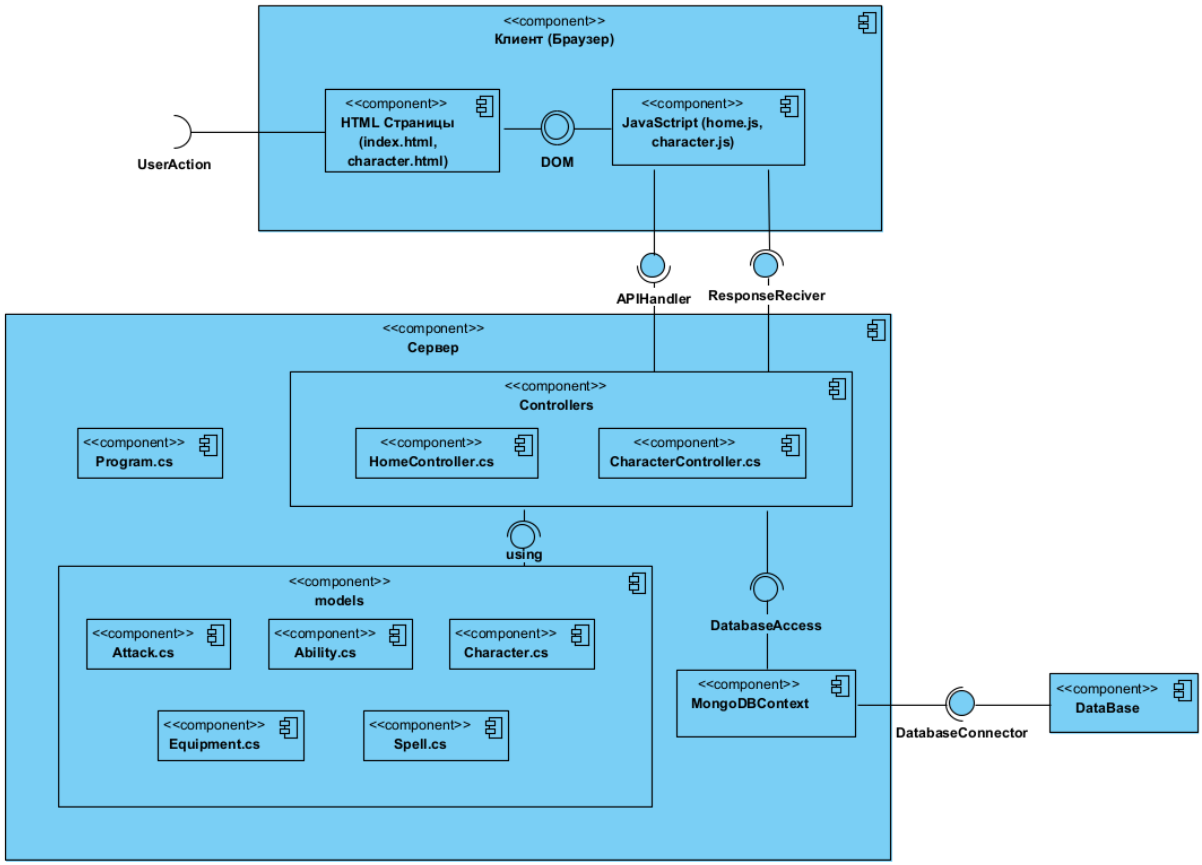

Рисунок 20 – Диаграмма компонентов

Компонент клиентской части состоит из HTML страниц, являющихся пользовательским интерфейсом, и JavaScript кода, который отвечает за логику серверной части и интерактивность интерфейса. Они взаимодействуют через объектную модель документа.

Контроллеры HomeController и CharacterController являются основными компонентами серверной части приложения, они используют Models при обработке API-запросов.

MongoDBContext используется контроллерами для получения доступа к базе данных и выполнения CRUD (create, read, update и delete) операций.

3.3 Реализация клиентской части приложения

На основе созданных ранее макетов, был реализован интерфейс для клиентской части приложения интерактивного листа персонажа. Клиентская часть приложения реализована как HTML страница с использованием CSS для стилизации и JavaScript для интерактивности. Помимо CSS при стилизации использовался фреймворк Bootstrap, который обеспечивает адаптивный дизайн и готовые компоненты интерфейса, такие как модальные окна, карточки и кнопки [14].

Интерфейс страницы со списком персонажей представлен на рисунке 21.

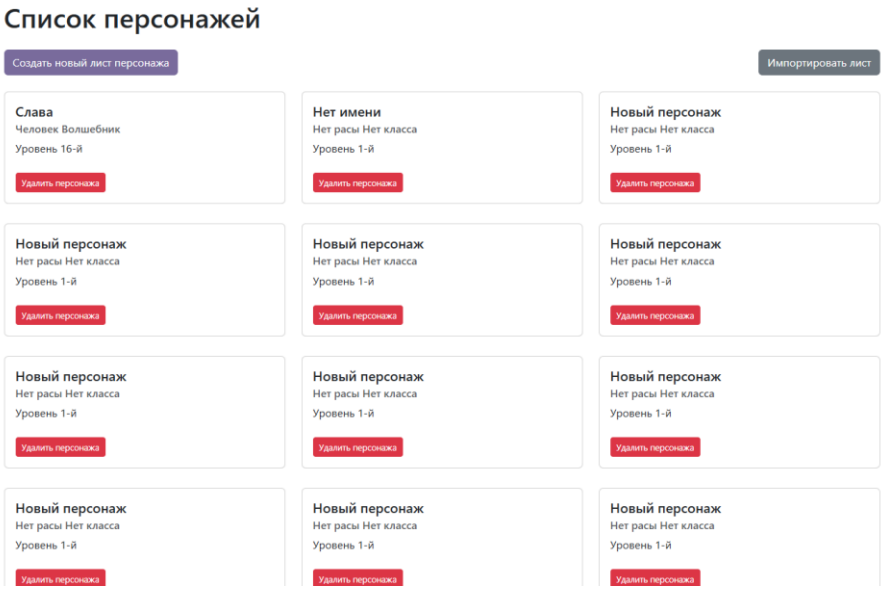

Рисунок 21 – Страница списка персонажей

Данная страница является стартовой при открытии приложения. При нажатии по кнопке «Создать новый лист персонажа» открывается модальное окно для создания персонажа, в котором можно выбрать способ создания персонажа через выпадающее меню.

Окно модального окна для создания листа персонажа представлено на рисунке 22.

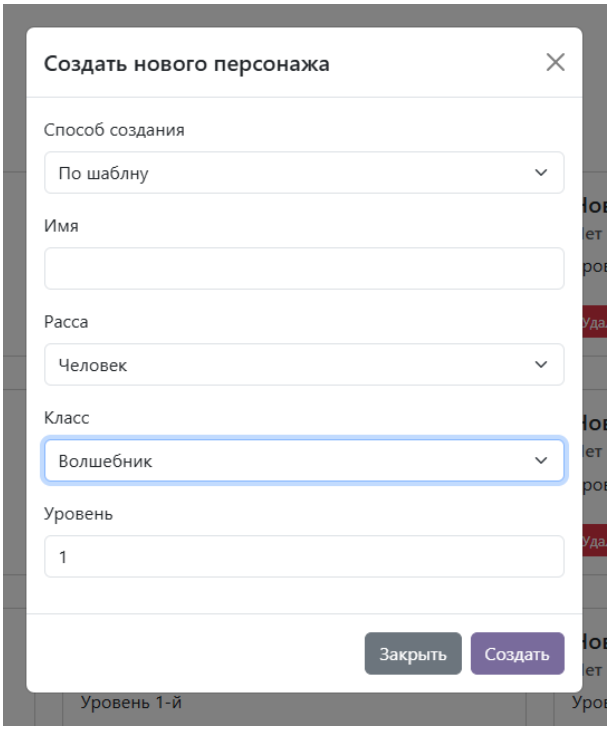

Рисунок 22 – Модальное окно создания персонажа

При нажатии по кнопке карточке персонажа откроется страница листа персонажа. Интерфейс страницы листа персонажа представлен на рисунке 23.

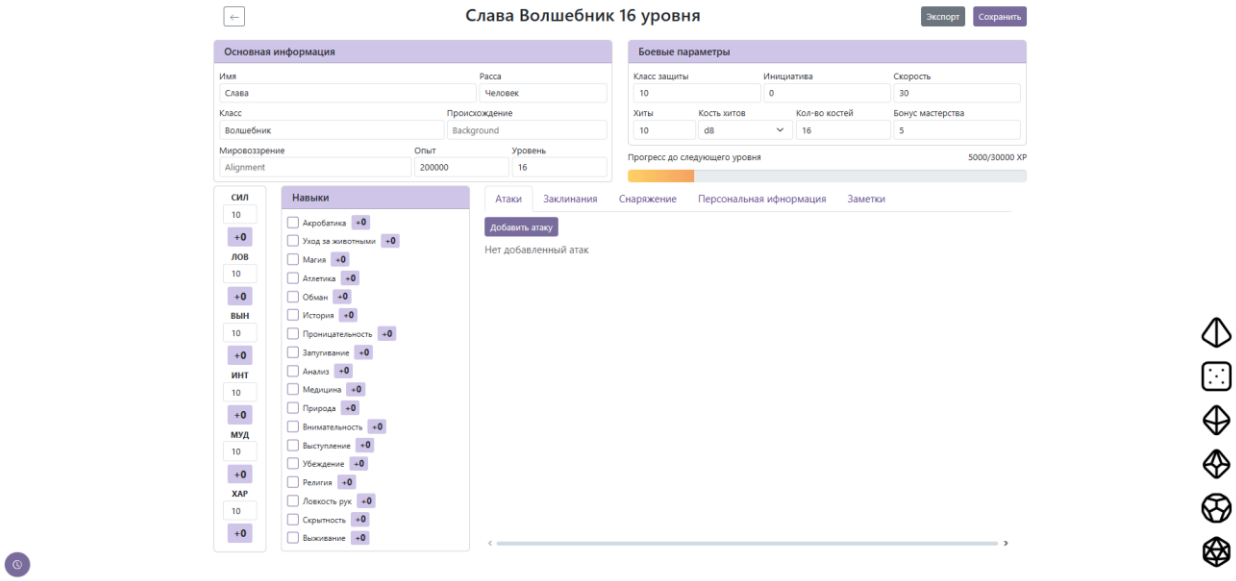

Рисунок 23 – Страница листа персонажа

При переходе нажатии на кнопку «Добавить атаку» во вкладке «Атаки», открывается модальное окно для добавления атаки в лист персонажа. Интерфейс модального окна представлен на рисунке 24.

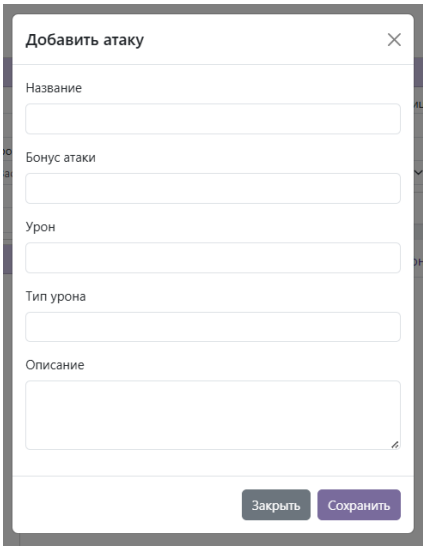

Рисунок 24 – Модально окно добавления атаки

При нажатии кнопки «Добавить заклинание» во вкладе «Заклинания», открывается модальное окно для добавления заклинания, которое представлено на рисунке 25.

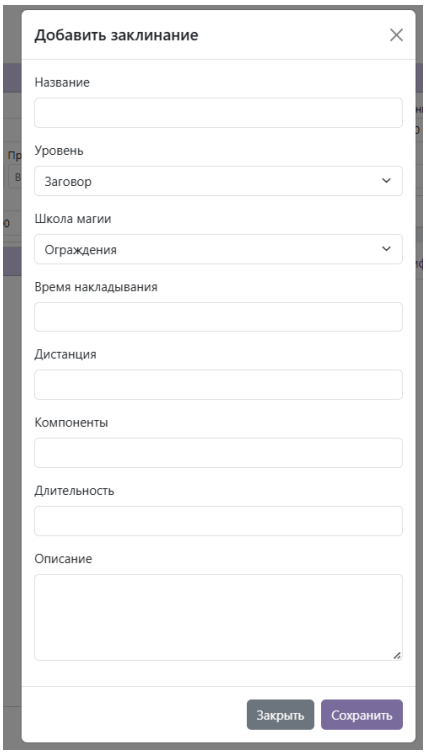

Рисунок 25 – Модально окно добавления заклинания

При нажатии по кнопке «Добавить снаряжение» во вкладке «Снаряжение», открывается модально окно для добавления снаряжение, которое представлено на рисунке 26.

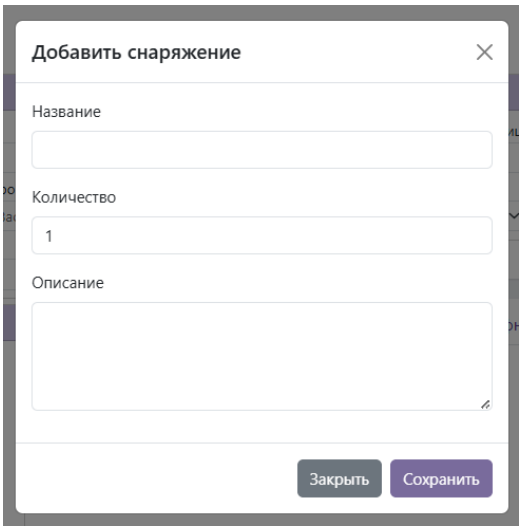

Рисунок 26 – Модально окно добавления снаряжения

Также на странице присутствуют кнопки для броска костей, результат которых отображается в блоке истории бросков. Интерфейс блока истории бросков представлен на рисунке 27.

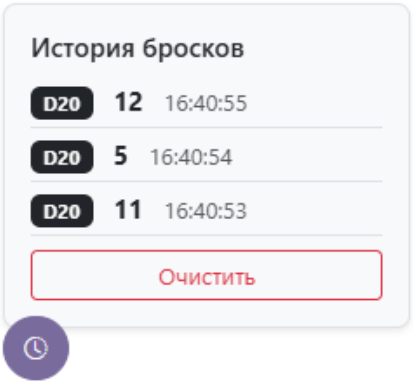

Рисунок 27 – Блок истории бросков

В результате был получен рабочий интерфейс для клиентской части приложения, который полностью удовлетворяет требованиям. Код HTML страниц, стилей и JavaScript клиентской части приложения придавлен на листингах В.1 – В.5 в приложении В [15].

4 Отладка и тестирование

4.1 Выбор режима тестирования

Для проверки корректности работы разработанного программного обеспечения и обеспечения соответствия заданным ранее требованиям было выполнено его тестирование. На данный момент существует множество способов тестирования, которые в общем подразделяются на функциональное и нефункциональное тестирование [16].

Функциональное тестирование предназначено для проверки соответствия системы заявленным требованиям и включает в себя:

– интеграционное тестирование;
– модульное тестирование;
– системное тестирование;
– приемочное тестирование.

Нефункциональное тестирование проверяет аспекты, которые не связаны с функционалом программного обеспечения, такие как:

– тестирование безопасности;
– тестирование нагрузки;
– тестирование производительности;
– тестирование удобства использования.

При тестировании веб-приложений построенных, созданных на платформе ASP.NET Core стоит уделить внимание проверке работы контроллеров, сервисов, моделей и взаимодействия с базой данных. Следует проверить корректность работы API-запросов от клиента к серверу, а также контроллеров, так как они являются главным узлом всей системы и отвечают за обработку HTTP-запросов. В процессе тестирования будут проверены все используемы в программе методы API запросов (GET, POST, DELETE) [17].

Для проверки удобства использования приложения применяется тестирование пользовательского интерфейса. При данном тестировании

следует уделить внимание корректности отображения визуальных элементов, их расположения на различных браузерах и устройствах.

Таким образом при тестировании приложения будет проверено его соответствие поставленным функциональным требования, корректная работа API-запросов к серверной части приложения, а также работа интерфейса в различных браузерах.

4.2 Функциональное тестирование

Для проведения функционального тестирования была выбрана функция программы по добавлению нового заклинания на лист персонажа.

При выполнении данной операции пользователь должен ввести такие данные как: название, уровень, школа магии, время накладывания, дистанция, компоненты, длительность и описание. При этом каждое из вводимы значений не должно быть пусты, и программа должна провести валидацию чтобы не допустить пустые поля при отправке формы.

Тестирование будет проводиться и использованием функциональных диаграмм и таблицы решений. Функциональная диаграмма представляет собой графическое отображение функции системы и строиться на основе причин, которые являются входными условиями в системе, и следствий, которые являются выходным результатом работы. Поля «Уровень» и «Школа магии» на форме являются выпадающем меню с готовым набором значений и не могут быть пустыми, следовательно не учитываются при тестировании.

На таблицах 5 и 6 представлен список причин и следствий для тестирования функции.

Таблица 5 – Причины

П1	Поле имени не пустое
П2	Поле время накладывания не пустое
П3	Поле дистанция не пустое
П4	Поле компонентов не пустое
П5	Поле длительность не пустое
П6	Поле описания не пустое

Таблица 6 – Следствия

С1	Сообщение: «Введите имя заклинания»
С2	Сообщение: «Введите время накладывания заклинания»
С3	Сообщение: «Введите дистанцию заклинания»
С4	Сообщение: «Введите компоненты заклинания»
С5	Сообщение: «Введите длительность заклинания»
С6	Сообщение: «Введите описание заклинания»
С7	Сообщение: «Заклинание успешно добавлено»

На основе данных таблиц была построена функциональная диаграмма, которая отражает процесс добавления заклинания (Рис. 28).

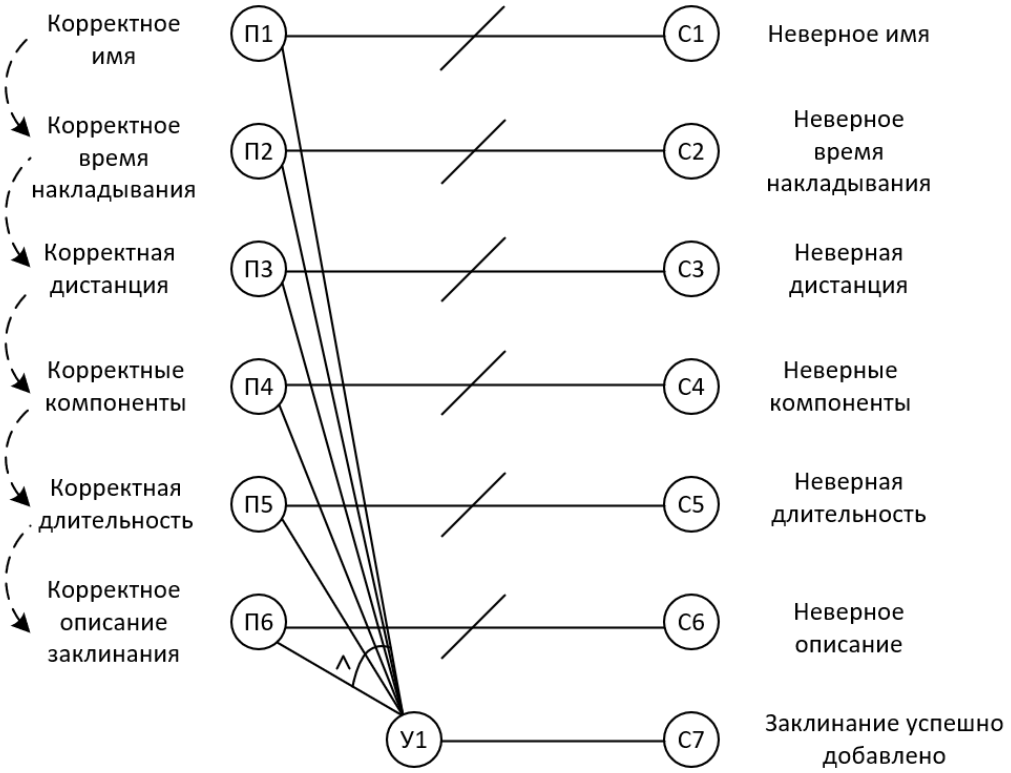

Рисунок 28 – Функциональная диаграмма добавления заклинания

Таблица решений, построенная на основе функциональной диаграммы, представлена на таблице 7.

Таблица 7 – Таблица решений

	1	2	3	4	5	6	7	8
П1	0	0	1	1	1	1	1	1
П2	0	1	0	1	1	1	1	1
П3	0	1	1	0	1	1	1	1
П4	0	1	1	1	0	1	1	1
П5	0	1	1	1	1	0	1	1
П6	0	1	1	1	1	1	0	1

Продолжение таблицы 7

	1	2	3	4	5	6	7	8
C1	1	1	0	0	0	0	0	0
C2	1	0	1	0	0	0	0	0
C3	1	0	0	1	0	0	0	0
C4	1	0	0	0	1	0	0	0
C5	1	0	0	0	0	1	0	0
C6	1	0	0	0	0	0	1	0
C7	0	0	0	0	0	0	0	1

На основе данной таблицы решений были разработаны тест-кейсы, которые представлены в таблице 8.

Таблица 8 – Тест-кейсы

№	Входные данные	Ожидаемый результат	Результат
1	Все поля пустые	Сообщение: «Заполните поля»	Тест пройден (рисунок Г.1)
2	Имя: Пустое Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Касание Компоненты: Вербальный Длительность: 1 час Описание: Предмет, которого вы коснулись, испускает яркий свет	Сообщение: «Введите имя заклинания»	Тест пройден (рисунок Г.2)
3	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: пустое Дистанция: Касание Компоненты: Вербальный Длительность: 1 час Описание: Предмет, которого вы коснулись, испускает яркий свет	Сообщение: «Введите время накладывания заклинания»	Тест пройден (рисунок Г.3)
4	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Пустое Компоненты: Вербальный Длительность: 1 час Описание: Предмет, которого вы коснулись, испускает яркий свет	Сообщение: «Введите дистанцию заклинания»	Тест пройден (рисунок Г.4)

Продолжение таблицы 8

№	Входные данные	Ожидаемый результат	Результат
5	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Касание Компоненты: Пустое Длительность: 1 час Описание: Предмет, которого вы коснулись, испускает яркий свет	Сообщение: «Введите компоненты заклинания»	Тест пройден (рисунок Г.5)
6	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Касание Компоненты: Вербальный Длительность: Пустое Описание: Предмет, которого вы коснулись, испускает яркий свет	Сообщение: «Введите длительность заклинания»	Тест пройден (рисунок Г.6)
7	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Касание Компоненты: Вербальный Длительность: 1 час Описание: Пустое	Сообщение: «Введите описание заклинания»	Тест пройден (рисунок Г7)
8	Имя: Свет Уровень: Заговор Школа магии: Воплощения Время накладывания: 1 действие Дистанция: Касание Компоненты: Вербальный Длительность: 1 час Описание: Предмет, которого вы коснулись, испускает яркий свет	Заклинание добавлено в лист персонажа	Тест пройден (рисунок Г.8)

В результате функциональное тестирование показало корректную работу функции добавления заклинания в интерактивный лист персонажа.

4.3 Тестирование серверной части приложения

При выполнении тестирования серверной части приложения применялся веб-сервис Postman. Данный сервис предназначен для тестирования работы API, документирования и проверки отправки-получения HTTP-запросов.

На рисунке 29 приведен пример использования сервиса Postman.

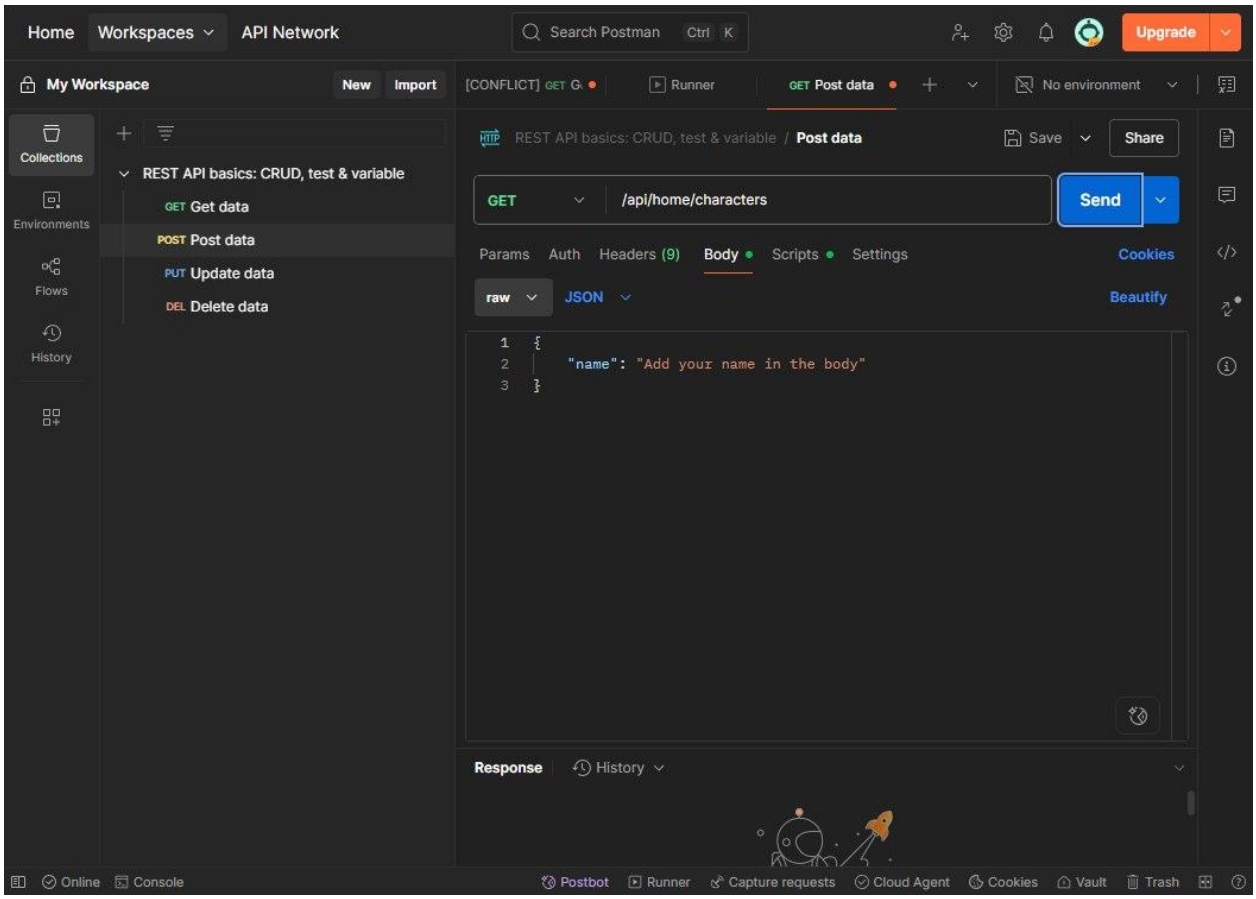

Рисунок 29 – Пример использования Postman

Процесс тестирования начинается с выбора метода запроса в левом меню. После выбора в текстовой строке указывается API-маршрут для отправки запроса. Если файл подразумевает отправку данных, то во вкладе Body вводятся тестовые данные для отправки в формате JavaScript, JSON, HTML или XML. После отправки запроса в окне Response отобразиться результат запроса в виде кода состояния HTTP. Если был выполнен запрос GET на выборку данных, то они также отобразятся в окне Response.

Тест кейсы для тестирования API методов представлены в таблице 9.

Таблица 9 – Тест кейсы API-запросов

API метод	Описание теста	Ожидаемый результат	Фактический результат
GET	Получение данных о всех персонажах по адресу: /api/home/characters	Тест пройден	Тест пройден (рисунок Г.9)
POST	Добавление персонажа по адресу: /api/home/characters	Тест пройден	Тест пройден (рисунок Г.10)
DELETE	Удаление персонажа по адресу /api/home/characters/{id}	Тест пройден	Тест пройден (рисунок Г.11)

В результате тестирования серверной части приложения не было выявлено ошибок при обработке API-запросов. Все методы успешно взаимодействую с базой данных по настроенным маршрутам и возвращают ожидаемые результаты.

4.4 Тестирование клиентской части приложения

В ходе тестирования клиентской части приложения был выполнен запуск разработанного веб-приложения на различных браузерах для тестирования совместимости интерфейса. Запуск приложения и проверка корректности работы пользовательского интерфейса выполнялась на персональном компьютере в браузерах: Opera, Google Chrome и Microsoft Edge. Также пользовательский интерфейс был протестирован на смартфоне с операционной системой Android 14.0 в браузере Google Chrome.

Список выполненных тест-кейсов представлен в таблице 10.

Таблица 10 – Тест-кейсы пользовательского интерфейса

№	Входные данные	Ожидаемый результат	Результат
1	Запуск приложения в браузере Google Chrome	Все элементы интерфейса в браузере отображены корректно	Тест пройден (Рисунок Г.9)
2	Запуск приложения в браузере Opera	Все элементы интерфейса в браузере отображены корректно	Тест пройден (Рисунок Г.10)
3	Запуск приложения в	Все элементы интерфейса в браузере отображены корректно	Тест пройден (Рисунок Г.11)

	браузере Microsoft Edge		
4	Запуск приложения на смартфоне	Все элементы интерфейса на смартфоне отображены корректно	Тест пройден (Рисунок Г.12)
5	Запуск приложения на смартфоне с альбомной ориентацией	Все элементы интерфейса смартфоне отображены корректно после смены ориентации на альбомную	Тест пройден (Рисунок Г.13)

Тестирование клиентской части приложения показало корректную работу пользовательского интерфейса на различных устройствах и браузерах.

5 Контроль качества

5.1 Метрики Холстеда

Метрики Холстеда – это набор метрик, предназначенных для оценки сложности программного кода на основе анализа путем анализа операторов и операндов программы [19].

Основные метрики Холстеда:

– число уникальных операторов;

– число уникальных операндов;

– общее количество операторов;

– общее количество операндов.

Для проведения оценки метрик Холстеда был выбран метод RemoveSpell класса CharacterController, код которого представлен в листинге 1.

Листинг 1 – Код метода RemoveSpell

```
public async Task<IActionResult> RemoveSpell(string characterId,
int spellIndex)
{
    try
    {
        //Получить персонажа из БД
        var character = await _context.Characters
            .Find(c => c.Id == characterId)
            .FirstOrDefaultAsync();
        //Проверка персонажа
        if (character == null)
            return NotFound("Character not found");

        //Поиск заклинания
        if (spellIndex < 0 || spellIndex >=
character.Spells.Count)
            return BadRequest("Invalid spell index");

        //Удаление заклинания в классе
        character.Spells.RemoveAt(spellIndex);

        //Загрузка заклинания в БД
        var filter = Builders<Character>.Filter.Eq(c => c.Id,
characterId);
        var update = Builders<Character>.Update.Set(c =>
c.Spells, character.Spells);
```

```
        var result = await
_context.Characters.UpdateOneAsync(filter, update);

        return Ok(new { success = true });
    }
    catch (Exception ex)
    {
        return StatusCode(500, new { error = ex.Message });
    }
}
```

Операторы метода RemoveSpell приведены в таблице 11.

Таблица 11 – Операторы метода RemoveSpell

Операторы	Число вхождений
Try	1
=	6
If	2
==	2
<	1
>=	4
Return	1
New	2
Catch	1
;	9
{}	4
.	18
n1 = 12	N1 = 51

Таблица 12 – Операнды метода RemoveSpell

Операнды	Число вхождений
character	5
characrerId	2
spellIndex	4
Spells	4
Filter	2
Update	2
Result	1
C	2
success	1
Error	1
500	1
n2 = 11	N2 = 25

Для расчета длинны программы применялась формула:

$$N = N1 + N2 = 51 + 25 = 76.$$

Расчет словаря программы рассчитывался по формуле:

$$n = n1 + n2 = 12 + 11 = 23.$$

Для расчета объема программы применялась формула:

$$V = N * \log2(n) = 76 * 4.52 = 343.52.$$

Потенциальный объем программы рассчитывался по формуле:

$$V^* = (2 + n2) * \log2(2 + n2) = 13 * 3.7 = 48.1.$$

Теоретический уровень программы вычислялся по формуле:

$$L = V^*/V = 48.1 / 343/52 = 0.14.$$

Для расчета сложности программы применялась формула:

$$HD = (n1 / 2) * (N2 / n2) = (12 / 2) * (25 / 11) = 6 * 2.27 = 13.62.$$

Таким образом был получен теоретический уровень программы со значением 0.14. Расчет объема программы показал, что программа имеет средний уровень сложности.

5.2 Метрики Чепина

Метрики Чепина предназначены для оценки информационной прочности части программного кода, при котором рассматривается использование переменных путем разбиения их на 4 группы:

- вводимые для выполнения расчетов и ввода-вывода (P);
- модифицируемые или создаваемые внутри программы (M);
- управляющие (C);
- неиспользуемые переменные (T).

При каждая переменная может быть охарактеризована несколькими группами. Расчет метрики Чепина выполняется по следующей формуле:

$$Q = a1 * P + a2 * M + a3 * C + a4 * T.$$

где a1, a2, a3, a4 – весовые коэффициенты [19].

Значения коэффициентов было принято следующее: a1 = 1, a2 = 2, a3 = 3, a4 = 0.5. Весовой коэффициент управляющих переменных наиболее высокий так как они влияют на поток управления программы. Неиспользуемые переменные хоть и не увеличивают сложность потока данных, но при этом затрудняют понимание самой программы. В рассматриваемой функции RemoveSpell присутствуют следующие группы

переменных: P =2, M = 1, C = 3, T = 0. Таким образом было получено следующие значение метрики Чепина:

$$Q = 1 * 2 + 2 * 1 + 3 * 3 + 0.5 * 0 = 13.$$

Полученное значение метрики показывает, что рассматриваемая функция имеет умеренную сложность.

5.3 Метрики SLOC

Метрика SLOC рассматривает объем кода, используя подсчет строк в исходном коде программы. Она применяется для оценки размера программы и анализа сложности, трудозатрат на разработку и поддержание кода [19].

В рассматриваемой функции RemoveSpell были подсчитаны следующие строки:

- физические строки: 32;
- логические строки: 22;
- строки комментариев: 5;
- пустые строки: 5.

Физические строки отражают общее число строк рассматриваемого кода, логические строки показывают количество операндов. Исходя из подсчитанных строк, получаем значение метрики SLOC = 22, что является приемлемым значением для рассмотренной функции.

Проведенный анализ кода с использование метрик Холстеда, Чепина и SLOC предоставил информацию о сложности и объеме разработанного приложения. Метрики Холстеда и Чепина показали средний уровень сложности программного кода, анализ SLOC также показал умеренное число логических строк. Общая структура кода достаточно понятная и читаема, что поспособствует дальнейшему развитию и поддержке приложения без значительных трудозатрат.

6 Расчет бюджета и разработки

При планировании разработки приложения и оценки его бюджета использовалась программа Microsoft Project. Данная программа предоставляет инструменты для планирования проектов, включая: планирование задач, распределение ресурсов, контроль сроков и загрузки сотрудников при работе над проектом. Работа над проектом была начата 8 апреля 2025 года.

В ходе планирования было выделено 26 задач, список которых представлен на рисунке 30.

	Название задачи	Длительн	Начало	Окончание	Предшественник	Названия ресурсов
1	Начало работы над проектом	0 дней	Вт 08.04.25	Вт 08.04.25		
2	**Анализ требований**	**7 дней**	**Вт 08.04.25**	**Ср 16.04.25**		**Бэкэнд разработчик[50%]**
3	Анализ предметной области	2 дней	Вт 08.04.25	Ср 09.04.25		Аналитик
4	Анализ аналогичных программ	3 дней	Ср 09.04.25	Пн 14.04.25	3	Аналитик
5	Постановка задачи на разработку	1 день	Пн 14.04.25	Вт 15.04.25	4	Аналитик
6	Анализ требований к программе	1 день	Вт 15.04.25	Ср 16.04.25	5	Бэкэнд разработчик;Аналитик
7	Анализ требований завершен	0 дней	Ср 16.04.25	Ср 16.04.25	6	
8	**Проектирование приложения**	**9 дней**	**Чт 17.04.25**	**Вт 29.04.25**		
9	Проектирование архитектуры приложения	2 дней	Ср 16.04.25	Пт 18.04.25	7	Бэкэнд разработчик[50%]
10	Проекитрование базы данных	3 дней	Пт 18.04.25	Ср 23.04.25	9	Бэкэнд разработчик[50%]
11	Проектирование пользовательского интерфейса	2 дней	Ср 23.04.25	Пт 25.04.25	10	Фронтэнд разработчик
12	Проектирование программных средств	1 день	Пн 28.04.25	Вт 29.04.25	11	Бэкэнд разработчик
13	Прокетирование приложения завершено	0 дней	Вт 29.04.25	Вт 29.04.25	12	
14	**Реализация приложения**	**18 дней**	**Вт 29.04.25**	**Пт 23.05.25**		
15	Реализация базы данных	4 дней	Вт 29.04.25	Пн 05.05.25	13	Бэкэнд разработчик[50%]
16	Реализация серверной части приложения	8 дней	Пн 05.05.25	Чт 15.05.25	15	Бэкэнд разработчик[50%]
17	Реализация клиентской части приложения	6 дней	Чт 15.05.25	Пт 23.05.25	16	Фронтэнд разработчик
18	Реализация приложения завершена	0 дней	Пт 23.05.25	Пт 23.05.25	17	
19	**Тестирование приложения**	**8 дней**	**Пт 23.05.25**	**Ср 04.06.25**		
20	Выбор мметода тестирования	2 дней	Пт 23.05.25	Вт 27.05.25	18	Аналитик
21	Тестирование серверной части приложения	1 день	Вт 27.05.25	Ср 28.05.25	20	Бэкэнд разработчик[50%]
22	Тестирование клиентской части приложения	1 день	Ср 28.05.25	Чт 29.05.25	21	Фронтэнд разработчик
23	Анализ метрик кода	1 день	Чт 29.05.25	Пт 30.05.25	22	Аналитик
24	Исправлене ошибок	3 дней	Пт 30.05.25	Ср 04.06.25	23	Бэкэнд разработчик[50%];Фрон
25	Тестирование завершено	0 дней	Ср 04.06.25	Ср 04.06.25	24	
26	Конец проекта	0 дней	Ср 04.06.25	Ср 04.06.25	25	

Рисунок 30 – Список задач

Для работы над проектом был определен список ресурсов, состоящий из трех сотрудников. На рисунке 31 представлен список используемы в проекте ресурсов.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочнь	Затраты на использ.	Начисление	Базовый календарь
1	Бэкэнд разработчик	Трудовой		Б		100%	400,00р./час	850,00р./час	0,00р.	Пропорциональное	Стандартный
2	Аналитик	Трудовой		А		100%	350,00р./час	750,00р./час	0,00р.	Пропорциональное	Стандартный
3	Фронтэнд разработчик	Трудовой		Ф		100%	370,00р./час	800,00р./час	0,00р.	Пропорциональное	Стандартный

Рисунок 31 – Список ресурсов

Полученная в результате планирования проекта диаграмма Ганта представлена на рисунке 32.

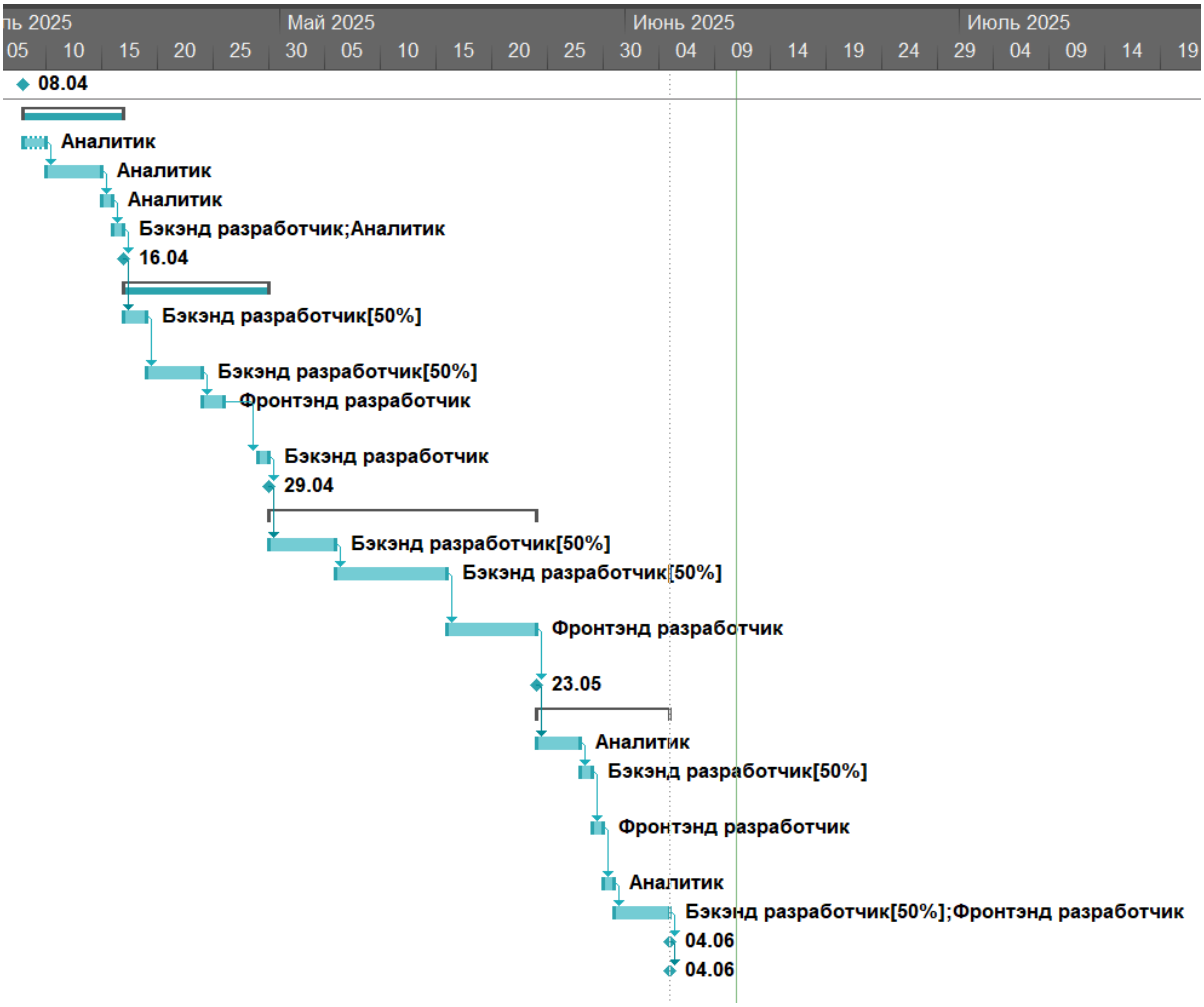

Рисунок 32 – Диаграмма Ганта

Также были подсчитаны трудозатраты человеко-часов, отражающее количество времени, затраченное каждым сотрудником на выполнение проекта. На рисунке 33 представлены трудозатраты каждого из сотрудников.

СОСТОЯНИЕ РЕСУРСОВ

Оставшиеся трудозатраты для всех трудовых ресурсов.

Название	Начало	Окончание	Оставшиеся трудозатраты
Бэкэнд разработчик	Вт 08.04.25	Ср 04.06.25	128 часов
Аналитик	Вт 08.04.25	Пт 30.05.25	80 часов
Фронтэнд разработчик	Ср 23.04.25	Ср 04.06.25	96 часов

Рисунок 33 – Трудозатраты сотрудников

Затраты на разработку проекта представлены на рисунке 34.

СВЕДЕНИЯ О ЗАТРАТАХ

Сведения о затратах для всех задач верхнего уровня.

Название	Фиксированные затраты	Фактические затраты	Оставшиеся затраты	Затраты	Базовые затраты	Отклонение по стоимости
Начало работы над проектом	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.	0,00р.
Анализ требований	0,00р.	0,00р.	34 000,00р.	34 000,00р.	0,00р.	34 000,00р.
Проектирование приложения	0,00р.	0,00р.	17 120,00р.	17 120,00р.	0,00р.	17 120,00р.
Реализация приложения	0,00р.	0,00р.	36 960,00р.	36 960,00р.	0,00р.	36 960,00р.
Тестирование приложения	0,00р.	0,00р.	26 640,00р.	26 640,00р.	0,00р.	26 640,00р.

Рисунок 34 – Затраты на разработку проекта

Из предоставленных выше данных был выявлено, что затраты на разработку проекта составят 114 720 рублей, и трудозатрат в размере 304 человеко-часов. Срок выполнения проекта составляет 57 дней.

Заключение

В рамках выполнения выпускной квалификационной работы было разработано веб-приложение для ведения листа персонажа в настольных ролевых играх.

На этапе анализа требование было проведено сравнение с существующими аналогами. На основе полученных данных были выделены функциональные и нефункциональные требования к приложению. Была создана диаграмма вариантов использования, которая отображает все функции интерактивного листа персонажа.

Для реализации проекта было выполнено планирование с оценкой сроков и бюджета разработки. Проектирование проекта включало в себя создание архитектуры приложения, проектирование базы данных, разработку пользовательского интерфейса и программных компонентов приложения.

Серверная часть приложения была выполнена с использованием платформы для разработки веб-приложений ASP.NET Core в среде разработки Visual Studio 2022. На этапе реализации была получена физическая модель базы данных в виде документа коллекции. MongoDB выступила в качестве базы данных в данном приложении.

Клиентская часть веб-приложения была выполнена на HTML, CSS, JavaScript. Также применялся фреймворк Bootstrap, который обеспечивал единый дизайн и адаптивность страниц.

Тестирование приложения показало его корректную работу, все необходимые тест-кейсы были успешно пройдены. Полученный метрики кода показали среднюю сложность кода, что обеспечит долгосрочную поддержку программы.

Таким образом, цели выпускной квалификационной работы были достигнуты, а полученный интерактивный лист персонажа соответствует заявленным требованиям.

Список использованных источников

1. Мирлс. М. Книга игрока. Dungeons & Dragons 5я редакция/ М. Мирс, Дж. Кроуфорд. – Renton, WA: Wizards of the Coast, 2014. – 320 с.

2. Айдин, С., & Боднар, А. (2024). Применение информационных технологий в сфере настольных ролевых игр. *Информатика и кибернетика*, (1(35), 24–32. извлечено от https://ojs.donntu.ru/index.php/infcyb/article/view/171

3. Роман Д., Виртуальные игровые столы, цифровые кости: как играть в настольные ролевые игры, не выходя из дома: DTF. 2020. [Электронный ресурс]. – URL: https://dtf.ru/boardgames/118417-virtualnye-stoly-cifrovye-kosti-kak-igrat-v-nastolnye-rolevye-igry-ne-vyhodya-iz-doma

4. Александр С., Павел И., Цифровые помощники и виртуальные столы: приложения и сервисы для настольных ролевых игр: Мир фантастики и фэнтези. 2020. [Электронный ресурс]. – URL: https://www.mirf.ru/games/cifrovye-pomoschniki-i-virtualnye-stoly-prilozheniya-i-servisy-dlya-nastolnyh-rolevyh-igr

5. Галиаскаров, Э.Г. Анализ и проектирование систем с использованием UML: учебник для вузов / Э. Г. Галиаскаров, А. С. Воробьев. – Москва: Издательство Юрайт, 2025. – 125 с.

6. Вигерс Карл И., Битти Джой. Разработка требований к программному обеспечению. 3-е изд., дополненное / Пер. с англ. – М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2014. – 736 с.

7. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с.

8. Осипов Д. Л. Технологии проектирования баз данных. – М.: ДМК Пресс, 2019. – 498 с.

9. Нереляционные данные и базы данных NoSQL [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data

10. Проектирование пользовательского интерфейса [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/windows/win32/appuistart/designing-a-user-interface

11. Полуэктова Н.Р. Разработка веб-приложений: учебник для вузов / Н. Р. Полуэктова – 2-е изд. – Москва: Издательство Юрайт, 2025. – 204 с.

12. Кайл Бэнкер. MongoDB в действии. /Пер. с англ. Слинкина А. А. – М.: ДМК Пресс, 2012. – 394 с.

13. Марк Прайс. C# 10 и .NET 6. Современная кросс-платформенная разработка. – СПб Питер, 2023. – 848 с.

14. Сильиво Морето. Bootstrap в примерах. /Пер. с англ. Рагимов Р. Н. / Науч. ред. Киселев А. Н. – М.: ДМК Пресс, 2017. – 314 с.

15. Флэнаган, Дэвид. JavaScript. Подробное руководство, 7-е изд. – Пер. с англ. – ООО «Диалектика», 2021. – 720 с.

16. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с

17. Тестирование служб и веб-приложений ASP.NET Core [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps

18. Postman для эффективного тестирования API [Электронный ресурс]. URL: https://habr.com/ru/companies/otus/articles/740884/

19. Евсеев В. В. Применение программных метрик кода на раннем этапе жизненного цикла программного обеспечения // ВЕЖПТ. 2011. №2 (49). URL: https://cyberleninka.ru/article/n/primenenie-programmnyh-metrik-koda-na-rannem-etape-zhiznennogo-tsikla-programmnogo-obespecheniya

20. Боронина, Л. Н. Основы управления проектами: [учеб. пособие] / Л. Н. Боронина, З. В. Сенук; М-во образования и науки Рос. Федерации, Урал. федер. ун-т. – Екатеринбург: Изд-во Урал. ун-та, 2015. – 112 с.

Приложение А

Листинг документа MongoDB

(рекомендуемо)

Листинг А.1 – Макет документа коллекции

```
{
  "_id": "",
  "Name": "",
  "Race": "",
  "Class": "",
  "Level": 1,
  "Background": "",
  "Alignment": "",
  "Experience": 0,
  "Abilities": [
    {
      "Name": "Strength",
      "Score": 10
    },
    {
      "Name": "Dexterity",
      "Score": 10
    },
    {
      "Name": "Constitution",
      "Score": 10
    },
    {
      "Name": "Intelligence",
      "Score": 10
    },
    {
      "Name": "Wisdom",
      "Score": 10
    },
    {
      "Name": "Charisma",
      "Score": 10
    }
  ],
  "ArmorClass": 10,
  "Initiative": 0,
  "Speed": 30,
  "HitPoints": 10,
  "HitDice": 1,
  "HitDiceType": "d6",
  "Skills": {
    "Acrobatics": false,
    "Animal Handling": false,
    "Arcana": false,
    "Athletics": false,
    "Deception": false,
    "History": false,
    "Insight": false,
    "Intimidation": false,
    "Investigation": false,
```

```
      "Medicine": false,
      "Nature": false,
      "Perception": false,
      "Performance": false,
      "Persuasion": true,
      "Religion": false,
      "Sleight of Hand": false,
      "Stealth": false,
      "Survival": false
    },
    "Proficiencies": "",
    "Features": "",
    "Attacks": [
      {
        "Name": "",
        "AttackBonus": "",
        "Damage": "",
        "DamageType": "",
        "Notes": ""
      }
    ],
    "Equipment": [],
    "Spells": [
      {
        "Name": "",
        "Level": 0,
        "School": "",
        "CastingTime": "",
        "Range": "",
        "Components": "",
        "Duration": "",
        "Description": ""
      }
    ],
    "PersonalityTraits": "",
    "Ideals": "",
    "Bonds": "",
    "Flaws": "",
    "Notes": "",
    "Appearance": ""
}
```

Приложение Б

Листинг серверной части приложения

(обязательное)

Листинг Б.1 – Файл Program

```
using DndCharacterSheet.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();
builder.Services.Configure<MongoDbSettings>(builder.Configuratio
n.GetSection("MongoDbSettings"));
builder.Services.AddSingleton<MongoDbContext>();

var app = builder.Build();

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseCors(policy =>
policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

app.UseRouting();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapControllerRoute(
    name: "character",
    pattern: "Character/Index/{id}",
    defaults: new { controller = "Character", action = "Index"
});

app.MapGet("/", async context => {
    context.Response.ContentType = "text/html";
    await
context.Response.SendFileAsync(Path.Combine(app.Environment.WebR
ootPath, "index.html"));
});

app.Use(async (context, next) =>
{
    Console.WriteLine($"Incoming request:
{context.Request.Method} {context.Request.Path}");
    await next();
});

app.UseCors(policy =>
policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

app.MapControllers();

app.Run();
```

Листинг Б.2 – Файл Controller/CharacterController

```
using DndCharacterSheet.Models;
using Microsoft.AspNetCore.Mvc;
using MongoDB.Bson;
using MongoDB.Driver;


namespace DndCharacterSheet.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CharacterController : Controller
    {
        private readonly MongoDbContext _context;

        public CharacterController(MongoDbContext context)
        {
            _context = context;
        }



        [HttpPost("index")]
        public async Task<IActionResult> Index(string id)
        {
            var character = await _context.Characters.Find(c =>
c.Id == id).FirstOrDefaultAsync();
            if (character == null)
            {
                return NotFound();
            }
            return View(character);
        }

        //Обновить данные о персонаже
        [HttpPost("update")]
        public async Task<IActionResult> Update([FromBody]
Character character)
        {
            if (character == null)
            {
                return BadRequest("Character data is null");
            }

            try
            {
                var filter = Builders<Character>.Filter.Eq(c =>
c.Id, character.Id);
                await
_context.Characters.ReplaceOneAsync(filter, character);

                return Ok(character);
```

```
            }
            catch (Exception ex)
            {
                return StatusCode(500, $"Internal server error:
{ex.Message}");
            }
        }

        //Добавить атаку
        [HttpPost("{characterId}/attacks")]
        public async Task<IActionResult> AddAttack(string
characterId, [FromBody] Attack attack)
        {
            if (string.IsNullOrEmpty(characterId))
            {
                return BadRequest("Character ID is required.");
            }

            if (attack == null)
            {
                return BadRequest("Attack data is required.");
            }

            Console.WriteLine("Received attack data: " +
System.Text.Json.JsonSerializer.Serialize(attack));

            var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
            var update = Builders<Character>.Update.Push(c =>
c.Attacks, attack);
            await _context.Characters.UpdateOneAsync(filter,
update);
            return Json(attack);
        }


        //Получить список атак
        [HttpGet("{characterId}/attacks")]
        public async Task<IActionResult> GetAttacks(string
characterId)
        {
            var character = await _context.Characters
                .Find(c => c.Id == characterId)
                .FirstOrDefaultAsync();

            if (character == null || character.Attacks == null)
            {
                return Ok(new List<Attack>());
            }

            return Ok(character.Attacks);
        }
```

```
        //Добавить заклинание
        [HttpPost("{characterId}/spells")]
        public async Task<IActionResult> AddSpell(string
characterId, [FromBody] Spell spell)
        {
            if (spell == null)
            {
                return BadRequest("Spell data is null");
            }

            try
            {
                var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
                var update = Builders<Character>.Update.Push(c
=> c.Spells, spell);

                var result = await
_context.Characters.UpdateOneAsync(filter, update);

                if (result.MatchedCount == 0)
                {
                    return NotFound($"Character with id
{characterId} not found");
                }

                return Ok(new
                {
                    success = true,
                    spell = spell
                });
            }
            catch (Exception ex)
            {
                return StatusCode(500, new
                {
                    success = false,
                    error = ex.Message
                });
            }
        }

        //Добавить снаряжение
        [HttpPost("{characterId}/equipment")]
        public async Task<IActionResult> AddEquipment(string
characterId, [FromBody] Equipment equipment)
        {
            var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
            var update = Builders<Character>.Update.Push(c =>
c.Equipment, equipment);
```

```
            await _context.Characters.UpdateOneAsync(filter,
update);
            return Json(equipment);
        }


        //Получить заклинания
        [HttpGet("{characterId}/spells")]
        public async Task<IActionResult> GetSpells(string
characterId)
        {
            try
            {
                var character = await _context.Characters
                    .Find(c => c.Id == characterId)
                    .FirstOrDefaultAsync();

                if (character == null)
                {
                    return NotFound($"Character with id
{characterId} not found");
                }

                return Ok(character.Spells ?? new
List<Spell>());
            }
            catch (Exception ex)
            {
                return StatusCode(500, new
                {
                    error = ex.Message,
                    stackTrace = ex.StackTrace
                });
            }
        }

        [HttpGet("export/{id}")]
        public async Task<IActionResult> Export(string id)
        {
            var character = await _context.Characters.Find(c =>
c.Id == id).FirstOrDefaultAsync();
            if (character == null)
            {
                return NotFound();
            }
            return Json(character);
        }

        //Получить персонажа
        [HttpGet("{id}")]
        public async Task<IActionResult> GetCharacter(string id)
        {
```

```
            var character = await _context.Characters.Find(c =>
c.Id == id).FirstOrDefaultAsync();
            if (character == null)
            {
                return NotFound();
            }


            if (character.Skills == null)
            {
                character.Skills = new Dictionary<string, bool>
        {
            {"Acrobatics", false},
            {"Animal Handling", false},
            {"Arcana", false},
            {"Athletics", false},
            {"Deception", false},
            {"History", false},
            {"Insight", false},
            {"Intimidation", false},
            {"Investigation", false},
            {"Medicine", false},
            {"Nature", false},
            {"Perception", false},
            {"Performance", false},
            {"Persuasion", false},
            {"Religion", false},
            {"Sleight of Hand", false},
            {"Stealth", false},
            {"Survival", false}
        };
            }

            return Ok(character);
        }

        //Импортировать персонажа
        [HttpPost("import")]
        public async Task<IActionResult> Import([FromBody]
Character character)
        {
            var existingCharacter = await
_context.Characters.Find(c => c.Id ==
character.Id).FirstOrDefaultAsync();

            if (existingCharacter != null)
            {
                character.Id =
ObjectId.GenerateNewId().ToString();
            }
            await _context.Characters.InsertOneAsync(character);
            return Json(character);
```

```
        }

        // Удаление атаки по индексу
        [HttpDelete("{characterId}/attacks/{attackIndex}")]
        public async Task<IActionResult> RemoveAttack(string
characterId, int attackIndex)
        {
            try
            {
                var character = await _context.Characters
                    .Find(c => c.Id == characterId)
                    .FirstOrDefaultAsync();

                if (character == null)
                    return NotFound("Character not found");

                if (attackIndex < 0 || attackIndex >=
character.Attacks.Count)
                    return BadRequest("Invalid attack index");

                character.Attacks.RemoveAt(attackIndex);

                var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
                var update = Builders<Character>.Update.Set(c =>
c.Attacks, character.Attacks);

                var result = await
_context.Characters.UpdateOneAsync(filter, update);

                return Ok(new { success = true });
            }
            catch (Exception ex)
            {
                return StatusCode(500, new { error = ex.Message
});
            }
        }

        // Удаление заклинания
        [HttpDelete("{characterId}/spells/{spellIndex}")]
        public async Task<IActionResult> RemoveSpell(string
characterId, int spellIndex)
        {
            try
            {
                //Получить персонажа из БД
                var character = await _context.Characters
                    .Find(c => c.Id == characterId)
                    .FirstOrDefaultAsync();
                //Проверка персонажа
                if (character == null)
```

```
                    return NotFound("Character not found");

                //Поиск заклинания
                if (spellIndex < 0 || spellIndex >=
character.Spells.Count)
                    return BadRequest("Invalid spell index");

                //Удаление заклинания в классе
                character.Spells.RemoveAt(spellIndex);

                //Загрузка заклинания в БД
                var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
                var update = Builders<Character>.Update.Set(c =>
c.Spells, character.Spells);

                var result = await
_context.Characters.UpdateOneAsync(filter, update);

                return Ok(new { success = true });
            }
            catch (Exception ex)
            {
                return StatusCode(500, new { error = ex.Message
});
            }
        }



        // Получить снаряжение персонажа
        [HttpGet("{characterId}/equipment")]
        public async Task<IActionResult> GetEquipment(string
characterId)
        {
            try
            {
                var character = await _context.Characters
                    .Find(c => c.Id == characterId)
                    .FirstOrDefaultAsync();

                if (character == null)
                {
                    return NotFound($"Character with id
{characterId} not found");
                }

                return Ok(character.Equipment ?? new
List<Equipment>());
            }
            catch (Exception ex)
            {
```

```
                return StatusCode(500, new
                {
                    error = ex.Message,
                    stackTrace = ex.StackTrace
                });
            }
        }

        // Удаление снаряжения по индексу
        [HttpDelete("{characterId}/equipment/{equipmentIndex}")]
        public async Task<IActionResult> RemoveEquipment(string
characterId, int equipmentIndex)
        {
            try
            {

                var character = await _context.Characters
                    .Find(c => c.Id == characterId)
                    .FirstOrDefaultAsync();

                if (character == null)
                    return NotFound("Character not found");

                if (equipmentIndex < 0 || equipmentIndex >=
character.Equipment.Count)
                    return BadRequest("Invalid equipment
index");

                character.Equipment.RemoveAt(equipmentIndex);

                var filter = Builders<Character>.Filter.Eq(c =>
c.Id, characterId);
                var update = Builders<Character>.Update.Set(c =>
c.Equipment, character.Equipment);

                var result = await
_context.Characters.UpdateOneAsync(filter, update);

                return Ok(new { success = true });
            }
            catch (Exception ex)
            {
                return StatusCode(500, new { error = ex.Message
});
            }
        }
    }
}
```

Листинг Б.3 – Файл Controller/HomeController

```
using DndCharacterSheet.Models;
using Microsoft.AspNetCore.Mvc;
```

```
using MongoDB.Bson;
using MongoDB.Driver;

[ApiController]
[Route("api/[controller]")]
public class HomeController : ControllerBase
{
    private readonly MongoDbContext _context;

    public HomeController(MongoDbContext context)
    {
        _context = context;
    }

    [HttpGet("characters")]
    public async Task<IActionResult> GetCharacters()
    {
        var characters = await _context.Characters.Find(_ =>
true).ToListAsync();
        return Ok(characters);
    }

    [HttpPost("characters")]
    public async Task<IActionResult> CreateCharacter([FromBody]
Character character)
    {
        try
        {
            if (string.IsNullOrEmpty(character.Id))
            {
                character.Id =
ObjectId.GenerateNewId().ToString();
            }
            character.Abilities ??= new List<Ability>
        {
            new Ability { Name = "Strength", Score = 10 },
            new Ability { Name = "Dexterity", Score = 10 },
            new Ability { Name = "Constitution", Score = 10 },
            new Ability { Name = "Intelligence", Score = 10 },
            new Ability { Name = "Wisdom", Score = 10 },
            new Ability { Name = "Charisma", Score = 10 }
        };

            character.Skills ??= new Dictionary<string, bool>
        {
            {"Acrobatics", false},
            {"Animal Handling", false},
            {"Arcana", false},
            {"Athletics", false},
            {"Deception", false},
            {"History", false},
            {"Insight", false},
```

```
            {"Intimidation", false},
            {"Investigation", false},
            {"Medicine", false},
            {"Nature", false},
            {"Perception", false},
            {"Performance", false},
            {"Persuasion", false},
            {"Religion", false},
            {"Sleight of Hand", false},
            {"Stealth", false},
            {"Survival", false}
        };

            character.Attacks ??= new List<Attack>();
            character.Spells ??= new List<Spell>();
            character.Equipment ??= new List<Equipment>();

            await _context.Characters.InsertOneAsync(character);
            return CreatedAtAction(nameof(GetCharacters), new {
id = character.Id }, character);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Ошибка создания персонажа:
{ex.Message}");
            Console.WriteLine($"Stack trace: {ex.StackTrace}");

            return StatusCode(500, new { error = "Internal
server error", message = ex.Message });
        }
    }

    [HttpDelete("characters/{id}")]
    public async Task<IActionResult> DeleteCharacter(string id)
    {
        await _context.Characters.DeleteOneAsync(c => c.Id ==
id);
        return NoContent();
    }
}
```

Листинг Б.4 – Файл Models/Ability

```
namespace DndCharacterSheet.Models
{
    public class Ability
    {
        public string Name { get; set; } = "";
        public int Score { get; set; }
        public int Modifier => (int)Math.Floor((Score - 10) /
2.0);
    }
}
```

Листинг Б.5 – Файл Models/Attack

```
namespace DndCharacterSheet.Models
{
    public class Attack
    {
        public string Name { get; set; } = "";
        public string AttackBonus { get; set; } = "";
        public string Damage { get; set; } = "";
        public string DamageType { get; set; } = "";
        public string Notes { get; set; } = "";
    }
}
```

Листинг Б.6 – Файл Models/Character

```
using System.Collections.Generic;

namespace DndCharacterSheet.Models
{
    public class Character
    {
        public string Id { get; set; } = "";
        public string Name { get; set; } = "";
        public string Race { get; set; } = "";
        public string Class { get; set; } = "";
        public int Level { get; set; } = 1;
        public string Background { get; set; } = "";
        public string Alignment { get; set; } = "";
        public int Experience { get; set; } = 0;

        public List<Ability> Abilities { get; set; } = new
List<Ability>
        {
            new Ability { Name = "Strength" },
            new Ability { Name = "Dexterity" },
            new Ability { Name = "Constitution" },
            new Ability { Name = "Intelligence" },
            new Ability { Name = "Wisdom" },
            new Ability { Name = "Charisma" }
        };
        public int ArmorClass { get; set; } = 10;
        public int Initiative { get; set; } = 0;
        public int Speed { get; set; } = 30;
        public int HitPoints { get; set; } = 10;
        public int HitDice { get; set; } = 1;
        public string HitDiceType { get; set; } = "d8";

        public Dictionary<string, bool> Skills { get; set; } =
new Dictionary<string, bool>
    {
        {"Acrobatics", false},
        {"Animal Handling", false},
```

```
            {"Arcana", false},
            {"Athletics", false},
            {"Deception", false},
            {"History", false},
            {"Insight", false},
            {"Intimidation", false},
            {"Investigation", false},
            {"Medicine", false},
            {"Nature", false},
            {"Perception", false},
            {"Performance", false},
            {"Persuasion", false},
            {"Religion", false},
            {"Sleight of Hand", false},
            {"Stealth", false},
            {"Survival", false}
        };

            public string Proficiencies { get; set; } = "";

            public string Features { get; set; } = "";

            public List<Attack> Attacks { get; set; } = new
List<Attack>();

            public List<Equipment> Equipment { get; set; } = new
List<Equipment>();

            public List<Spell> Spells { get; set; } = new
List<Spell>();

            public string PersonalityTraits { get; set; } = "";
            public string Ideals { get; set; } = "";
            public string Bonds { get; set; } = "";
            public string Flaws { get; set; } = "";

            public string Notes { get; set; } = "";

            public string Appearance { get; set; } = "";
        }
}
```

Листинг Б.7 - Файл Models/Equipmnet.cs

```
namespace DndCharacterSheet.Models
{
    public class Equipment
    {
        public string Name { get; set; } = "";
        public int Quantity { get; set; } = 1;
        public string Description { get; set; } = "";
    }
}
```

Листинг Б.9 – Файл Models/Spell

```
namespace DndCharacterSheet.Models
{
    public class Spell
    {
        public string Name { get; set; } = "";
        public int Level { get; set; }
        public string School { get; set; } = "";
        public string CastingTime { get; set; } = "";
        public string Range { get; set; } = "";
        public string Components { get; set; } = "";
        public string Duration { get; set; } = "";
        public string Description { get; set; } = "";
    }
}
```

Листинг Б.9 – Файл Models/MongoDBContext

```
using Microsoft.Extensions.Options;
using MongoDB.Driver;

namespace DndCharacterSheet.Models
{
    public class MongoDbContext
    {
        private readonly IMongoDatabase _database;

        public MongoDbContext(IOptions<MongoDbSettings>
settings)
        {
            var client = new
MongoClient(settings.Value.ConnectionString);
            _database =
client.GetDatabase(settings.Value.DatabaseName);
        }

        public IMongoCollection<Character> Characters =>
_database.GetCollection<Character>("Characters");
    }

    public class MongoDbSettings
    {
        public string ConnectionString { get; set; } = "";
        public string DatabaseName { get; set; } = "";
    }
}
```

Приложение В

Листинг клиентской части приложения

(обязательное)

Листинг В.1 – Код страницы списка персонажей

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>D&D Character Sheets</title>
    <link rel="stylesheet" href="/css/site.css">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boot
strap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container">
        <h1 class="my-4">Список персонажей</h1>

        <div class="d-flex justify-content-between mb-4">
            <button id="createCharacterBtn" class="btn btn-
primary">Создать новый лист персонажа</button>
            <div>
                <input type="file" id="importFile"
accept=".json" style="display: none;">
                <button id="importBtn" class="btn btn-
secondary">Импортировать лист</button>
            </div>
        </div>

        <div id="characterCards" class="row">

        </div>


        <div class="modal fade" id="createCharacterModal"
tabindex="-1">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Создать нового
персонажа</h5>
                        <button type="button" class="btn-close"
data-bs-dismiss="modal"></button>
                    </div>
                    <div class="modal-body">
                        <div class="mb-3">
                            <label class="form-label">Способ
создания</label>
                            <select id="creationMethod"
class="form-select">
                                <option value="empty">Пустой
лист</option>
```

```
                                <option value="template">По
шаблну</option>
                            </select>
                        </div>

                        <div id="templateFields" style="display:
none;">
                            <div class="mb-3">
                                <label class="form-
label">Имя</label>
                                <input type="text"
id="characterName" class="form-control">
                            </div>
                            <div class="mb-3">
                                <label class="form-
label">Раса</label>
                                <select id="characterRace"
class="form-select">
                                    <option
value="Человек">Человек</option>
                                    <option
value="Эльф">Эльф</option>
                                    <option
value="Дворф">Дворф</option>
                                    <option
value="Полурослик">Полурослик</option>
                                    <option
value="Драконорожденный">Драконорожденный</option>
                                    <option
value="Гном">Гном</option>
                                    <option
value="Полуэльф">Полуэльф</option>
                                    <option
value="Полуорк">Полуорк</option>
                                    <option
value="Тифлинг">Тифлинг</option>
                                </select>
                            </div>
                            <div class="mb-3">
                                <label class="form-
label">Класс</label>
                                <select id="characterClass"
class="form-select">
                                    <option
value="Варвар">Варвар</option>
                                    <option
value="Бард">Бард</option>
                                    <option
value="Жрец">Жрец</option>
                                    <option
value="Друид">Друид</option>
```

```
                                    <option
value="Воин">Воин</option>
                                    <option
value="Монах">Монах</option>
                                    <option
value="Паладин">Паладин</option>
                                    <option
value="Следопыт">Следопыт</option>
                                    <option
value="Плут">Плут</option>
                                    <option
value="Волшебник">Волшебник</option>
                                    <option
value="Колдун">Колдун</option>
                                    <option
value="Чародей">Чародей</option>
                                </select>
                            </div>
                            <div class="mb-3">
                                <label class="form-
label">Уровень</label>
                                <input type="number"
id="characterLevel" class="form-control" min="1" max="20"
value="1">
                            </div>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-
secondary" data-bs-dismiss="modal">Закрыть</button>
                        <button type="button"
id="confirmCreateBtn" class="btn btn-primary">Создать</button>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootst
rap.bundle.min.js"></script>
    <script src="/js/home.js"></script>
</body>
</html>
```

Листинг В.2 – Код страницы листа персонажа

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
```

```
    <title>Лист персонажа</title>
    <link rel="stylesheet" href="/css/site.css">
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.8.0/font/bootstrap-icons.css">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/boot
strap.min.css" rel="stylesheet">
</head>
<body>

    <div class="container my-4">
        <div id="topHead" class="d-flex justify-content-between
align-items-center mb-3">

            <a href="/index.html" class="btn btn-sm btn-outline-
secondary ms-3" title="Back to characters">
                <svg xmlns="http://www.w3.org/2000/svg"
width="16" height="16" fill="currentColor" class="bi bi-arrow-
left" viewBox="0 0 16 16">
                    <path fill-rule="evenodd" d="M15 8a.5.5 0 0
1-.5.5H2.707l3.147 3.146a.5.5 0 0 1-.707.708l-4-4a.5.5 0 0 1 0-
.708l4-4a.5.5 0 1 1 .707.708L2.707 7.5H14.5A.5.5 0 0 1 15 8z" />
                </svg>
            </a>
            <h3 id="characterNameDisplay">Лист персонажа</h3>
            <div>
                <button id="exportBtn" class="btn btn-sm btn-
secondary me-2">Экспорт</button>
                <button id="saveBtn" class="btn btn-sm btn-
primary">Сохранить</button>
            </div>
        </div>

        <div id="diceResult" class="dice-result alert alert-
success d-none position-fixed top-0 end-0 mt-3 me-3 rounded">
            <strong>Результат:</strong> <span
id="diceResultValue">0</span>
        </div>

        <div id="diceHistory" class="position-fixed bottom-0
start-0 mb-5 ms-3 p-3 bg-light rounded-3 shadow-sm d-none"
style="max-height: 200px; overflow-y: auto; width: 250px;">
            <h6 class="mb-2">История бросков</h6>
            <ul id="historyList" class="list-unstyled mb-
0"></ul>
            <button id="clearHistory" class="btn btn-sm btn-
outline-danger mt-2 w-100">Очистить</button>
        </div>
```

```
        <button id="toggleDiceBtn" class="position-fixed bottom-
0 end-0 m-3 btn btn-primary btn-sm rounded-circle" style="width:
40px; height: 40px;">
            <i class="bi bi-dice-6"></i>
        </button>


        <button id="toggleHistory" class="position-fixed bottom-
0 start-0 m-3 btn btn-primary btn-sm rounded-circle"
style="width: 40px; height: 40px;">
            <i class="bi bi-clock-history"></i>
        </button>

        <div id="diceContainer" class="dice-container position-
fixed bottom-0 end-0 mb-3 me-3 d-none">
            <div class="dice-container-vertical">
                <button type="button" class="btn-dice d4"
title="D4">
                    <img src="images/d4.svg" alt="D4">
                </button>
                <button type="button" class="btn-dice d6"
title="D6">
                    <img src="images/d6.svg" alt="D6">
                </button>
                <button type="button" class="btn-dice d8"
title="D8">
                    <img src="images/d8.svg" alt="D8">
                </button>
                <button type="button" class="btn-dice d10"
title="D10">
                    <img src="images/d10.svg" alt="D10">
                </button>
                <button type="button" class="btn-dice d12"
title="D12">
                    <img src="images/d12.svg" alt="D12">
                </button>
                <button type="button" class="btn-dice d20"
title="D20">
                    <img src="images/d20.svg" alt="D20">
                </button>
            </div>
        </div>

        <div class="row mb-3">
            <div class="col-md-6">
                <div class="card">
                    <div id="cardHeaderInfo" class="card-header
py-2">
                        <h6 class="mb-0">Основная
информация</h6>
                    </div>
                    <div class="card-body p-2">
```

```
                        <div class="row g-1">
                            <div class="col-8">
                                <label class="small">Имя</label>
                                <input type="text" id="name"
class="form-control form-control-sm" placeholder="Name">
                            </div>
                            <div class="col-4">
                                <label
class="small">Расса</label>
                                <input type="text" id="race"
class="form-control form-control-sm" placeholder="Race">
                            </div>
                            <div class="col-7">
                                <label
class="small">Класс</label>
                                <input type="text" id="class"
class="form-control form-control-sm" placeholder="Class">
                            </div>
                            <div class="col-5">
                                <label
class="small">Происхождение</label>
                                <input type="text"
id="background" class="form-control form-control-sm"
placeholder="Background">
                            </div>
                            <div class="col-6">
                                <label
class="small">Мировоззрение</label>
                                <input type="text"
id="alignment" class="form-control form-control-sm"
placeholder="Alignment">
                            </div>
                            <div class="col-3">
                                <label
class="small">Опыт</label>
                                <input type="number"
id="experience" class="form-control form-control-sm"
placeholder="XP">
                            </div>
                            <div class="col-3">
                                <label
class="small">Уровень</label>
                                <input type="number" id="level"
class="form-control form-control-sm" placeholder="Level" min="1"
max="20" readonly>
                            </div>

                        </div>
                    </div>
                </div>
            </div>
```

```
            <div class="col-md-6">
                <div class="card">
                    <div id="cardHeaderBattle" class="card-
header py-2">
                        <h6 class="mb-0">Боевые параметры</h6>
                    </div>
                    <div class="card-body p-2">
                        <div class="row g-1">
                            <div class="col-4">
                                <label class="small">Класс
защиты</label>
                                <input type="number"
id="armorClass" class="form-control form-control-sm">
                            </div>
                            <div class="col-4">
                                <label
class="small">Инициатива</label>
                                <input type="number"
id="initiative" class="form-control form-control-sm">
                            </div>
                            <div class="col-4">
                                <label
class="small">Скорость</label>
                                <input type="number" id="speed"
class="form-control form-control-sm">
                            </div>
                            <div class="col-2">
                                <label
class="small">Хиты</label>
                                <input type="number"
id="hitPoints" class="form-control form-control-sm">
                            </div>
                            <div class="col-3">
                                <label class="small">Кость
хитов</label>
                                <select id="hitDiceType"
class="form-select form-select-sm">
                                    <option
value="d6">d6</option>
                                    <option value="d8"
selected>d8</option>
                                    <option
value="d10">d10</option>
                                    <option
value="d12">d12</option>
                                </select>
                            </div>
                            <div class="col-3">
                                <label class="small">Кол-во
костей</label>
                                <input type="number"
id="hitDice" class="form-control form-control-sm" readonly>
```

```
                            </div>
                            <div class="col-4">
                                <label class="small">Бонус
мастерства</label>
                                <input type="number"
id="proficiencyBonus" class="form-control form-control-sm"
readonly>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="progress-container mt-2">
                    <div class="d-flex justify-content-between
align-items-center small mb-1">
                        <span class="text-nowrap">Прогресс до
следующего уровня </span>
                        <span id="xpProgressText" class="text-
nowrap ps-2">0 / 300 XP</span>
                    </div>
                    <div class="progress" style="height: 20px;">
                        <div id="experienceProgress"
class="progress-bar"
                             role="progressbar" style="width:
0%; background-color: #FFD166;"
                             aria-valuenow="0" aria-valuemin="0"
aria-valuemax="100"></div>
                    </div>
                </div>
            </div>

            <div class="row">
                <div class="col-sm-4">
                    <div class="d-flex flex-row flex-nowrap
overflow-auto">
                        <div class="card mb-1 ability-
countainer">
                            <div class="card-body p-2">
                                <div class="ability-row">
                                    <div class="ability-
name">СИЛ</div>
                                    <input type="number"
id="strength" class="form-control form-control-sm ability-
score">
                                    <div class="ability-
modifier" id="strengthMod">+0</div>
                                </div>
                                <div class="ability-row">
                                    <div class="ability-
name">ЛОВ</div>
                                    <input type="number"
id="dexterity" class="form-control form-control-sm ability-
score">
```

```
                                    <div class="ability-
modifier" id="dexterityMod">+0</div>
                                </div>
                                <div class="ability-row">
                                    <div class="ability-
name">ВЫН</div>
                                    <input type="number"
id="constitution" class="form-control form-control-sm ability-
score">
                                    <div class="ability-
modifier" id="constitutionMod">+0</div>
                                </div>
                                <div class="ability-row">
                                    <div class="ability-
name">ИНТ</div>
                                    <input type="number"
id="intelligence" class="form-control form-control-sm ability-
score">
                                    <div class="ability-
modifier" id="intelligenceMod">+0</div>
                                </div>
                                <div class="ability-row">
                                    <div class="ability-
name">МУД</div>
                                    <input type="number"
id="wisdom" class="form-control form-control-sm ability-score">
                                    <div class="ability-
modifier" id="wisdomMod">+0</div>
                                </div>
                                <div class="ability-row">
                                    <div class="ability-
name">ХАР</div>
                                    <input type="number"
id="charisma" class="form-control form-control-sm ability-
score">
                                    <div class="ability-
modifier" id="charismaMod">+0</div>
                                </div>
                            </div>
                        </div>

                        <div class="card skills-box flex-shrink-
0 mt-3">
                            <div id="cardHeaderSkills"
class="card-header py-2">
                                <h6 class="mb-0">Навыки</h6>
                            </div>
                            <div class="card-body p-2">
                                <div id="skillsList"
class="skills-container">

                                </div>
```

```
                            </div>
                        </div>
                    </div>
                    </div>

                    <div class="col-sm-8">
                        <ul class="nav nav-tabs mb-2"
id="characterTabs" role="tablist">
                            <li class="nav-item"
role="presentation">
                                <button class="nav-link active"
id="attacks-tab" data-bs-toggle="tab" data-bs-target="#attacks"
type="button">Атаки</button>
                            </li>
                            <li class="nav-item"
role="presentation">
                                <button class="nav-link"
id="spells-tab" data-bs-toggle="tab" data-bs-target="#spells"
type="button">Заклинания</button>
                            </li>
                            <li class="nav-item"
role="presentation">
                                <button class="nav-link"
id="equipment-tab" data-bs-toggle="tab" data-bs-
target="#equipment" type="button">Снаряжение</button>
                            </li>
                            <li class="nav-item"
role="presentation">
                                <button class="nav-link"
id="personality-tab" data-bs-toggle="tab" data-bs-
target="#personality" type="button">Персональная
ифнормация</button>
                            </li>
                            <li class="nav-item"
role="presentation">
                                <button class="nav-link"
id="notes-tab" data-bs-toggle="tab" data-bs-target="#notes"
type="button">Заметки</button>
                            </li>
                        </ul>

                        <div class="tab-content"
id="characterTabsContent">
                            <div class="tab-pane fade show
active" id="attacks" role="tabpanel">
                                <div class="mb-2">
                                    <button id="addAttackBtn"
class="btn btn-sm btn-primary">Добавить атаку</button>
                                </div>
                                <div id="attacksList" class="row
row-cols-1 row-cols-md-2 g-2">
                                </div>
```

```
                                <div class="modal fade"
id="attackModal" tabindex="-1" data-mode="add">
                                    <div class="modal-dialog">
                                        <div class="modal-
content">
                                            <div class="modal-
header">
                                                <h5
class="modal-title">Добавить атаку</h5>
                                                <button
type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
                                            </div>
                                            <div class="modal-
body">
                                                <div class="mb-
3">
                                                    <label
class="form-label">Название</label>
                                                    <input
type="text" id="attackName" class="form-control">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Бонус атаки</label>
                                                    <input
type="text" id="attackBonus" class="form-control">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Урон</label>
                                                    <input
type="text" id="attackDamage" class="form-control">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Тип урона</label>
                                                    <input
type="text" id="attackDamageType" class="form-control">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Описание</label>
                                                    <textarea
id="attackNotes" class="form-control" rows="3"></textarea>
                                                </div>
                                            </div>
```

```
                                                <div class="modal-
footer">
                                                    <button
type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Закрыть</button>
                                                    <button
type="button" id="saveAttackBtn" class="btn btn-
primary">Сохранить</button>
                                                </div>
                                            </div>
                                        </div>
                                    </div>

                                    <div class="tab-pane fade"
id="spells" role="tabpanel">
                                        <div class="mb-2">
                                            <button id="addSpellBtn"
class="btn btn-sm btn-primary">Добавить заклинание</button>
                                        </div>
                                        <div id="spellsList">

                                        </div>

                                        <div class="modal fade"
id="spellModal" tabindex="-1" data-mode="add">
                                            <div class="modal-dialog">
                                                <div class="modal-
content">
                                                    <div class="modal-
header">
                                                        <h5
class="modal-title">Добавить заклинание</h5>
                                                        <button
type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
                                                    </div>
                                                    <div class="modal-
body">
                                                        <div class="mb-
3">
                                                            <label
class="form-label">Название</label>
                                                            <input
type="text" id="spellName" class="form-control" required>
                                                        </div>
                                                        <div class="mb-
3">
                                                            <label
class="form-label">Уровень</label>
                                                            <select
id="spellLevel" class="form-select">
```

```
                                                    <option
value="0">Заговор</option>
                                                    <option
value="1">1</option>
                                                    <option
value="2">2</option>
                                                    <option
value="3">3</option>
                                                    <option
value="4">4</option>
                                                    <option
value="5">5</option>
                                                    <option
value="6">6</option>
                                                    <option
value="7">7</option>
                                                    <option
value="8">8</option>
                                                    <option
value="9">9</option>
                                                </select>
                                            </div>
                                            <div class="mb-
3">
                                                <label
class="form-label">Школа магии</label>
                                                <select
id="spellSchool" class="form-select">
                                                    <option
value="Ограждения">Ограждения</option>
                                                    <option
value="Призыва">Призыва</option>
                                                    <option
value="Прорицания">Прорицания</option>
                                                    <option
value="Очарования">Очарования</option>
                                                    <option
value="Воплощения">Воплощения</option>
                                                    <option
value="Иллюзии">Иллюзии</option>
                                                    <option
value="Некромантии">Некромантии</option>
                                                    <option
value="Преобразования">Преобразования</option>
                                                </select>
                                            </div>
                                            <div class="mb-
3">
                                                <label
class="form-label">Время накладывания</label>
                                                <input
type="text" id="spellCastingTime" class="form-control" required>
```

```
                                                    </div>
                                                    <div class="mb-
3">
                                                        <label
class="form-label">Дистанция</label>
                                                        <input
type="text" id="spellRange" class="form-control" required>
                                                    </div>
                                                    <div class="mb-
3">
                                                        <label
class="form-label">Компоненты</label>
                                                        <input
type="text" id="spellComponents" class="form-control" required>
                                                    </div>
                                                    <div class="mb-
3">
                                                        <label
class="form-label">Длительность</label>
                                                        <input
type="text" id="spellDuration" class="form-control" required>
                                                    </div>
                                                    <div class="mb-
3">
                                                        <label
class="form-label">Описание</label>
                                                        <textarea
id="spellDescription" class="form-control" rows="4"
required></textarea>
                                                    </div>
                                                </div>
                                                <div class="modal-
footer">
                                                    <button
type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Закрыть</button>
                                                    <button
type="submit" id="saveSpellBtn" class="btn btn-
primary">Сохранить</button>
                                                </div>
                                            </div>
                                        </div>
                                    </div>
                                </div>

                                <div class="tab-pane fade"
id="equipment" role="tabpanel">
                                    <div class="mb-2">
                                        <button id="addEquipmentBtn"
class="btn btn-sm btn-primary">Добавить снаряжение</button>
                                    </div>
                                    <div id="equipmentList">
```

```
                                </div>

                                <div class="modal fade"
id="equipmentModal" tabindex="-1" data-mode="add">
                                    <div class="modal-dialog">
                                        <div class="modal-
content">
                                            <div class="modal-
header">
                                                <h5
class="modal-title">Добавить снаряжение</h5>
                                                <button
type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
                                            </div>
                                            <div class="modal-
body">
                                                <div class="mb-
3">
                                                    <label
class="form-label">Название</label>
                                                    <input
type="text" id="equipmentName" class="form-control">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Количество</label>
                                                    <input
type="number" id="equipmentQuantity" class="form-control"
value="1" min="1">
                                                </div>
                                                <div class="mb-
3">
                                                    <label
class="form-label">Описание</label>
                                                    <textarea
id="equipmentDescription" class="form-control"
rows="4"></textarea>
                                                </div>
                                            </div>
                                            <div class="modal-
footer">
                                                <button
type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Закрыть</button>
                                                <button
type="button" id="saveEquipmentBtn" class="btn btn-
primary">Сохранить</button>
                                            </div>
                                        </div>
```

```
                                    </div>
                                </div>
                            </div>

                            <div class="tab-pane fade"
id="personality" role="tabpanel">
                                <div class="mb-2">
                                    <label
class="small">Личность</label>
                                    <textarea
id="personalityTraits" class="form-control form-control-sm"
rows="2"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Идеалы</label>
                                    <textarea id="ideals"
class="form-control form-control-sm" rows="2"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Привязанности</label>
                                    <textarea id="bonds"
class="form-control form-control-sm" rows="2"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Слабости</label>
                                    <textarea id="flaws"
class="form-control form-control-sm" rows="2"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Предыстория</label>
                                    <textarea id="appearance"
class="form-control form-control-sm" rows="2"></textarea>
                                </div>
                            </div>


                            <div class="tab-pane fade"
id="notes" role="tabpanel">
                                <div class="mb-2">
                                    <label
class="small">Особенности расы и класса</label>
                                    <textarea id="features"
class="form-control form-control-sm" rows="3"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Владения</label>
```

```
                                    <textarea id="proficiencies"
class="form-control form-control-sm" rows="3"></textarea>
                                </div>
                                <div class="mb-2">
                                    <label
class="small">Заметки</label>
                                    <textarea id="notesText"
class="form-control form-control-sm" rows="4"></textarea>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootst
rap.bundle.min.js"></script>
        <script src="/js/character.js"></script>
</body>
</html>
```

Листинг В.3 – Скрипты для главной страницы

```
document.addEventListener('DOMContentLoaded', function () {
    loadCharacters();

    // Create character button

document.getElementById('createCharacterBtn').addEventListener('
click', function () {
        const modal = new
bootstrap.Modal(document.getElementById('createCharacterModal'))
;
        modal.show();
    });


document.getElementById('importBtn').addEventListener('click',
function () {
        document.getElementById('importFile').click();
    });

document.getElementById('importFile').addEventListener('change',
function (e) {
        const file = e.target.files[0];
        if (!file) return;

        const reader = new FileReader();
        reader.onload = function (e) {
            try {
                const character = JSON.parse(e.target.result);
```

```
                fetch('/api/character/import', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify(character)
                })
                    .then(response => response.json())
                    .then(data => {
                        loadCharacters();
                    })
                    .catch(error => {
                        console.error('Error importing
character:', error);
                        alert('Error importing character');
                    });
            } catch (error) {
                console.error('Error parsing JSON:', error);
                alert('Invalid JSON file');
            }
        };
        reader.readAsText(file);
    });


document.getElementById('creationMethod').addEventListener('chan
ge', function () {
        const templateFields =
document.getElementById('templateFields');
        if (this.value === 'template') {
            templateFields.style.display = 'block';
        } else {
            templateFields.style.display = 'none';
        }
    });


document.getElementById('confirmCreateBtn').addEventListener('cl
ick', function () {
        const method =
document.getElementById('creationMethod').value;

        if (method === 'empty') {
            createEmptyCharacter();
        } else {
            createTemplateCharacter();
        }
    });
});

function createCharacterCard(character) {
    const col = document.createElement('div');
```

```
    col.className = 'col-md-4 mb-4';

    const card = document.createElement('div');
    card.className = 'card character-card';

    card.dataset.id = character.Id;

    const cardBody = document.createElement('div');
    cardBody.className = 'card-body';

    const title = document.createElement('h5');
    title.className = 'card-title';
    title.textContent = character.name || 'Нет имени';

    const subtitle = document.createElement('h6');
    subtitle.className = 'card-subtitle mb-2 text-muted';
    subtitle.textContent = `${character.race || 'Нет расы'}
${character.class || 'Нет класса'}`;

    const level = document.createElement('p');
    level.className = 'card-text';
    level.textContent = `Уровень ${character.level || '1'}-й`;

    const btnContainer = document.createElement('div');
    btnContainer.className = 'mt-auto d-flex justify-content-
between';

    const deleteBtn = document.createElement('button');
    deleteBtn.className = 'btn btn-danger btn-sm mt-2';
    deleteBtn.textContent = 'Удалить персонажа';
    console.log("Delete BUTTON")

    deleteBtn.addEventListener('click', function (e) {
        e.stopPropagation();
        deleteCharacter(character.id);
    });

    btnContainer.appendChild(deleteBtn);

    cardBody.appendChild(title);
    cardBody.appendChild(subtitle);
    cardBody.appendChild(level);
    cardBody.appendChild(btnContainer);
    card.appendChild(cardBody);
    col.appendChild(card);

    card.addEventListener('click', function () {
        window.location.href =
`/character.html?id=${character.id}`;
    });

    return col;
```

```
}

function deleteCharacter(id) {
    if (confirm('Вы уверены что хотите удалить персонажа?')) {
        fetch(`/api/home/characters/${id}`, {
            method: 'DELETE'
        })
            .then(response => {
                if (response.ok) {
                    loadCharacters();
                } else {
                    throw new Error('Failed to delete
character');
                }
            })
            .catch(error => {
                console.error('Error deleting character:',
error);
                alert('Error deleting character');
            });
    }
}

function createEmptyCharacter() {
    const character = {
        name: 'Новый персонаж',
        race: '',
        class: '',
        level: 1,
        background: '',
        alignment: '',
        experience: 0,
        armorClass: 10,
        initiative: 0,
        speed: 30,
        hitPoints: 10,
        hitDice: 1,
        hitDiceType: 'd8',
        abilities: [
            { name: "Strength", score: 10 },
            { name: "Dexterity", score: 10 },
            { name: "Constitution", score: 10 },
            { name: "Intelligence", score: 10 },
            { name: "Wisdom", score: 10 },
            { name: "Charisma", score: 10 }
        ],
        skills: {
            "Acrobatics": false,
            "Animal Handling": false,
            "Arcana": false,
            "Athletics": false,
            "Deception": false,
```

```
            "History": false,
            "Insight": false,
            "Intimidation": false,
            "Investigation": false,
            "Medicine": false,
            "Nature": false,
            "Perception": false,
            "Performance": false,
            "Persuasion": false,
            "Religion": false,
            "Sleight of Hand": false,
            "Stealth": false,
            "Survival": false
        },
        attacks: [],
        spells: [],
        equipment: [],
        personalityTraits: '',
        ideals: '',
        bonds: '',
        flaws: '',
        appearance: '',
        features: '',
        proficiencies: '',
        notes: ''
    };

    fetch('/api/home/characters', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(character)
    })
        .then(response => {
            if (!response.ok) {
                return response.json().then(err => { throw new
Error(err.message || 'Failed to create character'); });
            }
            return response.json();
        })
        .then(data => {
            const modal =
bootstrap.Modal.getInstance(document.getElementById('createChara
cterModal'));
            modal.hide();
            loadCharacters();
        })
        .catch(error => {
            console.error('Error creating character:', error);
            alert(`Error creating character: ${error.message}`);
        });
```

```
}

function createTemplateCharacter() {
    const character = {
        name: document.getElementById('characterName').value ||
'Новый персонаж',
        race: document.getElementById('characterRace').value,
        class: document.getElementById('characterClass').value,
        level:
parseInt(document.getElementById('characterLevel').value) || 1
    };

    character.abilities = [
        { name: "Strength", score: 10 },
        { name: "Dexterity", score: 10 },
        { name: "Constitution", score: 10 },
        { name: "Intelligence", score: 10 },
        { name: "Wisdom", score: 10 },
        { name: "Charisma", score: 10 }
    ];

    character.hitPoints = 10 +
Math.floor((character.abilities.find(a => a.name ===
"Constitution").score - 10) / 2);

    let diceSize =
        character.class === 'Wizard' || character.class ===
'Sorcerer' ? '6' :
            ['Bard', 'Cleric', 'Druid', 'Monk', 'Rogue',
'Warlock'].includes(character.class) ? '8' :
                ['Fighter', 'Paladin',
'Ranger'].includes(character.class) ? '10' : '12';

    character.hitDice = 1;
    character.hitDiceType = `d${diceSize}`;
    character.armorClass = 10;
    character.initiative = 0;
    character.speed = 30;
    character.experience = 0;
    character.background = '';
    character.alignment = '';
    character.skills = {
        "Acrobatics": false,
        "Animal Handling": false,
        "Arcana": false,
        "Athletics": false,
        "Deception": false,
        "History": false,
        "Insight": false,
        "Intimidation": false,
        "Investigation": false,
        "Medicine": false,
```

```
        "Nature": false,
        "Perception": false,
        "Performance": false,
        "Persuasion": false,
        "Religion": false,
        "Sleight of Hand": false,
        "Stealth": false,
        "Survival": false
    };
    character.attacks = [];
    character.spells = [];
    character.equipment = [];
    character.personalityTraits = '';
    character.ideals = '';
    character.bonds = '';
    character.flaws = '';
    character.appearance = '';
    character.features = '';
    character.proficiencies = '';
    character.notes = '';

    fetch('/api/home/characters', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(character)
    })
        .then(response => response.json())
        .then(data => {
            const modal =
bootstrap.Modal.getInstance(document.getElementById('createChara
cterModal'));
            modal.hide();
            loadCharacters();
        })
        .catch(error => {
            console.error('Error creating character:', error);
            alert('Error creating character');
        });
}

async function loadCharacters() {
    try {
        const response = await fetch('/api/home/characters');
        if (!response.ok) {
            throw new Error(`HTTP error! status:
${response.status}`);
        }
        const characters = await response.json();
        const container =
document.getElementById('characterCards');
```

```
        container.innerHTML = '';

        characters.forEach(character => {
            const card = createCharacterCard(character);
            container.appendChild(card);
        });
    } catch (error) {
        console.error('Error loading characters:', error);
        alert('Failed to load characters. See console for
details.');
    }
}
```

Листинг В.4 – Скрипты для страницы листа персонажа

```
document.addEventListener('DOMContentLoaded', function () {
    const urlParams = new
URLSearchParams(window.location.search);
    const characterId = urlParams.get('id');

    console.log('Loading character with ID:', characterId);

    if (!characterId) {
        alert("Character ID is missing from the URL");
        window.location.href = '/index.html';
        return;
    }

    const experienceTable = [
        0, 300, 900, 2700, 6500, 14000, 23000, 34000, 48000,
64000,
        85000, 100000, 120000, 140000, 165000, 195000, 225000,
265000, 305000, 355000
    ];

    function calculateLevelAndProficiencyBonus(xp) {
        let level = 1;
        for (let i = 1; i < experienceTable.length; i++) {
            if (xp >= experienceTable[i]) {
                level = i + 1;
            } else {
                break;
            }
        }
        const proficiencyBonus = Math.ceil(level / 4) + 1;
        return { level, proficiencyBonus };
    }


    // Запрос персонажа
    fetch(`/api/character/${characterId}`)
        .then(response => {
            if (!response.ok) {
```

```
                throw new Error(`HTTP error! status:
${response.status}`);
            }
            return response.json();
        })
        .then(data => {
            console.log('Character data:', data);
            populateCharacterSheet(data);
        })
        .catch(error => {
            console.error('Error loading character:', error);
            alert('Error loading character: ' + error.message);
        });


    document.getElementById('saveBtn').addEventListener('click',
saveCharacter);


document.getElementById('exportBtn').addEventListener('click',
exportCharacter);

    document.querySelectorAll('.ability-score').forEach(input =>
{
        input.addEventListener('change', function () {
            const ability = this.id;
            const modifierId = `${ability}Mod`;
            const score = parseInt(this.value) || 10;
            const modifier = Math.floor((score - 10) / 2);
            document.getElementById(modifierId).textContent =
modifier >= 0 ? `+${modifier}` : modifier;
        });
    });


document.getElementById('addAttackBtn').addEventListener('click'
, function () {
        const attackModal =
document.getElementById('attackModal');
        const modal = new bootstrap.Modal(attackModal);
        attackModal.dataset.mode = 'add';
        modal.show();
    });


document.getElementById('saveAttackBtn').addEventListener('click
', function () {
        const attack = {
            name: document.getElementById('attackName').value,
            attackBonus:
document.getElementById('attackBonus').value,
```

```
            damage:
document.getElementById('attackDamage').value,
            damageType:
document.getElementById('attackDamageType').value,
            notes: document.getElementById('attackNotes').value
        };

        console.log("Sending attack data:", attack);

        fetch(`/api/character/${characterId}/attacks`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(attack)
        })
            .then(response => {
                if (!response.ok) {
                    return response.text().then(text => {
                        throw new Error(`Failed to add attack:
${text}`);
                    });
                }
                return response.json();
            })
            .then(data => {
                loadAttacks();

bootstrap.Modal.getInstance(document.getElementById('attackModal
')).hide();
            })
            .catch(error => {
                console.error('Error adding attack:', error);
                alert('Error adding attack: ' + error.message);
            });
    });


document.getElementById('addSpellBtn').addEventListener('click',
function () {
        const modal = new
bootstrap.Modal(document.getElementById('spellModal'));
        document.getElementById('spellModal').dataset.mode =
'add';
        modal.show();
    });



document.getElementById('saveSpellBtn').addEventListener('click'
, function () {
        const spell = {
```

```
            name: document.getElementById('spellName').value,
            level:
parseInt(document.getElementById('spellLevel').value) || 0,
            school:
document.getElementById('spellSchool').value,
            castingTime:
document.getElementById('spellCastingTime').value,
            range: document.getElementById('spellRange').value,
            components:
document.getElementById('spellComponents').value,
            duration:
document.getElementById('spellDuration').value,
            description:
document.getElementById('spellDescription').value
        };

        if (!spell.name && !spell.castingTime &&
!spell.components &&
            !spell.duration && !spell.description &&
!spell.range) {
            alert("Заполните поля!");
            return;
            }

        if (!spell.name) {
            alert("Введите имя заклинания");
            return;
        }

        if (!spell.castingTime) {
            alert("Введите время накладывания заклинания");
            return;
        }

        if (!spell.components) {
            alert("Введите компоненты заклинания");
            return;
        }

        if (!spell.duration) {
            alert("Введите длительность заклинания");
            return;
        }

        if (!spell.description) {
            alert("Введите описание заклинания");
            return;
        }

        if (!spell.range) {
            alert("Введите дистанцию заклинания");
            return;
```

```
        }


        fetch(`/api/character/${characterId}/spells`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(spell)
        })
            .then(response => response.json())
            .then(data => {
                loadSpells();
                const modal =
bootstrap.Modal.getInstance(document.getElementById('spellModal'
));
                modal.hide();
                Alert("Заклинание успешно добавлено!");
            })
            .catch(error => {
                console.error('Error adding spell:', error);
                alert('Error adding spell');
            });
    });


document.getElementById('addEquipmentBtn').addEventListener('cli
ck', function () {
        const equipmentModal =
document.getElementById('equipmentModal');
        const modal = new bootstrap.Modal(equipmentModal);
        equipmentModal.dataset.mode = 'add';

        document.getElementById('equipmentName').value = '';
        document.getElementById('equipmentQuantity').value =
'1';
        document.getElementById('equipmentDescription').value =
'';

        modal.show();
    });


document.getElementById('saveEquipmentBtn').addEventListener('cl
ick', function () {
        const equipment = {
            name:
document.getElementById('equipmentName').value,
            quantity:
parseInt(document.getElementById('equipmentQuantity').value) ||
1,
```

```
            description:
document.getElementById('equipmentDescription').value
        };

        fetch(`/api/character/${characterId}/equipment`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(equipment)
        })
            .then(response => {
                if (!response.ok) throw new Error('Failed to add
equipment');
                return response.json();
            })
            .then(data => {
                loadEquipment();

bootstrap.Modal.getInstance(document.getElementById('equipmentMo
dal')).hide();
            })
            .catch(error => {
                console.error('Error adding equipment:', error);
                alert('Error adding equipment: ' +
error.message);
            });
    });

    function loadAttacks() {
        fetch(`/api/character/${characterId}/attacks`)
            .then(response => {
                if (!response.ok) {
                    throw new Error('Failed to load attacks');
                }
                return response.json();
            })
            .then(attacks => {
                const attacksList =
document.getElementById('attacksList');
                attacksList.innerHTML = '';

                if (attacks && attacks.length > 0) {
                    attacks.forEach((attack, index) => {
                        const attackElement =
createAttackElement(attack, index);
                        attacksList.appendChild(attackElement);
                    });
                } else {
                    attacksList.innerHTML = '<p class="text-
muted">Нет добавленный атак</p>';
                }
```

```
            })
            .catch(error => {
                console.error('Error loading attacks:', error);
                document.getElementById('attacksList').innerHTML
=
                    '<p class="text-danger">Не удалось загрузить
атаки</p>';
            });
    }

    function loadSpells() {
        fetch(`/api/character/${characterId}/spells`)
            .then(response => {
                if (!response.ok) {
                    return response.text().then(text => {
                        throw new Error(text || 'Failed to load
spells');
                    });
                }
                return response.json();
            })
            .then(spells => {
                const spellsList =
document.getElementById('spellsList');
                spellsList.innerHTML = '';

                if (spells && spells.length > 0) {
                    const spellsByLevel = spells.reduce((acc,
spell) => {
                        acc[spell.level] = acc[spell.level] ||
[];
                        acc[spell.level].push(spell);
                        return acc;
                    }, {});


                    const sortedLevels =
Object.keys(spellsByLevel).sort((a, b) => a - b);


                    sortedLevels.forEach(level => {
                        const levelHeader =
document.createElement('h5');
                        levelHeader.className = 'mt-3';
                        levelHeader.textContent = level === '0'
? 'Заговоры' : `Заклинания ${level} уровня`;
                        spellsList.appendChild(levelHeader);

                        spellsByLevel[level].forEach((spell,
index) => {
                            const spellElement =
createSpellElement(spell, index);
```

```
spellsList.appendChild(spellElement);
                        });
                    });
                } else {
                    spellsList.innerHTML = '<p class="text-
muted">Нет добавленных заклинаний</p>';
                }
            })
            .catch(error => {
                console.error('Error loading spells:', error);
                document.getElementById('spellsList').innerHTML
=
                    `<div class="alert alert-danger">Не удалось
загрузить зуклинания: ${error.message}</div>`;
            });
    }

    function loadEquipment() {
        fetch(`/api/character/${characterId}/equipment`)
            .then(response => {
                if (!response.ok) {
                    return response.text().then(text => {
                        throw new Error(text || 'Failed to load
equipment');
                    });
                }
                return response.json();
            })
            .then(equipment => {
                const equipmentList =
document.getElementById('equipmentList');
                equipmentList.innerHTML = '';

                if (equipment && equipment.length > 0) {
                    equipment.forEach((item, index) => {
                        const itemElement =
createEquipmentElement(item, index);
                        equipmentList.appendChild(itemElement);
                    });
                } else {
                    equipmentList.innerHTML = '<p class="text-
muted">Нет добавленного снаряжения.</p>';
                }
            })
            .catch(error => {
                console.error('Error loading equipment:',
error);

document.getElementById('equipmentList').innerHTML =
                    `<div class="alert alert-danger">Error
loading equipment: ${error.message}</div>`;
```

```
            });
    }

    function createEquipmentElement(equipment, index) {
        const itemDiv = document.createElement('div');
        itemDiv.className = 'card mb-2 position-relative';

        const cardBody = document.createElement('div');
        cardBody.className = 'card-body';

        const deleteBtn = document.createElement('button');
        deleteBtn.className = 'btn btn-sm btn-danger position-
absolute top-0 end-0 m-1';
        deleteBtn.innerHTML = '&times;';
        deleteBtn.onclick = (e) => {
            e.stopPropagation();
            deleteEquipment(index);
        };

        const title = document.createElement('h6');
        title.className = 'card-title';
        title.textContent = `${equipment.name}
(${equipment.quantity})`;

        const description = document.createElement('p');
        description.className = 'card-text';
        description.textContent = equipment.description || 'No
description';

        cardBody.appendChild(deleteBtn);
        cardBody.appendChild(title);
        cardBody.appendChild(description);
        itemDiv.appendChild(cardBody);

        return itemDiv;
    }

    function createAttackElement(attack, index) {
        const attackDiv = document.createElement('div');
        attackDiv.className = 'card mb-2 position-relative';

        const cardBody = document.createElement('div');
        cardBody.className = 'card-body';

        const deleteBtn = document.createElement('button');
        deleteBtn.className = 'btn btn-sm btn-danger position-
absolute top-0 end-0 m-1';
        deleteBtn.innerHTML = '&times;';
        deleteBtn.onclick = (e) => {
            e.stopPropagation();
            deleteAttack(index);
        };
```

```
        const title = document.createElement('h6');
        title.className = 'card-title';
        title.textContent = attack.name;

        const details = document.createElement('p');
        details.className = 'card-text';
        details.innerHTML = `<strong>Атака:</strong>
${attack.attackBonus}<br>
                        <strong>Урон:</strong> ${attack.damage}
${attack.damageType}<br>
                        <strong>Описание:</strong> ${attack.notes
|| 'None'}`;

        cardBody.appendChild(deleteBtn);
        cardBody.appendChild(title);
        cardBody.appendChild(details);
        attackDiv.appendChild(cardBody);

        return attackDiv;
    }

    function createSpellElement(spell, index) {
        const spellDiv = document.createElement('div');
        spellDiv.className = 'card mb-2 position-relative';

        const cardBody = document.createElement('div');
        cardBody.className = 'card-body';

        const deleteBtn = document.createElement('button');
        deleteBtn.className = 'btn btn-sm btn-danger position-
absolute top-0 end-0 m-1';
        deleteBtn.innerHTML = '&times;';
        deleteBtn.onclick = (e) => {
            e.stopPropagation();
            deleteSpell(index);
        };

        const title = document.createElement('h6');
        title.className = 'card-title';
        title.textContent = spell.name;

        const subtitle = document.createElement('p');
        subtitle.className = 'card-subtitle mb-2 text-muted';
        subtitle.textContent = `${spell.level === 0 ? 'Заговор'
: `Уровень ${spell.level}`} • ${(spell.school)}`;

        const details = document.createElement('p');
        details.className = 'card-text';
        details.innerHTML = `<strong>Время сотворения:</strong>
${spell.castingTime}<br>
```

```
                        <strong>Дистанция:</strong>
${spell.range}<br>
                        <strong>Компоненты:</strong>
${spell.components}<br>
                        <strong>Длительность:</strong>
${spell.duration}`;

        const description = document.createElement('p');
        description.className = 'card-text';
        description.textContent = spell.description;

        cardBody.appendChild(deleteBtn);
        cardBody.appendChild(title);
        cardBody.appendChild(subtitle);
        cardBody.appendChild(details);
        cardBody.appendChild(description);
        spellDiv.appendChild(cardBody);

        return spellDiv;
    }

    // Функция удаления атаки
    async function deleteAttack(index) {
        if (!confirm('Вы действительно хотите удалить эту
атаку?')) return;

        try {
            const response = await
fetch(`/api/character/${characterId}/attacks/${index}`, {
                method: 'DELETE'
            });

            if (!response.ok) {
                throw new Error('Failed to delete attack');
            }

            loadAttacks();
        } catch (error) {
            console.error('Error deleting attack:', error);
            alert('Error deleting attack: ' + error.message);
        }
    }

    // Функция удаления заклинания
    async function deleteSpell(index) {
        if (!confirm('Вы действительно хотите удалить это
заклинание?')) return;

        try {
            const response = await
fetch(`/api/character/${characterId}/spells/${index}`, {
                method: 'DELETE'
```

```
            });

            if (!response.ok) {
                throw new Error('Failed to delete spell');
            }

            loadSpells();
        } catch (error) {
            console.error('Error deleting spell:', error);
            alert('Error deleting spell: ' + error.message);
        }
    }

    function createEquipmentElement(equipment, index) {
        const itemDiv = document.createElement('div');
        itemDiv.className = 'card mb-2 position-relative';

        const cardBody = document.createElement('div');
        cardBody.className = 'card-body';

        const deleteBtn = document.createElement('button');
        deleteBtn.className = 'btn btn-sm btn-danger position-
absolute top-0 end-0 m-1';
        deleteBtn.innerHTML = '&times;';
        deleteBtn.onclick = (e) => {
            e.stopPropagation();
            deleteEquipment(index);
        };

        const title = document.createElement('h6');
        title.className = 'card-title';
        title.textContent = `${equipment.name}
(${equipment.quantity})`;

        const description = document.createElement('p');
        description.className = 'card-text';
        description.textContent = equipment.description || 'Нет
описания';

        cardBody.appendChild(deleteBtn);
        cardBody.appendChild(title);
        cardBody.appendChild(description);
        itemDiv.appendChild(cardBody);

        return itemDiv;
    }

    //Функция удаления снаряжения
    async function deleteEquipment(index) {
        if (!confirm('Вы действительно хотите удалить этот
предмет?')) return;
```

```
        try {
            const response = await
fetch(`/api/character/${characterId}/equipment/${index}`, {
                method: 'DELETE'
            });

            if (!response.ok) {
                throw new Error('Failed to delete equipment');
            }

            loadEquipment();
        } catch (error) {
            console.error('Error deleting equipment:', error);
            alert('Error deleting equipment: ' + error.message);
        }
    }

    //Загрузка информации о персонаже на страницу
    function populateCharacterSheet(data) {


document.getElementById('characterNameDisplay').textContent =
data.name + ' ' + data.class + ' ' + data.level + ' ' + 'уровня'
|| 'Character Sheet';
        document.getElementById('name').value = data.name || '';
        document.getElementById('race').value = data.race || '';
        document.getElementById('class').value = data.class ||
'';
        document.getElementById('level').value = data.level ||
1;
        document.getElementById('background').value =
data.background || '';
        document.getElementById('alignment').value =
data.alignment || '';
        document.getElementById('experience').value =
data.experience || 0;

        const currentXP = data.experience || 0;
        const { level, proficiencyBonus } =
calculateLevelAndProficiencyBonus(currentXP);

        updateXPDisplay(currentXP, level);

        const nextLevelXP = experienceTable[level] ||
experienceTable[experienceTable.length - 1];
        const prevLevelXP = experienceTable[level - 1] || 0;
        const progress = ((currentXP - prevLevelXP) /
(nextLevelXP - prevLevelXP)) * 100;


document.getElementById('experienceProgress').style.width =
`${progress}%`;
```

```
        document.getElementById('xpProgressText').textContent =
            `${currentXP - prevLevelXP}/${nextLevelXP -
prevLevelXP} XP`;
        document.getElementById('proficiencyBonus').value =
proficiencyBonus;
        document.getElementById('level').value = level;

        if (data.abilities) {
            data.abilities.forEach(ability => {
                const input =
document.getElementById(ability.name.toLowerCase());
                if (input) {
                    input.value = ability.score || 10;
                    const modifierId =
`${ability.name.toLowerCase()}Mod`;
                    const modifier = Math.floor((ability.score -
10) / 2);

document.getElementById(modifierId).textContent = modifier >= 0
? `+${modifier}` : modifier;
                }
            });
        }

        document.getElementById('armorClass').value =
data.armorClass || 10;
        document.getElementById('initiative').value =
data.initiative || 0;
        document.getElementById('speed').value = data.speed ||
30;
        document.getElementById('hitPoints').value =
data.hitPoints || 10;
        document.getElementById('hitDice').value = data.hitDice
|| '1d8';


        const skillsList =
document.getElementById('skillsList');
        skillsList.innerHTML = '';

        if (data.skills) {
            const sortedSkills =
Object.keys(data.skills).sort();

            sortedSkills.forEach(skill => {
                const skillDiv = document.createElement('div');
                skillDiv.className = 'skill-item';

                const checkbox =
document.createElement('input');
                checkbox.type = 'checkbox';
```

```
                checkbox.id = `skill-${skill.replace(/\s+/g, '-
').toLowerCase()}`;
                checkbox.checked = data.skills[skill] || false;
                checkbox.addEventListener('change', function ()
{
                    data.skills[skill] = this.checked;
                });

                const label = document.createElement('label');
                label.htmlFor = checkbox.id;
                label.textContent =
translateSkillToRussian(skill);
                label.className = 'skill-label';

                const bonusSpan =
document.createElement('span');
                bonusSpan.className = 'skill-bonus ms-2';
                bonusSpan.textContent = calculateSkillBonus(
                    skill,
                    checkbox.checked,
                    getAbilityModifierForSkill(skill),
                    proficiencyBonus
                );

                skillDiv.appendChild(checkbox);
                skillDiv.appendChild(label);
                skillDiv.appendChild(bonusSpan);
                skillsList.appendChild(skillDiv);
            });
        }

        document.getElementById('personalityTraits').value =
data.personalityTraits || '';
        document.getElementById('ideals').value = data.ideals ||
'';
        document.getElementById('bonds').value = data.bonds ||
'';
        document.getElementById('flaws').value = data.flaws ||
'';
        document.getElementById('appearance').value =
data.appearance || '';

        document.getElementById('features').value =
data.features || '';
        document.getElementById('proficiencies').value =
data.proficiencies || '';
        document.getElementById('notesText').value = data.notes
|| '';

        loadAttacks();
        loadSpells();
        loadEquipment();
```

```
    }

    //Сохранение персонажа в БД
    async function saveCharacter() {
        try {

            const response = await
fetch(`/api/character/${characterId}`);
            if (!response.ok) {
                throw new Error('Failed to load character
data');
            }
            const currentCharacter = await response.json();


            const updatedCharacter = {
                ...currentCharacter,

                name: document.getElementById('name').value,
                race: document.getElementById('race').value,
                class: document.getElementById('class').value,
                level:
parseInt(document.getElementById('level').value) || 1,
                background:
document.getElementById('background').value,
                alignment:
document.getElementById('alignment').value,
                experience:
parseInt(document.getElementById('experience').value) || 0,
                armorClass:
parseInt(document.getElementById('armorClass').value) || 10,
                initiative:
parseInt(document.getElementById('initiative').value) || 0,
                speed:
parseInt(document.getElementById('speed').value) || 30,
                hitPoints:
parseInt(document.getElementById('hitPoints').value) || 10,
                hitDice:
parseInt(document.getElementById('level').value) || 1,
                hitDiceType:
document.getElementById('hitDiceType').value,

                skills:
getUpdatedSkills(currentCharacter.skills),

                personalityTraits:
document.getElementById('personalityTraits').value,
                ideals: document.getElementById('ideals').value,
                bonds: document.getElementById('bonds').value,
                flaws: document.getElementById('flaws').value,
                appearance:
document.getElementById('appearance').value,
```

```
                features:
document.getElementById('features').value,
                proficiencies:
document.getElementById('proficiencies').value,
                notes:
document.getElementById('notesText').value
            };

            updatedCharacter.abilities = [
                { name: "Strength", score:
parseInt(document.getElementById('strength').value) || 10 },
                { name: "Dexterity", score:
parseInt(document.getElementById('dexterity').value) || 10 },
                { name: "Constitution", score:
parseInt(document.getElementById('constitution').value) || 10 },
                { name: "Intelligence", score:
parseInt(document.getElementById('intelligence').value) || 10 },
                { name: "Wisdom", score:
parseInt(document.getElementById('wisdom').value) || 10 },
                { name: "Charisma", score:
parseInt(document.getElementById('charisma').value) || 10 }
            ];

            const saveResponse = await
fetch('/api/character/update', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(updatedCharacter)
            });

            if (!saveResponse.ok) {
                throw new Error('Failed to save character');
            }

            alert('Персонаж сохранен!');
        } catch (error) {
            console.error('Error saving character:', error);
            alert(`Error saving character: ${error.message}`);
        }
    }

    function getUpdatedSkills(currentSkills = {}) {
        const skills = {
            "Acrobatics": false,
            "Animal Handling": false,
            "Arcana": false,
            "Athletics": false,
            "Deception": false,
            "History": false,
            "Insight": false,
```

```
            "Intimidation": false,
            "Investigation": false,
            "Medicine": false,
            "Nature": false,
            "Perception": false,
            "Performance": false,
            "Persuasion": false,
            "Religion": false,
            "Sleight of Hand": false,
            "Stealth": false,
            "Survival": false
        };

        const skillCheckboxes =
document.querySelectorAll('#skillsList input[type="checkbox"]');
        skillCheckboxes.forEach(checkbox => {
            const russianSkillName =
checkbox.nextElementSibling.textContent;
            const englishSkillName =
Object.keys(skillTranslations).find(
                key => skillTranslations[key] ===
russianSkillName
            ) || russianSkillName;
            skills[englishSkillName] = checkbox.checked;
        });

        return skills;
    }


document.getElementById('experience').addEventListener('change',
function () {
        const xp = parseInt(this.value) || 0;
        const { level, proficiencyBonus } =
calculateLevelAndProficiencyBonus(xp);

        document.getElementById('level').value = level;
        document.getElementById('proficiencyBonus').value =
proficiencyBonus;

        updateXPDisplay(xp, level);
    });

    function exportCharacter() {
        fetch(`/api/character/export/${characterId}`)
            .then(response => {
                if (!response.ok) {
                    throw new Error(`HTTP error! status:
${response.status}`);
                }
                return response.json();
            })
```

```
            .then(data => {
                if (!data) {
                    throw new Error("No data received from
server");
                }

                console.log('Data received from server:', data);

                if (!data.name) {
                    console.warn('Character name is missing in
the data');
                    data.name = 'character';
                }

                const blob = new Blob([JSON.stringify(data,
null, 2)], { type: 'application/json' });
                const url = URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.href = url;
                a.download = `${data.name || 'character'}.json`;
                document.body.appendChild(a);
                a.click();
                document.body.removeChild(a);
                URL.revokeObjectURL(url);
            })
            .catch(error => {
                console.error('Error exporting character:',
error);
                alert(`Error exporting character:
${error.message}`);
            });
    }

    function calculateSkillBonus(skillName, isProficient,
abilityModifier, proficiencyBonus) {
        const skillAbilityMap = {
            "Acrobatics": "dexterity",
            "Animal Handling": "wisdom",
            "Arcana": "intelligence",
            "Athletics": "strength",
            "Deception": "charisma",
            "History": "intelligence",
            "Insight": "wisdom",
            "Intimidation": "charisma",
            "Investigation": "intelligence",
            "Medicine": "wisdom",
            "Nature": "intelligence",
            "Perception": "wisdom",
            "Performance": "charisma",
            "Persuasion": "charisma",
            "Religion": "intelligence",
            "Sleight of Hand": "dexterity",
```

```
            "Stealth": "dexterity",
            "Survival": "wisdom"
        };

        const ability = skillAbilityMap[skillName] ||
'strength';
        const modifier = abilityModifier || 0;
        const bonus = isProficient ? modifier + proficiencyBonus
: modifier;

        return bonus >= 0 ? `+${bonus}` : `${bonus}`;
    }

    function getAbilityModifierForSkill(skillName) {
        const skillAbilityMap = {
            "Acrobatics": "dexterity",
            "Animal Handling": "wisdom",
            "Arcana": "intelligence",
            "Athletics": "strength",
            "Deception": "charisma",
            "History": "intelligence",
            "Insight": "wisdom",
            "Intimidation": "charisma",
            "Investigation": "intelligence",
            "Medicine": "wisdom",
            "Nature": "intelligence",
            "Perception": "wisdom",
            "Performance": "charisma",
            "Persuasion": "charisma",
            "Religion": "intelligence",
            "Sleight of Hand": "dexterity",
            "Stealth": "dexterity",
            "Survival": "wisdom"
        };

        const ability = skillAbilityMap[skillName] ||
'strength';
        const modifierElement =
document.getElementById(`${ability}Mod`);
        if (modifierElement) {
            const modifierText = modifierElement.textContent;
            return parseInt(modifierText) || 0;
        }
        return 0;
    }

    function updateXPDisplay(currentXP, currentLevel) {
        const xpForNextLevel = experienceTable[currentLevel] ||
experienceTable[experienceTable.length - 1];


        const formatXP = (xp) => xp.toLocaleString();
```

```
        document.getElementById('xpProgressText').textContent =
            `${formatXP(currentXP)} /
${formatXP(xpForNextLevel)} XP`;


        const progress = (currentXP / xpForNextLevel) * 100;

document.getElementById('experienceProgress').style.width =
`${progress}%`;

        if (currentLevel >= 20) {

document.getElementById('xpProgressText').textContent =
`${formatXP(currentXP)} XP (Макс. уровень)`;

document.getElementById('experienceProgress').style.width =
"100%";
        }
    }


    document.querySelectorAll('.ability-score').forEach(input =>
{
        input.addEventListener('change', function () {
            const ability = this.id;
            const modifierId = `${ability}Mod`;
            const score = parseInt(this.value) || 10;
            const modifier = Math.floor((score - 10) / 2);
            document.getElementById(modifierId).textContent =
modifier >= 0 ? `+${modifier}` : modifier;

            updateAllSkillBonuses();
        });
    });

    function updateAllSkillBonuses() {
        const skillsList =
document.getElementById('skillsList');
        const skillItems = skillsList.querySelectorAll('.skill-
item');
        const proficiencyBonus =
parseInt(document.getElementById('proficiencyBonus').value) ||
2;

        skillItems.forEach(skillItem => {
            const label = skillItem.querySelector('label');
            const skillName = label.textContent;
            const checkbox =
skillItem.querySelector('input[type="checkbox"]');
```

```
            const bonusSpan = skillItem.querySelector('.skill-
bonus');

            bonusSpan.textContent = calculateSkillBonus(
                skillName,
                checkbox.checked,
                getAbilityModifierForSkill(skillName),
                proficiencyBonus
            );
        });
    }

    document.addEventListener('DOMContentLoaded', function () {
        const textareas = document.querySelectorAll
            ('#personalityTraits, #ideals, #bonds, #flaws,
#appearance, #features, #proficiencies, #notesText');

        textareas.forEach(textarea => {
            textarea.addEventListener('input', function () {
                this.style.height = 'auto';
                this.style.height = (this.scrollHeight) + 'px';
            });

            textarea.dispatchEvent(new Event('input'));
        });
    });

    // Функция для броска кубика
    function rollDice(max) {
        return Math.floor(Math.random() * max) + 1;
    }

    let diceHistory = [];

    // Показать результат броска
    function showDiceResult(value, diceType) {
        const resultElement =
document.getElementById('diceResult');
        const valueElement =
document.getElementById('diceResultValue');

        valueElement.textContent = `${value} (${diceType})`;
        resultElement.classList.remove('d-none');
        resultElement.classList.add('show');

        addToHistory(value, diceType);

        setTimeout(() => {
            resultElement.classList.remove('show');
            setTimeout(() => {
                resultElement.classList.add('d-none');
            }, 300);
```

```
        }, 3000);
    }

    // Добавление в историю
    function addToHistory(value, diceType) {
        const timestamp = new Date().toLocaleTimeString();
        diceHistory.push({ value, diceType, timestamp });

        if (diceHistory.length > 20) {
            diceHistory.shift();
        }

        updateHistoryUI();
    }

    function updateHistoryUI() {
        const historyList =
document.getElementById('historyList');
        historyList.innerHTML = '';

        diceHistory.slice().reverse().forEach(roll => {
            const li = document.createElement('li');
            li.className = 'py-1 border-bottom';
            li.innerHTML = `
            <span class="badge bg-dark">${roll.diceType}</span>
            <strong class="mx-2">${roll.value}</strong>
            <small class="text-muted">${roll.timestamp}</small>
        `;
            historyList.appendChild(li);
        });
    }


document.getElementById('clearHistory').addEventListener('click'
, () => {
        diceHistory = [];
        updateHistoryUI();
    });


document.getElementById('toggleHistory').addEventListener('click
', () => {
        const historyElement =
document.getElementById('diceHistory');
        historyElement.classList.toggle('d-none');
    });

    function animateDiceRoll(button) {
        button.classList.add('dice-rolling');
        setTimeout(() => {
            button.classList.remove('dice-rolling');
        }, 1000);
```

```
}

// Обработчики для кнопок кубиков
document.querySelectorAll('.btn-dice').forEach(button => {
    button.addEventListener('click', function () {
        const diceType = this.title;
        let max;

        switch (diceType) {
            case 'D4': max = 4; break;
            case 'D6': max = 6; break;
            case 'D8': max = 8; break;
            case 'D10': max = 10; break;
            case 'D12': max = 12; break;
            case 'D20': max = 20; break;
            case 'D100': max = 100; break;
            default: max = 20;
        }

        animateDiceRoll(this);
        setTimeout(() => {
            const result = rollDice(max);
            showDiceResult(result, diceType);
            console.log(`Бросок ${diceType}: ${result}`);
        }, 300);
    });
});

const skillTranslations = {
    "Acrobatics": "Акробатика",
    "Animal Handling": "Уход за животными",
    "Arcana": "Магия",
    "Athletics": "Атлетика",
    "Deception": "Обман",
    "History": "История",
    "Insight": "Проницательность",
    "Intimidation": "Запугивание",
    "Investigation": "Анализ",
    "Medicine": "Медицина",
    "Nature": "Природа",
    "Perception": "Внимательность",
    "Performance": "Выступление",
    "Persuasion": "Убеждение",
    "Religion": "Религия",
    "Sleight of Hand": "Ловкость рук",
    "Stealth": "Скрытность",
    "Survival": "Выживание"
};

function translateSkillToRussian(englishSkill) {
    const skillTranslations = {
        "Acrobatics": "Акробатика",
```

```
            "Animal Handling": "Уход за животными",
            "Arcana": "Магия",
            "Athletics": "Атлетика",
            "Deception": "Обман",
            "History": "История",
            "Insight": "Проницательность",
            "Intimidation": "Запугивание",
            "Investigation": "Анализ",
            "Medicine": "Медицина",
            "Nature": "Природа",
            "Perception": "Внимательность",
            "Performance": "Выступление",
            "Persuasion": "Убеждение",
            "Religion": "Религия",
            "Sleight of Hand": "Ловкость рук",
            "Stealth": "Скрытность",
            "Survival": "Выживание"
        };

        return skillTranslations[englishSkill] || englishSkill;
    }

    const toggleDiceBtn =
document.getElementById('toggleDiceBtn');
    const diceContainer =
document.getElementById('diceContainer');
    toggleDiceBtn.addEventListener('click', function () {
        diceContainer.classList.toggle('show');
        diceContainer.classList.toggle('d-none');

        const icon = this.querySelector('i');
        if (diceContainer.classList.contains('show')) {
            icon.classList.remove('bi-dice-6');
            icon.classList.add('bi-x');
        } else {
            icon.classList.remove('bi-x');
            icon.classList.add('bi-dice-6');
        }
    });
    document.addEventListener('click', function (e) {
        if (!diceContainer.contains(e.target) &&
            e.target !== toggleDiceBtn &&
            !toggleDiceBtn.contains(e.target)) {
            diceContainer.classList.remove('show');
            diceContainer.classList.add('d-none');
            const icon = toggleDiceBtn.querySelector('i');
            icon.classList.remove('bi-x');
            icon.classList.add('bi-dice-6');
        }
    });

});
```

Листинг В.5 – Код CSS стилей

```
.row {
    margin-right: -5px;
    margin-left: -5px;
}

.col-md-2, .col-md-3, .col-md-7 {
    padding-right: 5px;
    padding-left: 5px;
}

.ability-row {
    display: flex;
    align-items: center;
    flex-direction: column;
    margin-bottom: 0.4rem;
}

.ability-name {
    width: 30px;
    font-size: 0.8rem;
    font-weight: bold;
    text-align: center;
    flex-shrink: 0;
    margin-bottom: 2px;
    gap: 0.5rem;
}

.ability-score {
    margin: 0 0.4rem;
    font-size: 0.8rem;
    text-align: center;
    flex-shrink: 0;
}

.ability-modifier {
    width: 40px;
    text-align: center;
    font-weight: bold;
    background-color: #cfc5e8;
    border-radius: 3px;
    margin-top: 4pt;
    padding: 0.2rem;
    flex-shrink: 0;
}

.skills-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 0.2rem;
    max-height: 600px;
    overflow-y: auto;
```

```
}

.skill-item {
    display: flex;
    align-items: center;
    padding: 0.1rem;
    font-size: 0.8rem;
}

.skill-item input[type="checkbox"] {
    margin-right: 0.3rem;
    width: 1em;
    height: 1em;
}

#saveBtn, #addAttackBtn, #createCharacterBtn, #confirmCreateBtn,
#addSpellBtn, #addEquipmentBtn, #saveAttackBtn, #saveSpellBtn,
#saveEquipmentBtn {
    background-color: #796b9c;
    border-color: #796b9c;
}

.ability-countainer {

    margin-top: 16px;
}

.skills-box {
    margin-left: 10px;
}


@media (max-width: 767.98px) {

    .ability-countainer {
        width: 80px;
        flex-shrink: 0;
        margin-top: 5px;
    }

    .skills-box {
        flex-grow: 1;
        margin-left: 10px;
    }
}

.progress {
    height: 20px;
    margin-top: 10px;
}

.progress-bar {
```

```
    background-color: #28a745;
}

.miniText {
    font-size: 10pt;
    margin: inherit;
    padding-left: 6px;
}

.bts {
    background-color: mediumpurple;
}

#cardHeaderInfo, #cardHeaderBattle, #cardHeaderSkills {
    background-color: #cfc5e8;
}

#attacks-tab, #spells-tab, #equipment-tab, #personality-tab,
#notes-tab {
    color: #584191;
}

.skill-bonus {
    display: inline-block;
    width: 30px;
    text-align: center;
    font-weight: bold;
    font-size: 10pt;
    background-color: #cfc5e8;
    border-radius: 3px;
    padding: 0.1rem;
    margin-left: 0.3rem;
}

.card {
    margin-bottom: 5px;
}

.character-card {
    transition: transform 0.2s;
    cursor: pointer;
    background-color: #f8f9fa;
}

.character-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

.character-card .btn {
    font-size: 0.8rem;
    padding: 0.25rem 0.5rem;
```

```
}

#personalityTraits, #ideals, #bonds, #flaws, #appearance,
#features, #proficiencies, #notesText {
    resize: vertical;
    min-height: 50px;
    overflow-y: auto;
    height: auto;
}

.tab-content {
    height: 530px;
    overflow-y: auto;
}

#equipmentList {
    height: calc(100% - 40px);
    overflow-y: auto;
}

.btn-dice {
    width: 50px;
    height: 50px;
    border: none;
    background: none;
    cursor: pointer;
    margin-bottom: 10px;
    display: flex;
    align-items: center;
    justify-content: center;
}

.btn-dice img {
    width: 100%;
    height: 100%;

}

.btn-dice:hover {
    transform: scale(1.1);
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
}
.dice-result {
    position: fixed;
    top: 20px;
    right: 20px;
    z-index: 1050;
    min-width: 180px;
    padding: 10px 15px;
    font-size: 1rem;
    border-radius: 8px;
    box-shadow: 0 3px 10px rgba(0, 0, 0, 0.15);
```

```
    opacity: 0;
    transform: translateY(-20px);
    transition: all 0.3s ease;
    pointer-events: none;
    background-color: #d1e7dd;
    border: 1px solid #badbcc;
}

.dice-result.show {
    opacity: 1;
    transform: translateY(0);
}

.dice-container {
    position: fixed;
    bottom: 60px;
    right: 20px;
    z-index: 1040;
}

.dice-container-vertical {
    display: flex;
    flex-direction: column;
    align-items: center;
}

#diceHistory {
    transition: all 0.3s ease;
    z-index: 1060;
    border: 1px solid #dee2e6;
}

#historyList li {
    transition: background-color 0.2s;
}

#historyList li:hover {
    background-color: #f8f9fa;
}

#toggleHistory {
    z-index: 1070;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #796b9c;
    border-color: #796b9c;
}

.bg-primary {
    background-color: #796b9c;
    border-color: #796b9c;
```

```
}

.progress-container {
    margin-top: 1rem;
}

.progress {
    background-color: #e9ecef;
    border-radius: 0.25rem;
    box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.1);
}

.progress-bar {
    background: linear-gradient(90deg, #FFD166, #F4A261);
    transition: width 0.6s ease;
}

.skill-item input[type="checkbox"] {
    position: absolute;
    opacity: 0;
    width: 0;
    height: 0;
}

.skill-item {
    position: relative;
    padding-left: 20px;
    cursor: pointer;
    display: flex;
    align-items: center;
    margin-bottom: 0px;
}


.skill-item label:before {
    content: "";
    position: absolute;
    left: 0;
    top: 50%;
    transform: translateY(-50%);
    width: 18px;
    height: 18px;
    border: 2px solid #796b9c;
    border-radius: 3px;
    background-color: white;
    transition: all 0.2s ease;
}

.skill-item label:after {
    content: "";
    position: absolute;
    left: 6px;
```

```
    top: 50%;
    transform: translateY(-50%) rotate(45deg);
    width: 5px;
    height: 10px;
    border: solid white;
    border-width: 0 2px 2px 0;
    opacity: 0;
    transition: all 0.2s ease;
}

.skill-item:hover label:before {
    border-color: #5e4d8a;
    box-shadow: 0 0 5px rgba(121, 107, 156, 0.5);
}

.skill-item input[type="checkbox"]:checked + label:before {
    background-color: #796b9c;
    border-color: #796b9c;
}

.skill-item input[type="checkbox"]:checked + label:after {
    opacity: 1;
}

.skill-item input[type="checkbox"]:focus + label:before {
    box-shadow: 0 0 5px rgba(121, 107, 156, 0.8);
}

.skill-label {
    margin-left: 5px;
    cursor: pointer;
    user-select: none;
    transition: color 0.2s ease;
}

.skill-item:hover .skill-label {
    color: #584191;
}

.skill-item input[type="checkbox"]:active + label:before {
    transform: translateY(-50%) scale(0.95);
}


.dice-container {
    transition: all 0.3s ease;
    transform: translateY(20px);
    opacity: 0;
    pointer-events: none;
}

.dice-container.show {
```

```
    transform: translateY(0);
    opacity: 1;
    pointer-events: all;
}

#toggleDiceBtn {
    z-index: 1070;
    transition: all 0.3s ease;
}

#toggleDiceBtn:hover {
    transform: scale(1.1);
}

.btn-dice {
    transition: all 0.2s ease;
    transform: scale(0.8);
    opacity: 0;
}

.dice-container.show .btn-dice {
    transform: scale(1);
    opacity: 1;
}

.dice-container.show .d4 {
    transition-delay: 0.05s;
}

.dice-container.show .d6 {
    transition-delay: 0.1s;
}

.dice-container.show .d8 {
    transition-delay: 0.15s;
}

.dice-container.show .d10 {
    transition-delay: 0.2s;
}

.dice-container.show .d12 {
    transition-delay: 0.25s;
}

.dice-container.show .d20 {
    transition-delay: 0.3s;
}
```

Приложение Г

Результаты тестирования

(рекомендуемо)

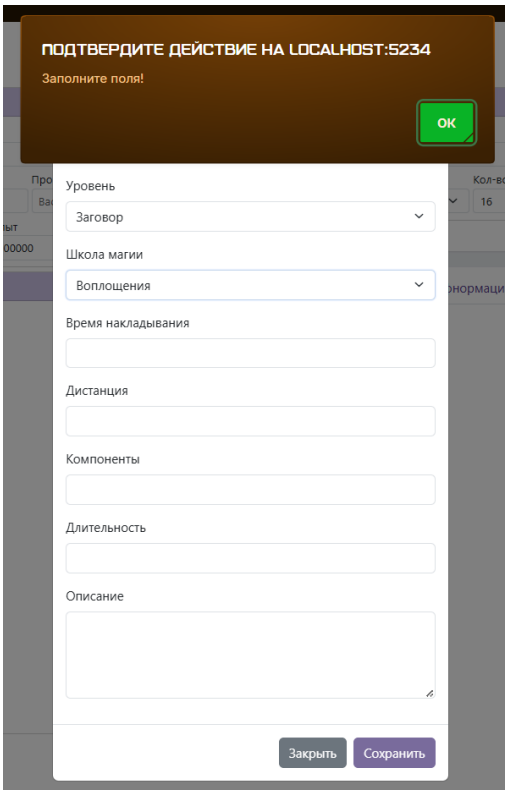

Рисунок Г.1 – Результат тест кейса 1

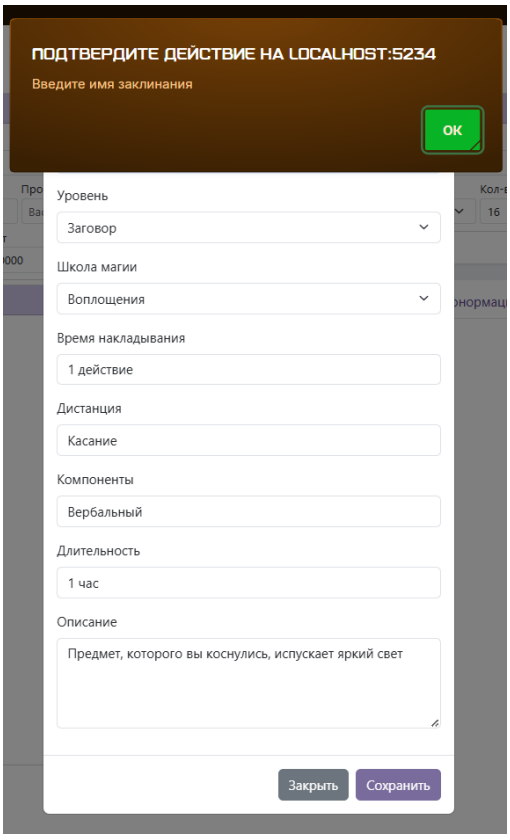

Рисунок Г.2 – Результат тест кейса 2

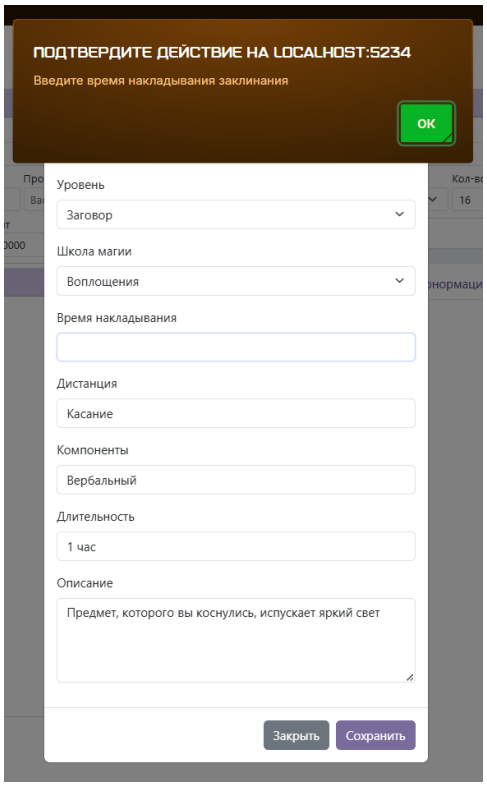

Рисунок Г.3 – Результат тест кейса 3

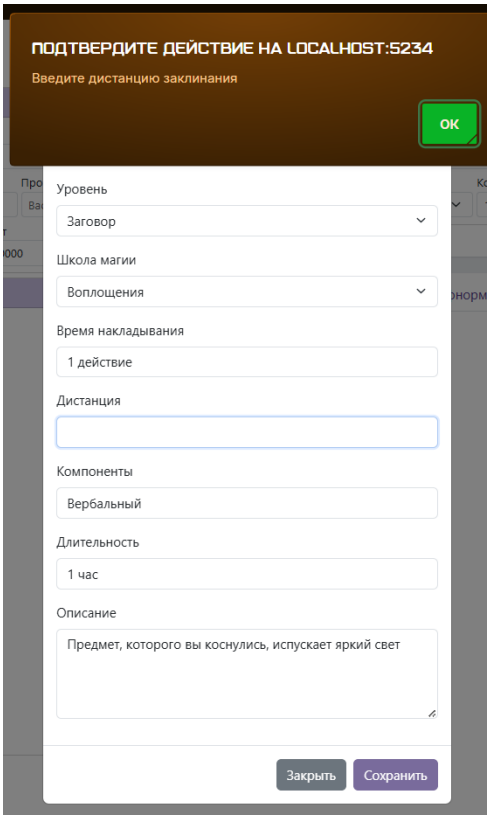

Рисунок Г.4 – Результат тест кейса 4

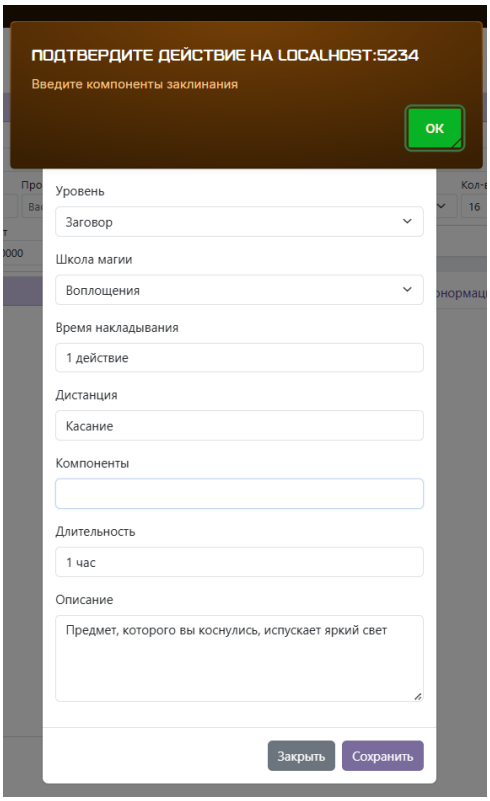

Рисунок Г.5 – Результат тест кейса 5

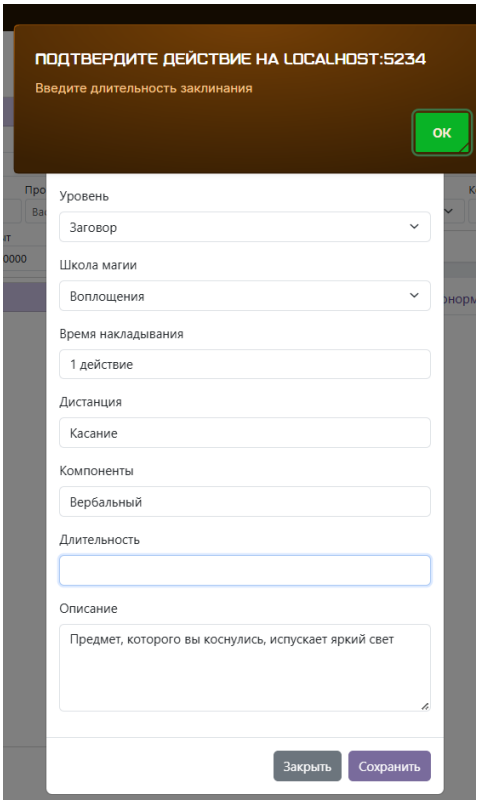

Рисунок Г.6 – Результат тест кейса 6

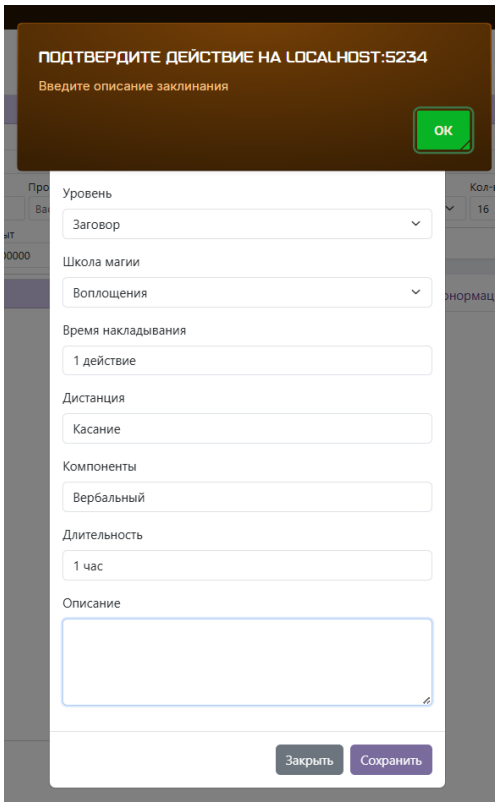

Рисунок Г.7 – Результат тест кейса 7

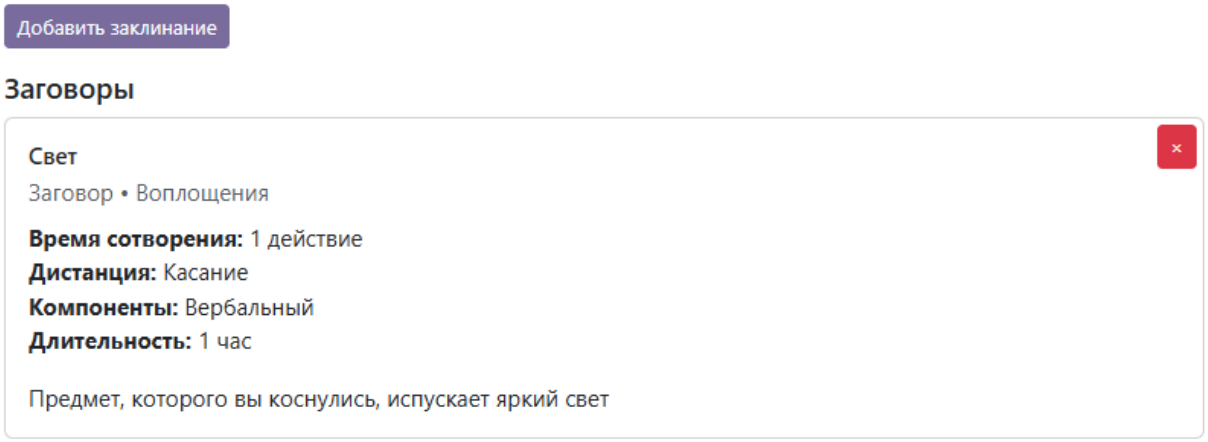

Рисунок Г.8 – Результат тест кейса 8

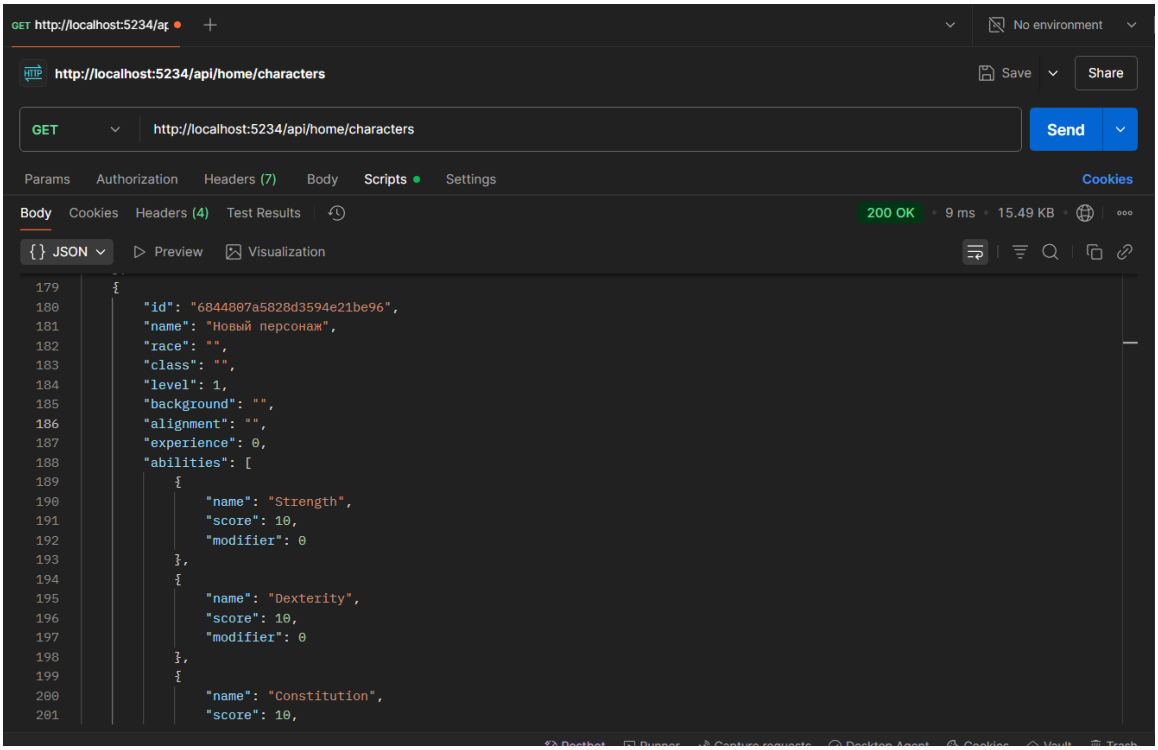

Рисунок Г.9 – Тестирование GET-метода

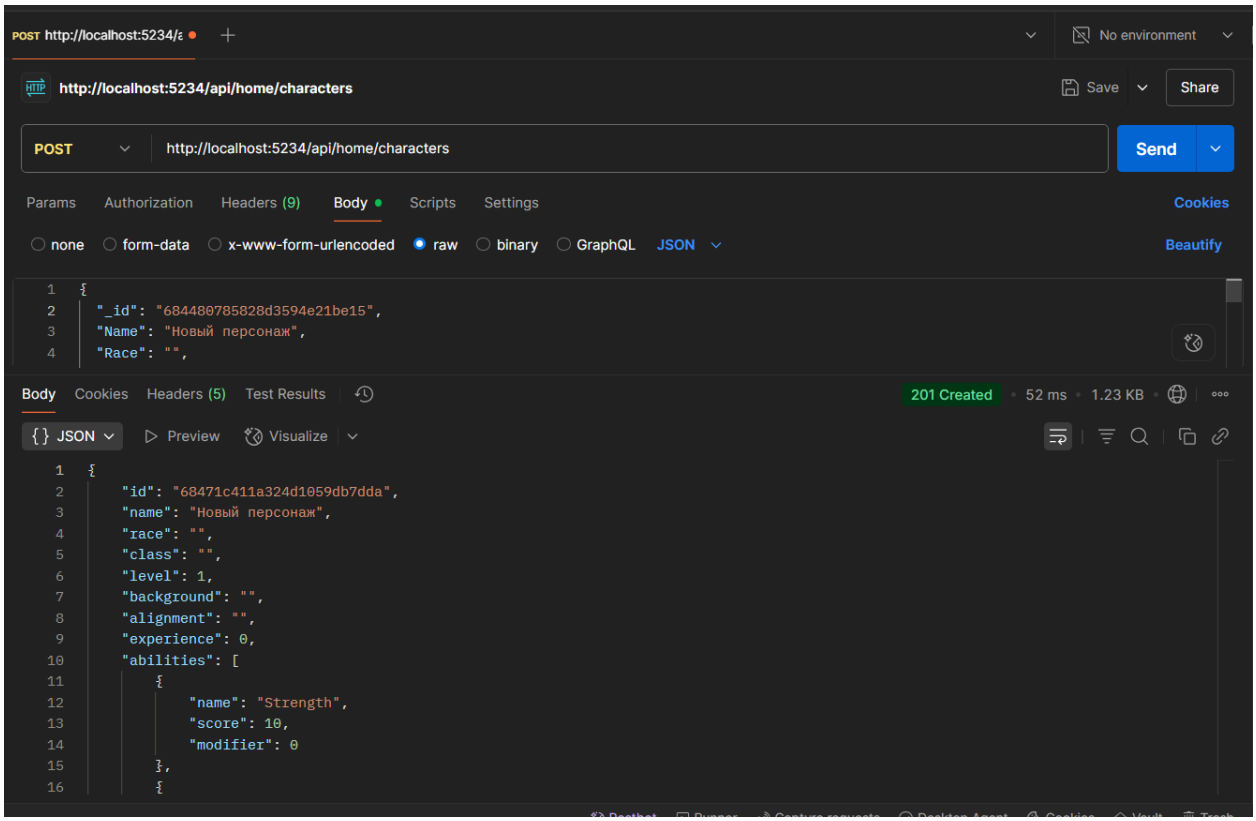

Рисунок Г.10 – Тестирование POST-метода

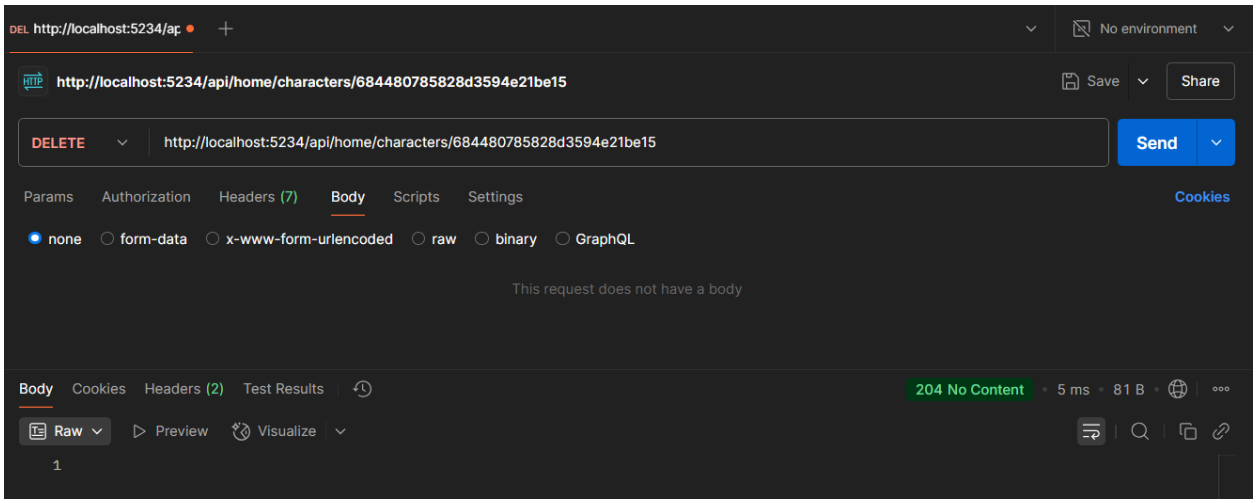

Рисунок Г.11 – Тестирование DELETE-метода

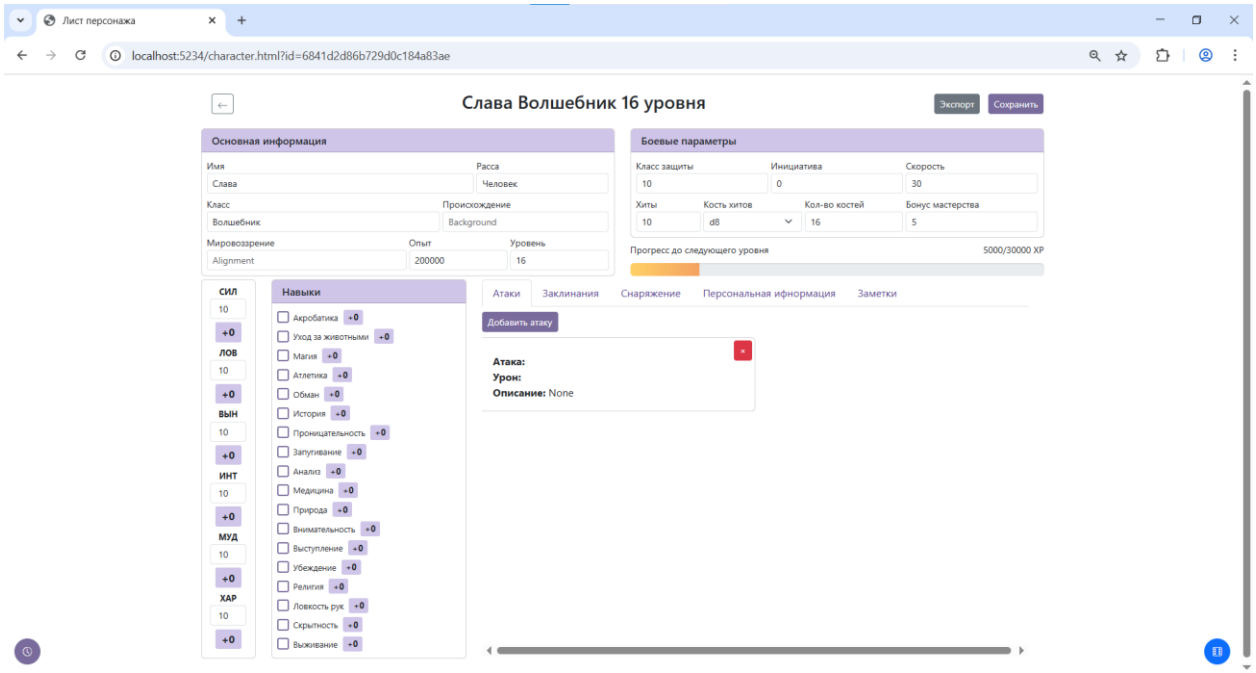

Рисунок Г.12 – Тестирование интерфейса в браузере Google Chrome

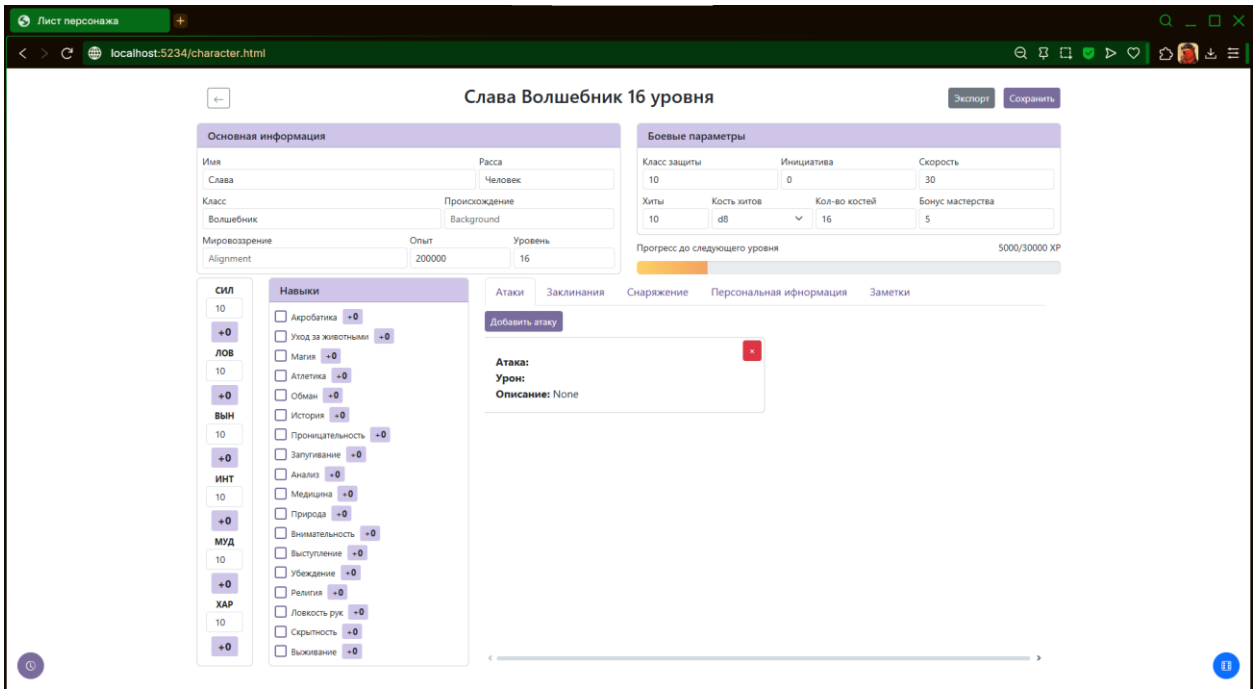

Рисунок Г.13 – Тестирование интерфейса в браузере Opera

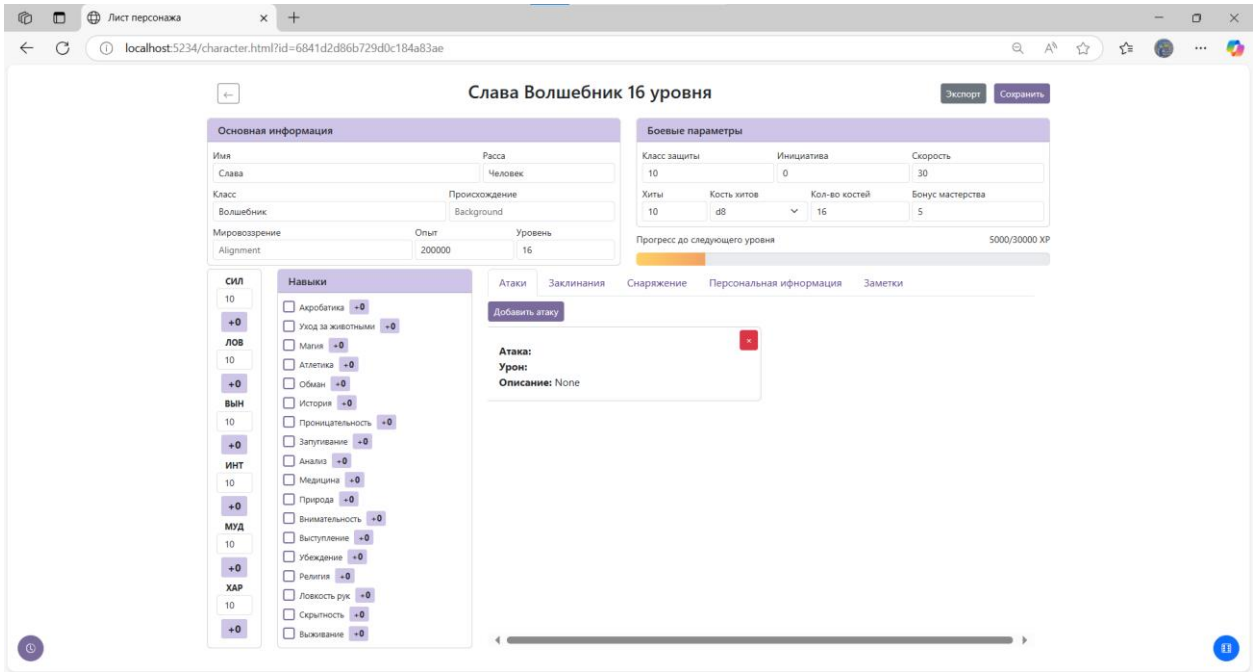

Рисунок Г.14 – Тестирование интерфейса в браузере Microsoft Edge

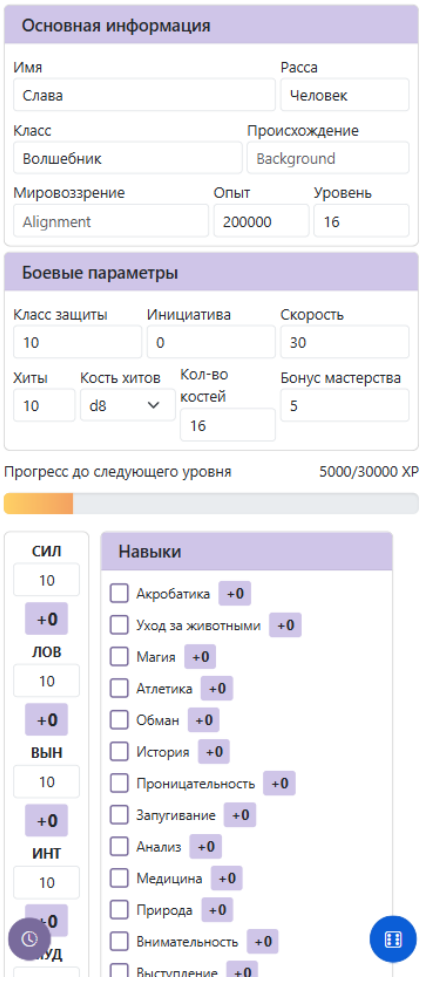

Рисунок Г.15 – Тестирование интерфейса на смартфоне

Рисунок Г.16 – Тестирование интерфейса на смартфоне с альбомной ориентацией