

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ
по дисциплине «Введение в нереляционные базы данных»
Тема: БД для фанатов Поттерианы

Студент гр. 2383

Иваницкий И.А.

Студент гр. 2383

Исаев Э.Н.

Студентка гр. 2384

Лавренова Ю.Д

Студентка гр. 2384

Соц Е.А.

Студент гр. 2384

Цыганков Р.М.

Преподаватель

Заславский М.М

Санкт-Петербург

2025

ЗАДАНИЕ

Студент Иваницкий И.А. 2383

Студент Исаев Э.Н. 2383

Студентка Лавренова Ю.Д. 2384

Студентка Соц Е.А. 2384

Студент Цыганков Р.М. 2384

Тема работы: БД для фанатов Поттерианы

Исходные данные:

Необходимо сделать сервис для фанатов, где они смогут ознакомится с разными героями / объектами / событиями из вселенной Гарри Поттера, а также визуализировать их взаимосвязи в виде графа.

Используемая база данных – Neo4j.

Содержание пояснительной записи:

«Введение», «Сценарий использования», «Модель данных»,
«Разработанное приложение», «Выводы», «Приложения», «Литература»

Предполагаемый объем пояснительной записи:

Не менее 10 страниц.

Дата выдачи задания: 05.02.2025

Дата сдачи реферата: 20.05.2025

Дата защиты реферата: 20.05.2025

Студент гр. 2383

Иваницкий И.А.

Студент гр. 2383

Исаев Э.Н.

Студентка гр. 2384

Лавренова Ю.Д

Студентка гр. 2384

Соц Е.А.

Студент гр. 2384

Цыганков Р.М.

Преподаватель

Заславский М.М

АННОТАЦИЯ

В рамках дисциплины «Введение в нереляционные базы данных» разработано приложение в команде. Выбрана тема: «БД для фанатов Поттерианы», используя базу данных Neo4j. Во внимание принимаются такие аспекты как производительность приложения и удобство пользования.

SUMMARY

As part of the course "Introduction to Non-relational Databases", an application was developed in a team. The topic chosen was "DB for Potteriana fans", using the Neo4j database. Such aspects as application performance and usability are taken into account.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1. Актуальность решаемой проблемы.....	6
2. Постановка задачи.....	6
3. Предлагаемое решение.....	6
4. Качественные требования к решению.....	6
1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ.....	8
1.1. Макет UI.....	8
1.2. Сценарий использования.....	19
1.3. Преобладающая операция.....	24
2. МОДЕЛЬ ДАННЫХ.....	26
2.1. Нереляционная модель.....	26
2.2. Реляционная модель.....	34
2.3. Сравнение моделей.....	44
3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ.....	46
3.1. Краткое описание.....	46
3.2. Использованные технологии.....	46
3.3. Снимки экрана приложения.....	47
ВЫВОДЫ.....	48
4.1. Достигнутые результаты.....	48
4.2. Недостатки и пути для улучшения полученного решения.....	48
4.3. Будущее развитие решения.....	48
ЗАКЛЮЧЕНИЕ.....	49
ПРИЛОЖЕНИЯ.....	50
ЛИТЕРАТУРА.....	52

ВВЕДЕНИЕ

1. Актуальность решаемой проблемы

Фанатское сообщество, посвященное вселенной Гарри Поттера, насчитывает миллионы людей по всему миру. При этом большинство существующих ресурсов предоставляет информацию о героях, заклинаниях и событиях в разрозненном и неудобном виде, что затрудняет глубокое погружение в мир Поттерианы и поиск нужной информации. Отсутствует единое, структурированное и интуитивно понятное хранилище, которое позволило бы поклонникам эффективно исследовать взаимосвязи между персонажами, артефактами и событиями.

2. Постановка задачи

Необходимо разработать веб-приложение, которое обеспечит удобный доступ к структурированной информации по вселенной Гарри Поттера, позволит выполнять гибкий поиск объектов, фильтрацию данных и визуализацию взаимосвязей между сущностями.

3. Предлагаемое решение

Решено создать веб-приложение для фанатов Поттерианы, в котором пользователи смогут просматривать подробную информацию о персонажах, объектах и событиях волшебного мира, а также изучать их связи в наглядной форме — с помощью интерактивного графа. Приложение будет обеспечивать интуитивно понятный интерфейс и высокую скорость отклика при работе с большими объемами данных.

4. Качественные требования к решению

- Высокая производительность при работе с данными и визуализациях
- Удобный и понятный пользовательский интерфейс

- Возможность расширения функционала (импорт/экспорт данных, статистика, настройка графа)
- Надежность и устойчивость к ошибкам при взаимодействии с базой данных
- Актуальность и полнота отображаемой информации

1. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Перед началом разработки мы подробно проанализировали, как пользователи будут взаимодействовать с сайтом. В этом разделе представлены ключевые сценарии использования — типовые маршруты, по которым пользователи будут достигать своих целей.

Сценарии основаны на реальных потребностях целевой аудитории и учитывают навигационную структуру, элементы интерфейса и возможные действия. Такой подход позволяет нам спроектировать понятный и удобный интерфейс, в котором все функции работают логично и последовательно.

1.1. Макет UI



Рисунок 1.1 – Экран при старте сайта

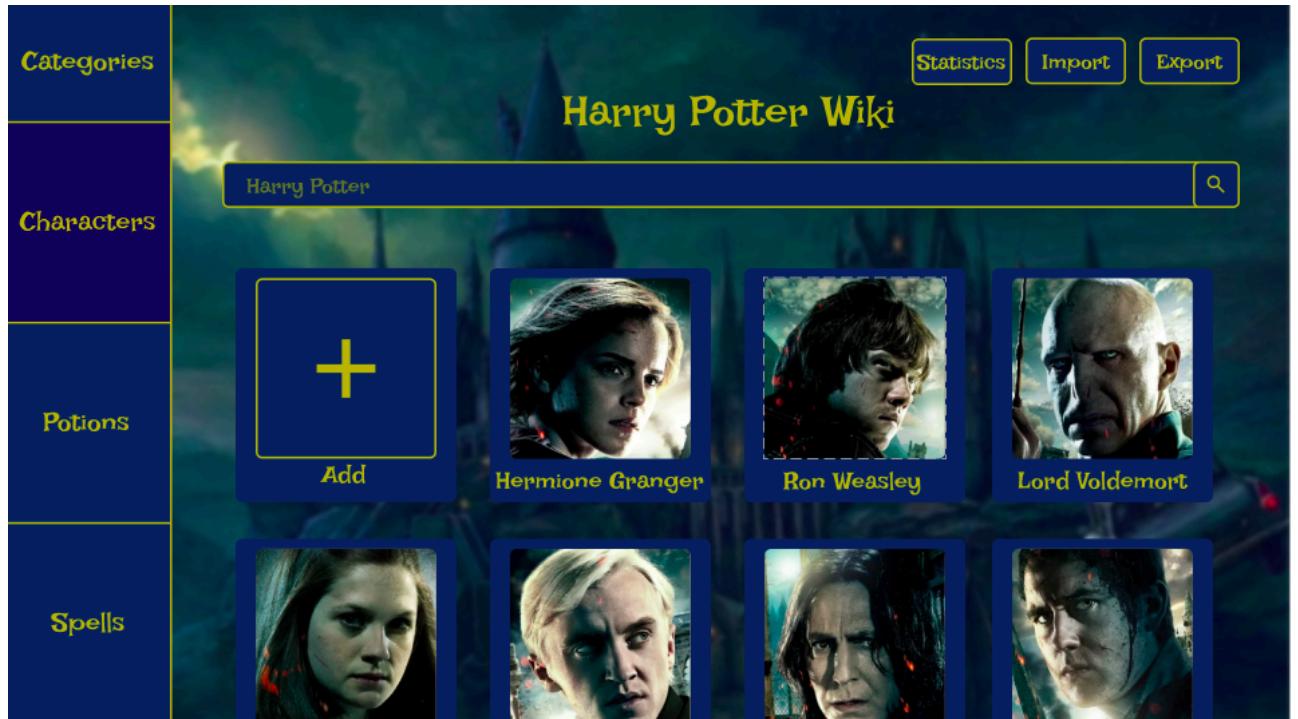


Рисунок 1.2 – Экран просмотра всех персонажей

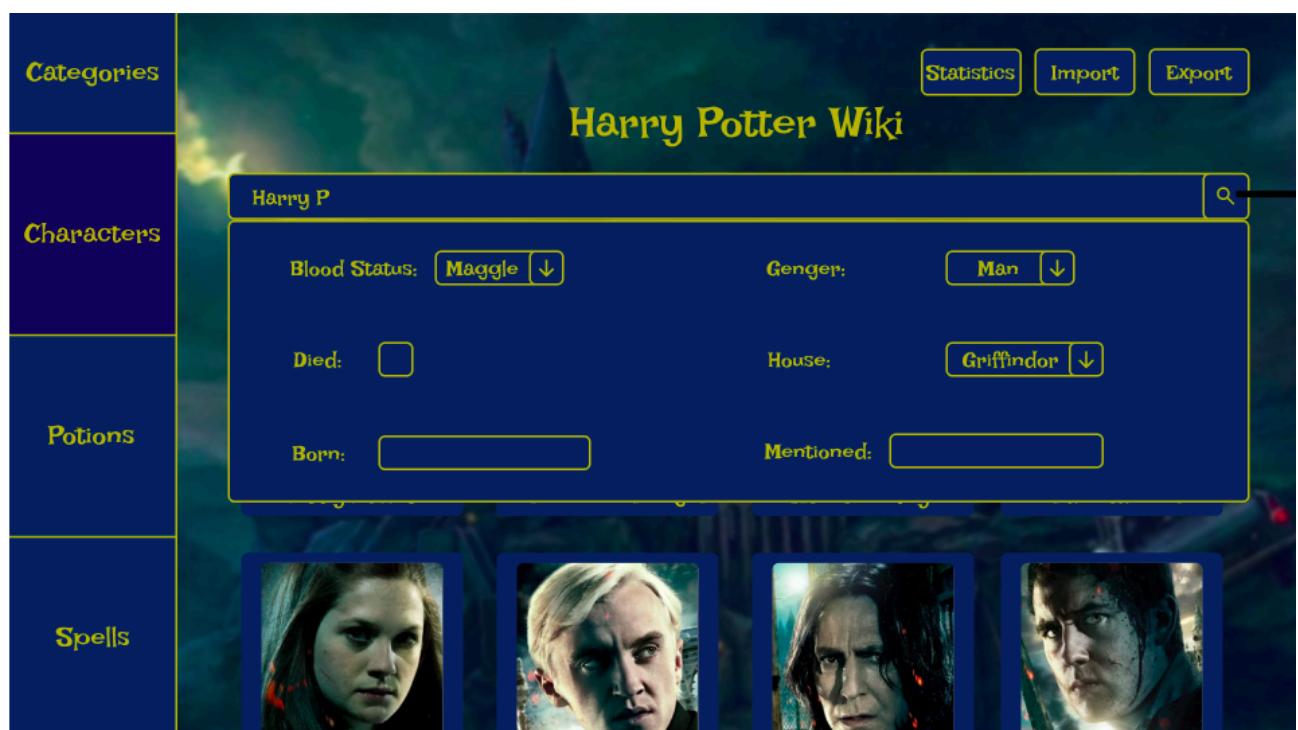


Рисунок 1.3 – Экран поиска персонажа с фильтрами



Рисунок 1.4 – Экран с результатом поиска персонажа

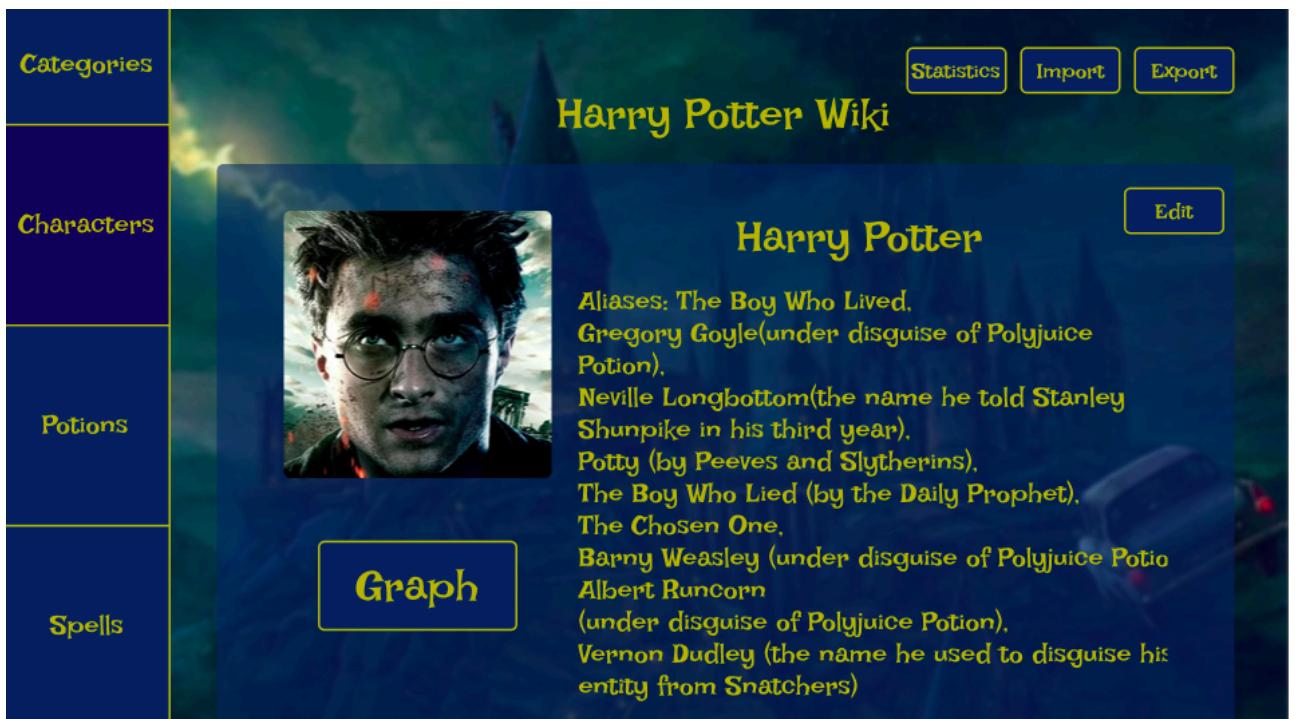


Рисунок 1.5 – Экран с подробной информацией о персонаже

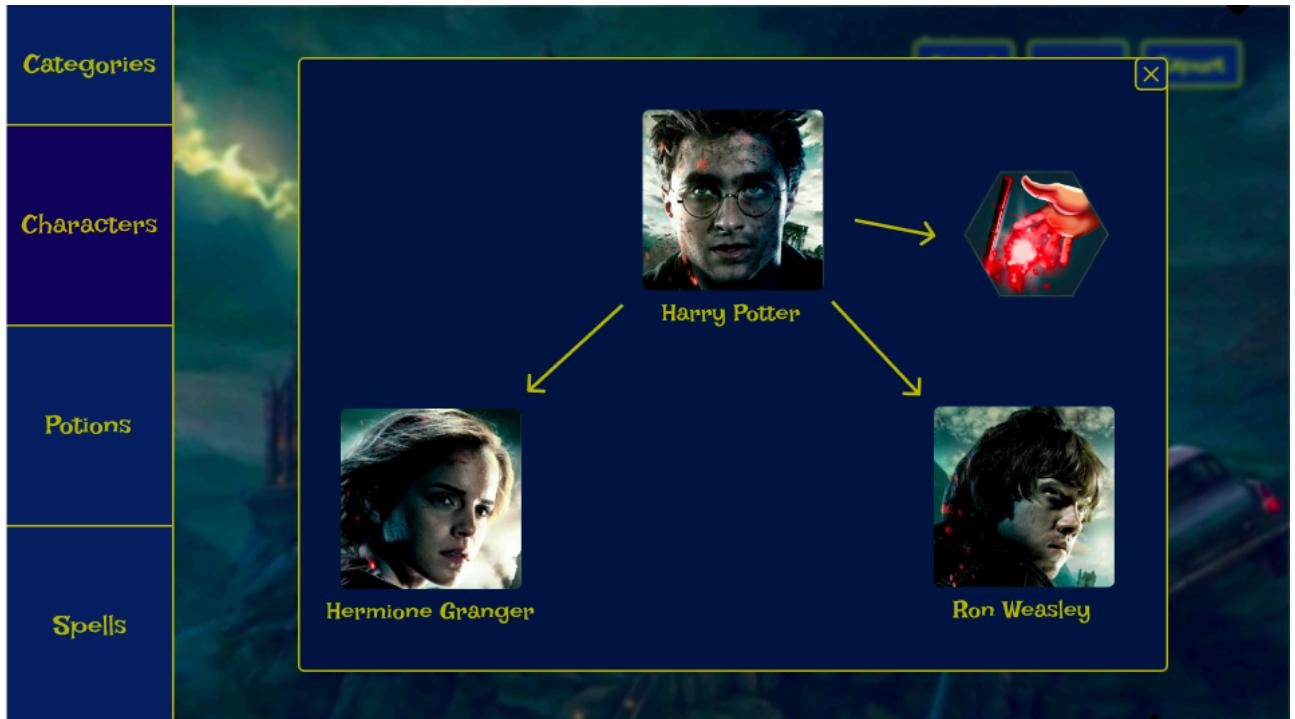


Рисунок 1.6 – Экран с графом связей для выбранного персонажа

The screenshot shows the Harry Potter Wiki character editing screen for Harry Potter. At the top right are buttons for "Statistics", "Import", and "Export". The main area contains the following fields:

- Name:** Harry Potter
- Description:** The Boy Who Lived.
- Blood Status:** Half-blood
- Gender:** Male
- House:** Gryffindor
- Born:** 1980
- Died:**
- Image URL:** https://static.wikia.nocookie.net/harrypotter/images/c/cd/Harry_Potter_DHP.jpg

Below these fields is a "Relations" section with a "Relation Type" dropdown, a "Select Target" dropdown, and an "Add" button. A list of existing relations is shown:

- enemy: Lord Voldemort
- friend: Hermione Granger
- friend: Ron Weasley
- Brewed: Polyjuice Potion
- Knows: Stupefy
- Knows: Expecto Patronum
- Knows: Expelliarmus

On the far right, there is a column of "Remove" buttons for each relation entry. At the bottom right is a "Save" button.

Рисунок 1.7 – Экран редактирования данных конкретного персонажа



Рисунок 1.8 – Экран просмотра всех зелий

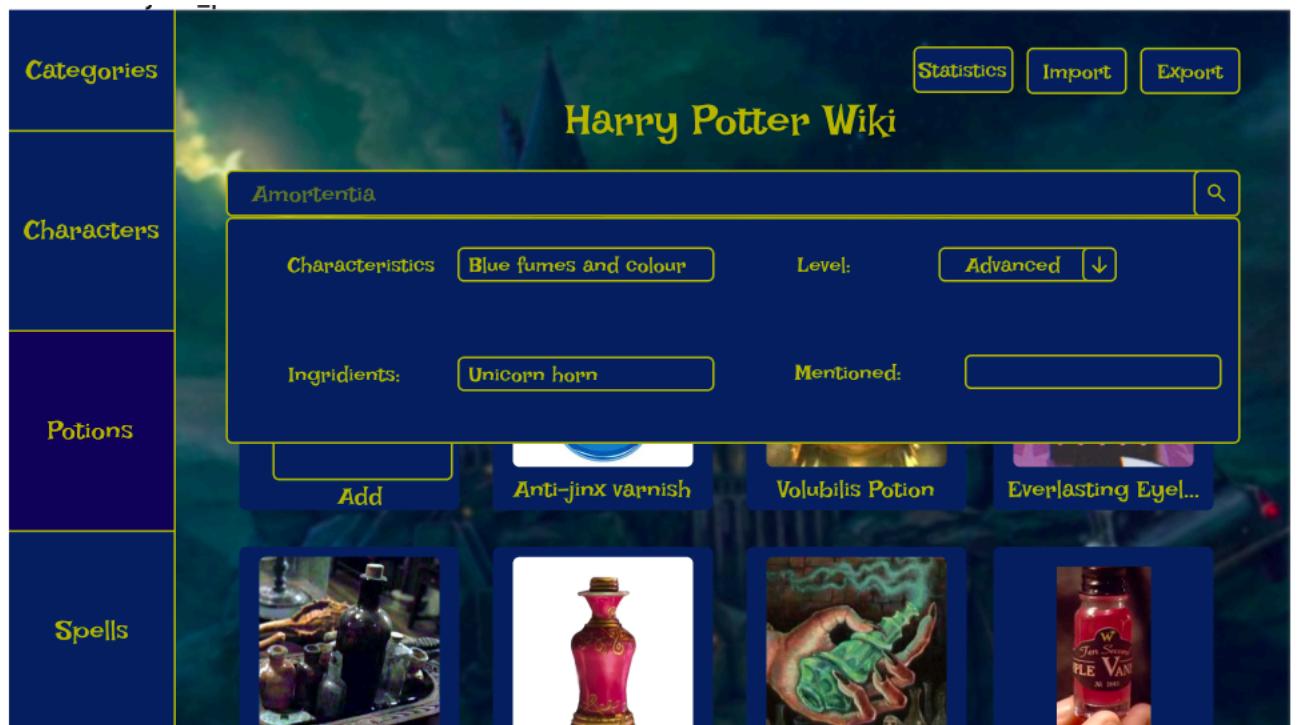


Рисунок 1.9 – Экран поиска зелий с фильтрами

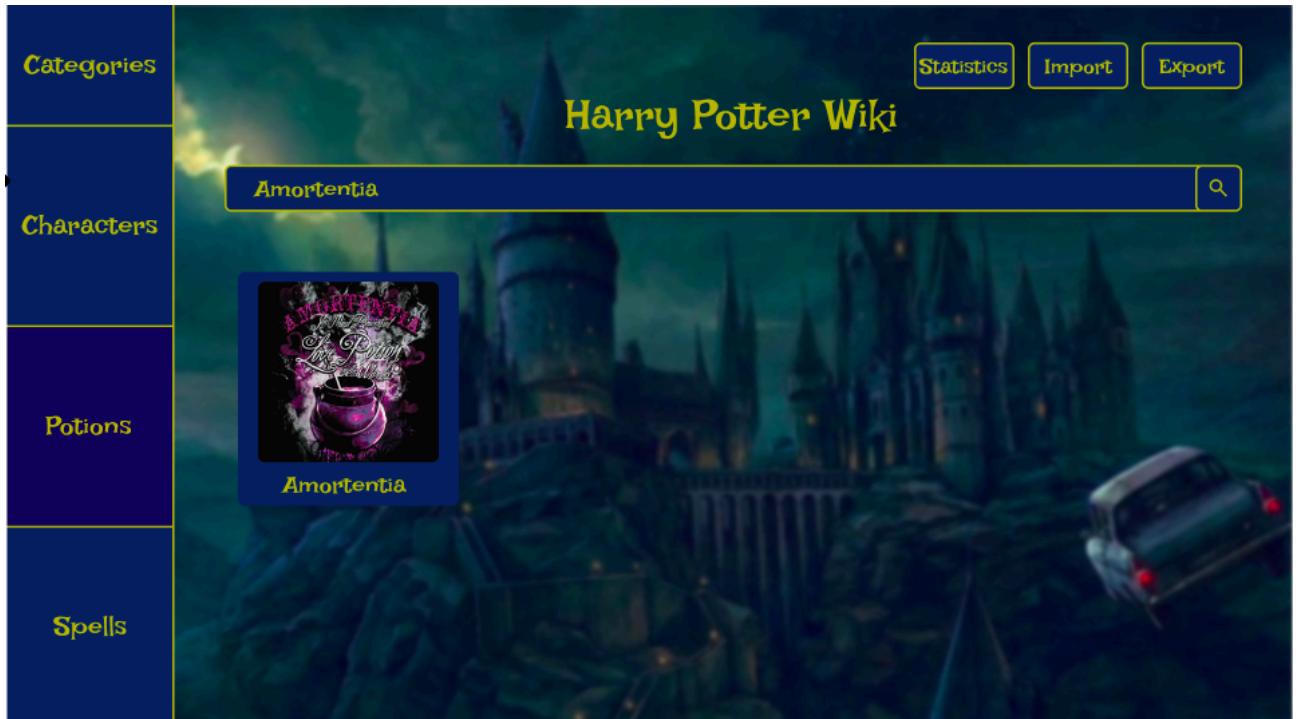


Рисунок 1.10 – Экран с результатом поиска зелий

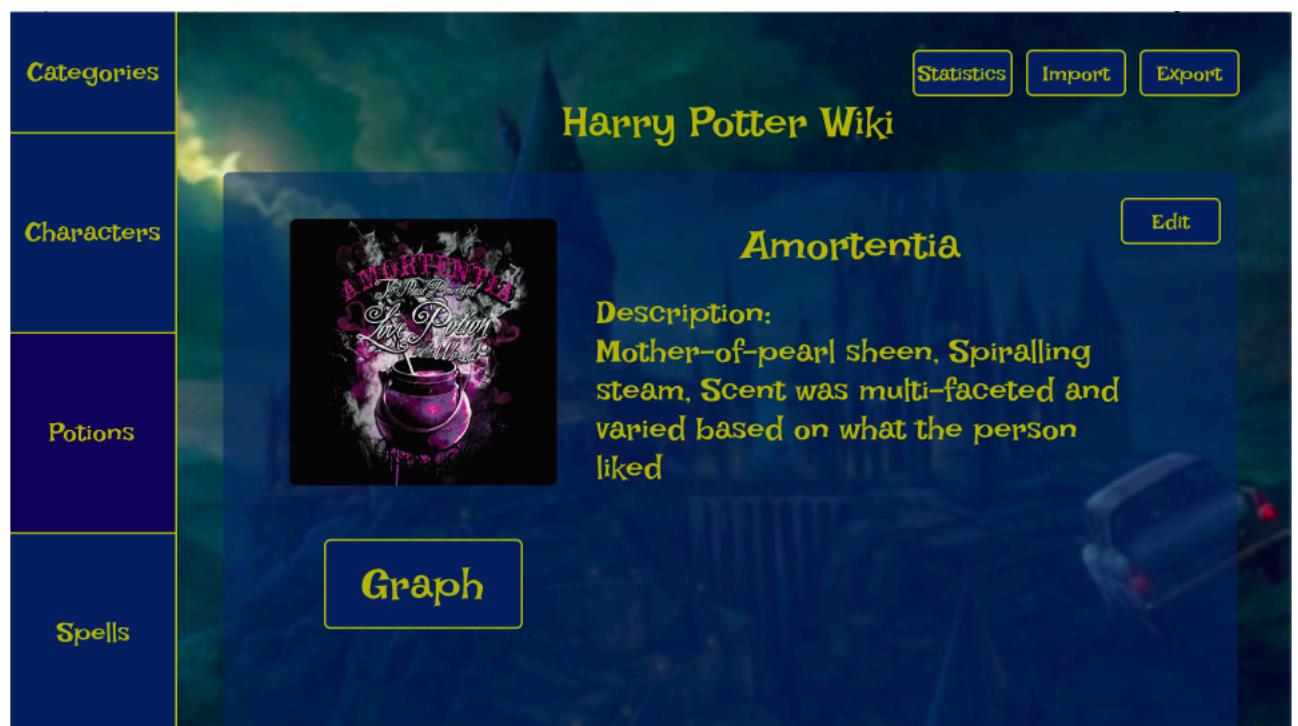


Рисунок 1.11 – Экран с подробной информацией о зелии



Рисунок 1.12 – Экран с графиком связей для выбранного зелья

The screenshot shows the Harry Potter Wiki edit screen for 'Polyjuice Potion'. The sidebar on the left lists categories: Character, Potions, and Spell. The main area contains fields for Name (Polyjuice Potion), Difficulty (Advanced), Effect (Transform into Someone else), Ingredients (a list of items including Ashwinder egg, Bezoar, Bone, Jobberknoll feathers, Lacewing flies, Mandrake root, Pearl dust, Shrivingel, Valerian root, Wolfsbane, belladonna, leeches, mistletoe, peppermint, powdered bloom horn, regrowth solution, rose thorns, and squill bulb), and Image URL (https://static.wikia.nocookie.net/harrypotter/images/1/1b/B2C12M2_Polyjuice_Potion_ready.jpg). There are also 'Statistics', 'Import', and 'Export' buttons at the top right.

Рисунок 1.13 – Экран редактирования данных конкретного зелья

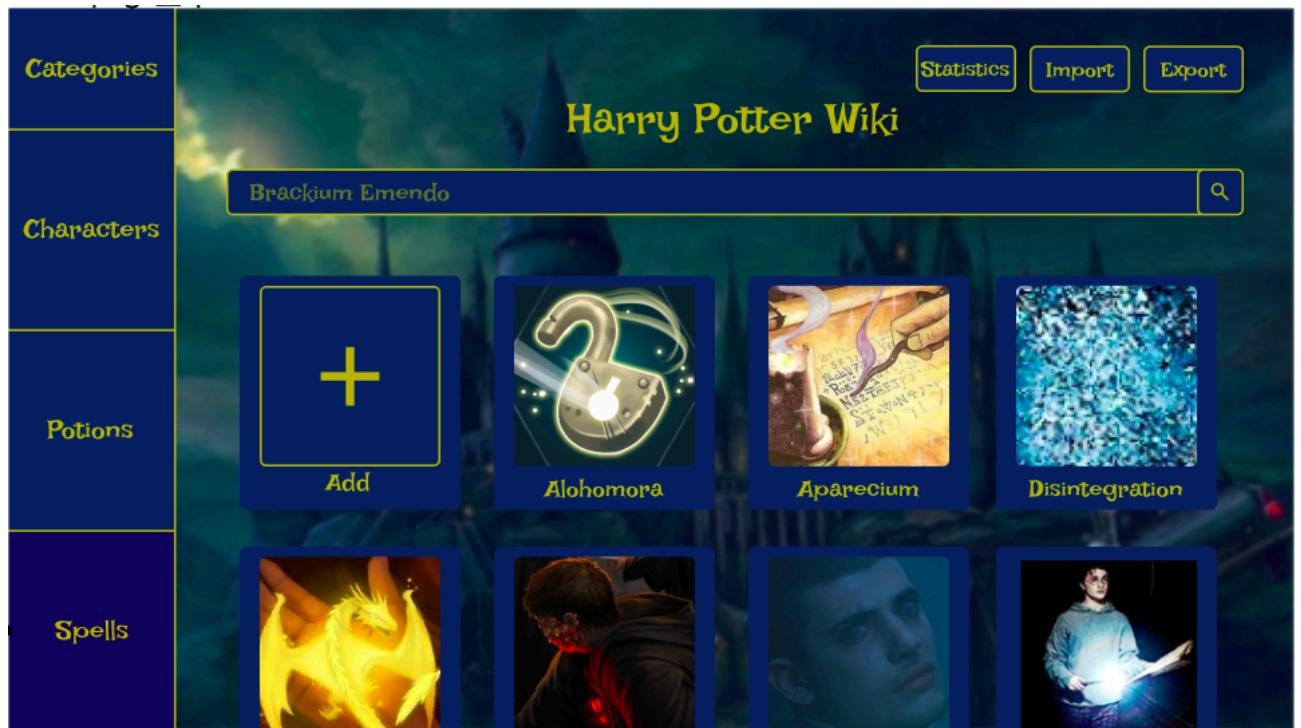


Рисунок 1.14 – Экран просмотра всех заклинаний

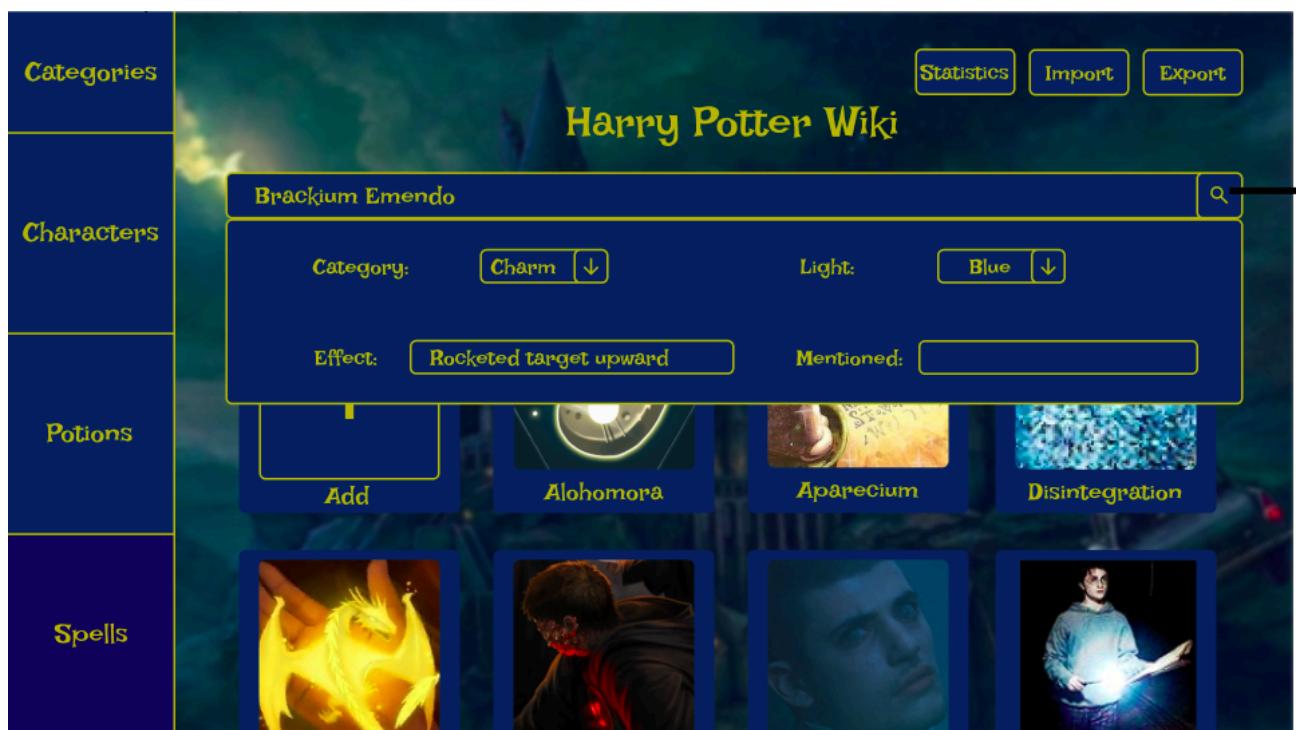


Рисунок 1.15 – Экран поиска заклинаний с фильтрами



Рисунок 1.16 – Экран с результатом поиска заклинания

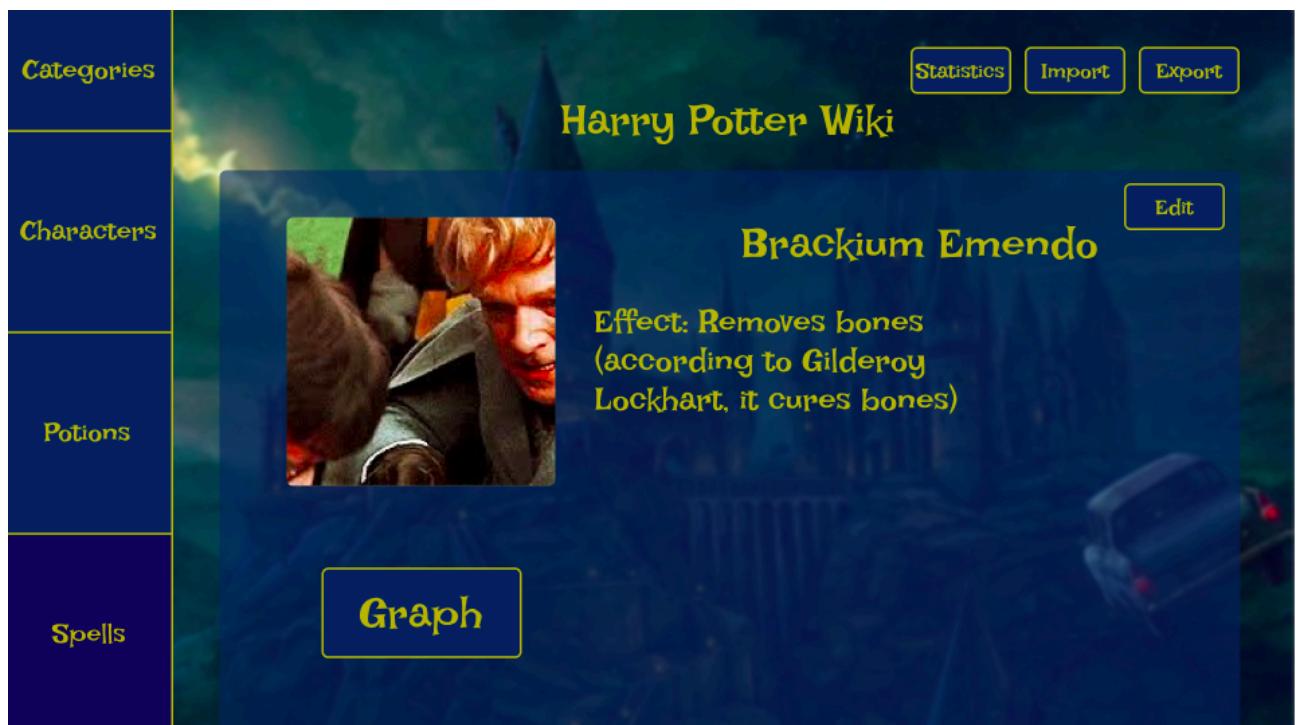


Рисунок 1.17 – Экран с подробной информацией о конкретном заклинании

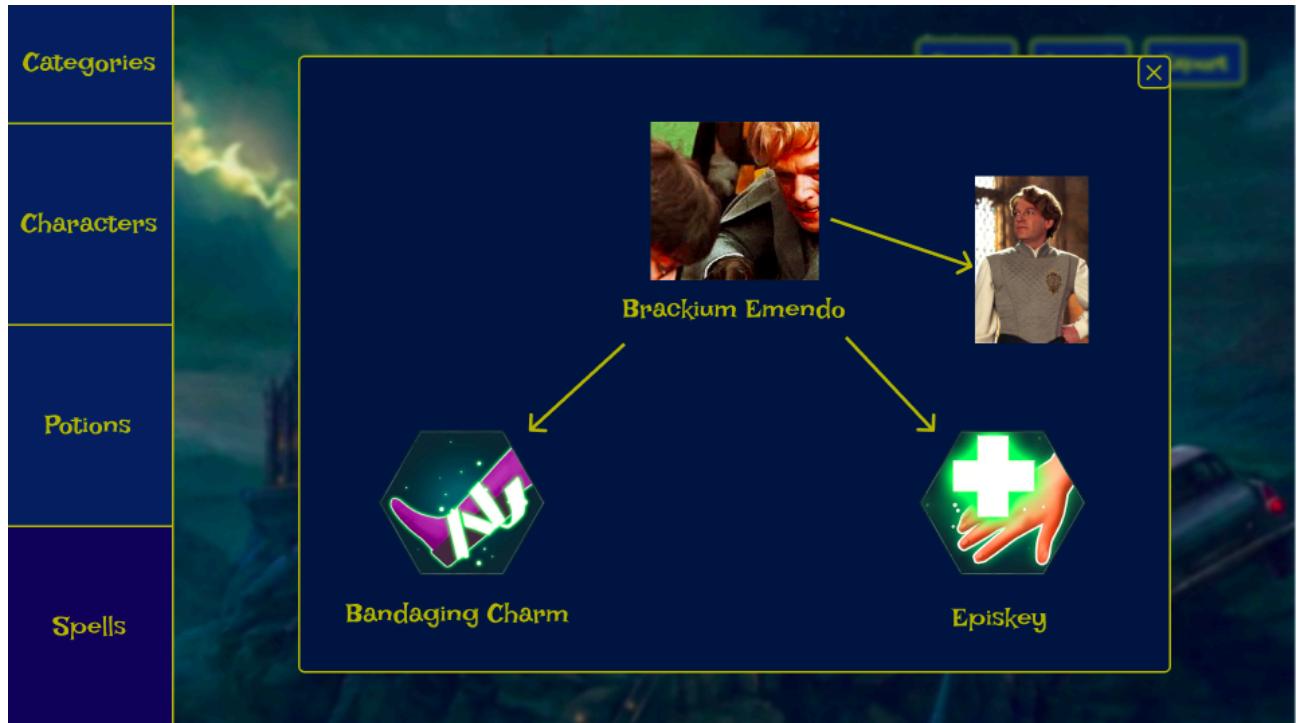


Рисунок 1.18 – Экран с графиком связей конкретного заклинания

The screenshot shows the Harry Potter Wiki edit screen for the 'Disarm' spell. On the left, there's a sidebar with 'Categories' (Character, Potions, Spell). The 'Character' category is selected. The main area contains form fields for editing the spell: 'Name:' (Expelliarmus), 'Effect:' (Disarms opponent), 'Light:' (Red), 'Category:' (Charm), and 'Image URL:' (https://static.wikia.nocookie.net/harrypotter/images/9/96/Snape__disarming__Lockhart__.COS.gif). At the top right, there are buttons for 'Statistics', 'Import', and 'Export'. The background features a dark, magical-themed image of Hogwarts castle.

Рисунок 1.19 – Экран редактирования данных конкретного заклинания

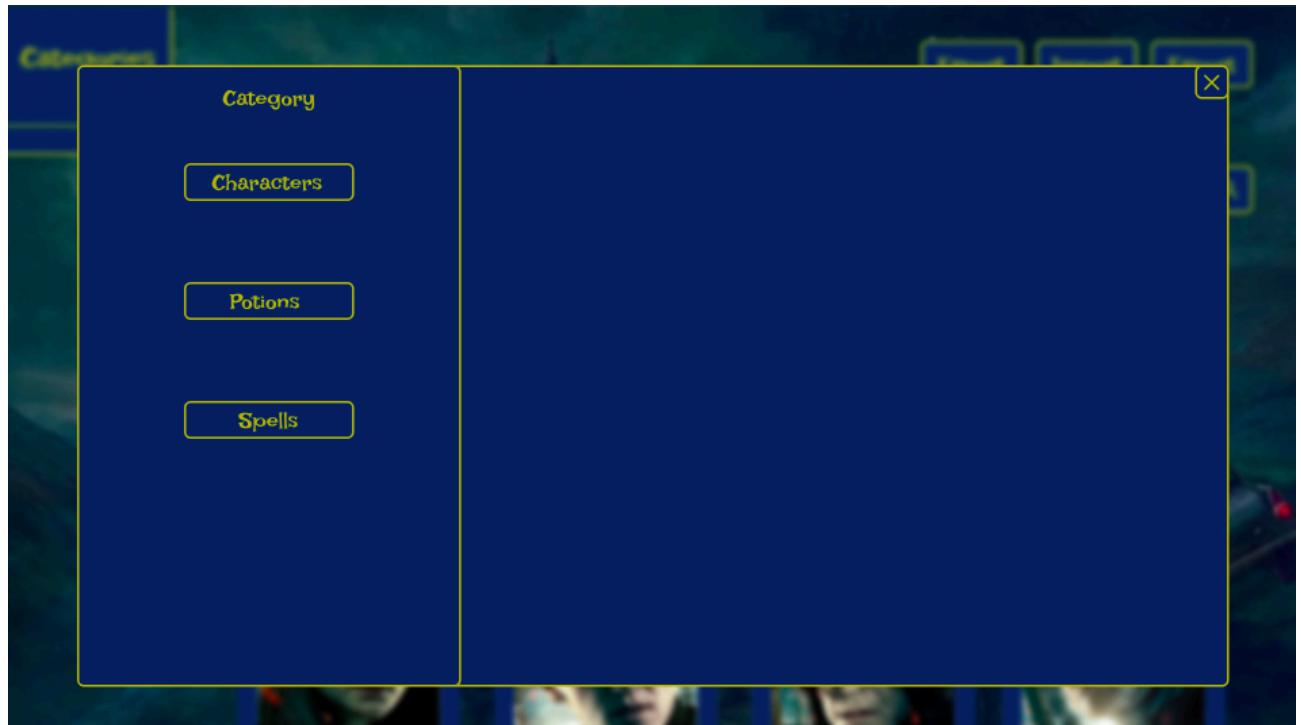


Рисунок 1.20 – Экран выбора категорий для статистики

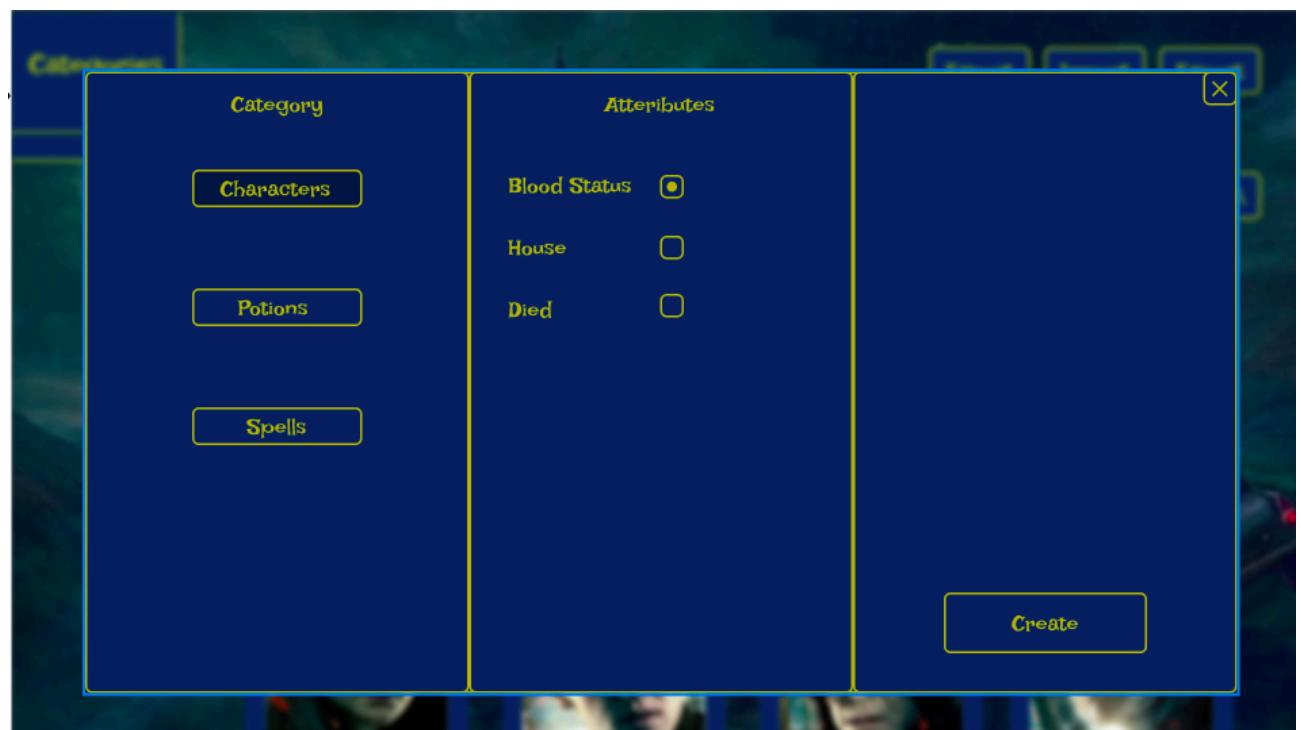


Рисунок 1.21 – Экран выбора атрибутов для конкретной категории

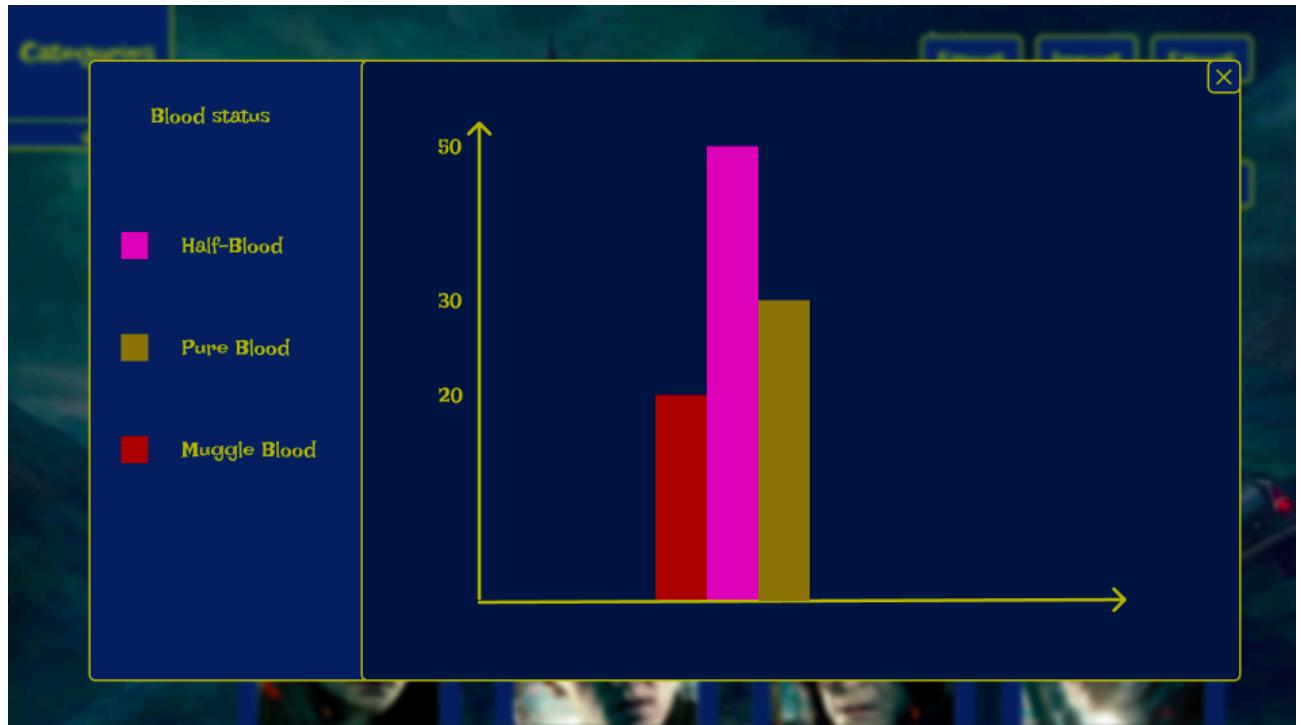


Рисунок 1.22 – Экран с отображением статистики по заданным параметрам

1.2. Сценарий использования

1. Категории

Описание: Пользователь на главной странице и видит приветствие.

Действующее лицо: Пользователь.

Предусловия: Пользователь зашел на сайт.

Основной сценарий:

1. Пользователь видит главную страницу.
2. Пользователь нажимает на стрелку категорий (Categories).
3. Разворачивается список категорий.
4. Видит основные категории (Characters, Poisons, Spells).
5. Может нажать на любую категорию, чтобы перейти к ее содержимому.

Альтернативный сценарий:

- Если сайт загружается медленно, пользователь ждет или обновляет страницу.

Результат: Пользователь может продолжить навигацию по сайту.

2. Поиск объекта

Описание: Пользователь вводит имя объекта или использует фильтры для поиска.

Действующее лицо: Пользователь.

Предусловия: Пользователь находится в категории (например, "Characters").

Основной сценарий:

1. Пользователь вводит имя объекта в строку поиска (например, "Harry Potter").
2. Либо выбирает фильтры (например, "Blood Status", "House", "Gender").
3. Нажимает кнопку поиска.
4. Отображается список объектов, соответствующих критериям.

Альтернативный сценарий:

- Если ничего не найдено, в списке объектов выводится строка "Nothing was found".
- При вводе названия объекта появляются подсказки с подходящими вариантами.
 - Пользователь может выбрать объект из подсказок и сразу перейти на его страницу описания.

Результат: Пользователь видит отфильтрованный список объектов.

3. Выбор объекта из списка

Описание: Пользователь выбирает конкретный объект из списка найденных.

Действующее лицо: Пользователь.

Предусловия: Пользователь видит список объектов (например, список персонажей после поиска).

Основной сценарий:

1. Пользователь нажимает на объект (например, "Harry Potter").
2. Загружается страница с описанием объекта.

Альтернативный сценарий:

- Если объект не загружается, выводится сообщение об ошибке.

Результат: Пользователь попадает на страницу описания объекта.

4. Просмотр описания объекта

Описание: Пользователь читает описание выбранного объекта.

Действующее лицо: Пользователь.

Предусловия: Пользователь выбрал объект из списка.

Основной сценарий:

1. Открывается страница с описанием объекта.
2. Пользователь читает текстовое описание, видит изображение объекта.
3. Может нажать на кнопку "Graph", чтобы увидеть связи объекта.

Альтернативный сценарий:

- Если описание не загружается, выводится сообщение об ошибке.

Результат: Пользователь ознакомился с информацией об объекте.

5. Использование графа связей

Описание: Пользователь изучает связи объекта с другими элементами.

Действующее лицо: Пользователь.

Предусловия: Пользователь находится на странице описания объекта.

Основной сценарий:

1. Пользователь нажимает кнопку "Graph".
2. Открывается визуализация связей объекта с другими элементами (например, связи персонажа с друзьями, учителями и врагами).

Альтернативный сценарий:

- Если график не загружается, выводится сообщение об ошибке.

Результат: Пользователь может изучить связи объекта и продолжить навигацию.

6. Изменение параметров графа связей

Описание: Пользователь может настроить параметры отображения графа связей, выбрав, какие типы связей будут показаны.

Действующее лицо: Пользователь.

Предусловия: Пользователь открыл график связей для объекта.

Основной сценарий:

1. Пользователь нажимает кнопку "Customize" в графике связей.
2. Открывается меню с опциями выбора типов связей (например, семейные связи, принадлежность к дому, свойства заклинаний, а также связи с другими категориями).
3. Пользователь выбирает нужные параметры.
4. Нажимает кнопку "Apply".
5. Граф обновляется с учетом новых параметров.

Альтернативный сценарий

- Если настройки не загружаются, выводится сообщение об ошибке.
- Если пользователь снимает все галочки и применяет настройки, выводится пустой график.

Результат: Пользователь получает обновленный график связей с учетом выбранных параметров.

7. Импорт JSON-файла с новыми данными для БД

Действующие лица: Пользователь, Система.

Цель: Импортировать JSON-файл с новыми данными для БД.

Предусловия: 1. Открыт сайт 2. Доступна кнопка "Import"

Основной поток:

1. Пользователь нажимает на кнопку “Import” на сайте.
2. Открывается диалоговое окно выбора файла.
3. Пользователь выбирает файл в формате .json и нажимает "Открыть".
4. Система проверяет формат и содержимое файла на соответствие требованиям.
5. Система загружает данные из файла в базу данных.
6. Система отображает сообщение об успешной загрузке данных.
7. Происходит обновление страницы, на основе новых данных.

Альтернативный поток:

- Если пользователь открыл файл не в формате .json, система выводит сообщение об ошибке и предлагает выбрать другой файл.
- Если данные в файле не соответствуют структуре БД, система отображает сообщение об ошибке.

Результат: На сайте отображена информация из новой БД.

8. Экспорт JSON-файла с данными текущей БД

Действующие лица: Пользователь, Система.

Цель: Экспортировать JSON-файл с данными текущей БД.

Предусловия:

1. Открыт сайт
2. Доступна кнопка “Import”

Основной поток:

1. Пользователь нажимает на кнопку “Export” на сайте.
2. Система формирует запрос к базе данных для получения данных.
3. Система экспортирует данные в JSON формат.
4. Система создает файл с данными и предоставляет его для скачивания.
5. Пользователь получает уведомление о готовности файла и возможность скачать его.
6. Файл загружается на устройство пользователя.

Альтернативный поток:

- Если не удалось сохранить файл на компьютер, система отображает сообщение об ошибке.

Результат: БД успешна загружена на компьютер пользователя.

9. Построение кастомизируемой статистики

Действующие лица: Пользователь, Система.

Цель: Построение кастомизируемой статистики по многокритериальной фильтрации данных.

Предусловия:

1. Открыт сайт
2. Доступна кнопка “Statistics”

Основной поток:

1. Пользователь нажимает на кнопку “Statistics” на сайте.
2. Открывается окно, в котором пользователь выбирает категории и параметры для многокритериальной фильтрации данных.
3. Система формирует запрос к базе данных для получения данных.
4. Система отображает диаграмму, заданную пользователем.

Альтернативный поток:

- Если не удалось отобразить статистику, система отображает сообщение об ошибке.

Результат: Пользователь видит статистику,строенную на заданных критериях фильтрации.

1.3. Преобладающая операция

В предлагаемом решении преобладающими будут операции чтения:

- 1) Навигация по категориям, поиск объектов, просмотр описаний и графов связей — все эти действия связаны с извлечением данных из базы без их изменения.
- 2) Просмотр графов и кастомизация их отображения также основаны на получении уже существующих связей и объектов, то есть являются операциями чтения.
- 3) Построение статистики реализуется через многокритериальные запросы к базе, что также представляет собой чтение.

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель

A. Графическое представление

Модель данных представляет собой ориентированный граф, где узлы соответствуют сущностям вселенной Гарри Поттера, а рёбра - отношениям между ними. Графическое представление отображено на рисунке 2.1.1.

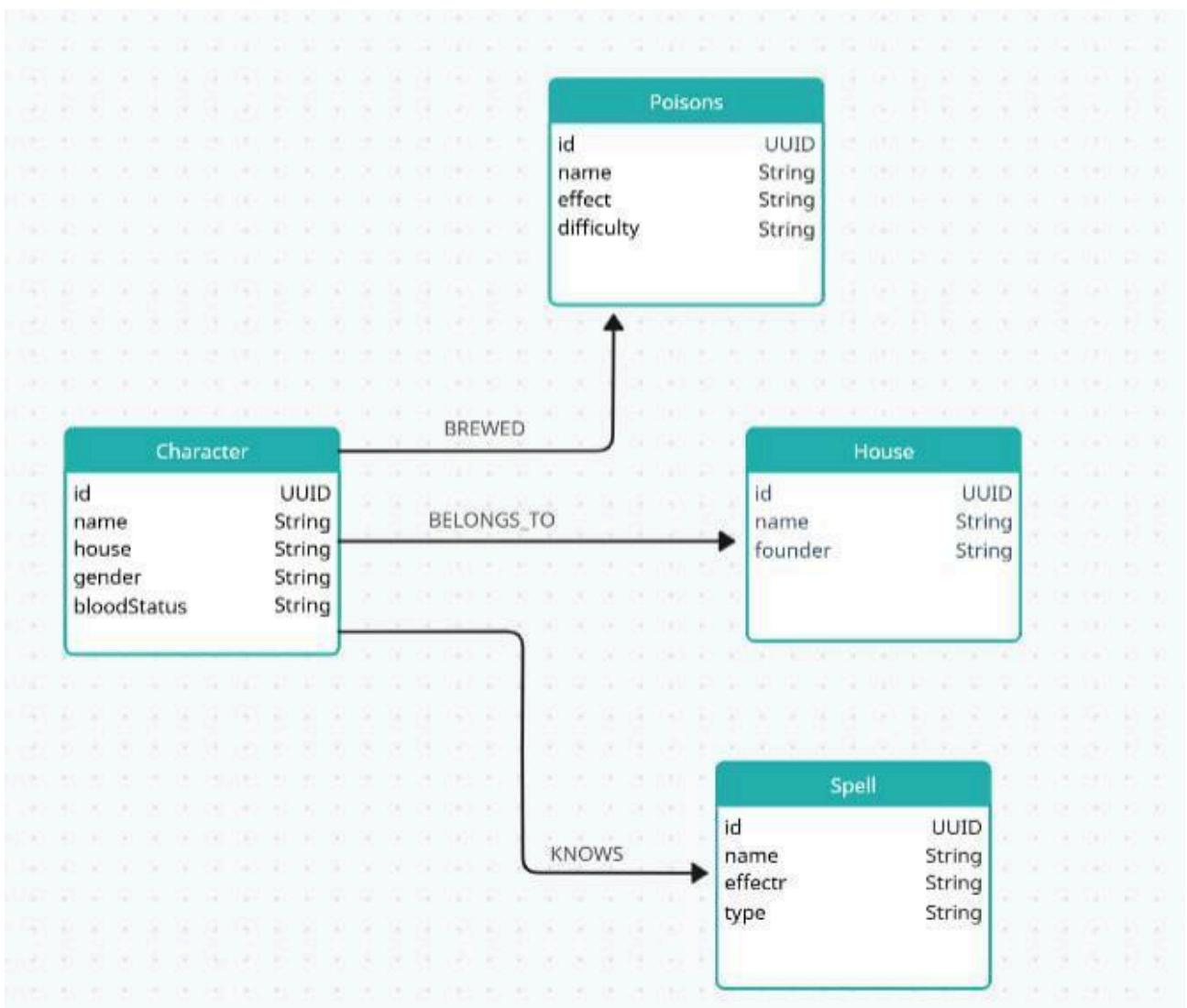


Рисунок 2.1.1 – Графическое представление

Граф содержит следующие типы узлов и связей:

```
[Character] -[:BELONGS_TO]-> [House]  
[Character] -[:KNOWS]-> [Spell]
```

```
[Character] - [:BREWED] -> [Poison]
[Character] - [:HAS_RELATIONSHIP_WITH {type: 'Friend'}] -> [Character]
[Character] - [:HAS_RELATIONSHIP_WITH {type: 'Enemy'}] -> [Character]
[Character] - [:HAS_RELATIONSHIP_WITH {type: 'Teacher'}] -> [Character]
[Character] - [:HAS_RELATIONSHIP_WITH {type: 'Family'}] -> [Character]
```

B. Описание модели

1. Коллекции и их назначения:

- **characters** - содержит всех персонажей с их атрибутами (имя, факультет, статус крови и т.д.)
- **poisons** - хранит информацию о магических зельях и их свойствах
- **spells** - содержит заклинания с описанием их эффектов
- **houses** - факультеты Хогвартса

2. Типы данных и их размеры

Типы данных с размерами отражены в таблице 1.

Таблица 1 – Типы данных

Сущность	Поле	Тип данных	Размер (байт)	Описание
Character	id	UUID	16	Уникальный идентификатор
	name	String	50	Имя персонажа
	house	String	50	Название факультета
	bloodStatus	String	20	Чистота крови
	gender	String	10	Пол
	description	Text	500	Описание
Poison	id	UUID	16	Уникальный идентификатор
	name	String	50	Название зелья
	effect	Text	500	Эффект зелья
	difficulty	String	20	Сложность приготовления
Spell	id	UUID	16	Уникальный идентификатор
	name	String	50	Название заклинания
	effect	Text	500	Эффект заклинания
	type	String	50	Тип заклинания
House	id	UUID	16	Уникальный идентификатор
	name	String	50	Название факультета

	founder	String	100	Основатель факультета
--	---------	--------	-----	-----------------------

3. Сущности и их назначения:

- **Character** - представляет персонажа вселенной Гарри Поттера. Содержит биографические данные и связи с другими сущностями. Пример: Гарри Поттер, Гермиона Грейндженер, Северус Снейп.
- **Poison** - магическое зелье с описанием его свойств. Пример: "Оборотное зелье", "Зелье живого смерти".
- **Spell** - заклинание с описанием его эффекта. Пример: "Экспеллиармус", "Левиоса".
- **House** - факультет Хогвартса. Пример: "Гриффиндор", "Слизерин".

C. Оценка объема информации

Исходные данные для расчета отражены в таблице 2.

Таблица 2 – Исходные данные для расчета

Сущность	Количество	Среднее количество связей
Character (C)	50	-
Poison (P)	30	-
Spell (S)	100	-
House (H)	4	-
BELONGS_TO	-	1 на персонажа
KNOWS	-	10 на персонажа
BREWED	-	0,4 на персонажа
HAS_RELATIONSHIP_WITH	-	5 на персонажа

Расчет объема:

- Размер узлов:
 - *Character*: $50 \times (16 + 50 + 50 + 20 + 10 + 500) = 50 \times 646 = 32.300$ байт
 - *Poison*: $30 \times (16 + 50 + 500 + 20) = 30 \times 586 = 17.580$ байт
 - *Spell*: $100 \times (16 + 50 + 500 + 50) = 100 \times 616 = 61.600$ байт
 - *House*: $4 \times (16 + 50 + 100) = 4 \times 166 = 664$ байт
- Размер связей (каждая связь: $16 \times 2 + 20 = 52$ байта):

- *BELONGS_TO*: $50 \times 52 = 2.600$ байт
- *KNOWS*: $50 \times 10 \times 52 = 26.000$ байт
- *BREWED*: $20 \times 52 = 1.040$ байт
- *HAS_RELATIONSHIP_WITH*: $50 \times 5 \times 52 = 13.000$ байт

Общая формула объема:

Упрощенная формула через количество персонажей (C):

$$V(C) = (646 + 52 + 520 + 20,8 + 260) \times C + 17.580 + 61.600 + 664$$

$$V(C) = 1.498,8 \times C + 79.844 \text{ байт}$$

Для C=50:

$$V(50) = 1498,8 \times 50 + 79.844 = 154.784 \text{ байт}$$

D. Избыточность данных

- Чистый объем данных (только полезная информация):

$$V_{clean}(C) = (50+50+20+10+500) \times C + (50+500+20) \times 30 + (50+500+50) \times 100 + (50+100) \times 4 =$$

$$= 630 \times C + 17.100 + 60.000 + 600 = 630 \times C + 77.700 \text{ байт}$$

- Фактический объем:

$$V(C) = 1.498,8 \times C + 79.844 \text{ байт}$$

- Формула избыточности:

$$E(C) = V(C)/V_{clean}(C) = (1.498,8 \times C + 79.844)/(630 \times C + 77.700) \xrightarrow[C \rightarrow \infty]{} 2,38$$

Для C=50:

$$E(50) = (74.940 + 79.844)/(31.500 + 77.700) \approx 1,42$$

(42% избыточности)

E. Направление роста модели

Полная формула зависимости объема от количества сущностей:

$$V(C, P, S, H, R) = 1.498,8 \times C + 586 \times P + 616 \times S + 166 \times H + 79.844$$

байт

Анализ влияния параметров:

1. Количество персонажей (C):

- Наибольшее влияние (коэффициент 1.498,8)
- Каждый новый персонаж добавляет ~1,5 КБ данных
- Обусловлено множественными связями персонажа

2. Количество заклинаний (S):

- Значительное влияние (коэффициент 616)
- Каждое новое заклинание добавляет ~0,6 КБ

3. Количество зелий (P):

- Среднее влияние (коэффициент 586)
- Каждое новое зелье добавляет ~0,6 КБ

4. Факультеты (H):

- Минимальное влияние
- Добавление нового факультета - ~166 байт

F. Примеры данных

Пример данных изображен на рисунке 2.1.2.

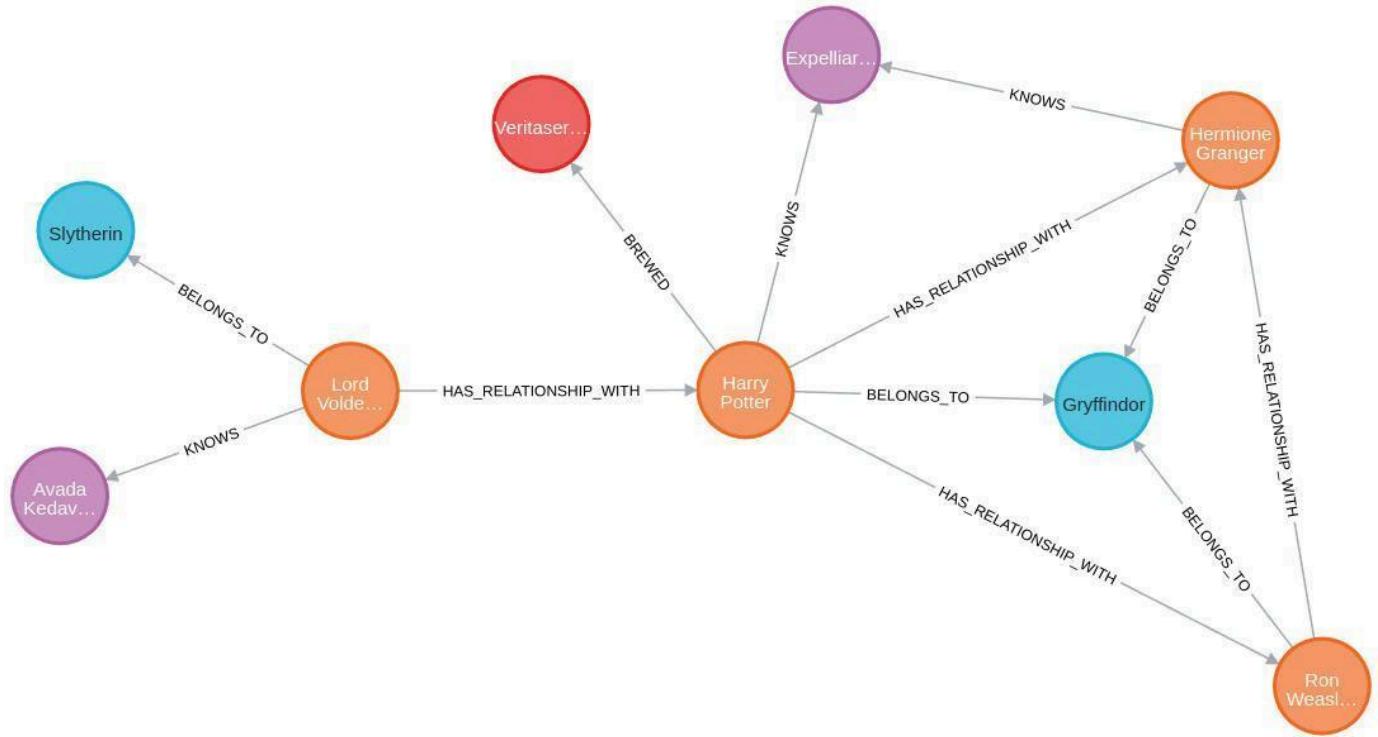


Рисунок 2.1.2 – Пример данных

G. Примеры запросов

1. Поиск всех друзей персонажа:

```

MATCH (c:Character{name:"Имя_персонажа"}) -[:HAS_RELATIONSHIP_WITH
{type:'Friend'}]->(friend:Character)
RETURN friend.name AS friend_name, friend.house AS house
    
```

- Количество запросов:
 - 1 запрос для получения всех друзей для 1 персонажа
- Сложность:
 - $O(k)$, где k — количество друзей

Зависит от числа отношений типа *Friend* у персонажа
- Задействованные коллекции:
 - Узлы:

Character (исходный + друзья)
 - Связи:

HAS_RELATIONSHIP_WITH

2. Поиск всех зелий, созданных персонажем:

```

MATCH (c:Character {name: "Имя_персонажа"}) -[:BREWED]-> (p:Poison)
RETURN p.name AS poison_name, p.effect AS effect, p.difficulty AS difficulty
ORDER BY p.name

```

- Количество запросов:
 - 1 запрос для получения всех зелий для 1 персонажа
- Сложность:
 $O(m)$, где m — количество зелий, созданных персонажем.

- Задействованные коллекции:

- Узлы:

Character

Poison.

- Связи:

BREWED.

3. Поиск всех заклинаний определенного типа:

```

MATCH (c:Character {name: "Имя_персонажа"}) -[:KNOWS]-> (s:Spell {type:
"Тип_заклинания"})
RETURN s.name AS spell_name, s.effect AS effect

```

- Количество запросов:
 - 1 запрос, но может быть неэффективным без индексов.
- Сложность:
 $O(n * t)$, где: n — количество заклинаний, которые знает персонаж, t — количество заклинаний с указанным типом.

- Задействованные коллекции:

- Узлы:

Character,

Spell.

- Связи:

KNOWS.

4. Все противоречия фразе "враг моего врага - мой друг":

```

MATCH (a:Character)-[r1:HAS_RELATIONSHIP_WITH {type: 'Friend'}]-(b:Character), (b)-[r2:HAS_RELATIONSHIP_WITH {type: 'Enemy'}]-(c:Character),
(a)-[r3:HAS_RELATIONSHIP_WITH {type: 'Friend'}]-(c)
RETURN a.name AS person1, b.name AS person2, c.name AS person3

```

- Количество запросов:

- Сложность:

$O(n^3)$ - поиск отношений двух уровней на каждом персонаже.

- Задействованные коллекции:

- Узлы:

Character.

- Связи:

HAS_RELATIONSHIP_WITH {type: 'Friend'},

HAS_RELATIONSHIP_WITH {type: 'Enemy'}

5. Факультеты, с чьими учениками меньше всего конфликтов:

```

MATCH
(h:House)<-[:BELONGS_TO]-(c:Character)-[r:HAS_RELATIONSHIP_WITH
{type: 'Enemy'}]-(other:Character)
WITH h, COUNT(r) AS conflict_count
ORDER BY conflict_count ASC
LIMIT 1
RETURN h.name AS house_name, conflict_count

```

- Количество запросов:

- Сложность:

$O(p + q)$, где p – количество персонажей, q – количество вражеских отношений,

$O(q \log q)$ для сортировки.

- Задействованные коллекции:

- Узлы:

House,

Character.

- Связи:

BELONGS_TO,

HAS_RELATIONSHIP_WITH {type: 'Enemy'}.

6. "Уникальные" заклинания по факультетам:

```
MATCH (h:House)-[:BELONGS_TO]-(c:Character)-[:KNOWS]->(s:Spell)
WITH h, s, COUNT(c) AS house_count
OPTIONAL MATCH (other:Character)-[:KNOWS]->(s)
WHERE NOT (other)-[:BELONGS_TO]->(h)
WITH h, s, house_count, COUNT(other) AS other_count
WHERE other_count = 0 OR house_count > other_count
RETURN h.name AS house, s.name AS spell,
CASE WHEN other_count = 0 THEN 'Уникальное' ELSE 'Чаще всего
используется' END AS type
ORDER BY house, type, spell
```

- Количество запросов:

- Сложность:

$O(s \times h \times c)$, где s - заклинания, h - факультеты, c -- персонажи:
перебор всех заклинаний, для каждого заклинания перебор всех
факультетов и знающих это заклинание студентов этого факультета
и незнающих другого.

- Задействованные коллекции:

- Узлы:

House,

Spell,

Character.

- Связи:

KNOWS,

BELONGS_TO,

HAS_RELATIONSHIP_WITH {type: 'Enemy'}.

2.2. Реляционная модель

A. Графическое представление

Реляционная модель изображена на рисунке 2.2, она состоит из 5 основных таблиц и 3 таблиц связей многие-ко-многим:

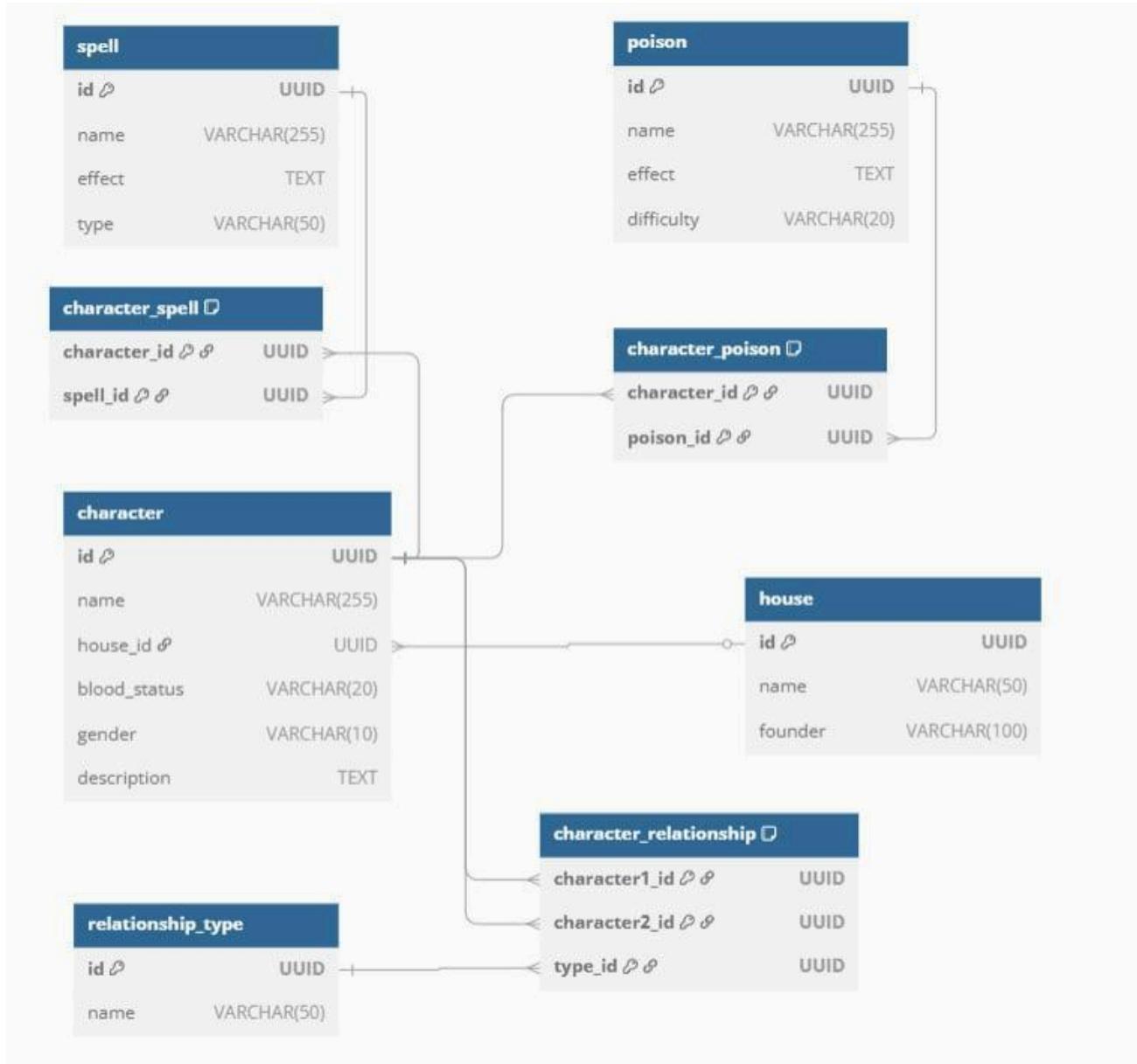


Рисунок 2.2 – Реляционная модель

B. Описание модели

1. Таблицы и их назначения:

- **character** - основная таблица персонажей
- **poison** - таблица магических зелий
- **spell** - таблица заклинаний
- **house** - таблица факультетов Хогвартса
- **relationship_type** - справочник типов отношений
- **character_spell** - таблица связей персонажей и заклинаний

- **character_poison** - таблица связей персонажей и зелий
- **character_relationship** - таблица отношений между персонажами

2. Типы данных и их размеры

Типы данных и их размеры отражены в таблице 3.

Таблица 3 – Типы данных и их размеры

Таблица	Столбец	Тип данных	Размер (байт)	Описание
character	id	UUID	16	Первичный ключ
	name	VARCHAR(255)	255	Имя персонажа
	house_id	UUID	16	Внешний ключ
	blood_status	VARCHAR(20)	20	Статус крови
	gender	VARCHAR(10)	10	Пол
	description	TEXT	500	Описание
poison	id	UUID	16	Первичный ключ
	name	VARCHAR(255)	255	Название зелья
	effect	TEXT	500	Эффект
	difficulty	VARCHAR(20)	20	Сложность
spell	id	UUID	16	Первичный ключ
	name	VARCHAR(255)	255	Название заклинания
	effect	TEXT	500	Эффект
	type	VARCHAR(50)	50	Тип заклинания
house	id	UUID	16	Первичный ключ
	name	VARCHAR(50)	50	Название факультета
	founder	VARCHAR(100)	100	Основатель
relationship_type	id	UUID	16	Первичный ключ
	name	VARCHAR(50)	50	Тип отношения
character_spell	character_id	UUID	16	Внешний ключ
	spell_id	UUID	16	Внешний ключ
character_poison	character_id	UUID	16	Внешний ключ
	poison_id	UUID	16	Внешний ключ
character_relationship	character1_id	UUID	16	Внешний ключ
	character2_id	UUID	16	Внешний ключ
	type_id	UUID	16	Внешний ключ

С. Оценка объема информации

Исходные данные (аналогичны NoSQL модели) отражены в таблице

4.

Таблица 4 – Исходные данные

Сущность	Количество
Character (C)	50
Poison (P)	30
Spell (S)	100
House (H)	4

RelationshipType 5

- Размер строк:
 - *character*: $16 + 255 + 16 + 20 + 10 + 500 = 817$ байт
 - *poison*: $16 + 255 + 500 + 20 = 791$ байт
 - *spell*: $16 + 255 + 500 + 50 = 821$ байт
 - *house*: $16 + 50 + 100 = 166$ байт
 - *relationship_type*: $16 + 50 = 66$ байт
 - *character_spell*: $16 + 16 = 32$ байт
 - *character_poison*: $16 + 16 = 32$ байт
 - *character_relationship*: $16 + 16 + 16 = 48$ байт
- Общий объем:
 - *character*: $50 \times 817 = 40.850$ байт
 - *poison*: $30 \times 791 = 23.730$ байт
 - *spell*: $100 \times 821 = 82.100$ байт
 - *house*: $4 \times 166 = 664$ байт
 - *relationship_type*: $5 \times 66 = 330$ байт
 - *character_spell*: $50 \times 10 \times 32 = 16.000$ байт
 - *character_poison*: $20 \times 32 = 640$ байт
 - *character_relationship*: $50 \times 5 \times 48 = 12.000$ байт

Общая формула объема:

Упрощенная формула через количество персонажей (C):

$$V(C) = (817 + 320 + 12,8 + 240) \times C + 23.730 + 82.100 + 664 + 330$$

$$V(C) = 1.389,8 \times C + 106.824 \text{ байт}$$

Для C=50: $V(50) = 176.314 \approx 172,2 \text{ КБ}$

D. Избыточность данных

Чистый объем данных (только полезная информация):

$$\begin{aligned}V_{clean}(C) &= (255+50+20+10+500) \times C + (255+500+20) \times 30 + (255+500+50) \times \\&\times 100 + (50 + 100) \times 4 + 50 \times 5 = \\&= 835 \times C + 23.250 + 80.500 + 600 + 250 = 835 \times C + 104.600\end{aligned}$$

байт

Фактический объем:

$$V(C) = 1.389,8 \times C + 106.824 \text{ байт}$$

Формула избыточности:

$$E(C) = V(C)/V_{clean}(C) = (1.389,8 \times C + 106.824)/(835 \times C + 104.600) \xrightarrow{C \rightarrow \infty} 1,66$$

Для C=50: $E(50) = (69.490 + 106.824)/(41.750 + 104.600) \approx 1,21$
(21% избыточности)

E. Направление роста модели

Полная формула зависимости объема от количества сущностей:

$$\begin{aligned}V(C,P,S,H,R) &= 1.389,8 \times C + 791 \times P + 821 \times S + 166 \times H + 66 \times R + 106.824 \\&\text{байт}\end{aligned}$$

Анализ влияния параметров:

1. Количество персонажей (C):

- Основной фактор роста (коэффициент 1.389,8)
- Каждый новый персонаж добавляет ~1,4 КБ данных

2. Количество заклинаний (S):

- Значительное влияние (коэффициент 821)
- Каждое новое заклинание добавляет ~0,8 КБ

3. Количество зелий (P):

- Среднее влияние (коэффициент 791)
- Каждое новое зелье добавляет ~0,8 КБ

4. Факультеты (H) и типы связей (R):

- Минимальное влияние
- Новый факультет - ~166 байт
- Новый тип связи - ~66 байт

F. Примеры данных

1. Таблица house

id	name	founder
550e8400-...-446655440000	Gryffindor	Godric Gryffindor
550e8400-...-446655440001	Slytherin	Salazar Slytherin

2. Таблица character

id	name	house_id	blood_status	gender	description
770e8400-...-446655440000	Harry Potter	550e8400-...-446655440000	Half-blood	Male	The Boy Who Lived
770e8400-...-446655440001	Hermione Granger	550e8400-...-446655440001	Muggle-born	Female	Brightest witch of her age

3. Таблица spell

id	name	effect	type
880e8400-...-446655440000	Expelliarmus	Disarms opponent	Charm
880e8400-...-446655440002	Avada Kedavra	Kills instantly	Unforgivable Curse

4. Таблица poison

id	name	effect	difficulty
990e8400-...-446655440000	Veritaserum	Forces truth-telling	Very Hard
990e8400-...-446655440001	Polyjuice Potion	Transforms drinker	Extremely Difficult

5. Таблица relationship_type

id	name
660e8400-...-446655440000	Friend
660e8400-...-446655440001	Enemy

6. Таблица character_spell

character_id	spell_id
770e8400-...-446655440000	880e8400-...-446655440000
770e8400-...-446655440001	880e8400-...-446655440003

7. Таблица character_poison

character_id	poison_id
770e8400-...-446655440001	990e8400-...-446655440001
770e8400-...-446655440004	990e8400-...-446655440000

8. Таблица character_relationship

character1_id	character2_id	type_id
770e8400-...-446655440000	770e8400-...-446655440001	660e8400-...-446655440000
770e8400-...-446655440000	770e8400-...-446655440003	660e8400-...-446655440001

G. Примеры запросов

1. Получение статистики по факультетам:

```

SELECT
    h.name AS house_name,
    COUNT(c.id) AS members_count,
    AVG(LENGTH(c.description)) AS avg_description_length
FROM
    house h LEFT JOIN character c ON h.id = c.house_id
GROUP BY h.id, h.name
ORDER BY members_count DESC;

```

- Количество запросов:

- 1 запрос с использованием агрегатных функций (*COUNT*, *AVG*) и группировки.
- Сложность:

$O(N + M)$, где: N — количество факультетов (*house*), M — количество персонажей (*character*).

Группировка (*GROUP BY*) добавляет $O(M \log M)$ при больших данных.

- Задействованные таблицы:

- Основные:

house, character

- Связи:

house.id = character.house_id (JOIN)

2. Поиск 10-ти самых популярных заклинаний:

```

SELECT s.name AS spell_name, s.type AS spell_type,
COUNT(cs.character_id) AS usage_count
FROM spell s JOIN character_spell cs ON s.id = cs.spell_id
GROUP BY s.id, s.name, s.type
ORDER BY usage_count DESC
LIMIT 10;

```

- Количество запросов:

- 1 запрос с подсчетом через *COUNT* и сортировкой.

- Сложность:

$O(K + L)$, где: K — количество заклинаний (*spell*), L — количество записей в *character_spell*. Сортировка (*ORDER BY*) с *LIMIT* требует $O(L \log L)$.

- Задействованные таблицы:

- Основные:

spell, character_spell

- Связи:

spell.id = character_spell.spell_id (JOIN)

3. Поиск сложных зелий, созданных персонажами:

```
SELECT
    c.name AS character_name,
    p.name AS poison_name,
    p.difficulty AS difficulty
FROM character c
    JOIN character_poison cp ON c.id = cp.character_id
    JOIN poison p ON cp.poison_id = p.id
WHERE p.difficulty IN ('сложное', 'очень сложное')
ORDER BY p.difficulty DESC, c.name;
```

- Количество запросов:

- 1 запрос с двумя *JOIN* и фильтрацией.

- Сложность:

$O(P + Q + R)$, где: P — количество персонажей (*character*), Q — количество зелий (*poison*), R — количество записей в *character_poison*.

- Задействованные таблицы:

- Основные:

character, poison, character_poison

- Связи:

character.id = character_poison.character_id

poison.id = character_poison.poison_id

4. Все противоречия фразе "враг моего врага - мой друг":

```
WITH friend_pairs AS (
    SELECT cr.character1_id AS a_id, cr.character2_id AS b_id
```

```

        FROM character_relationship cr
        JOIN relationship_type rt ON cr.type_id = rt.id
        WHERE rt.name = 'Friend'
),
enemy_pairs AS (
    SELECT cr.character1_id AS b_id, cr.character2_id AS c_id
    FROM character_relationship cr
    JOIN relationship_type rt ON cr.type_id = rt.id
    WHERE rt.name = 'Enemy'
)
SELECT
    c1.name AS person1,
    c2.name AS person2,
    c3.name AS person3,
    'Друзья, хотя должны быть врагами' AS conflict_type
FROM friend_pairs fp1
JOIN enemy_pairs ep ON fp1.b_id = ep.b_id
JOIN friend_pairs fp2 ON fp1.a_id = fp2.a_id AND ep.c_id =
fp2.b_id
JOIN character c1 ON fp1.a_id = c1.id
JOIN character c2 ON fp1.b_id = c2.id
JOIN character c3 ON ep.c_id = c3.id;

```

- Количество запросов:

- Сложность:

$O(f * e * f) \sim O(n^3)$ в худшем случае.

- Задействованные таблицы:

- *character_relationship, relationship_type, character*

5. Факультеты, с чьими учениками меньше всего конфликтов:

```

SELECT h.name, COUNT(*) AS conflicts
FROM character_relationship cr
JOIN relationship_type rt ON cr.type_id = rt.id AND rt.name =
'Enemy'
JOIN character c ON cr.character1_id = c.id
JOIN house h ON c.house_id = h.id
GROUP BY h.id, h.name
ORDER BY conflicts ASC
LIMIT 1

```

- Количество запросов:

- Сложность:

$O(e) + O(h \log h)$ для сортировки.

- Задействованные таблицы:

 - *character_relationship, relationship_type, character, house*

6. "Уникальные" заклинания по факультетам

```

WITH house_spells AS (
    SELECT h.id AS house_id, h.name AS house_name,
           s.id AS spell_id, s.name AS spell_name,
           COUNT(DISTINCT cs.character_id) AS house_count
      FROM house h
     JOIN character c ON h.id = c.house_id
     JOIN character_spell cs ON c.id = cs.character_id
     JOIN spell s ON cs.spell_id = s.id
    GROUP BY h.id, h.name, s.id, s.name
),
other_usage AS (
    SELECT hs.house_id, hs.spell_id,
           COUNT(DISTINCT cs.character_id) AS other_count
      FROM house_spells hs
     JOIN character_spell cs ON hs.spell_id = cs.spell_id
     JOIN character c ON cs.character_id = c.id AND c.house_id != hs.house_id
    GROUP BY hs.house_id, hs.spell_id
)
SELECT hs.house_name, hs.spell_name,
       CASE WHEN ou.other_count = 0 THEN 'Уникальное'
             WHEN hs.house_count > ou.other_count THEN
                 'Доминирующее'
             ELSE 'Обычное' END AS spell_type
  FROM house_spells hs
 LEFT JOIN other_usage ou ON hs.house_id = ou.house_id AND
 hs.spell_id = ou.spell_id
 WHERE ou.other_count = 0 OR hs.house_count > ou.other_count

```

- Количество запросов:

 - Сложность:

$$O(h \times s \times c).$$

- Задействованные таблицы:

 - *house, character, character_spell, spell*

2.3. Сравнение моделей

A. Удельный объем информации

Удельный объем информации для двух моделей отражен в таблице 5.

Таблица 5 – Удельный объем информации

Параметр	NoSQL (Neo4j)	SQL (Реляционная)
Формула объема	$V_1(C) = 1.498,8 \times C + 80.174$	$V_2(C) = 1.389,8 \times C + 106.824$
Объем при C=50	151,5 КБ	172,2 КБ
Избыточность	1,42 (42%)	1,1 (21%)
Зависимость от C	Линейная (1.498,8)	Линейная (1.389,8)

Анализ:

- При малом количестве персонажей ($C < 75$) Neo4j требует меньше памяти
- При $C > 75$ реляционная модель становится более экономной
- Neo4j имеет более высокую избыточность из-за хранения метаданных о связях
- SQL модель более эффективна для хранения "плоских" данных

B. Производительность запросов

1. Запросы на получение связей

Сравнение запросов на получение связей отражено в таблице 6.

Таблица 6 – Запросы на получение связей

Параметр	Neo4j	SQL
Пример запроса	<code>MATCH path=(c:Character)-[*1..3]-(related)</code>	Многотабличные JOIN с подзапросами
Сложность	$O(k)$, где k - длина пути	$O(n^k)$ для k -уровневых связей
Коллекции/ Таблицы	Только задействованные узлы	Все таблицы в JOIN
Индексация	Автоматическая по связям	Требуются составные индексы

2. Агрегационные запросы

Сравнение агрегационных запросов представлено в таблице 7.

Таблица 7 – Агрегационные запросы

Параметр	Neo4j	SQL
Пример запроса	MATCH (c:Character) RETURN c.house, COUNT(c)	SELECT house_id, COUNT(*) FROM character GROUP BY house_id
Производительность	Средняя, требует полного сканирования	Высокая, оптимизирована
Использование индексов	Только для фильтрации	Эффективно для GROUP BY

C. Вывод

Графовая модель Neo4j идеально подходит для представления сложных взаимосвязей между сущностями, таких как отношения между персонажами, связи с заклинаниями и зельями. Она предоставляет большую гибкость, удобные запросы и лучше масштабируется для задач, связанных с анализом графов.

Однако, реляционная модель остается более компактной и простой в использовании для табличных данных.

Для каталога вселенной Гарри Поттера с учетом требований и сценариев использования оптимальным выбором является NoSQL модель на Neo4j, несмотря на несколько большую избыточность данных.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Приложение представляет собой современное веб-решение, построенное на микросервисной архитектуре и состоящее из следующих основных компонентов:

1) Frontend (клиентская часть):

- Модуль отображения данных (компоненты Vue.js)
- Модуль визуализации графа связей
- Модуль поиска и фильтрации
- Модуль управления состоянием

2) Backend (серверная часть):

- API-сервер на Flask
- Сервисный слой для бизнес-логики
- Модуль работы с базой данных
- Модуль обработки запросов

3) База данных:

- Neo4j для хранения графовых данных
- Модуль визуализации графа

3.2. Использованные технологии

Frontend:

- Vue.js 3 (фреймворк)
- Vite (сборщик)
- JavaScript/TypeScript
- Современные веб-технологии (HTML5, CSS3)

Backend:

- Python
- Flask (веб-фреймворк)
- REST API
- Neo4j Python Driver

Инфраструктура:

- Docker
- Docker Compose
- Neo4j (графовая база данных)
- Git (система контроля версий)

3.3. Снимки экрана приложения

Экраны приложения и переходы между ними отображены на рисунке 3.

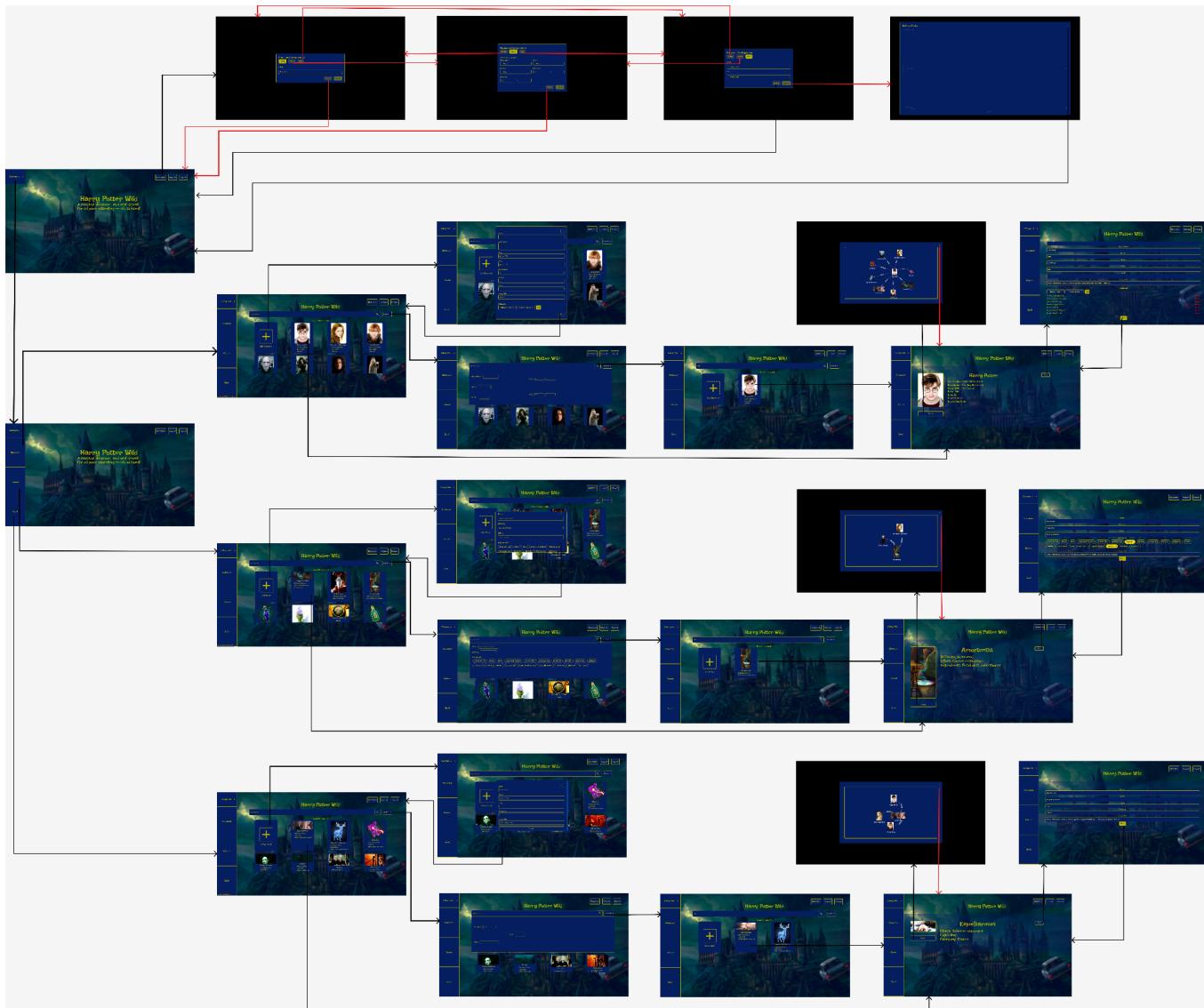


Рисунок 3 – Снимки экрана

ВЫВОДЫ

4.1. Достигнутые результаты

- Создано полнофункциональное веб-приложение с интуитивно понятным интерфейсом
- Реализована эффективная система хранения и обработки графовых данных
- Обеспечена высокая производительность при работе с большими объемами данных
- Реализована удобная система поиска и фильтрации информации
- Создана интерактивная визуализация связей между сущностями

4.2. Недостатки и пути для улучшения полученного решения

- Возможность добавления системы аутентификации и авторизации
- Расширение функционала для работы с изображениями и медиа-контентом
- Улучшение системы кэширования для оптимизации производительности
- Добавление системы уведомлений и обратной связи
- Расширение возможностей визуализации графа

4.3. Будущее развитие решения

- Интеграция с социальными сетями
- Добавление системы комментариев и обсуждений
- Создание мобильной версии приложения
- Расширение базы данных новыми сущностями и связями
- Внедрение системы рекомендаций на основе анализа графа
- Добавление многоязычности
- Реализация системы тегов и категорий
- Интеграция с внешними API для получения дополнительной информации

ЗАКЛЮЧЕНИЕ

В ходе разработки веб-приложения была успешно реализована современная система для работы с данными вселенной Гарри Поттера. Приложение демонстрирует эффективное использование графовой базы данных Neo4j для хранения и анализа сложных взаимосвязей между персонажами, объектами и событиями.

Ключевыми достижениями проекта являются:

1. Создание интуитивно понятного пользовательского интерфейса, обеспечивающего удобный доступ к информации
2. Реализация эффективной системы хранения и обработки графовых данных
3. Разработка гибкой архитектуры, позволяющей легко расширять функциональность
4. Успешное применение современных технологий и практик разработки

Проект показал, что использование графовой базы данных является оптимальным решением для работы со сложными взаимосвязями в контексте вселенной Гарри Поттера. Микросервисная архитектура и контейнеризация обеспечили надежность и масштабируемость системы.

Несмотря на достигнутые результаты, проект имеет потенциал для дальнейшего развития. Планируемые улучшения, такие как добавление системы аутентификации, расширение функционала работы с медиа-контентом и интеграция с социальными сетями, позволят сделать приложение еще более полезным и привлекательным для пользователей.

В целом, проект успешно демонстрирует возможности современных веб-технологий в создании интерактивных систем для работы со сложными данными и их визуализацией.

ПРИЛОЖЕНИЯ

1. Документация по сборке и развертыванию приложения

1. Системные требования

- Операционная система: Ubuntu 22.04 или новее
- Docker и Docker Compose

2. Подготовка к установке

Клонирование репозитория:

```
git clone https://github.com/moevm/nosql1h25-hogwarts.git  
cd nosql1h25-hogwarts
```

Настройка переменных окружения: отредактируйте файл .env в соответствии с вашими требованиями

3. Сборка и запуск

Сборка контейнеров:

```
docker-compose build --no-cache
```

Запуск приложения:

```
docker-compose up
```

4. Доступ к сервисам

Frontend: <http://localhost:3000>

Backend API: <http://localhost:5000>

Neo4j Browser: <http://localhost:7474>

5. Остановка приложения

Остановка контейнеров:

```
docker-compose down
```

Остановка и удаление контейнеров (включая данные БД)

```
docker-compose down -v
```

2. Инструкция для пользователя

1. Начало работы

- Откройте приложение в браузере по адресу <http://localhost:3000>
- На главной странице вы увидите основные разделы приложения

2. Основные функции

1) Поиск персонажей и объектов

- Используйте поисковую строку для быстрого поиска
- Применяйте фильтры для уточнения результатов
- Просматривайте детальную информацию о выбранном объекте

2) Работа с графом связей

- Выберите интересующий вас персонаж или объект
- Исследуйте связи с другими сущностями
- Используйте инструменты масштабирования и навигации

3. Советы по использованию

- Для эффективного поиска используйте ключевые слова
- При работе с графом начинайте с конкретного персонажа
- Используйте фильтры для уточнения результатов поиска
- Сохраняйте интересные находки для последующего изучения

4. Возможные проблемы и их решение

- 1) Если график не отображается, проверьте подключение к интернету
- 2) В случае ошибок обновите страницу или очистите кэш браузера

ЛИТЕРАТУРА

Официальная документация:

- 1) Vue.js 3 Documentation – <https://vuejs.org/guide/introduction.html>
- 2) Flask Documentation – <https://flask.palletsprojects.com/>
- 3) Neo4j Documentation – <https://neo4j.com/docs/>
- 4) Docker Documentation – <https://docs.docker.com/>
- 5) Vite Documentation – <https://vitejs.dev/guide/>

Ссылка на репозиторий с проектом –

<https://github.com/moevm/nosql1h25-hogwarts>