**AI-BOOK**   
*Бот, который знает, что тебе понравится раньше, чем ты сам*

**ТРЕБОВАНИЯ К ПРОЕКТУ**

Автор: Новгородов Иван

**Содержание:**

1. **Бизнес-требования……………………………………………………………………...3**

**1.1 User Story………………………………………………………………………………...3**

**1.2 Use Case…………………………………………………………………………………..3**

**1.3 Макет……………………………………………………………………………………..5**

**1.4 BPMN…………………………………………………………………………………….6**

**2. Функциональные требования………………………………………………………….7**

**2.1 Архитектура……………………………………………………………………………..7**

**2.2 Модель данных…………………...……………………………………………………..7**

**2.3 ERD-диаграмма………………………………………………………………………...8**

**2.4 Диаграмма последовательности……………………………………………………...9**

**2.5** **REST...................................................................................................................................9**

**2.6 Swagger………………………………………………………………………………….10**

**3. Нефункциональные требования……………………………………………………...11**

**3.1 Критерии приемки…………………………………………………………………….11**

**3.2 Нефункциональные требования…………………………………………………….11**

1. **Бизнес-требования**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**User Story**

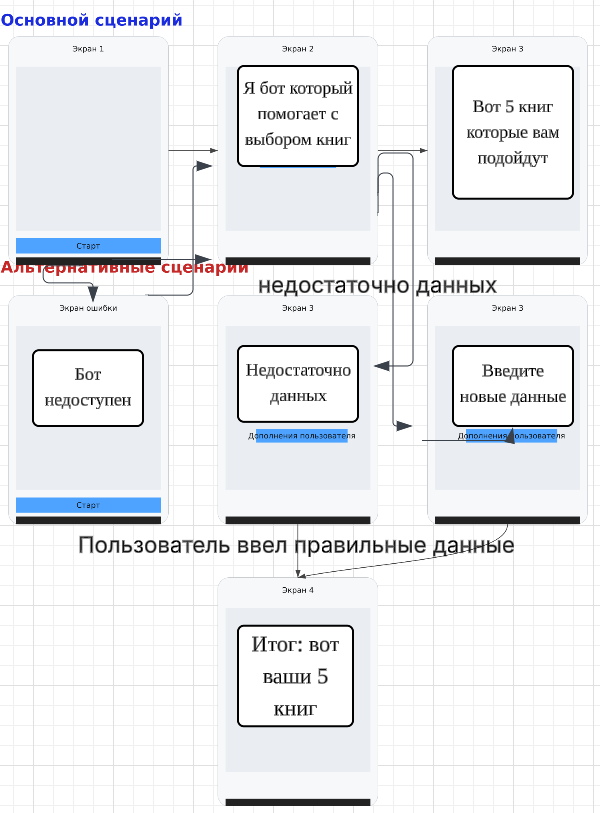
Я, как читатель, хочу получать персональные рекомендации книг от AI-бота в Telegram, основанные на моих интересах, языковых предпочтениях и предыдущих оценках, чтобы экономить время и находить литературу, которая мне действительно нравится.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Use Case**

| **Наименование структуры** | **Описание** |
| --- | --- |
| **Заголовок** | Получение персонализированных рекомендаций книг |
| **Акторы** | Основной: пользователь Telegram. Вспомогательные: LLM (модель рекомендаций), backend AI-BOOK, Telegram API |
| **Предусловие** | У пользователя установлен Telegram и выполнена авторизация в боте |
| **Ограничения** | Поддержка русского и английского языков; Контент - только рекомендательный; |
| **Триггер** | Пользователь вводит команду /start или выбирает «Получить рекомендации» |
| **Основной сценарий** | 1. Бот приветствует пользователя и предлагает заполнить краткий профиль. 2. Пользователь указывает любимые жанры, авторов, язык. 3. Система сохраняет данные в БД. 4. Пользователь отправляет запрос «Что почитать?» или /recommend. 5. Бот обращается к LLM-модели, генерирует рекомендации и объяснение выбора. 6. Система учитывает выбор пользователя для последующих рекомендаций. |
| **Альтернативный сценарий** | Пользователь не заполняет профиль → выдаются случайные рекомендации по популярным книгам. |
| **Исключительный сценарий** | Ошибка соединения с LLM или Telegram API → система показывает сообщение «Попробуйте позже». |
| **Результат** | Пользователь получает релевантную рекомендацию и может оценить её, улучшая точность ИИ. |

**Макет**



**BPMN**

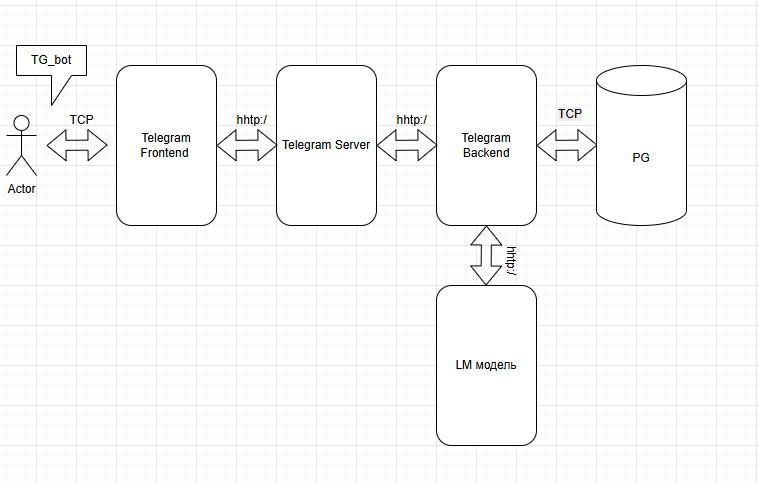
(диаграмма взаимодействия: пользователь → бот → LLM → база данных → ответ)

1. **Функциональные требования**

**Архитектура:**

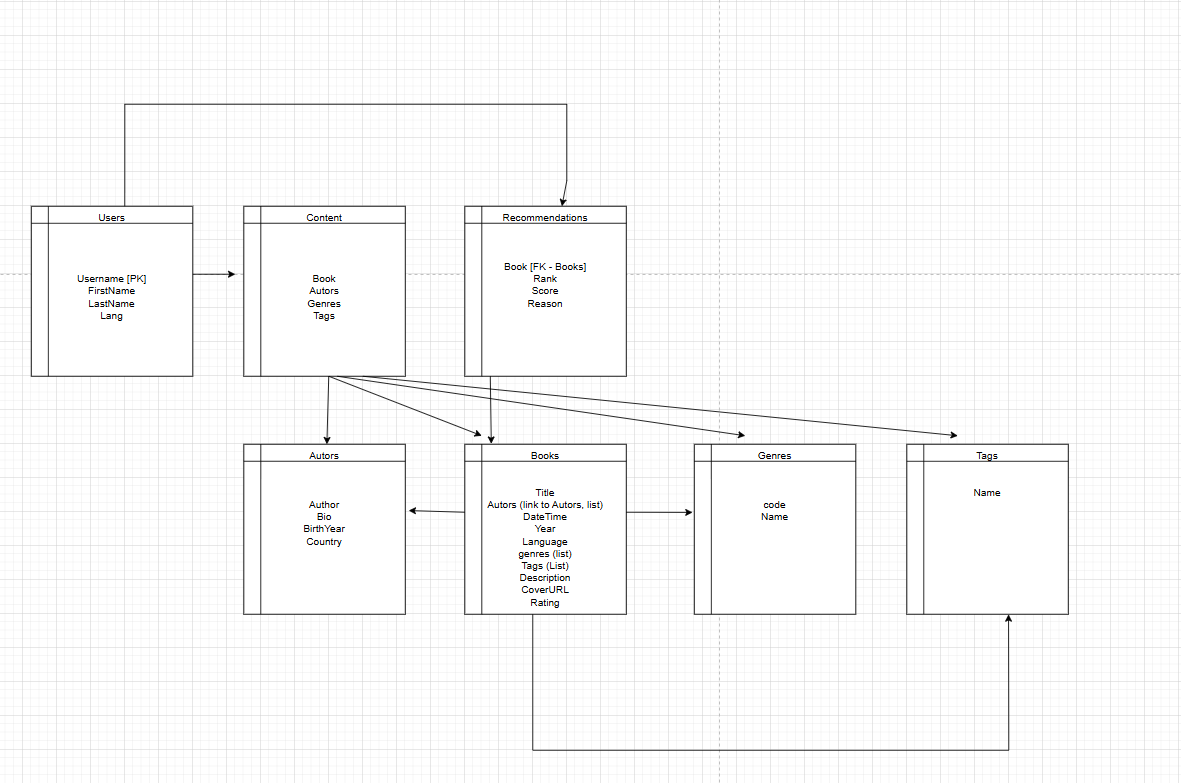
* **Frontend:** Telegram интерфейс (Bot API)
* **Backend:** FastAPI
* **Database:** PostgreSQL
* **ML-модуль:** LLM (например, GPT-4 или аналог, через REST API)
* **Интеграции:** Telegram API, Google Books API (опционально)

**Архитектура**

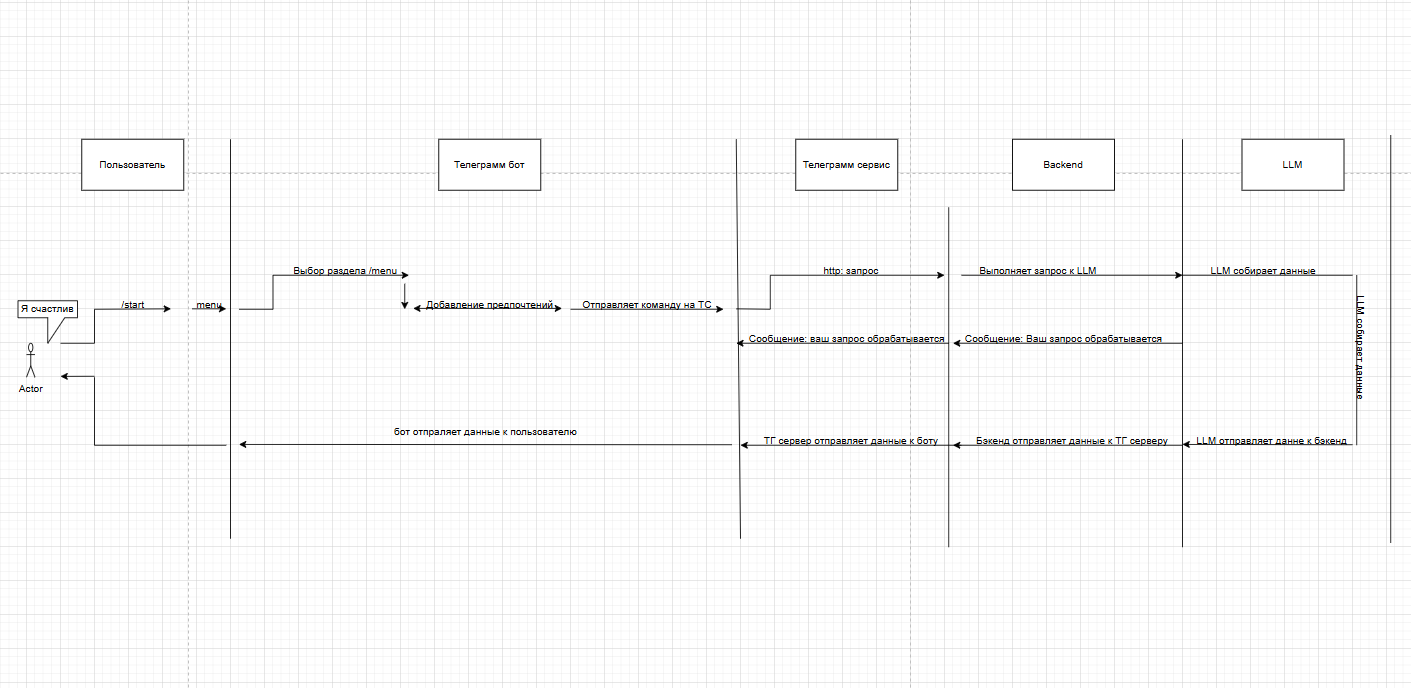


**Модель данных**

| **Сущность** | **Атрибут** | **Описание** |
| --- | --- | --- |
| **Users** | user\_id | Telegram ID пользователя |
|  | username | Имя или никнейм |
|  | language | Язык интерфейса |
| **Profiles** | favorite\_genres | Любимые жанры |
|  | favorite\_authors | Любимые авторы |
|  | created\_at | Дата создания |
| **Books** | book\_id, title, author, genre, description | Каталог книг |
| **Reactions** | reaction\_id, user\_id, book\_id, reaction | Оценка пользователем |

**ERD-диаграмма**

**Диаграмма последовательности**

****

**7. REST**

**GET /v1/users/{telegram\_id}**  
→ Получить профиль пользователя.

**POST /v1/users/{user\_id}/recommendations**  
→ Получить рекомендации книг.

**POST /v1/reactions**  
→ Сохранить рекомендацию пользователей.

8. **Swagger (OpenAPI 3.0.3)**



* 1. **Нефункциональные требования**

**Критерии приёмки:**

| **№** | **Критерий** | **Описание** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| AC-1 | Создание профиля | Пользователь вводит предпочтения → бот сохраняет данные |

|  |  |  |
| --- | --- | --- |
| AC-2 | Получение рекомендаций | /recommend возвращает 3 книги ≤ 5 сек |

|  |  |  |
| --- | --- | --- |
| AC-3 | Оценка рекомендаций | Лайки влияют на следующие выдачи |

|  |  |  |
| --- | --- | --- |
| AC-4 | Объяснение выбора | Каждая книга сопровождается текстом «Почему она тебе подойдёт» |

|  |  |  |
| --- | --- | --- |
| AC-5 | Ошибки соединения | При сбое выводится понятное сообщение пользователю |

**Нефункциональные требования**

* ⏱ **Время отклика API:** ≤ 3 сек
* 🧠 **LLM-ответ:** ≤ 10 сек
* 💾 **Uptime:** ≥ 99%
* 🔐 **Безопасность:** аутентификация через Telegram ID
* 📊 **Масштабируемость:** до 5000 пользователей одновременно
* 💡 **Наблюдаемость:** логирование, трейсинг, мониторинг