**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ПП.09.02.07-1.25.221.19

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

|  |  |  |
| --- | --- | --- |
| Руководитель от предприятия:  М.П. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.З. Зайцева) |
| Руководитель от техникума: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Е.С. Фролова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Ю.К. Терентьев) |

Иркутск 2025

Содержание

# Введение

# Предпроектное исследование

# Описание предметной области

# Анализ инструментов

# Проектирование программного модуля

# Проектирование базы данных

# Проектирование интерфейса

# Разработка программного модуля

# Разработка интерфейса программного модуля

# Создание базы данных

# Разработка программного модуля

# Тестирование программного модуля

# Документирование программного модуля

# Руководство пользователя

# Оценка возможности модернизации программного модуля

# Производственные задачи

# Заключение

# Список использованных источников

# Приложение А Листинг – Работа с созданием статьи

# Введение

Веб-приложение "Wiki по вселенной «Ведьмака» – это информационная платформа, предоставляющая пользователям доступ к структурированной базе знаний о мире, созданном Анджеем Сапковским. Приложение позволяет изучать информацию о персонажах, локациях, монстрах, магии и ключевых событиях, а также взаимодействовать с контентом через удобный интерфейс. В современном мире, где фэнтези-вселенные становятся частью массовой культуры, подобные ресурсы пользуются большим спросом среди фанатов, исследователей и новых зрителей, желающих глубже погрузиться в lore «Ведьмака».

Разработка качественной вики-энциклопедии – это сложная задача, требующая внимания к нескольким ключевым аспектам:

* Обширный и структурированный контент: успешная вики должна содержать детальную информацию по всем элементам вселенной, что требует продуманной системы хранения, категоризации и поиска данных.
* Удобный и интуитивный интерфейс: приложение должно быть простым в использовании как для обычных читателей, так и для редакторов, позволяя легко находить нужные материалы и при необходимости вносить правки.
* Гибкость и масштабируемость: поскольку вселенная «Ведьмака» постоянно расширяется (книги, игры, сериал), система должна позволять добавлять новый контент без нарушения существующей структуры.

Преимущества веб-приложения:

* Централизованный доступ к информации: пользователи могут быстро находить проверенные данные без необходимости поиска по разрозненным источникам.
* Возможность совместного редактирования: зарегистрированные пользователи могут дополнять и уточнять статьи, поддерживая актуальность базы знаний.
* Мультимедийность: интеграция изображений, карт, цитат и ссылок на первоисточники делает изучение вселенной более наглядным и увлекательным.

Цель производственной практики: разработка функционального веб-приложения на Django, которое обеспечит удобный доступ к информации о вселенной «Ведьмака» и позволит пользователям взаимодействовать с контентом.

Основные задачи:

* Анализ предметной области: изучение структуры вселенной «Ведьмака» и определение ключевых сущностей для базы данных (персонажи, локации, магия и т.д.).
* Выбор инструментов разработки: обоснование использования Django, PostgreSQL (или SQLite), HTML/CSS/JavaScript и дополнительных библиотек.
* Проектирование базы данных: создание ER-диаграммы и схемы таблиц с учетом связей между сущностями.
* Разработка backend-части:
* Настройка моделей Django для хранения данных.
* Реализация CRUD-функционала для статей.
* Создание системы аутентификации и разграничения прав (например, модераторы/обычные пользователи).
* Разработка frontend-части:
* Верстка шаблонов с адаптивным дизайном.
* Добавление интерактивных элементов (поиск, фильтры, пагинация).
* Тестирование: проверка работоспособности всех функций, включая обработку пользовательского ввода и отображение контента.
* Документирование: описание архитектуры проекта, руководство для пользователей и разработчиков.

1.Предпроектное исследование

1.1Описание предметной области

Областью применения веб-платформы «Wiki по вселенной Ведьмака» является создание централизованной информационной системы для сбора, систематизации и предоставления знаний о мире, созданном Анджеем Сапковским.

Современная фэнтези-культура требует структурированного подхода к хранению и представлению обширной информации о художественных вселенных. Вселенная Ведьмака, включающая книги, игры, сериалы и другие медиа, содержит множество взаимосвязанных элементов: персонажей, локаций, событий, магических систем и исторических фактов, которые требуют систематизации для удобства изучения и анализа фанатами и исследователями.

Веб-платформа предназначена для автоматизации следующих процессов:

* Создание и редактирование статей о персонажах, локациях, событиях и артефактах вселенной;
* Категоризация и структурирование информации по тематическим разделам;
* Установление связей между различными элементами вселенной через систему ссылок;
* Загрузка и управление медиа-контентом (изображения, карты, схемы);
* Модерация контента и контроль качества информации;
* Поиск и навигация по базе знаний;
* Формирование тематических коллекций и путеводителей.

Пользователи системы являются:

* Читатели — посетители, изучающие информацию о вселенной Ведьмака для развлечения или исследований;
* Редакторы — зарегистрированные пользователи, создающие и редактирующие статьи, добавляющие новую информацию;
* Модераторы — опытные участники, контролирующие качество контента, разрешающие споры и следящие за соблюдением правил;
* Администраторы — технические специалисты, управляющие настройками платформы, правами доступа и обеспечивающие работоспособность системы.

Основные функции модуля:

* Создание и редактирование wiki-статей с поддержкой разметки и мультимедиа;
* Система категорий и тегов для классификации контента;
* Механизм версионности для отслеживания изменений в статьях;
* Интерактивные карты локаций с возможностью детализации;
* Генеалогические древа персонажей и хронологические линии событий;
* Система обсуждений и комментариев к статьям;
* Инструменты поиска с фильтрацией по различным критериям.

Требования к системе включают в себя:

* Масштабируемость: способность обрабатывать большие объемы контента и одновременных пользователей, оптимизация производительности при росте базы данных.
* Удобство использования: интуитивный интерфейс для редактирования, удобная навигация, адаптивный дизайн для различных устройств.
* Надежность: сохранность данных при сбоях, регулярное резервное копирование, система восстановления контента.
* Безопасность: защита от вандализма, система модерации, контроль прав доступа, защита от спама и злоупотреблений.
* Совместимость: поддержка различных форматов медиа-файлов, кроссбраузерность, интеграция с внешними источниками данных.

# 1.2 Обзор инструментальных средств для разработки

Инструменты разработки программного продукта определяют будущий результат.

Проектировать структуру веб-приложения более удобно через инструменты разработки такие как: PhpMyAdmin и Draw.io, а дизайн – через Figma.

Веб-приложения будет состоять из двух частей – клиентская и серверная. Для реализации клиентской части отлично подойдет PyCharm Community Edition .

Серверная часть будет со стандартной базой данных PhpMyAdmin.

PhpMyAdmin – это веб-интерфейс для администрирования баз данных MySQL и MariaDB. Управление базами данных, таблицами, пользователями, импорт и экспорт данных, выполнение SQL-запросов, настройка прав доступа.

Draw.io – это онлайн-инструмент для создания схем, диаграмм и блок-схем. Это бесплатное приложение, которое широко используется для визуализации процессов, архитектуры систем, ER-диаграмм баз данных и других видов диаграмм.

Figma – это онлайн-платформа для дизайна интерфейсов и прототипирования. Она позволяет создавать интерфейсы приложений, веб-сайтов и других цифровых продуктов. Создание макетов интерфейсов, прототипов и взаимодействий между элементами, командная работа в реальном времени. Поддерживает векторную графику и работу с компонентами, что удобно для разработки интерфейсов.

PyCharm Community Edition – это бесплатная интегрированная среда разработки (IDE) для языка программирования Python. Она разработана компанией JetBrains и предоставляет все необходимые базовые инструменты для написания, отладки и запуска кода.

Python – это универсальный язык программирования, широко используемый в различных областях, таких как веб-разработка, анализ данных, машинное обучение, разработка программного обеспечения и многое другое. Он известен своей простотой и читабельностью, что делает его отличным выбором как для начинающих, так и для опытных разработчиков.

Мобильное приложение будет содержать в себе информацию её необходимо хранить, изменять, структурировать и использовать. Это реализуется благодаря базе данных. Были рассмотрены следующие варианты реализации СУБД:

* MySQL;
* MongoDB;
* PostgreSQL.

MySQL – это реляционная система управления базами данных (СУБД), разработанная в 1995 году и ныне принадлежащая Oracle. Реляционные СУБД основаны на моделировании данных в виде таблиц (сущностей) с определенными взаимосвязями между ними. В MySQL данные организованы в виде таблиц с строками и столбцами, а также использует SQL (Structured Query Language) для создания, чтения, обновления и удаления данных (CRUD-операции). SQL предоставляет богатый набор команд для работы с данными и управления структурой базы данных.

MySQL использует архитектуру клиент-сервер, где сервер обрабатывает запросы от клиентов и возвращает им результаты. Это позволяет использовать MySQL в распределенных системах, где приложения взаимодействуют с сервером базы данных по сети.MySQL предлагает несколько движков хранения, такие как InnoDB (с поддержкой транзакций и внешних ключей) и MyISAM (более быстрый, но без поддержки транзакций).

MongoDB – это система управления базами данных NoSQL, разработанная в 2009 году компанией MongoDB Inc. В отличие от реляционных СУБД, MongoDB хранит данные в виде документов BSON (бинарная версия JSON), что позволяет создавать сложные и вложенные структуры данных.

Документы в MongoDB организованы в коллекции (аналог таблиц в реляционных базах данных). Каждый документ представляет собой JSON-подобную структуру, состоящую из пар «ключ-значение». Это позволяет хранить данные с разными структурами в одной коллекции.

MongoDB не требует заранее определенной схемы, что делает её гибкой и позволяет изменять структуру данных на лету. Это упрощает работу с изменяющимися данными и быстрое прототипирование. MongoDB использует собственный язык запросов, основанный на операторах и фильтрах для поиска, обновления и агрегации данных. Это позволяет выполнять сложные операции, такие как группировка и фильтрация данных.

PostgresSQL – это реляционная система управления базами данных с открытым исходным кодом, впервые выпущенная в 1989 году. Это одна из самых мощных СУБД, поддерживающая как традиционные реляционные структуры, так и современные расширения, такие как JSON и геопространственные данные.

PostgreSQL поддерживает широкий спектр возможностей SQL, таких как сложные запросы, подзапросы, оконные функции, общие табличные выражения (CTE) и агрегатные функции, что делает его подходящим для аналитических задач и работы с большими объемами данных.

PostgreSQL поддерживает пользовательские типы данных, хранимые процедуры, и даже хранение сложных структур данных, таких как JSONB (бинарный JSON). Это делает его полезным как для традиционных реляционных задач, так и для хранения неструктурированных данных. PostgreSQL поддерживает репликацию (синхронную и асинхронную), что позволяет создавать резервные копии и настраивать отказоустойчивость. Также поддерживает распределенные транзакции и параллельные запросы.

Для наглядности сравнения вариантов реализации базы данных была составлена таблица 1.

Таблица 1 - Сравнение средств реализации базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Название БД | MySQL | MongoDB | PostgreSQL |
| Большое кол-во типов данных | + | - | + |
| Популярность | + | + | + |
| Отказоустойчивость | + | + | + |
| Гибкость в моделировании данных | - | + | - |
| Не требует строгой схемы | - | + | - |
| Поддержка ACID | + | - | + |
| Масштабируемость | + | + | + |
| Подходит для аналитики | - | - | + |
| Простота установки | + | + | - |
| Сообщество и документация | + | + | + |

Таким образом, несмотря на то, что MySQL обладает преимуществами в простоте установки и скорости операций чтения, PostgreSQL лучше соответствует требованиям проекта по надежности, гибкости и функциональности. Это делает PostgreSQL разумным выбором для разработки масштабируемого, безопасного и функционально богатого веб-приложения для систематизации веб-приложения.

Для наглядности сравнения вариантов среди языков программирования была составлена таблица 2.

Таблица 2 - Сравнение языков программирования для разработки веб-приложения

| Критерии | Python | Java | С# |
| --- | --- | --- | --- |
| Простота изучения | Высокая | Средняя | Средняя |
| Экосистема и библиотеки | Богатая, включая Django, Flask | Широкая, корпоративные фреймворки | Хорошо, интеграция с .NET |
| Производительность | Средняя | Высокая | Высокая |
| Разработка веб-приложений | Отличная | Широкая | Хорошая |
| Безопасность | Хорошая | Высокая | Хорошая |

Python был выбран в качестве базового языка благодаря высокой простоте изучения и быстроте разработки, а также богатой экосистеме и мощной поддержке веб-приложений через фреймворк Django. Это обеспечивает эффективное создание надёжных, масштабируемых и легко сопровождаемых решений для систематизации веб-приложения.

PyCharm Community Edition — это бесплатная версия интегрированной среды разработки (IDE) PyCharm от компании JetBrains, предназначенная для инновационных разработчиков и небольших проектов на языке Python. Она предоставляет базовые инструменты для написания, отладки и тестирования кода на Python, включая поддержку таких языков и форматов, как HTML, XML, JSON, YAML и Markdown.

Visual Studio Code (VS Code) — это бесплатный, простой и видимый редактор исходного кода от компании Microsoft, доступный для Windows, macOS и Linux [1](https://sky.pro/wiki/profession/visual-studio-code-opisanie-i-funkcii/)[4](https://visualstudio.microsoft.com/ru/) . Он поддерживает множество языков программирования «из коробки», Python, JavaScript, Java, C++ и других, а также позволяет расширять функционал с помощью большого количества плагинов и расширений.

IntelliJ IDEA — это мощная интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для профессиональной разработки программного обеспечения на Java, Kotlin, а также поддерживающая множество других языков, включая JavaScript и Python. Она кроссплатформенная и работает на Windows, macOS и Linux.

Для наглядности сравнения вариантов среды разработки была составлена таблица 3.

Таблица 3 - Сравнение IDE сред для проектирования веб-приложения

| Критерии | PyCharm Community Efition | Visual Studio Code | IntelliJ IDEA |
| --- | --- | --- | --- |
| Поддержка Python | Отличная | Очень хорошая (через расширение) | Хорошая |
| Удобство и функционал | Богатый функционал | Лёгкая, гибкая | Мощная |
| Бесплатно | Бесплатно | Бесплатно | Платная (есть бесплатная версия) |

Для разработки на Python и работы с платформером Django наиболее распространенным является выбор среды PyCharm и Visual Studio Code. Обеспечиваем бесплатную версию с полным набором инструментов для эффективного написания, отладки и тестирования кода. PyCharm выделяет мощный функционал, специально ориентированный на Python-разработчиков, включая интеллектуальное автодополнение и встроенный отладчик, что обеспечивает глубокую интеграцию с Python-экосистемой. Visual Studio Code, в свою очередь, предлагает легкую и гибкую среду с поддержкой расширения Python, что делает ее удобной и быстрой для настройки. IntelliJ IDEA, хоть и мощная и универсальная IDE с поддержкой Python через плагины, является платной (за исключением ограниченной версии) и ориентирована больше на мультиязычную разработку. Таким образом, PyCharm и Visual Studio Code предлагают бесплатные, функционально насыщенные и удобные решения для эффективной работы с Python и Django Для проведения анализа языков и средств разработки выбор был сделан с использованием языка Python и редактора PyCharm, поскольку они позволяют легко и удобно создавать веб-приложения. PyCharm предлагает широкий набор инструментов для написания, отладки и внедрения кода на Python, что значительно улучшает разработку и повышает качество конечного продукта.

1. Для визуального проектирования структуры базы данных используется бесплатный и открытый инструмент PhpMyAdmin, который легко интегрируется с различными потоками управления контентом и позволяет эффективно управлять данными веб-приложений.
2. Для построения структурных схем, контекстных диаграмм и декомпозиции диаграмм применяются CASE-инструменты, такие как Draw.io, благодаря его простому и доступному понятному интерфейсу, доступности онлайн и удобству использования даже для пользователей без традиционных технических знаний.
3. Для создания прототипа веб-интерфейса был выбран онлайн-редактор Figma, который предоставляет бесплатную базовую версию, кроссплатформенность (работает в браузере и в приложении), большое количество шаблонов и возможность расширения функционала с помощью плагинов. Это позволяет быстро и эффективно разрабатывать дизайн и интерактивные прототипы веб-приложений.

Веб-приложения, разработанные с использованием Python и современных веб-фреймворков, обладают следующими преимуществами:

Удобство в использовании. Веб-приложения обеспечивают доступ к функциям и сервисам через любой браузер с адаптивным и стандартным интерфейсом.

Высокая производительность. Оптимизация серверной и клиентской части позволяет быстро обрабатывать запросы и обеспечивать стабильную работу приложения.

Персонализация. Использование данных пользователя, таких как предпочтения и поведение, позволяет создавать индивидуальные пользовательские опыты и рекомендации.

Таким образом, сочетание Python и PyCharm вместе с современными инструментами проектирования и прототипирования обеспечивает ресурс и качественную разработку современных веб-приложений.

2.Проектирование программного модуля

Одним из важнейших этапов разработки является проектирование диаграмм, которые помогут лучше понять структуру нашего приложения и работоспособность в целом.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

Диаграмма прецедентов для музыкального приложения будет включать одного актера – Пользователь. Прецедент включает регистрацию и авторизацию, поиск треков, функционал взаимодействия с треками, просмотр добавленных треков и создание плейлистов.

На рисунке 1 изображена Use Case View, которая показывает структурную схему веб-приложения «Wiki по Ведьмаку» для ролей «Читатель», «Редактор», «Модератор», «Администратор».

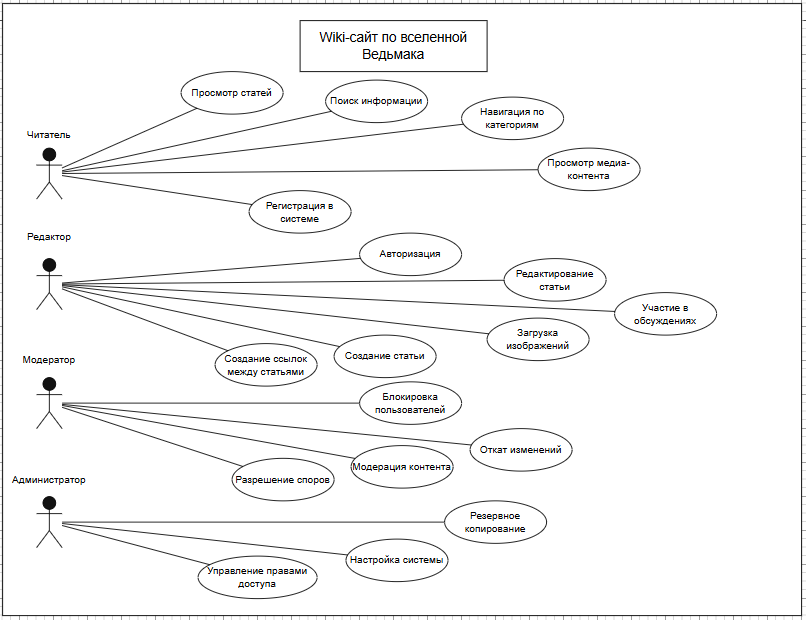


Рисунок 1 – Диаграмма прецедентов

Диаграмма деятельности - UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов - вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

На рисунке 2 изображена диаграмма деятельности веб-приложения «Wiki по Ведьмаку».

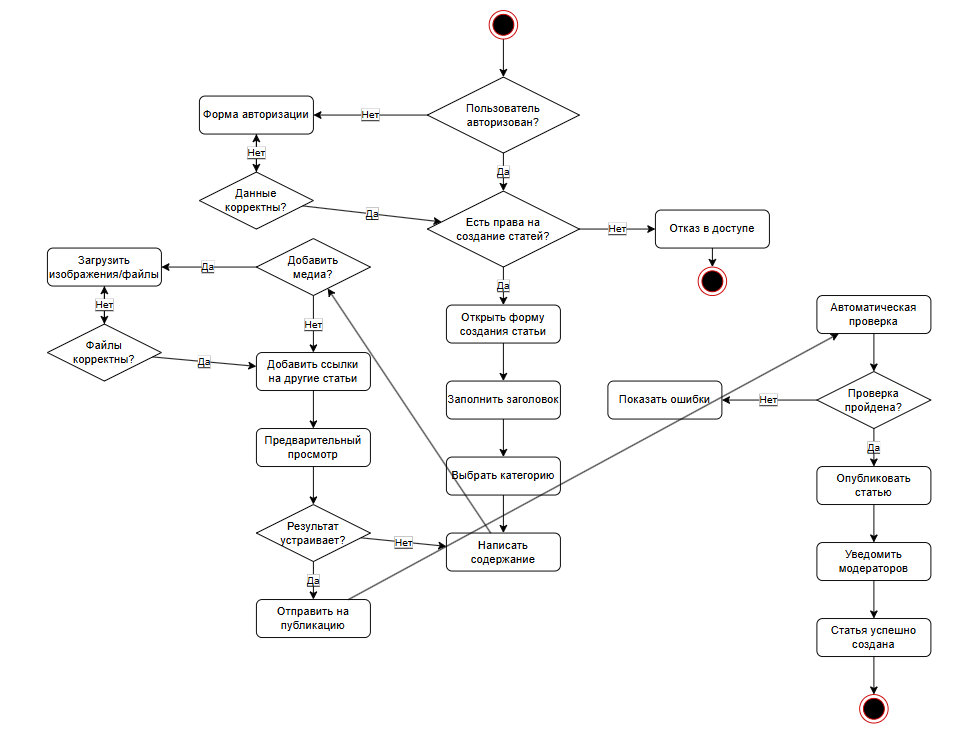


Рисунок 2 – Диаграмма деятельности

В ходе проектирования диаграммы были выделены все действия пользователя при использовании мобильного приложения.

## 2.1Проектирование базы данных

Инфологическая модель базы данных — это абстрактное представление структуры данных, которое описывает основные понятия предметной области и связи между ними. Она служит основой для проектирования базы данных и определяет, какие данные будут храниться, их атрибуты и взаимосвязи.

Основные элементы инфологической модели:

Сущности (объекты) — это основные понятия предметной области, которые представляют собой реальные или абстрактные объекты, о которых собирается информация. Например, в системе учёта товаров сущностями могут быть товары, поставщики, склады и т.д.

Атрибуты — это характеристики сущностей, которые описывают их свойства. Например, для сущности «товар» атрибутами могут быть название, цена, количество на складе и т.д.

Связи — это отношения между сущностями, которые показывают, как они взаимодействуют друг с другом. Связи могут быть многих-ко-многим, один-ко-многим или один-к-одному.

Процесс создания инфологической модели:

Анализ предметной области — это изучение основных понятий и процессов, которые происходят в предметной области.

Выделение сущностей — это определение основных понятий, которые будут храниться в базе данных.

Проверка модели — это анализ модели на предмет её соответствия предметной области и требованиям пользователей.

Инфологическая модель базы данных является основой для проектирования физической модели базы данных, которая будет реализована в конкретной системе управления базами данных(СУБД).

На рисунке 3 представлена инфологическая модель базы данных «Веб-приложение».

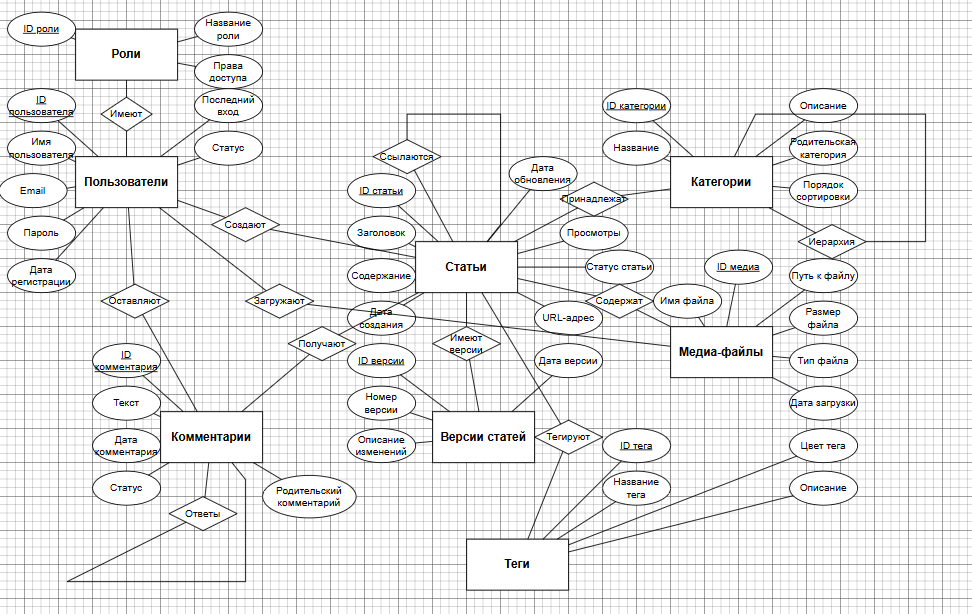


Рисунок 3 – Инфологическая модель базы данных

На модели видно восемь сущностей, которые имеют свои уникальные атрибуты. Также сущности связаны между собой и в сумме получается структура под названием – инфологическая модель.

Проектирование базы данных не ограничивается только этим этапом разработки, а проходит на протяжении всей разработки, до того момента пока в системе не появятся данные, которые нельзя потерять. Результатом проектирования базы данных является ER-модель.

ER-модель (Entity-Relationship model, модель «сущность-связь») — это графическое представление структуры данных, которое используется для проектирования баз данных. Она позволяет описать сущности (объекты) в системе, их атрибуты (свойства) и связи между ними.

Определение ER-модели:

ER-модель представляет собой диаграмму, состоящую из сущностей (прямоугольников) и связей (линий) между ними. Каждая сущность имеет набор атрибутов, которые описывают её характеристики.

Описание ER-модели:

Сущности (Entities):

- Сущности представляют объекты в системе, которые имеют уникальные идентификаторы.

- Каждая сущность имеет набор атрибутов, которые определяют её характеристики.

- Примеры сущностей: треки, альбомы, плейлисты.

Атрибуты (Attributes):

- Атрибуты описывают характеристики сущностей.

- Каждый атрибут имеет имя и тип данных (например, строка, число, дата).

- Примеры атрибутов: имя пользователя, email, password.

Связи (Relationships):

- Связи определяют отношения между сущностями.

- Связи могут быть одного-к-одному (1:1), одного-ко-многим (1:N) или многие-ко-многим (M:N).

- Примеры связей:один трек может быть добавлен в несколько плейлистов.

Ключи (Keys):

- Ключи используются для идентификации сущностей.

- Первичный ключ уникально идентифицирует каждую сущность в базе данных.

- Внешний ключ связывает сущности через связи.

Ограничения (Constraints):

- Ограничения определяют правила для данных в базе данных.

- Примеры ограничений: пользователь должен иметь уникальный идентификатор, альбом должен содержать треки.

ER-модель используется для проектирования баз данных. Она помогает определить структуру данных и связи между ними. ER-модель может быть преобразована в физическую модель базы данных, которая будет использоваться для хранения и обработки данных.

ER-модель позволяет:

- Определить структуру данных.

- Определить связи между данными.

- Определить правила для данных.

- Создать физическую модель базы данных.

Она является важным инструментом для проектирования баз данных, который помогает обеспечить эффективность и надёжность хранения данных.

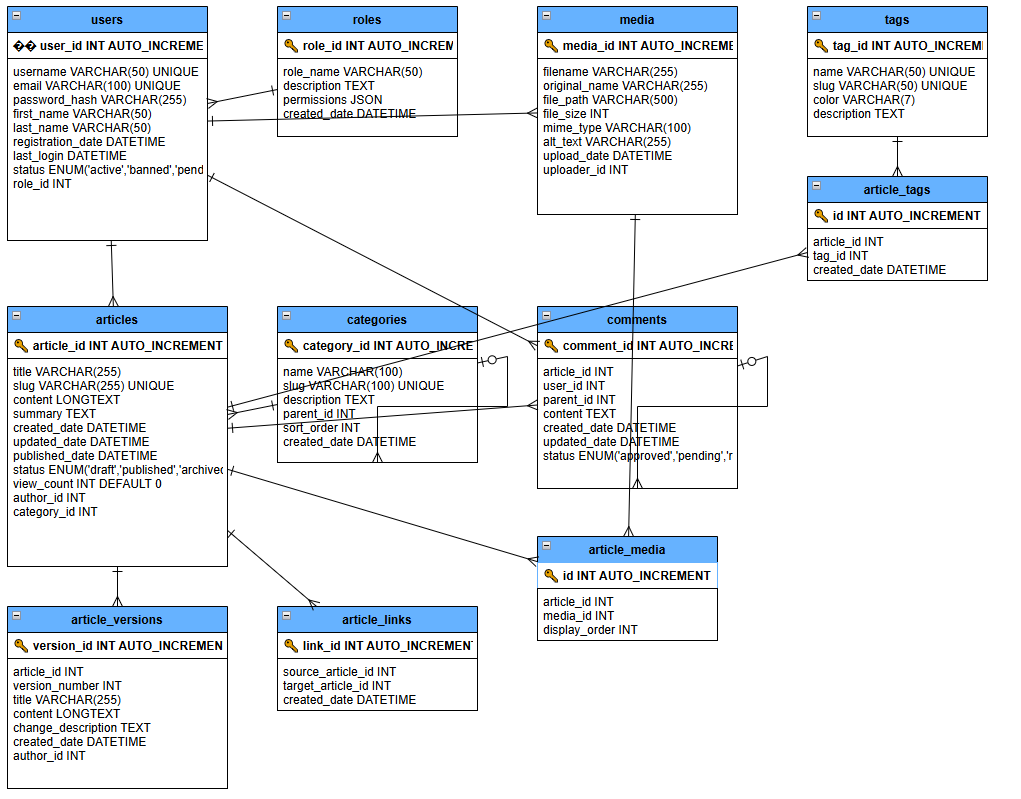


Рисунок 4 – ER-модель базы данных

Всего она содержит 11 таблиц:

Таблица «users», которая предназначена для хранения таких данных как:

* user\_id(уникальный идентификатор пользователя);
* username(уникальный логин пользователя);
* email(уникальная электронная почта);
* password\_hash(хэш пароля пользователя);
* first\_name(имя пользователя);
* last\_name(фамилия пользователя);
* registration\_date(дата регистрации);
* last\_login(дата последнего входа);
* status(статус аккаунта);
* role\_id(ссылка на роль пользователя);

Таблица «roles», предназначенная для назначения ролей пользователей, хранит в себе такие данные как:

* role\_id(уникальный идентификатор роли);
* role\_name(название роли);
* description(описание роли и её полномочий);
* permissions(права доступа в формате JSON);
* created\_at(дата создания роли);

Таблица «articles», предназначенная для хранения статей, содержит в себе такие данные:

* article\_id(уникальный идентификатор статьи);
* title(заголовок статьи);
* slug(ЧПУ-ссылка статьи);
* content(основное содержимое статьи);
* summary(краткое описание/аннотация);
* created\_date(дата создания);
* updated\_date(дата последнего обновления);
* published\_date(дата публикации);
* status(статус статьи);
* view\_count(количество просмотров);
* author\_id(id автора статьи);
* category\_id(id категории);

Таблица «categories», хранит в себе данные категорий для статей:

* category\_id(уникальный идентификатор категории);
* name(название категории);
* slug(ЧПУ-ссылка);
* description(описание категории);
* parent\_id(id родительской категории);
* sort\_order(порядок сортировки в списке);
* created\_date(дата создания);

Таблица «article\_versions», хранит в себе данные об изменение статьи:

* version\_id(уникальный идентификатор версии);
* article\_id(ссылка на статью);
* version\_number(номер версии);
* title(заголовок на момент версии);
* content(содержимое на момент версии);
* change\_description(описание изменений);
* created\_date(дата создания версии);
* author\_id(автор изменений);

Таблица «media», предназначена для хранения медиафайлов:

* media\_id(уникальный идентификатор файла);
* filename(имя файла в системе);
* original\_name(оригинальное имя файла);
* file\_path(путь к файлу);
* file\_size(размер файла);
* mime\_type(тип файла);
* alt\_text(альтернативный текст);
* upload\_date(дата загрузки);
* uploader\_id(пользователь, загрузивший файл);

Таблица «comments», предназначена для хранения данных комментариев статей:

* comment\_id(уникальный идентификатор комментария);
* article\_id(ссылка на статью);
* user\_id(автор комментария);
* parent\_id(родительский комментарий);
* content(текст комментария);
* created\_date(дата создания);
* updated\_date(дата обновления);
* status(статус модерации);

Таблица «tags», предназначена для хранения тегов:

* tag\_id(уникальный идентификатор тега);
* name(название тега);
* slug(ЧПУ-версия названия);
* color(цвет тега);
* description(описание тега);
* created\_date(дата создания);

Таблица «article\_tags», предназначена для хранения связи статей и тегов:

* id(уникальный идентификатор связи);
* article\_id(ссылка на статью);
* tag\_id(ссылка на тег);
* created\_date(дата создания связи);

Таблица «article\_links», предназначена для связи между статьями:

* link\_id(уникальный идентификатор связи);
* source\_article\_id(исходная статья);
* target\_article\_id(целевая статья);
* created\_date(дата создания связи);

Таблица «article\_media», предназначена для связи статей и медиафайлов:

* id(уникальный идентификатор связи);
* article\_id(ссылка на статью);
* media\_id(ссылка на медиафайл);
* display\_order(порядок отображения);

Сама база данных была доведена до третьей нормальной формы. Но прежде, чем она стала таковой пришлось пройти три этапа, это:

− Нормализация до первой нормальной формы.

− Нормализация до второй нормальной формы.

* Нормализация до третьей нормальной формы.

Нормальная форма — это характеристика отношения в реляционной модели данных, которая описывает его с точки зрения избыточности, способной привести к логическим ошибкам при выборке или изменении данных. Нормальная форма определяется набором требований, которым должно соответствовать отношение. Существует шесть нормальных форм, и в данной базе данных достигнута третья.

При нормализации до первой нормальной формы были выполнены условия, обеспечивающие её соответствие. В частности, все данные должны быть атомарными, то есть каждое поле должно содержать только одно значение. Это условие было соблюдено, и таблица находится в первой нормальной форме.

Для перехода ко второй нормальной форме необходимо, чтобы таблица уже соответствовала первой нормальной форме, а также чтобы таблицы были взаимосвязаны. Это позволяет избежать дублирования данных между таблицами. Если записи повторяются в разных таблицах, то при изменении данных потребуется обновлять их в нескольких местах, что нежелательно.

После приведения базы данных ко второй нормальной форме был выполнен следующий этап нормализации - переход к третьей нормальной форме. Этот процесс включал в себя дополнительные преобразования структуры данных для дальнейшего устранения избыточности и потенциальных аномалий при работе с данными.

Результатами приведения базы данных к третьей нормальной форме стали:

* Полностью устранены транзитивные зависимости между атрибутами;
* Каждая таблица содержит только те данные, которые логически относятся к её сущности;
* Все неключевые атрибуты зависят исключительно от первичного ключа своей таблицы;
* Связи между таблицами осуществляются только через внешние ключи.

Таким образом, представлена вся необходимая информация для понимания системы хранения данных.

## 2.2Проектирование интерфейса

Проектирование пользовательского интерфейса — это процесс создания удобного и интуитивно понятного интерфейса для взаимодействия пользователя с продуктом или системой.

Интуитивно понятный дизайн: интерфейс должен быть интуитивно понятным, чтобы пользователи могли легко ориентироваться в нём и выполнять нужные действия без затруднений.

Логика и последовательность: элементы интерфейса должны быть расположены логично и последовательно. Это поможет пользователям быстро находить нужную информацию и выполнять действия.

На рисунке 5 представлена главная страница веб-приложения.



Рисунок 5 – Главная страница сайта

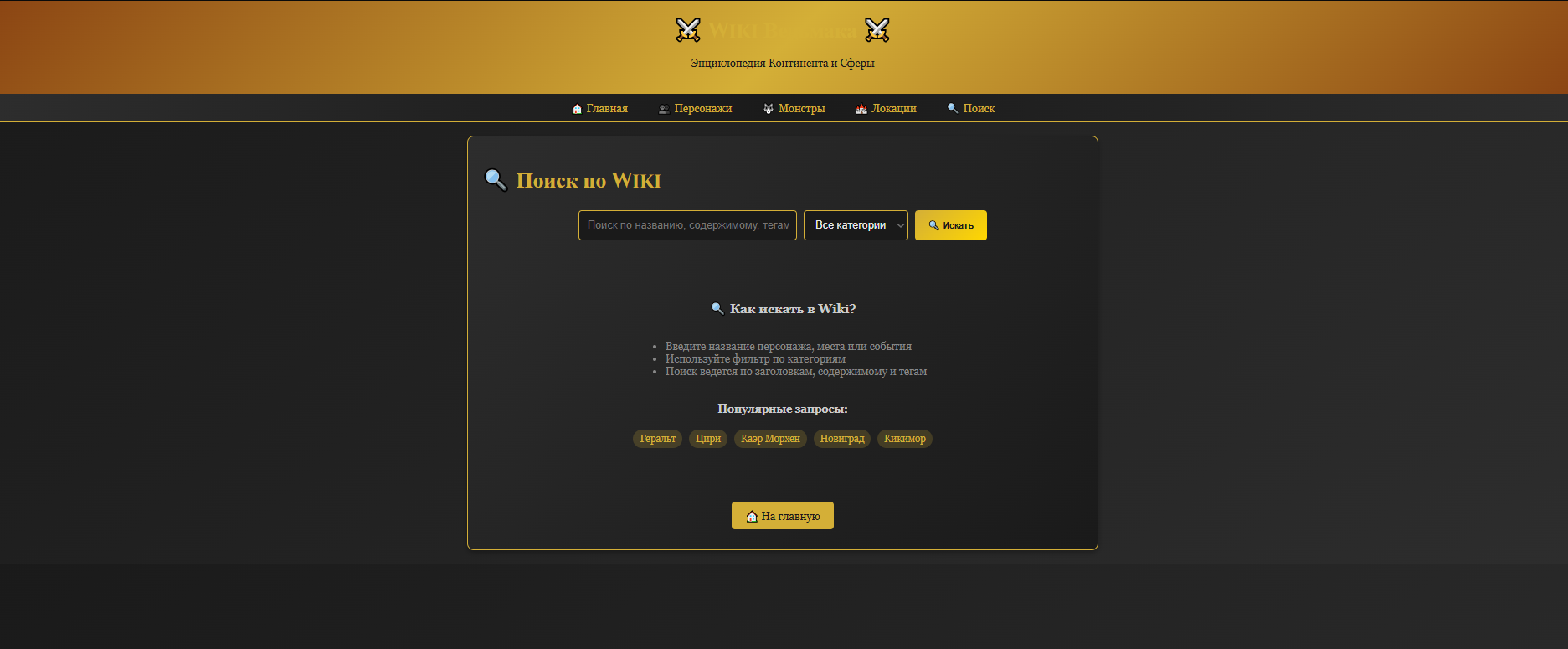


Рисунок 6 – Страница поиска

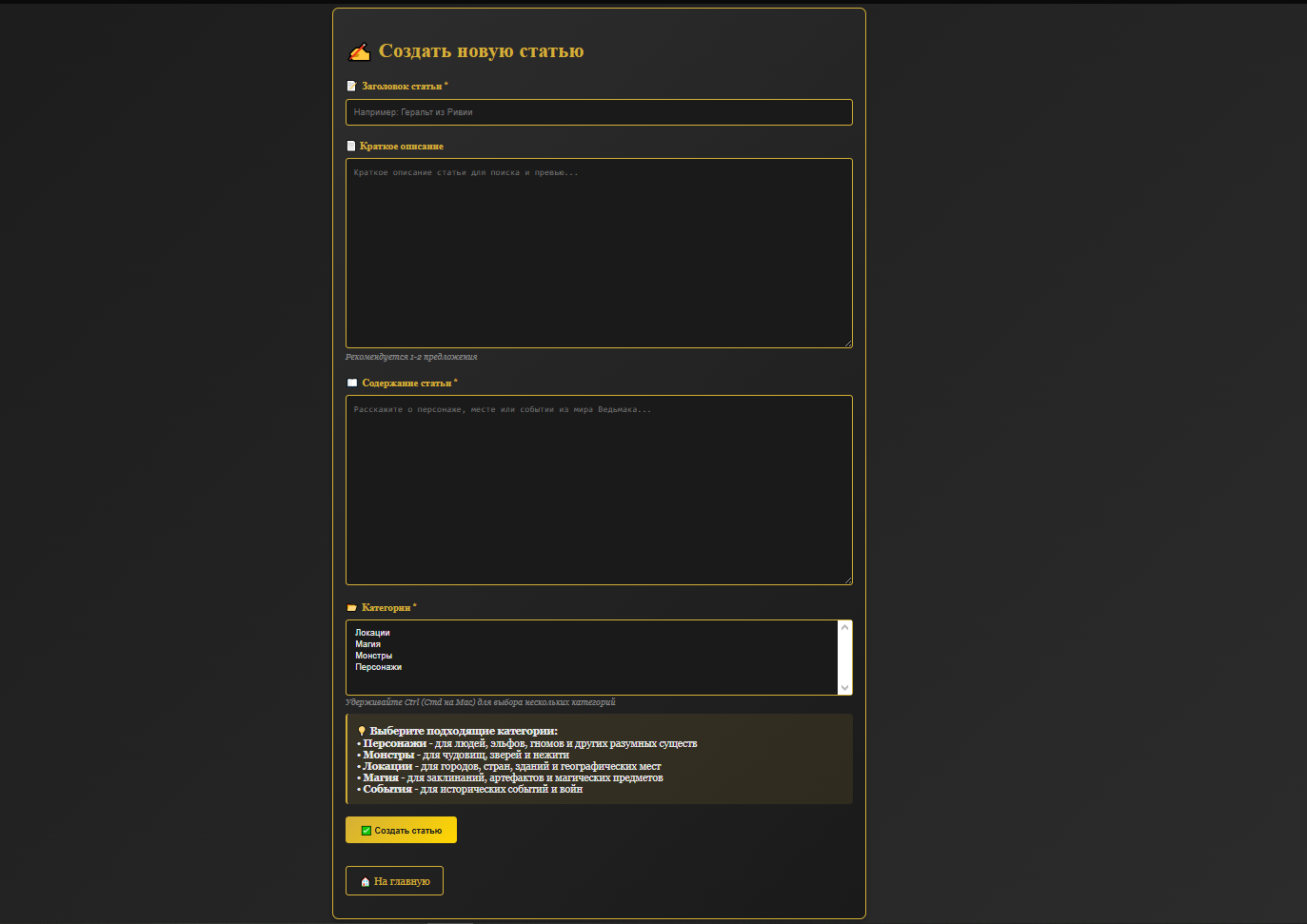


Рисунок 7 – Страница создания статьи

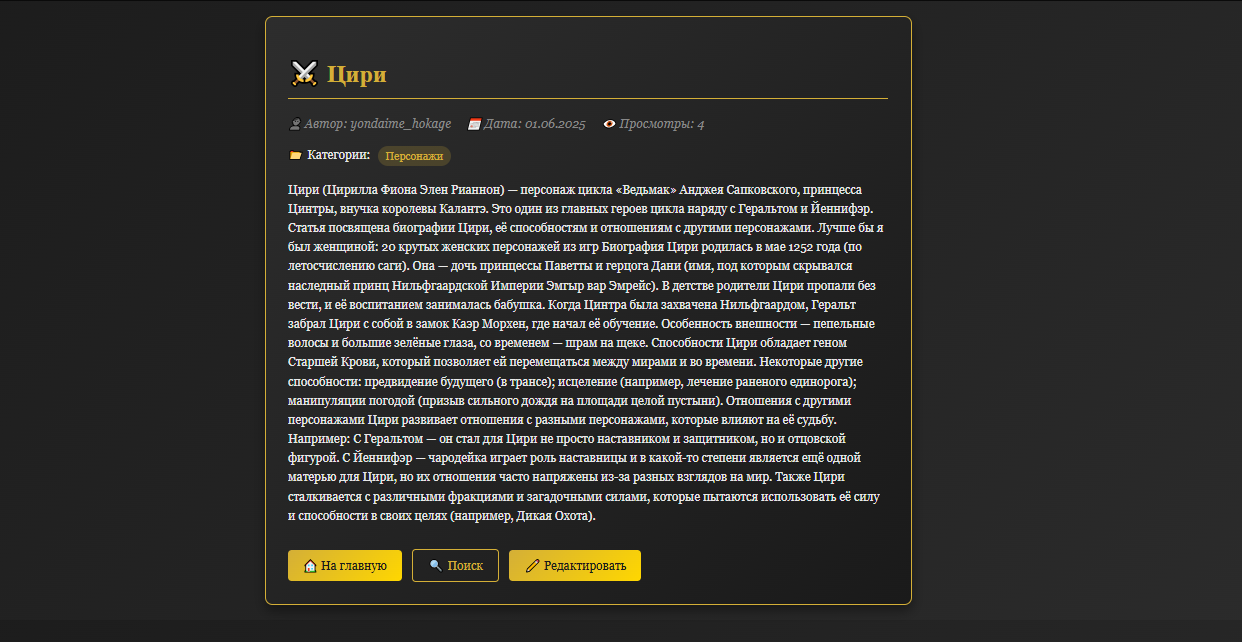


Рисунок 8 – Страница статьи

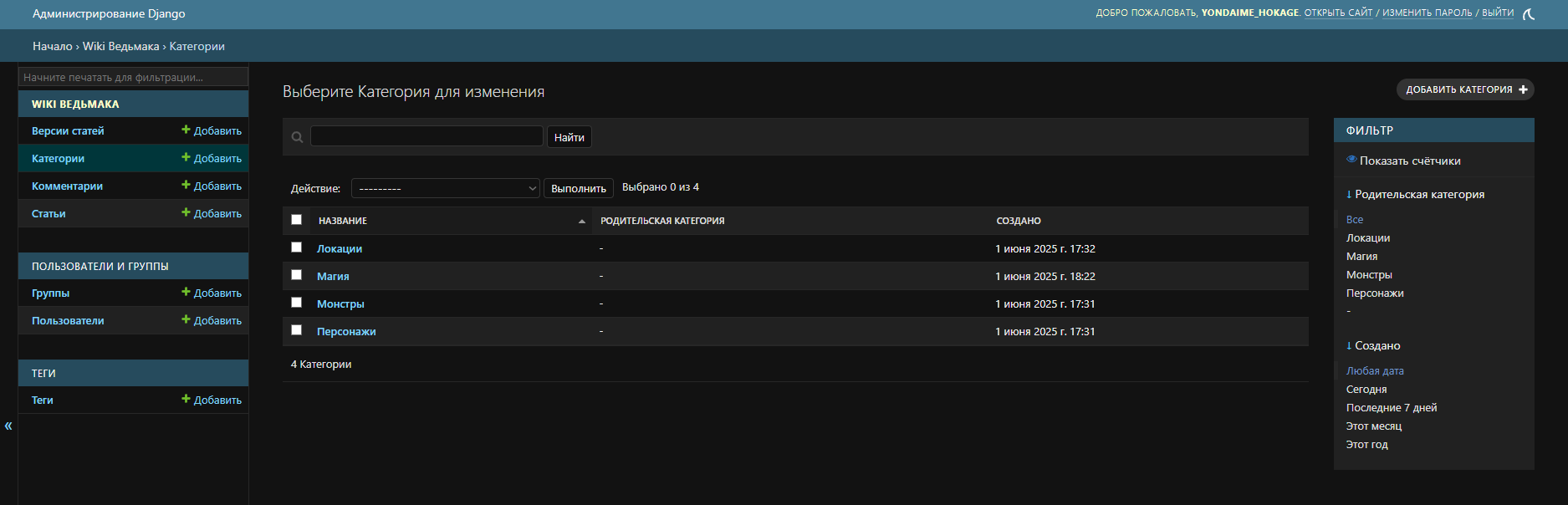


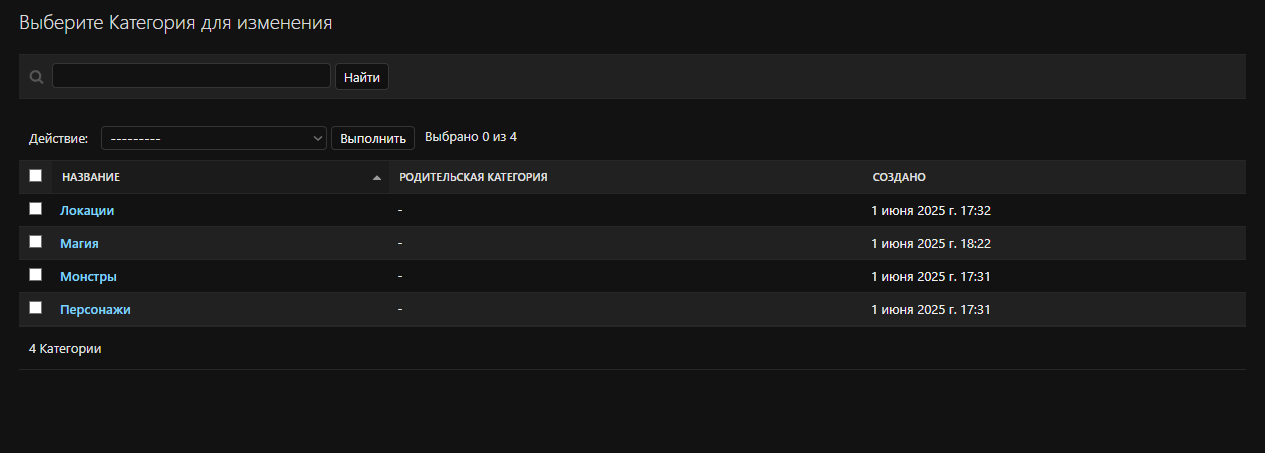
Рисунок 9 – Страница администрирования Django 

Рисунок 10 – Страница создания категорий

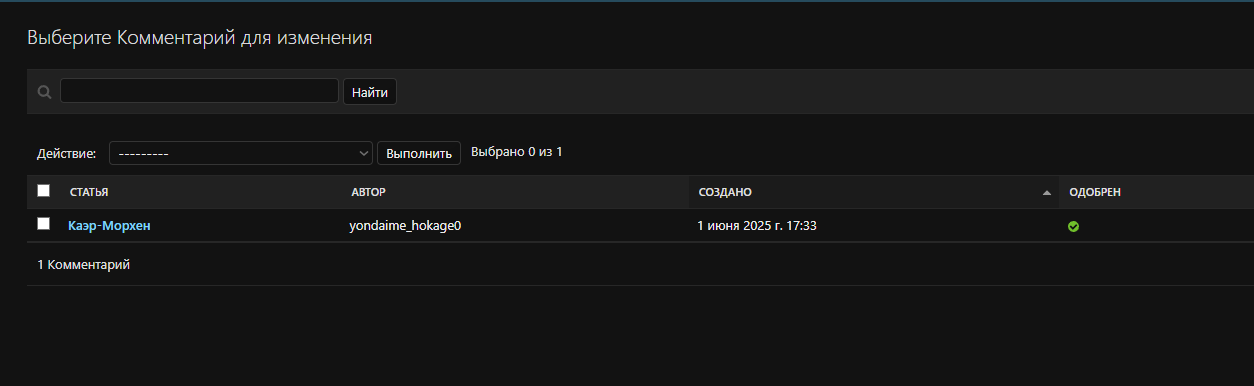


Рисунок 11 – Страница добавления комментария

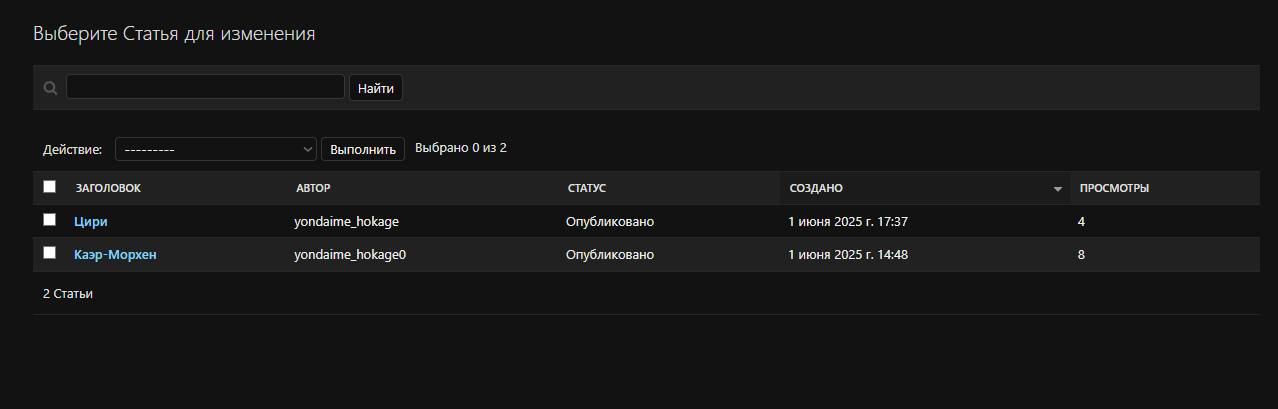


Рисунок 12 – Страница изменения статей

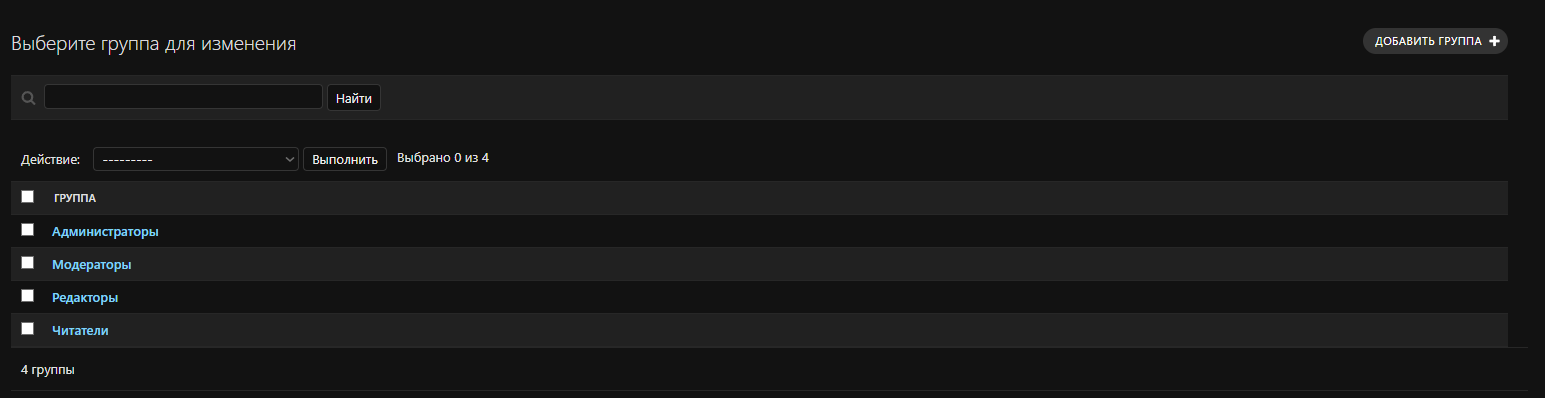


Рисунок 13 – Страница просмотра и создания групп пользователей

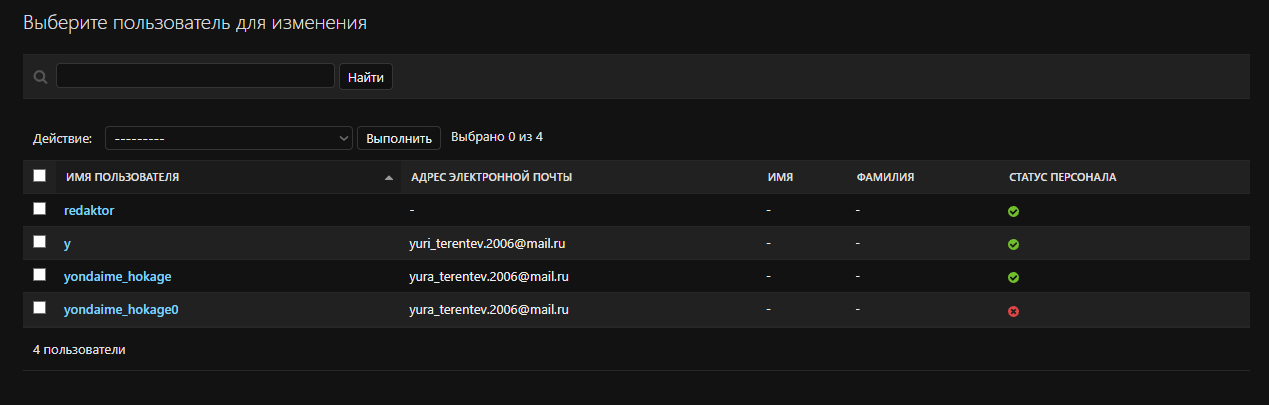


Рисунок 14 – Страница просмотра и редактирования пользователей

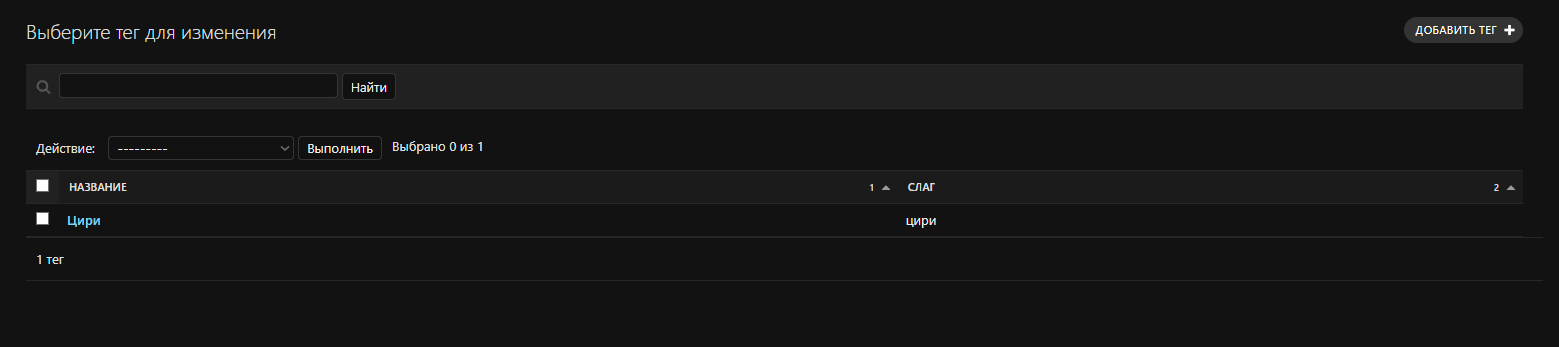


Рисунок 15 – Страница создания тегов

## 3.Разработка программного модуля

Разработка интерфейса – это процесс создания пользовательского интерфейса для мобильного приложения. Интерфейс включает в себя разработку дизайна экрана, расположение элементов управления, выбор цветов и шрифтов, а также тестирование интерфейса на удобство использования.

На рисунке 16 изображен код интерфейса для главной страницы веб-приложения.



Рисунок 16 – Код интерфейса главной страницы(main.py/views)

На рисунке 17 изображен код интерфейса для страницы поиска статей в веб-приложении.

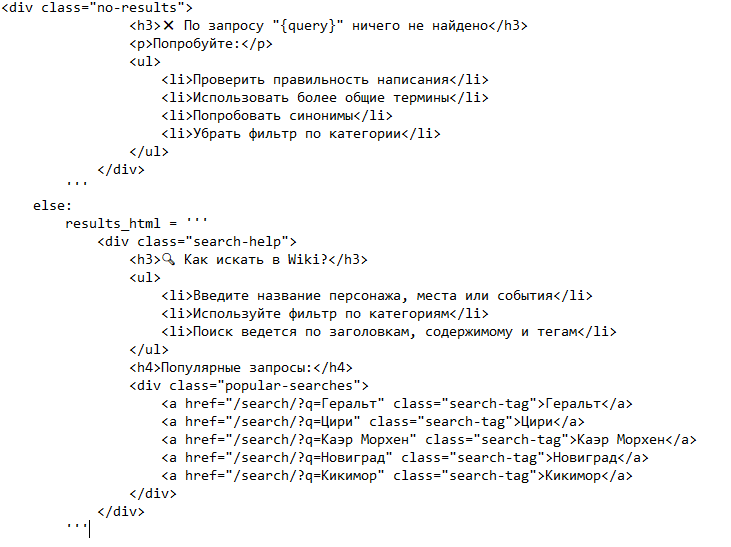


Рисунок 17 – Код интерфейса страницы поиска статей(wiki.py)

На рисунке 18 изображен код интерфейса для страницы создания статей в веб-приложении.



Рисунок 18 – Код интерфейса страницы создания статей(wiki.py)

## 3.2Создание базы данных

Процесс разработки базы данных для веб-приложения «Wiki по Ведьмаку» включал следующие этапы:

Проектирование схемы:

Определены основные сущности: читатели, редакторы, модераторы, администраторы, статьи.

Разработана структура таблиц и связи между ними.

Реализация структуры производилась с помощью создания таблиц: roles, users, categories, articles, article\_versions, media, comments, tags, article\_tags, article\_links, article\_media. Определены поля для каждой таблицы, включая первичные и внешние ключи.

После реализации программного продукта было произведено тестирование: проверка корректности создания таблиц и связей и тестирование основных операций с данными.

Этот процесс обеспечивает создание эффективной базы данных для хранения всей необходимой информации музыкального плеера, учитывая сложные взаимосвязи между различными типами данных. База данных веб-приложения «Wiki по Ведьмаку» состоит из 11 таблиц.

Структура таблиц соответствует схеме базы данных из пункта 2.2.

Таблица «roles» представлена на рисунке 19.

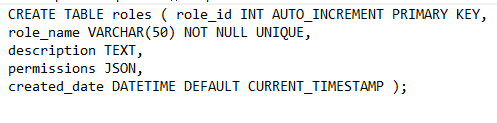


Рисунок 19 – Таблица «roles»

Таблица «users» представлена на рисунке 20.

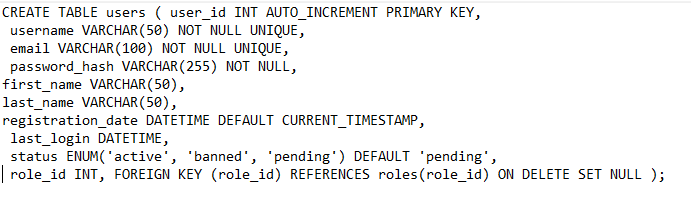


Рисунок 20 – Таблица «users»

Таблица «categories» представлена на рисунке 21.

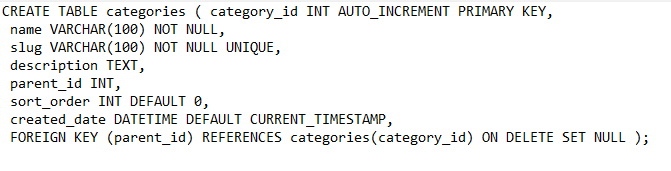


Рисунок 21 – Таблица «categories»

Таблица «articles» представлена на рисунке 22.

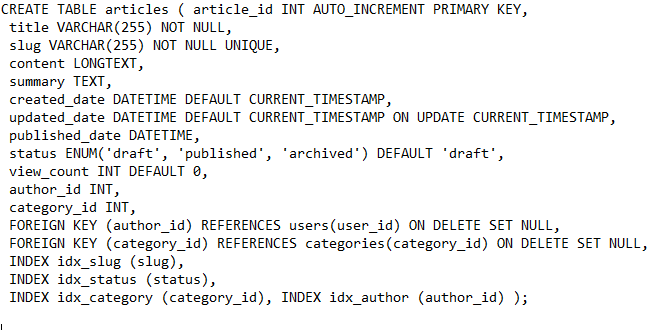


Рисунок 22 – Таблица «articles»

Таблица «article\_versions» представлена на рисунке 23.

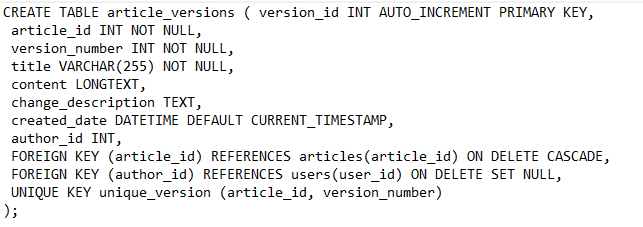


Рисунок 23 – Таблица «article\_versions»

Таблица «media» представлена на рисунке 24.

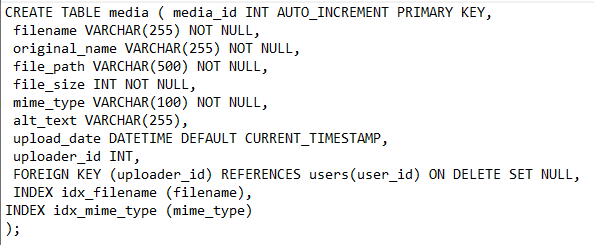


Рисунок 24 – Таблица «media»

Таблица «comments» представлена на рисунке 25.

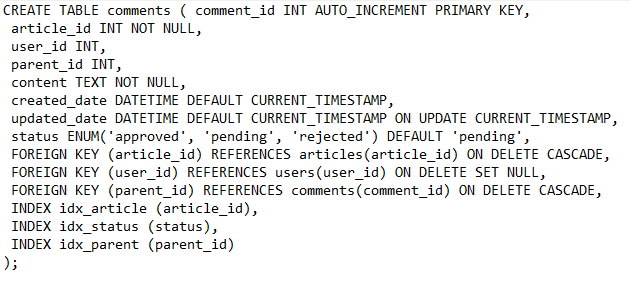


Рисунок 25 – Таблица «comments»

Таблица «tags» представлена на рисунке 26.

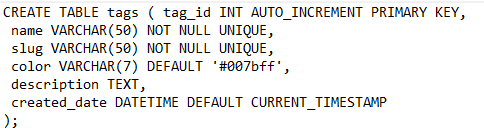


Рисунок 26 – Таблица «tags»

Таблица «article\_tags» представлена на рисунке 27.

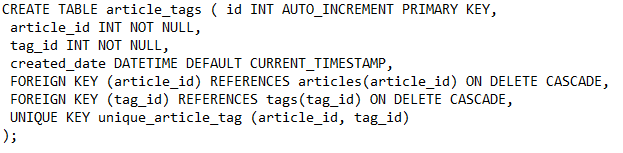
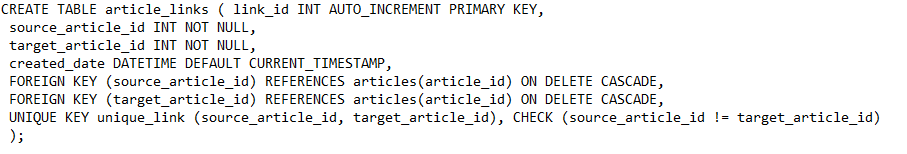


Рисунок 27 – Таблица «article\_tags»

Таблица «article\_links» представлена на рисунке 28.



## 3.3Разработка программного модуля

В соответствии с заданием, необходимо реализовать следующие функции:

Создание статьи.

Просмотр статьи.

Поиск статей.

Разработка функционала веб-приложения «Wiki по Ведтьмаку» была реализована на Python, база данных и работа с ней также реализована на Python. Взаимодействие с базой данных протекает через witcher\_wiki.py.

Создание статьи показано на рисунке 29.

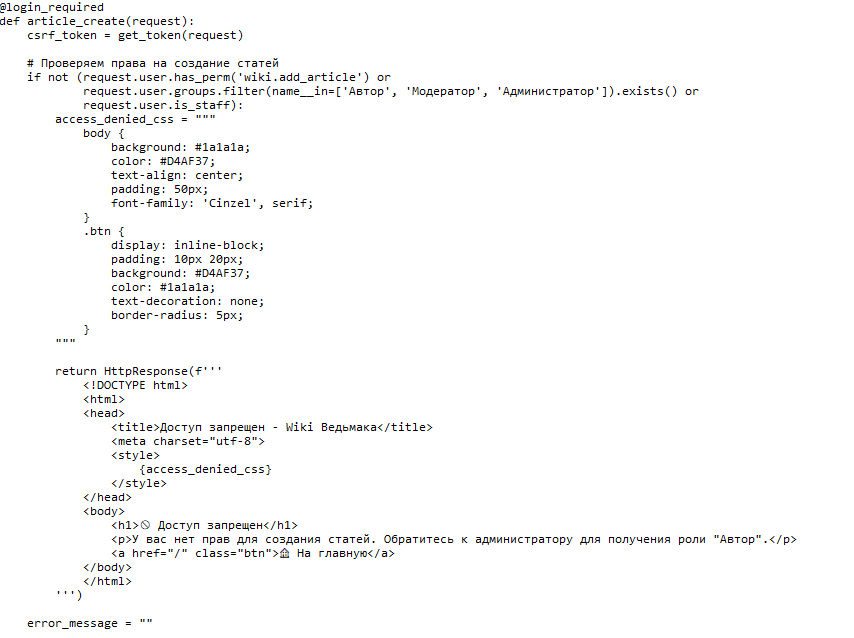


Рисунок 29 – Код создания статьи

Просмотр статьи показан на рисунке 30.



Рисунок 30 – Код просмотра статьи

Поиск статьи по сайту показан на рисунке 31.



Рисунок 31 – Код поиска статьи

## 4.Тестирование программного модуля

Тестирование веб-приложения является критически важным этапом разработки, позволяющим выявить потенциальные ошибки и удостовериться в корректности работы всех функций системы. В данном документе представлены тестовые сценарии, включающие как успешные, так и неудачные кейсы, которые демонстрируют поведение системы при различных условиях. Тестирование охватывает ключевые функциональные модули приложения, включая авторизацию пользователей и управление контентом. Результаты тестирования помогут оценить стабильность работы приложения и выявить области, требующие доработки.

**Сценарий 1**

Таблица 4 – Сценарий тестирования на валидацию данных при регистрации/авторизации

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 07.06.2025 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Проверка на валидацию при регистрации |
| Этапы теста | Пользователь заходит на сайт.  Нажимает на кнопку регистрации.  Заполняет свои данные: почта, никнейм, пароль, подтверждение пароля, выбирает роль. |
| Тестовые данные | [qwe@mail.ru](mailto:qwe@mail.ru), qweqwe, qweqwe123, qweqwe123 |
| Ожидаемый результат | Валидация и проверка действительно существующий почты не позволит пользователю зарегистрироваться. |
| Фактический результат | Пользователь не зарегистрирован. |

Таблица 5 – Чек-лист безуспешной регистрации пользователя

**Чек-лист 1**

|  |  |
| --- | --- |
| Текстовый пример # | 1 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Регистрация пользователя |
| Краткое изложение теста | Во время проведение теста, пользователю необходимо ввести личные данные: никнейм, почта, пароль. |
| Этапы теста | Пользователь заходит на сайт.  Нажимает на кнопку регистрации.  Заполняет свои данные: почта, никнейм, пароль, подтверждение пароля, выбирает роль. |
| Тестовые данные | [qwe@mail.ru](mailto:qwe@mail.ru), qweqwe, qweqwe123, qweqwe123 |
| Ожидаемый результат | Валидация и проверка действительно существующий почты не позволит пользователю зарегистрироваться. |
| Статус | Незачёт |
| Предварительное условие | Пользователю необходимо вводить действительные данные |
| Постусловие | После успешной регистрации система направляет пользователя на главную страницу. |

**Сценарий 2**

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 07.06.2025 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Проверка внесения данных пользователя в базу данных |
| Этапы теста | Пользователь заходит на сайт.  Нажимает на кнопку регистрации.  Заполняет свои данные: почта, никнейм, пароль, подтверждение пароля, выбирает роль. |
| Тестовые данные | [Ivan\_petrov1102@mail.ru](mailto:Ivan_petrov1102@mail.ru), i.petrov, 18929999, 18929999. |
| Ожидаемый результат | Данные успешно внесутся. |
| Фактический результат | Успешно |

**Чек-лист 2**

|  |  |
| --- | --- |
| Текстовый пример # | 2 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Проверка внесения данных пользователя в базу данных |
| Краткое изложение теста | Во время проведение теста, пользователю необходимо ввести личные данные: никнейм, почта, пароль. |
| Этапы теста | Пользователь заходит на сайт.  Нажимает на кнопку регистрации.  Заполняет свои данные: почта, никнейм, пароль, подтверждение пароля, выбирает роль. |
| Тестовые данные | [Ivan\_petrov1102@mail.ru](mailto:Ivan_petrov1102@mail.ru), i.petrov, 18929999, 18929999. |
| Ожидаемый результат | Данные успешно внесутся. |
| Статус | Зачет |
| Предварительное условие | Пользователю необходимо вводить действительные данные |
| Постусловие | После успешной регистрации система направляет пользователя на главную страницу. |

**Сценарий 3**

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 07.06.2025 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Проверка создания статьи на панели Django-администрирования |
| Этапы теста | Администратор заходит на сайт.  Нажимает на кнопку создания статьи.  Заполняет название статьи, краткое описание, содержание статьи, прикрепляет медиа файл, выбирает категорию статьи и публикует. |
| Тестовые данные | Цирилла… Цири (Цирилла Фиона Элен Рианнон) — персонаж цикла «Ведьмак» Анджея Сапковского, принцесса Цинтры, внучка королевы Калантэ.  Персонажи. |
| Ожидаемый результат | Данные успешно будут отображаться на панели Django. |
| Фактический результат | Успешно |

**Чек-лист 3**

|  |  |
| --- | --- |
| Текстовый пример # | 3 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Проверка создания статьи на панели Django-администрирования |
| Краткое изложение теста | Во время проведения теста администратору необходимо полностью создать статью. |
| Этапы теста | Администратор заходит на сайт.  Нажимает на кнопку создания статьи.  Заполняет название статьи, краткое описание, содержание статьи, прикрепляет медиа файл, выбирает категорию статьи и публикует. |
| Тестовые данные | Цирилла…  Цири (Цирилла Фиона Элен Рианнон) — персонаж цикла «Ведьмак» Анджея Сапковского, принцесса Цинтры, внучка королевы Калантэ.  Персонажи. |
| Ожидаемый результат | Данные успешно будут отображаться на панели Django. |
| Статус | Зачет |
| Предварительное условие | Администратору необходимо полностью заполнить статью. |
| Постусловие | После успешного создания статьи, можно будет просмотреть её, увидеть автора, дату создания и количество её просмотров. |

**Сценарий 4**

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 07.06.2025 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Ограничение прав пользователей |
| Этапы теста | Редактор заходит на сайт.  Хочет создать статью.  При нажатии на кнопку создания статьи система автоматически проверит его права. |
| Тестовые данные | - |
| Ожидаемый результат | Редактор не сможет создать статью. |
| Фактический результат | Редактор не создал статью. |

**Чек-лист 4**

|  |  |
| --- | --- |
| Текстовый пример # | 4 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Ограничение прав пользователей |
| Краткое изложение теста | Во время проведения теста редактор попытается создать статью. |
| Этапы теста | Редактор заходит на сайт.  Хочет создать статью.  При нажатии на кнопку создания статьи система автоматически проверит его права. |
| Тестовые данные | - |
| Ожидаемый результат | Редактор не сможет создать статью. |
| Статус | Незачет |
| Предварительное условие | Редактору необходимо получить права администратора. |
| Постусловие | После попытки создания статьи, система покажет сообщение о недостатке прав для выполнения данной операции. |

## 5.Документирование программного модуля

## 5.1Руководство пользователя

Веб-приложение «Wiki по Ведьмаку» имеет простой и удобный интерфейс, что позволяет пользователю легко ориентироваться в его возможностях. При открытии сайта пользователя встречает главная страница, где, для просмотра статей необходимо авторизоваться. Страница авторизации представлена на рисунке 32.

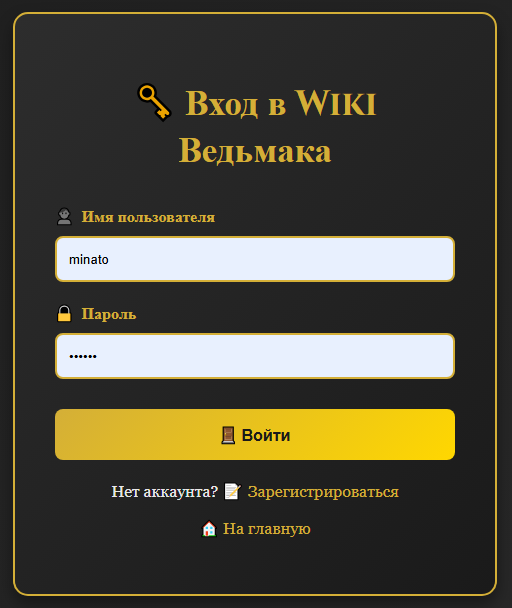


Рисунок 32 – Страница авторизации

Так как пользователь не зарегистрирован, система не даст ему авторизоваться. Для регистрации необходимо нажать кнопку регистрации, страница которой представлена на рисунке 33.

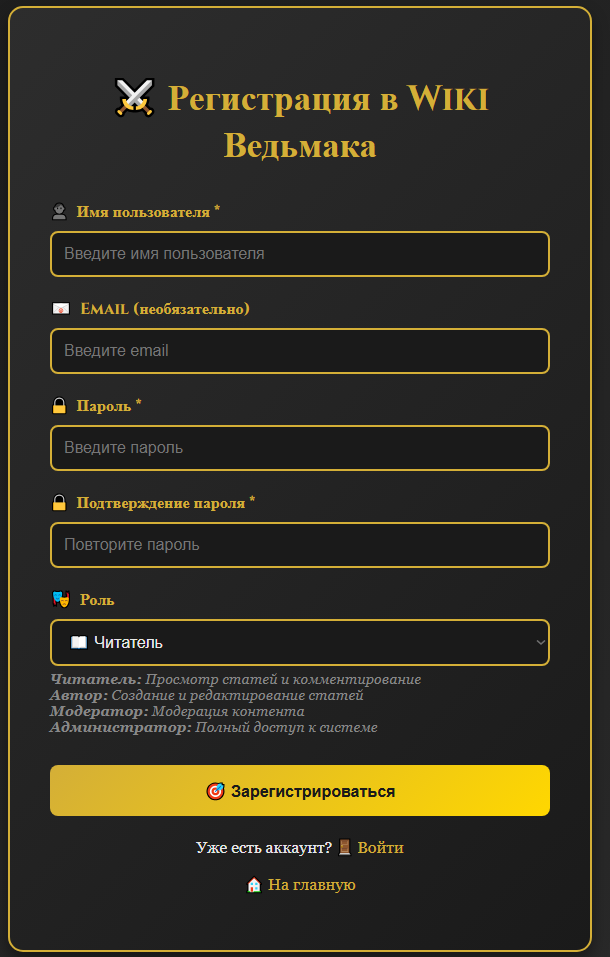


Рисунок 33 – Страница регистрации

После успешной регистрации пользователь может перейти на главную страницу сайта, представленной на рисунке 34.

Рисунок 33 – Страница регистрации

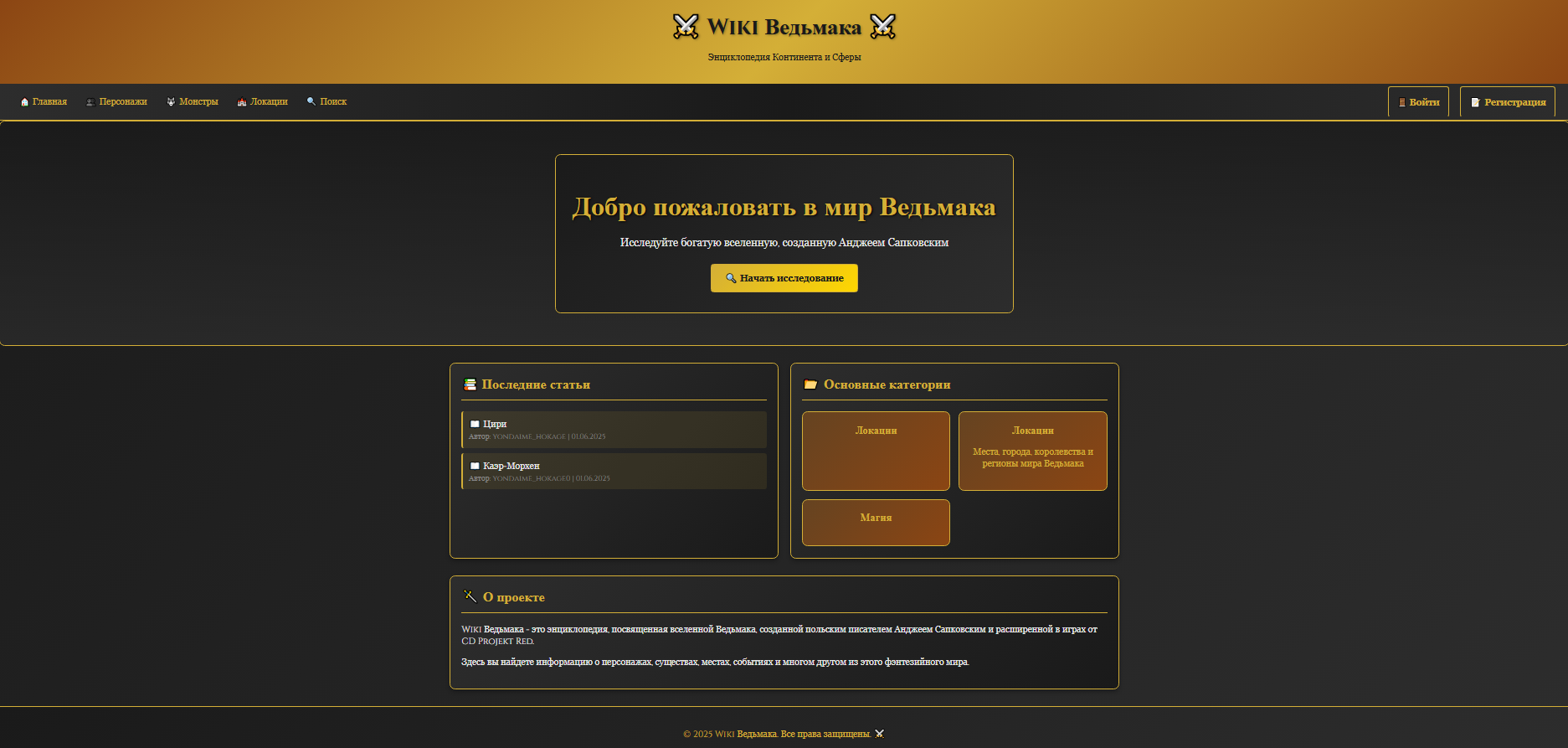


Рисунок 34 – Главная страница

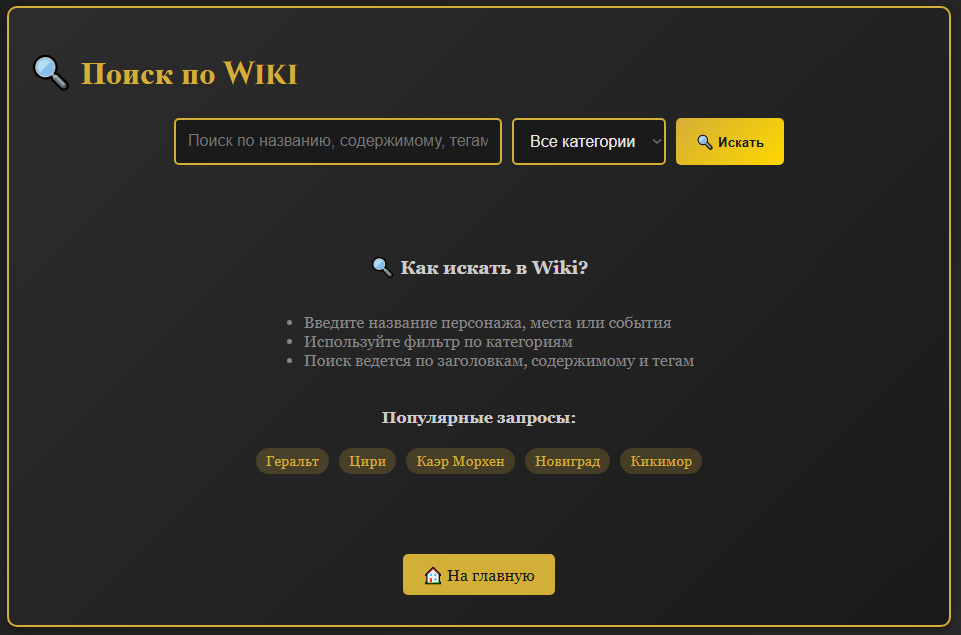
На главной странице пользователь может нажать на кнопку поиска интересующих его статей, страница поиска представлена на рисунке 35. 

Рисунок 35 – Страница поиска статей

Введя в строку поиска интересующую статью, пользователь может открыть её, страница статьи представлена на рисунке 36.

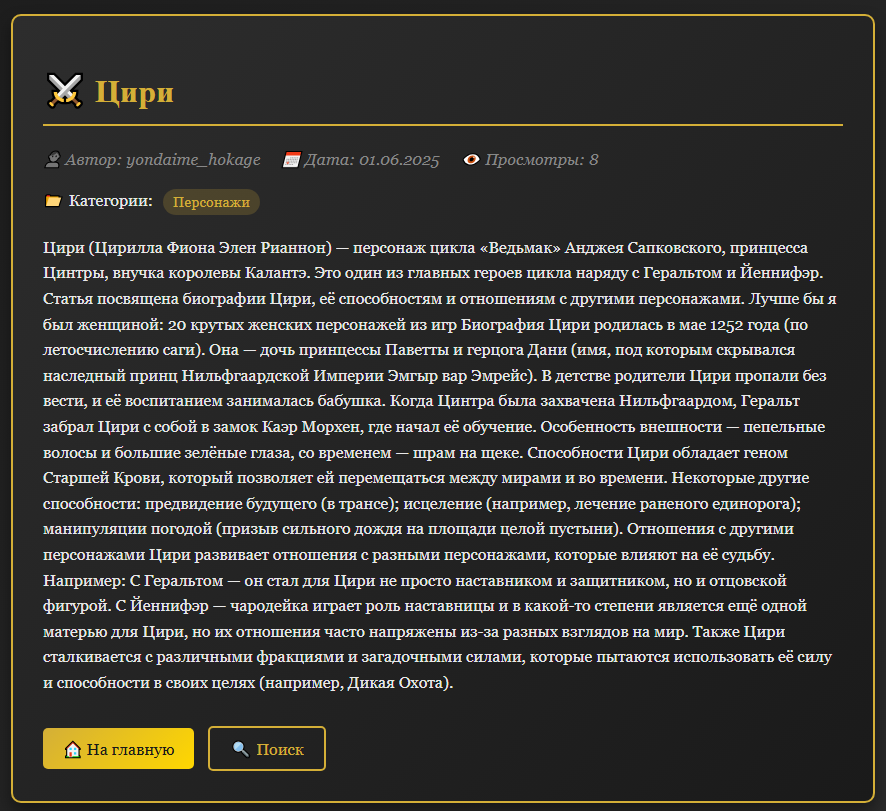


Рисунок 36 – Страница просмотра статьи.

6.Оценка возможности модернизации программного модуля

Оценка возможности модернизации сайта Wiki по вселенной «Ведьмака» показывает высокий потенциал для развития. Существующая структура базы данных, соответствующая 3НФ, обеспечивает надежную основу для внедрения новых функций. Возможна реализация системы рекомендаций на основе предпочтений пользователей, использующая анализ их активности. Добавление интерактивной карты мира с маркерами ключевых локаций значительно улучшит навигацию. Интеграция API игр и сериала позволит автоматически обновлять контент. Внедрение системы рейтингов и обсуждений повысит вовлеченность аудитории. Модернизация дизайна с адаптивными шаблонами обеспечит удобство использования на мобильных устройствах. Добавление мультиязычной поддержки расширит целевую аудиторию. Возможно создание раздела с фанатскими теориями и анализом сюжета. Внедрение инструментов для модерации контента упростит управление сайтом. Оптимизация поиска с использованием полнотекстовых индексов ускорит работу. Реализация REST API позволит разрабатывать мобильные приложения на основе данных вики. Эти улучшения сделают ресурс более современным и удобным для пользователей.

7.Производственные задачи

ПК 01.01. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием

ПК 01.02. Разрабатывать программные модули в соответствии с техническим заданием

ПК 01.03. Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 01.04. Выполнять тестирование программных модулей

ПК 01.05. Осуществлять рефакторинг и оптимизацию программного кода

ПК 01.06. Разрабатывать модули программного обеспечения для мобильных платформ

*Этапы:*

1. Изучить предметную область, для которой необходимо разработать программный модуль. Определить требования к функциональности, надежности, безопасности и другим аспектам системы.
2. Разработать концепцию программного модуля, включающую ее цели, функциональность, структуру и архитектуру. Определить технические требования к модулю, выбрать подходящие технологии и инструменты для его разработки.
3. Создать проектные документы, включающие описание предметной области, архитектуры и схемы баз данных, алгоритмы, прототипы интерфейсов и другие важные аспекты программного модуля.

**Задание на 26.05.25**

1. Исследовать предметную область будущего программного продукта.
2. Провести анализ инструментальных средств, обосновать свой выбор.

* Провести анализ и выбрать язык программирования, инструмент проектирования и среду разработки.
* Определить не менее 3 критериев сравнения.
* Для каждого выбранного инструмента сделать вывод.
* **Задание на 28.05.25**

1. Провести проектирование программного продукта.

* Построить диаграмму прецедентов, деятельности.
* **Задание на 29.05.25**
* Создание ER-диаграммы (сущности, связи, атрибуты).
* Нормализация БД (1NF →2NF → 3NF).
* Написание SQL-скриптов для создания таблиц.
* **Задание на 30.05.25**

1. Прототипирование интерфейса.

* Прототипы и макеты 5 основных окон программного продукта (не считая авторизации/регистрации)
* Описание выбора цвета и типографики
* Описание интерфейса

**Задание на 31.05.-02.06.25**

1. Разработать программный продукт.

* Разработка клиентской части на основе макетов
* Верстка интерфейса
* ПП должен соответствовать выбранной предметной области.
* ПП должен выполнять основные операции (просмотр, добавление, изменение и удаление данных).
* **Задание на 03.06.-06.06.25**

1. Разработка серверной части

* Работа с данными

1. Интеграция и отладка

* Связывание frontend и backend.
* Тестирование взаимодействия.

1. Проверить программный код на предмет соответствия стандартам кодирования:

* Определить стандарты кодирования.
* Структурировать проект и программный код.
* Наименовать переменные соответствующим стандартам.
* Наименовать функции соответствующим стандартам.
* Продемонстрировать комментарии, поясняющие смысл написанного кода.

1. Руководство пользователя (титульный лист, содержание (автособираемое), нумерация страниц, скриншоты функционала ПП, инструкции).

Заключение

В ходе курсовой работы было разработано веб-приложение «Wiki по Ведьмаку». Поставленная цель и входящие в её состав задачи были выполнены. Приложение даёт возможность пользователю искать статьи, просматривать статьи, отправлять свои статьи на проверку модератору, комментировать статьи, добавлять понравившиеся статьи себе в избранное.

Цель работы предполагала решение следующих задач:

* Анализ предметной области.
* Выбор инструментов разработки.
* Проектирование базы данных.
* Разработка backend-части:
* Настройка моделей Django для хранения данных.
* Реализация CRUD-функционала для статей.
* Создание системы аутентификации и разграничения прав (например, модераторы/обычные пользователи).
* Разработка frontend-части:
* Верстка шаблонов с адаптивным дизайном.
* Добавление интерактивных элементов (поиск, фильтры, пагинация).
* Тестирование: проверка работоспособности всех функций, включая обработку пользовательского ввода и отображение контента.
* Документирование: описание архитектуры проекта, руководство для пользователей и разработчиков.

[Список](file:///G:/%D0%9A%D0%9F__%D0%98%D0%BB%D1%8C%D0%B8%D0%BD%20%D0%90%D1%80%D1%82%D1%83%D1%80.docx#_Toc89852065) используемых источников

* 1. Habr / Работа с Django в Pycharm – URL: <https://habr.com/ru/articles/336510/> (дата обращения: 01.06.2025)
  2. Metanit / Основы Django в Pycharm – URL: https://metanit.com/py/pycharm/14.1.php(дата обращения: 02.06.2025)
  3. Unetway / Python Синтаксис – URL: [https://unetway.com/tutorial/sqlite–syntax?ysclid=la0uq3qfnr275255001](https://unetway.com/tutorial/sqlite-syntax?ysclid=la0uq3qfnr275255001) (дата обращения: 02.06.2025). – Текст: электронный.
  4. Developer pycharm / pycharm – URL: <https://developer.pycharm.com/studio> (дата обращения: 03.06.2025). – Текст: электронный.
  5. Pycharm Community Edition / pycharmcommunityedition – URL: [https:// pycharmcommunityedition.org/dl/](https://sqlitebrowser.org/dl/) (дата обращения: 03.06.2025). – Текст: электронный.
  6. Tproger / Руководство по Pycharm– URL: [https://tproger.ru/articles/android–studio–guide/](https://tproger.ru/articles/android-studio-guide/) (дата обращения: 04.06.2025)
  7. Proglib / Основы Python – URL: https://proglib.io/p/python-basics (дата обращения: 04.06.2025)
  8. Blog.skillfactory / Как пользоваться Pycharm – URL: [Pycharm: как пользоваться – начало работы и настройка (skillfactory.ru)](https://blog.skillfactory.ru/kak-polzovatsya-android-studio/?ysclid=lbbvjrnuy5880863937) (дата обращения: 05.06.2025). – Текст: электронный.
  9. Pycharm.ru / Работа с базами данных в Pycharm – URL: <https://pycharm.ru/courses/basics/19–rabota–s–bazami–dannyx–v–pycharm/> (дата обращения: 05.06.2025)
  10. Practicum.yandex / Pycharm: как пользоваться – установка, настройка и установка – URL: https://practicum.yandex.ru/blog/kak–polzovatsya–pycharm/ (дата обращения: 06.07.2025). – Текст: электронный.

# Приложение А Листинг – Работа с созданием статьи

@login\_required

def article\_create(request):

csrf\_token = get\_token(request)

# Проверяем права на создание статей

if not (request.user.has\_perm('wiki.add\_article') or

request.user.groups.filter(name\_\_in=['Автор', 'Модератор', 'Администратор']).exists() or

request.user.is\_staff):

access\_denied\_css = """

body {

background: #1a1a1a;

color: #D4AF37;

text-align: center;

padding: 50px;

font-family: 'Cinzel', serif;

}

.btn {

display: inline-block;

padding: 10px 20px;

background: #D4AF37;

color: #1a1a1a;

text-decoration: none;

border-radius: 5px;

}

"""

return HttpResponse(f'''

<!DOCTYPE html>

<html>

<head>

<title>Доступ запрещен - Wiki Ведьмака</title>

<meta charset="utf-8">

<style>

{access\_denied\_css}

</style>

</head>

<body>

<h1>🚫 Доступ запрещен</h1>

<p>У вас нет прав для создания статей. Обратитесь к администратору для получения роли "Автор".</p>

<a href="/" class="btn">🏠 На главную</a>

</body>

</html>

''')

error\_message = ""

if request.method == 'POST':

title = request.POST.get('title', '').strip()

content = request.POST.get('content', '').strip()

excerpt = request.POST.get('excerpt', '').strip()

category\_ids = request.POST.getlist('categories')

if title and content:

# Создаем slug из заголовка

try:

from unidecode import unidecode

slug = slugify(unidecode(title))

except ImportError:

# Простая транслитерация если unidecode не установлен

translit\_dict = {

'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e', 'ё': 'yo',

'ж': 'zh', 'з': 'z', 'и': 'i', 'й': 'y', 'к': 'k', 'л': 'l', 'м': 'm',

'н': 'n', 'о': 'o', 'п': 'p', 'р': 'r', 'с': 's', 'т': 't', 'у': 'u',

'ф': 'f', 'х': 'h', 'ц': 'ts', 'ч': 'ch', 'ш': 'sh', 'щ': 'sch',

'ъ': '', 'ы': 'y', 'ь': '', 'э': 'e', 'ю': 'yu', 'я': 'ya'

}

title\_lower = title.lower()

for ru, en in translit\_dict.items():

title\_lower = title\_lower.replace(ru, en)

slug = slugify(title\_lower)

# Проверяем уникальность slug

if Article.objects.filter(slug=slug).exists():

counter = 1

original\_slug = slug

while Article.objects.filter(slug=slug).exists():

slug = f"{original\_slug}-{counter}"

counter += 1

# Создаем статью

article = Article(

title=title,

content=content,

excerpt=excerpt,

slug=slug,

author=request.user,

status='published'

)

article.save()

# Добавляем категории

if category\_ids:

categories = Category.objects.filter(id\_\_in=category\_ids)

article.categories.set(categories)

return redirect(f'/article/{article.slug}/')

else:

error\_message = "Пожалуйста, заполните все обязательные поля."

# Получаем все категории для формы

categories = Category.objects.all()

categories\_options = ''.join([f'<option value="{c.id}">{c.name}</option>' for c in categories])

create\_css = """

body {

font-family: 'Georgia', serif;

margin: 0;

padding: 0;

background: linear-gradient(135deg, #1a1a1a 0%, #2d2d2d 100%);

color: #e8e8e8;

}

.header {

background: linear-gradient(135deg, #8B4513 0%, #D4AF37 50%, #8B4513 100%);

color: #1a1a1a;

padding: 20px;

text-align: center;

}

.header h1 {

font-family: 'Cinzel', serif;

margin: 0;

}

.container {

max-width: 800px;

margin: 20px auto;

padding: 20px;

background: linear-gradient(135deg, #2d2d2d 0%, #1a1a1a 100%);

border-radius: 10px;

border: 2px solid #D4AF37;

box-shadow: 0 4px 8px rgba(0,0,0,0.3);

}

.form-group {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 8px;

font-weight: bold;

color: #D4AF37;

font-family: 'Cinzel', serif;

}

input[type="text"],

textarea,

select {

width: 100%;

padding: 12px;

border: 2px solid #D4AF37;

border-radius: 5px;

box-sizing: border-box;

background: #1a1a1a;

color: #e8e8e8;

font-size: 14px;

}

textarea {

height: 300px;

resize: vertical;

}

select[multiple] {

height: 120px;

}

button {

padding: 12px 24px;

background: linear-gradient(135deg, #D4AF37 0%, #FFD700 100%);

color: #1a1a1a;

border: none;

border-radius: 5px;

cursor: pointer;

font-weight: bold;

transition: all 0.3s ease;

}

button:hover {

transform: translateY(-2px);

box-shadow: 0 4px 8px rgba(212, 175, 55, 0.4);

}

.error-message {

color: #ff6b6b;

margin-bottom: 15px;

padding: 10px;

background: rgba(255, 107, 107, 0.1);

border-radius: 5px;

border-left: 3px solid #ff6b6b;

}

h1 {

color: #D4AF37;

font-family: 'Cinzel', serif;

}

small {

color: #888;

font-style: italic;

}

.btn {

display: inline-block;

padding: 10px 20px;

background: transparent;

color: #D4AF37;

text-decoration: none;

border: 2px solid #D4AF37;

border-radius: 5px;

margin-top: 20px;

transition: all 0.3s ease;

}

.btn:hover {

background: #D4AF37;

color: #1a1a1a;

}

.category-help {

background: rgba(212, 175, 55, 0.1);

padding: 15px;

border-radius: 5px;

margin-top: 10px;

border-left: 3px solid #D4AF37;

}

"""

return HttpResponse(f'''

<!DOCTYPE html>

<html>

<head>

<title>Создать статью - Wiki Ведьмака</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://fonts.googleapis.com/css2?family=Cinzel:wght@400;600;700&display=swap" rel="stylesheet">

<style>

{create\_css}

</style>

</head>

<body>

<div class="header">

<h1> Wiki Ведьмака </h1>

<p>Энциклопедия Континента и Сферы</p>

</div>

<div class="container">

<h1> Создать новую статью</h1>

{f'<div class="error-message">❌ {error\_message}</div>' if error\_message else ''}

<form method="post">

<input type="hidden" name="csrfmiddlewaretoken" value="{csrf\_token}">

<div class="form-group">

<label for="title"> Заголовок статьи \*</label>

<input type="text" id="title" name="title" required placeholder="Например: Геральт из Ривии">

</div>

<div class="form-group">

<label for="excerpt"> Краткое описание</label>

<textarea id="excerpt" name="excerpt" rows="3" placeholder="Краткое описание статьи для поиска и превью..."></textarea>

<small>Рекомендуется 1-2 предложения</small>

</div>

<div class="form-group">

<label for="content"> Содержание статьи \*</label>

<textarea id="content" name="content" required placeholder="Расскажите о персонаже, месте или событии из мира Ведьмака..."></textarea>

</div>

<div class="form-group">

<label for="categories"> Категории \*</label>

<select id="categories" name="categories" multiple required>

{categories\_options}

</select>

<small>Удерживайте Ctrl (Cmd на Mac) для выбора нескольких категорий</small>

<div class="category-help">

<strong> Выберите подходящие категории:</strong><br>

• <strong>Персонажи</strong> - для людей, эльфов, гномов и других разумных существ<br>

• <strong>Монстры</strong> - для чудовищ, зверей и нежити<br>

• <strong>Локации</strong> - для городов, стран, зданий и географических мест<br>

• <strong>Магия</strong> - для заклинаний, артефактов и магических предметов<br>

• <strong>События</strong> - для исторических событий и войн

</div>

</div>

<button type="submit"> Создать статью</button>

</form>

<p><a href="/" class="btn"> На главную</a></p>

</div>

</body>

</html> ''')