

Санкт-Петербургский Политехнический Университет Петра Великого  
Физико-Механический институт

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

**«Генерация диаграмм UML»**

По дисциплине «Автоматизация научных исследований»

Выполнили  
студенты гр. № 5040102/50201  
Зинкин С.В., Гордейко Н.Л., Усманов И.Р.

Преподаватель: Новиков Ф.А.

Санкт-Петербург  
2025 г.

## Генерация диаграмм UML

Лабораторная работа состоит из двух частей: в первой части мы просим строить диаграмму вариантов использования (**Use Case Diagram**), а во второй – диаграмму классов (**Class Diagram**).

В качестве объекта для исследований возьмём программную систему «**Система управления библиотекой**», для которой нам нужно эти диаграммы построить.

Пусть у нас есть простенькая система, в которой есть следующие действующие лица: читатель, библиотекарь, администратор.

Читатель может:

- 1) бронировать книги
- 2) продлевать срок пользования книгой
- 3) добавлять книгу в избранное
- 4) смотреть каталог книг

Библиотекарь может:

- 1) выдавать книги
- 2) возвращать книги
- 3) формировать отчёт по книгам
- 4) добавлять книги в каталог

Администратор может всё то же, что и библиотекарь, а также:

- 1) просматривать отчёты
- 2) удалять книги из каталога
- 3) мониторить состояние системы

Также возможна система штрафов (за просрочку, несвоевременный возврат книги, например)

В качестве ответа будем ожидать от нейронок PlantUML-код (с помощью PlantUML пишут код, описывая, что мы хотим видеть в диаграмме UML, и программа генерирует эту самую диаграмму).

Вместо заранее заданных уточняющих промптов было принято решение придумывать уточняющие промпты «на ходу», в зависимости от того, что именно в ответе нам не нравится.

## **Список промптов для диаграммы вариантов использования**

### **A1 (минимальный):**

Сгенерируй PlantUML код для диаграммы вариантов использования системы управления библиотекой. Укажи действующие лица, перечисли основные случаи использования.

### **A2 (базовый):**

Сгенерируй PlantUML код для диаграммы вариантов использования системы управления библиотекой. Укажи действующие лица: читатель, библиотекарь, администратор. Перечисли основные случаи использования, такие как просмотр каталога, бронирование книг, продление срока пользования, добавление в избранное, выдача книг, принятие возврата, формирование отчёта, добавление книги в каталог, удаление книги из каталога, и др.

### **A3 (расширенный):**

Сгенерируй PlantUML код для диаграммы вариантов использования системы управления библиотекой. Укажи действующие лица и перечисли случаи использования по ролям.

Для читателя перечисли действия, связанные с взаимодействием с книгами: поиск книг, просмотр каталога, бронирование, продление срока, управление избранными. Для библиотекаря перечисли действия, связанные с обслуживанием читателей: оформление выдачи, приём возврата, создание отчётов и работа с каталогом. Для администратора перечисли действия, связанные с управлением системой: контроль каталога, просмотр системной информации и доступ к отчётам.

Отобрази все связи между действующими лицами и соответствующими вариантами использования. Включи границу системы.

### **A4 (продвинутый):**

Сгенерируй PlantUML код для диаграммы вариантов использования системы управления библиотекой. Укажи действующие лица: Читатель, Библиотекарь, Администратор (Администратор наследует все функции Библиотекаря). Отобрази границу системы и все связи действующих лиц с вариантами использования.

Варианты использования для читателя:

- Просмотреть каталог
- Искать книгу
- Бронировать книгу
  - include → Проверка наличия книги
- Продлить срок пользования
  - include → Проверка задолженности
- Добавить книгу в избранное
- Просмотреть свои бронирования

Варианты использования для библиотекаря:

- Выдать книгу
  - include → Проверка наличия книги
  - extend → Резервирование книги (если книга недоступна)
- Принять возврат книги
  - include → Проверка задолженности
  - extend → Начисление штрафа за просрочку (если книга возвращена с опозданием)
- Добавить книгу в каталог
- Формировать отчёт по книгам
  - include → Просмотр задолженностей читателей

Варианты использования для администратора:

- Просматривать отчёты
- Удалять книги из каталога
- Мониторить состояние системы

## **Список промптов для диаграммы классов**

### **B1 (минимальный):**

Сгенерируй PlantUML код для диаграммы классов системы управления библиотекой. Укажи основные классы, покажи связи между ними.

### **B2 (базовый):**

Сгенерируй PlantUML код для диаграммы классов системы управления библиотекой. Укажи такие классы, как Человек, Читатель, Библиотекарь, Администратор, Книга, Выдача, Бронирование, и т.д. Покажи связи между ними и укажи базовые атрибуты классов (например, имя пользователя, название книги, дата выдачи).

### **B3 (расширенный):**

Сгенерируй PlantUML код для диаграммы классов системы управления библиотекой. Укажи классы и наследование: Человек → Читатель, Человек → Библиотекарь, Библиотекарь → Администратор, выдача книги, штраф, книга, бронирование, отчёт. Покажи связи между ними (ассоциация, агрегация, композиция, ассоциация-класс).

### **B4 (продвинутый):**

Сгенерируй PlantUML код для диаграммы классов системы управления библиотекой. Укажи следующие классы:

- Человек
- Читатель
- Библиотекарь
- Администратор
- Книга
- Экземпляр книги
- Выдача книги
- Бронирование
- Штраф
- Отчёт

Наследование:

- Человек → Читатель
- Человек → Библиотекарь
- Библиотекарь → Администратор

Атрибуты читателя: ФИО, дата рождения, статус задолженности, пол

Атрибуты библиотекаря: ФИО, дата рождения, список обработанных выдач, пол

Атрибуты администратора: ФИО, дата рождения, список управляемых отчётов, пол

Атрибуты книги: название, автор, ISBN, жанр, год издания

Экземпляр книги: уникальный инвентарный номер, статус

Выдача книги: дата выдачи, дата возврата

Бронирование: дата бронирования, срок действия Штраф:

сумма, дата начисления

Отчёт: дата создания, тип

Укажи типы данных атрибутов: string, date, int, boolean, или даже enumeration (для атрибута «статус», например, нужно будет создать enumeration со значениями «Доступна», «Выдана», «Забронирована»).

Покажи связи между классами (композиция, агрегация, ассоциация, ассоциация- класс), а также кратности (0, 1, \*, 0..\*, 0..1 и др.)

Все коды, которые сгенерировали ИИ, и все уточняющие промпты вы сможете увидеть в материалах к этой лабораторной, в примерах кода для каждой части были использованы варианты от GPT5-mini от промптов A4 и B4 соответственно

# Часть 1. Анализ генерации диаграммы вариантов использования

Пример кода:

```
@startuml  
left to right direction  
skinparam packageStyle rectangle  
  
actor Читатель  
actor Библиотекарь  
actor Администратор as Admin  
  
Admin -> Библиотекарь  
  
package "Система управления библиотекой" {  
  
usecase "Просмотреть каталог" as UC_ViewCatalog  
usecase "Искать книгу" as UC_SearchBook  
usecase "Бронировать книгу" as UC_Book  
usecase "Проверка наличия книги" as UC_CheckAvailability  
usecase "Продлить срок пользования" as UC_Renew  
usecase "Проверка задолженности" as UC_CheckDebt  
usecase "Добавить книгу в избранное" as UC_AddFavorite  
usecase "Просмотреть свои бронирования" as UC_ViewReservations  
  
usecase "Выдать книгу" as UC_LendBook  
usecase "Резервирование книги" as UC_ReservelfUnavailable  
usecase "Принять возврат книги" as UC_ReturnBook  
usecase "Начисление штрафа за просрочку" as UC_FineLate  
usecase "Добавить книгу в каталог" as UC_AddToCatalog  
usecase "Формировать отчёт по книгам" as UC_Report  
usecase "Просмотр задолженностей читателей" as UC_ViewDebts  
  
usecase "Просматривать отчёты" as UC_AdminReports  
usecase "Удалять книги из каталога" as UC_DeleteBooks
```

usecase "Мониторить состояние системы" as UC\_Monitor

' Связи Читателя

Читатель --> UC\_ViewCatalog

Читатель --> UC\_SearchBook

Читатель --> UC\_Book

UC\_Book --> UC\_CheckAvailability : <<include>>

Читатель --> UC\_Renew

UC\_Renew --> UC\_CheckDebt : <<include>>

Читатель --> UC\_AddFavorite

Читатель --> UC\_ViewReservations

' Связи Библиотекаря

Библиотекарь --> UC\_LendBook

UC\_LendBook --> UC\_CheckAvailability : <<include>>

UC\_LendBook --> UC\_ReservelfUnavailable : <<extend>>

Библиотекарь --> UC\_ReturnBook

UC\_ReturnBook --> UC\_CheckDebt : <<include>>

UC\_ReturnBook --> UC\_FineLate : <<extend>>

Библиотекарь --> UC\_AddToCatalog

Библиотекарь --> UC\_Report

UC\_Report --> UC\_ViewDebts : <<include>>

' Связи Администратора

Admin --> UC\_AdminReports

Admin --> UC\_DeleteBooks

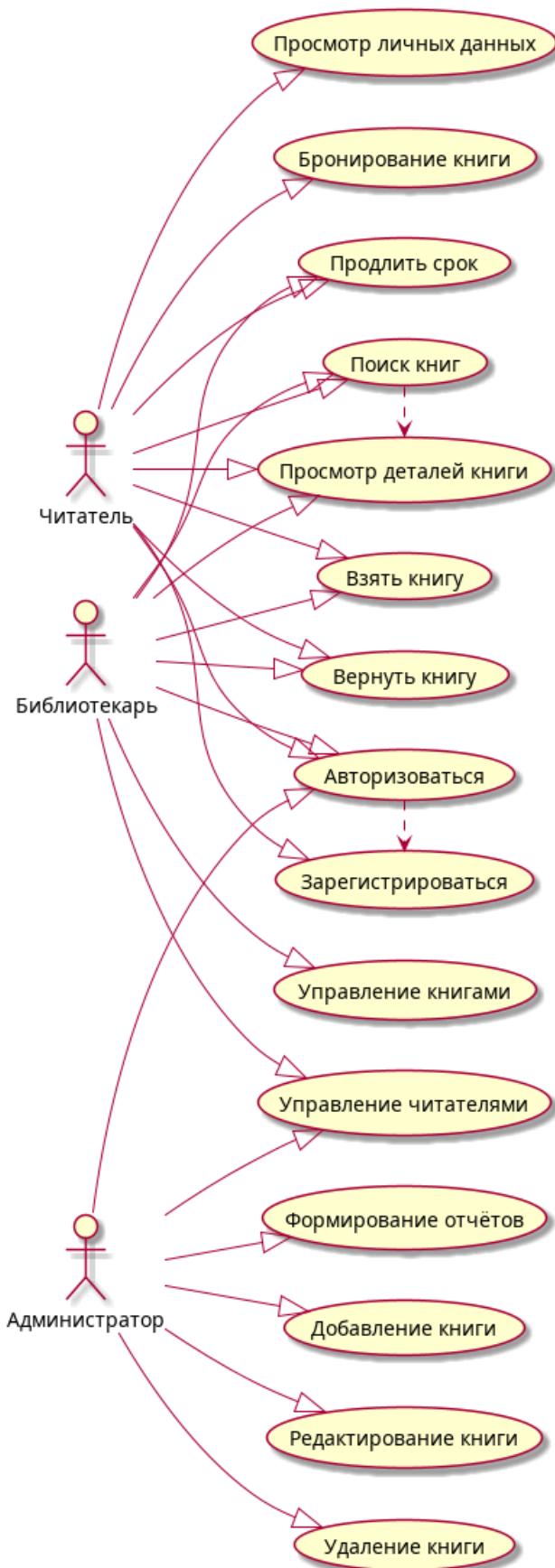
Admin --> UC\_Monitor

}

@enduml

## 1.1. Эксперимент с промптом A1 (Минимальный)

### 1.1.1. Модель: Алиса (Yandex)



- **Анализ:** При первой попытке получен **Syntax Error**. После исправлений допущена логическая ошибка, применив наследование к **вариантам использования**. Также присутствуют непонятные стрелки между вариантами использования.
- **Выводы:**
  - . **Работоспособность кода:** Низкая (требует исправлений).
  - . **Отсутствие ошибок:** НЕ соответствует (Syntax Error).
  - . **Соответствие логике UML:** НЕ соответствует (ошибки в типе отношений include).
  - . **Сложность и стиль:** Низкая (базовый синтаксис).
  - . **Соответствие содержанию ТЗ:** Соответствует.

#### **1.1.2. Модель: DeepSeek**



- Анализ:** Код содержал ошибку, связанную с !theme plain, вызвавшую ошибку рендера.
- Выводы:**
- Работоспособность кода:** Низкая (ошибка рендера).
- Отсутствие ошибок:** НЕ соответствует (ошибка темы).
- Соответствие логике UML:** Соответствует.
- Сложность и стиль:** Средняя (попытка использовать темы).
- Соответствие содержанию ТЗ:** Соответствует.

### 1.1.3. Модель: GPT5-mini



- **Анализ:** Сгенерировала простой, рабочий код с первого раза.
- **Выводы:**
  - . **Работоспособность кода:** Высокая.
  - . **Отсутствие ошибок:** Соответствует.
  - . **Соответствие логике UML:** Соответствует.
  - . **Сложность и стиль:** Базовый (чистый код).
  - . **Соответствие содержанию ТЗ:** Соответствует.

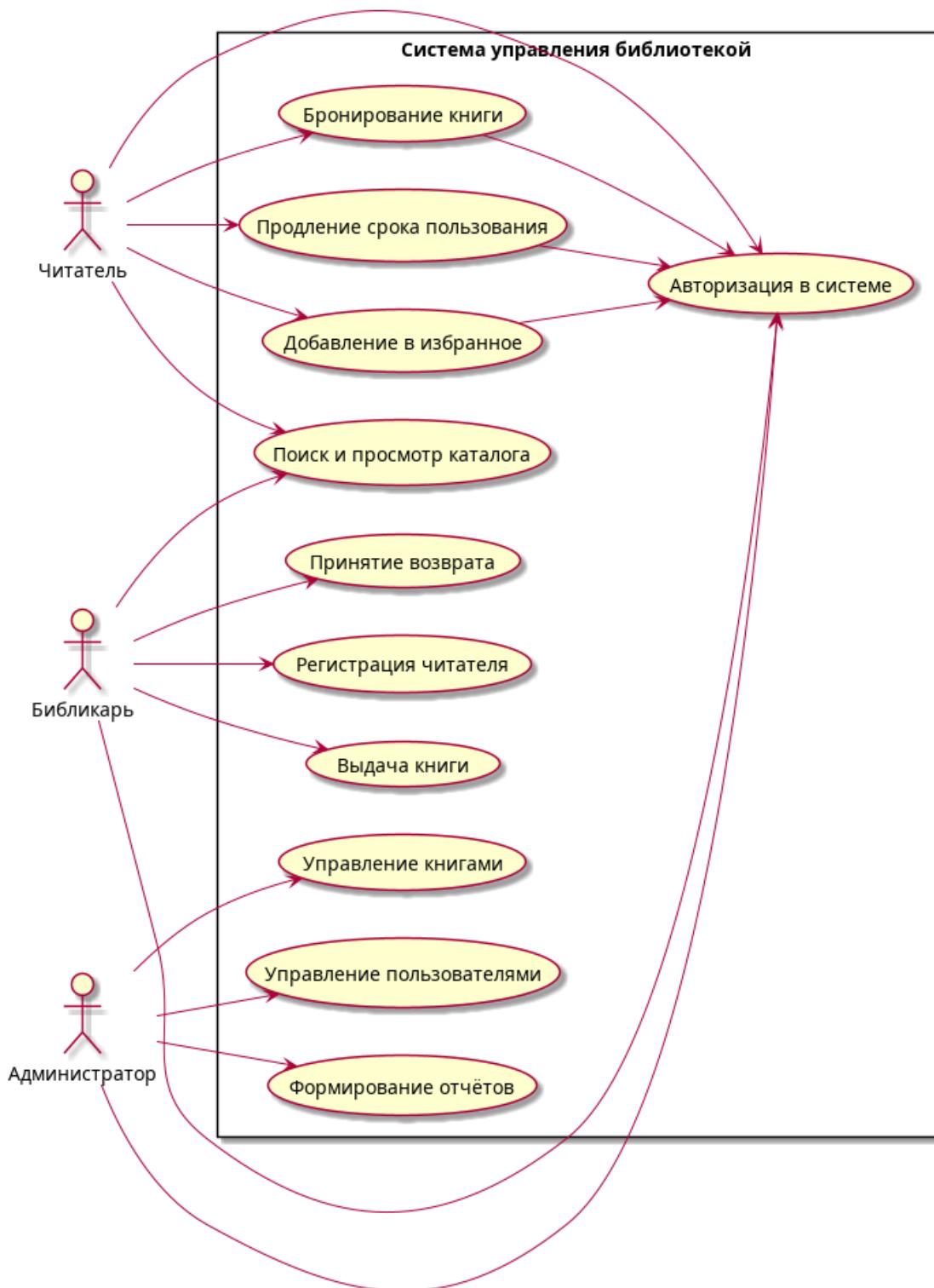
## 1.2. Эксперимент с промптом A2 (Базовый)

### 1.2.1. Модель: Алиса (Yandex)

- **Анализ:** Снова выдала **Syntax Error** на объявлении use case. Получить валидный код не получилось даже после 3 уточняющих промптов.
- **Выводы:**
  - . **Работоспособность кода:** Низкая (код не компилируется).
  - . **Отсутствие ошибок:** НЕ соответствует (повторяющиеся ошибки).
  - . **Соответствие логике UML:** -.

- Сложность и стиль: -.
- Соответствие содержанию ТЗ: -.

### 1.2.2. Модель: DeepSeek



- **Анализ:** Код скомпилировался, но диаграмма получилась **визуально хаотичной** и перегруженной, что мешает восприятию логики. Также нужно дописать "include" от "добавление в избранное", "продление срока пользования" и "бронирование книги" к "авторизация к системе" и поставить соответствующие пунктирные стрелки.
- **Выводы:**
  - . **Работоспособность кода:** Средняя (рабочий, но нечитаемый результат).
  - . **Отсутствие ошибок:** Соответствует.
  - . **Соответствие логике UML:** Соответствует.
  - . **Сложность и стиль:** Низкое качество визуализации.
  - . **Соответствие содержанию ТЗ:** Полное.

#### 1.2.3. Модель: GPT5-mini

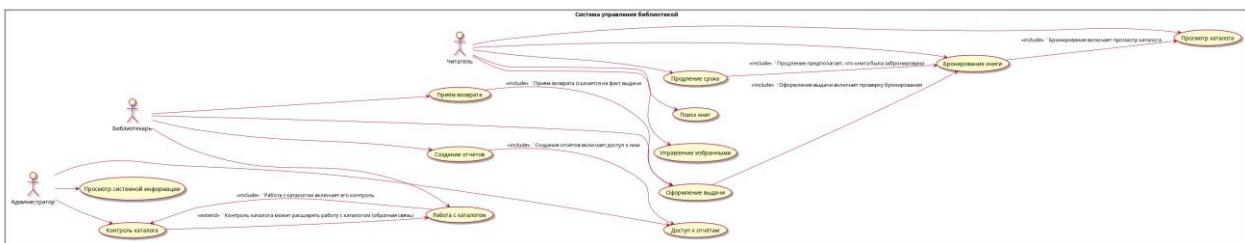


- **Анализ:** Сгенерировала чистый код. Связи распределены логично. Ошибка лишь в том, что отсутствует стрелка к варианту использования "Регистрация читателя"
- **Выводы:**
  - . **Работоспособность кода:** Высокая.
  - . **Отсутствие ошибок:** Соответствует.

- Соответствие логике UML:** Частично соответствует (нет стрелочки к регистрации читателя).
- Сложность и стиль:** Соответствует.
- Соответствие содержанию ТЗ:** Полное.

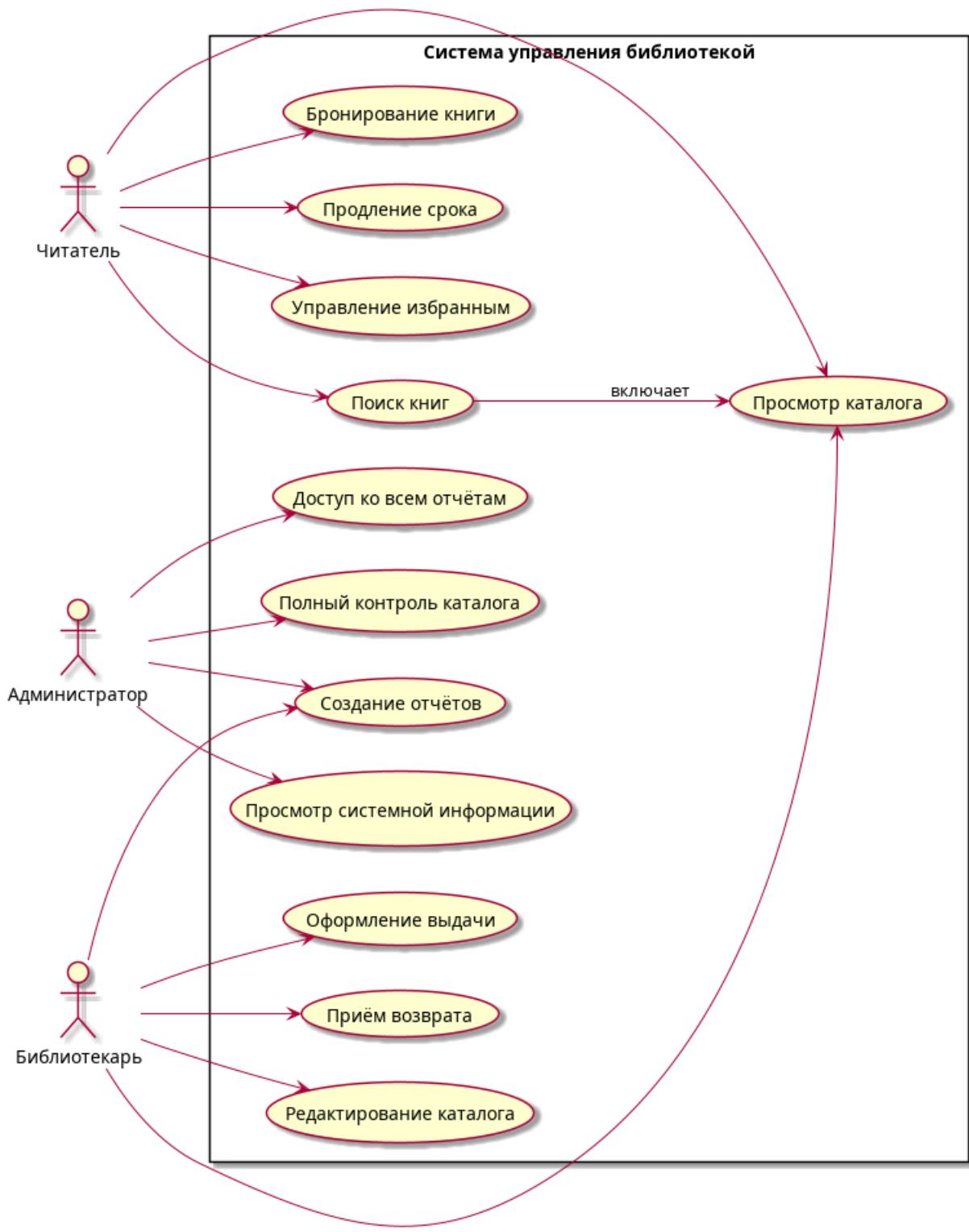
## 1.3. Эксперимент с промптом А3 (Расширенный)

### 1.3.1. Модель: Алиса (Yandex)



- Анализ:** Модель допустила **логическую ошибку** в интерпретации границ системы или группировки, которая потребовала ручного уточнения. Несмотря на отсутствие синтаксических ошибок, результат с первого раза был неполным.
- Выводы:**
  - Работоспособность кода:** Средняя (требует логической правки).
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Частичное (проблема с границами или группировкой).
  - Сложность и стиль:** Соответствует.
  - Соответствие содержанию ТЗ:** Неполное (на первом этапе).

### 1.3.2. Модель: DeepSeek



- Анализ:** Модель успешно реализовала требование по отображению границы системы и структурированию функций по ролям. Код скомпилировался, но допущена ошибка: должно быть "include", вместо "включает" и пунктирная стрелка.
- Выводы:**

- . **Работоспособность кода:** Высокая.
- . **Отсутствие ошибок:** Соответствует.
- . **Соответствие логике UML:** Частичное соответствие.
- . **Сложность и стиль:** Соответствует.
- . **Соответствие содержанию ТЗ:** Полное.

### **1.3.3. Модель: GPT5-mini**

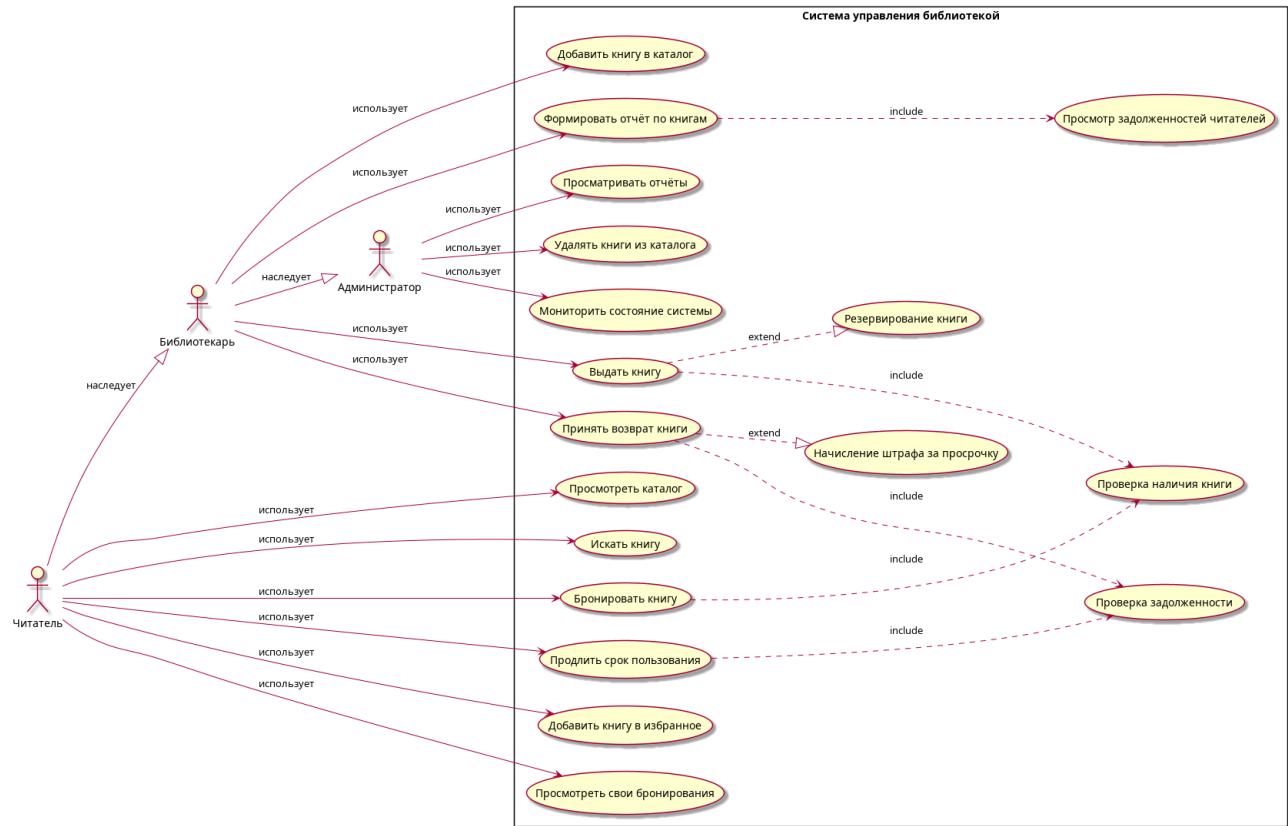


- **Анализ:** Успешно справилась с группировкой и границей системы, без необходимости уточнений.
- **Выводы:**
  - **Работоспособность кода:** Высокая.
  - **Отсутствие ошибок:** Соответствует.

- Соответствие логике UML:** Соответствует.
- Сложность и стиль:** Высокий.
- Соответствие содержанию ТЗ:** Полное.

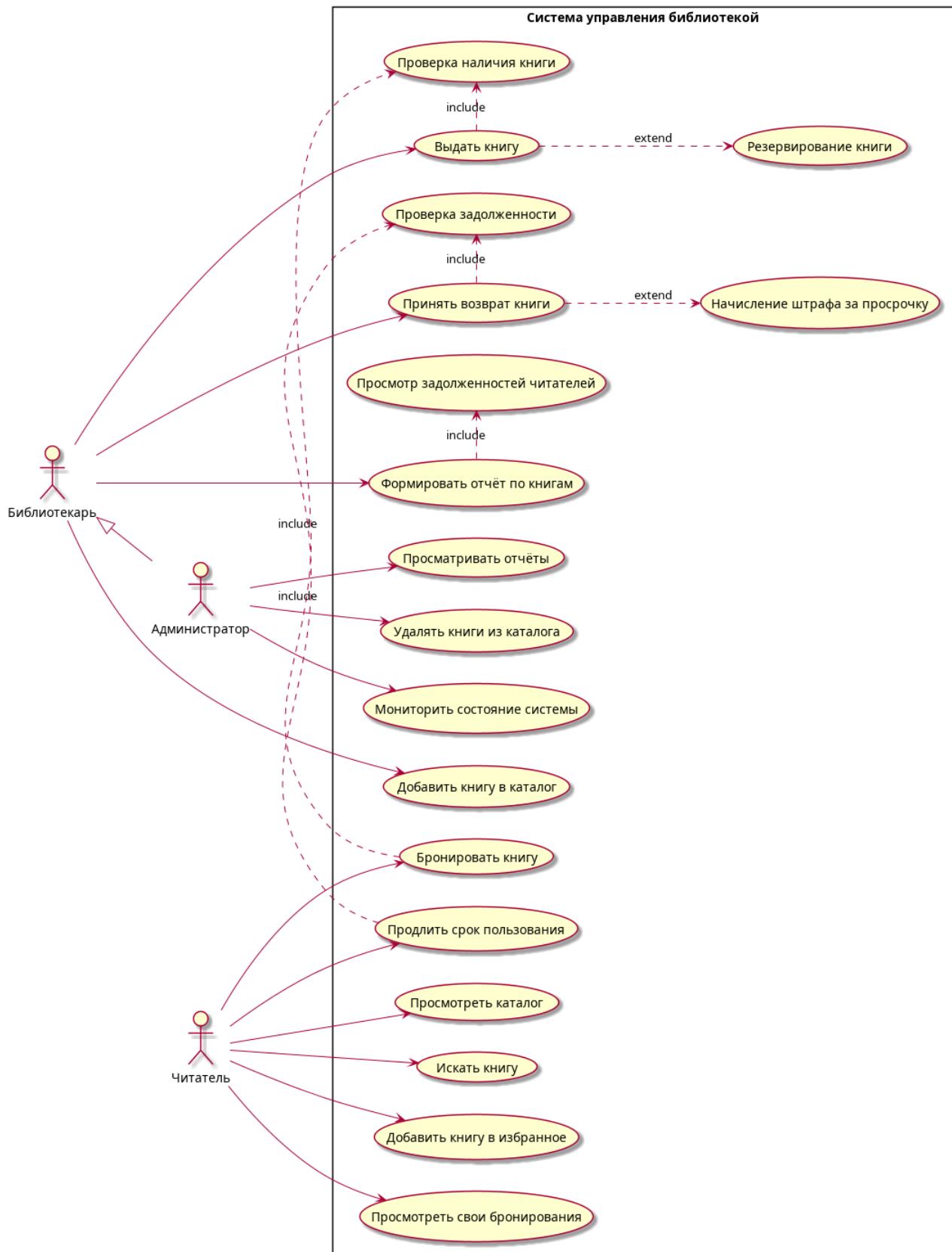
## 1.4. Эксперимент с промптом A4 (Продвинутый)

### 1.4.1. Модель: Алиса (Yandex)



- Анализ:** Сгенерировала валидный код с первого запроса. Некорректно указаны наследования (должны быть в другую сторону, ведь, например, не может читатель наследовать все права библиотекаря), а также немного неверные стрелки у extend/include.
- Выводы:**
  - Работоспособность кода:** Высокая.
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Частичное соответствие.
  - Сложность и стиль:** Соответствует.
  - Соответствие содержанию ТЗ:** Полное.

### 1.4.2. Модель: DeepSeek



- **Анализ:** Успешно выполнила все условия, реализовав наследование акторов символами UML. Допустила лишь ошибку в направлении стрелки extend.

- **Выводы:**
  - . **Работоспособность кода:** Высокая.
  - . **Отсутствие ошибок:** Соответствует.
  - . **Соответствие логике UML:** Частичное соответствие.
  - . **Сложность и стиль:** Высокий (продвинутый синтаксис).
  - . **Соответствие содержанию ТЗ:** Полное.

#### **1.4.3. Модель: GPT5-mini**



- **Анализ:** С первого раза выдала валидный код, точно следуя всем условиям. Допустила такие ошибки: стрелки `include` и `extend` должны быть пунктирными, а также стрелка `extend` должна быть направлена в другую сторону.
  - **Выводы:**

- . **Работоспособность кода:** Высокая.
- . **Отсутствие ошибок:** Соответствует.
- . **Соответствие логике UML:** Частичное соответствие.
- . **Сложность и стиль:** Высокий.
- . **Соответствие содержанию ТЗ:** Полное.

## Часть 2. Анализ генерации диаграммы классов

**Пример кода:**

@startuml

```
' --- Перечисления ---
enum BookStatus {
    Available
    Issued
    Reserved
}

' --- Классы ---
class Человек {
    +ФИО: string
    +ДатаРождения: date
    +Пол: string
}

class Читатель {
    +СтатусЗадолжности: boolean
}

class Библиотекарь {
    +СписокОбработанныхВыдач: List<ВыдачаКниги>
}

class Администратор {
    +СписокУправляемыхОтчетов: List<Отчет>
}
```

```
class Книга {  
    +Название: string  
    +Автор: string  
    +ISBN: string  
    +Жанр: string  
    +ГодИздания: int  
}  
  
class ЭкземплярКниги {  
    +УникальныйИнвентарныйНомер: string  
    +Статус: BookStatus  
}  
  
class ВыдачаКниги {  
    +ДатаВыдачи: date  
    +ДатаВозврата: date  
}  
  
class Бронирование {  
    +ДатаБронирования: date  
    +СрокДействия: int  
}  
  
class Штраф {  
    +Сумма: int  
    +ДатаНачисления: date  
}  
  
class Отчет {  
    +ДатаСоздания: date  
    +Тип: string  
}  
  
' --- Наследование ---
```

Человек <|-- Читатель  
Человек <|-- Библиотекарь  
Библиотекарь <|-- Администратор

' --- Связи ---

Читатель "0..\*" --> "0..\*" ВыдачаКниги : делает  
Читатель "0..\*" --> "0..\*" Бронирование : оформляет  
Читатель "0..\*" --> "0..\*" Штраф : получает

Библиотекарь "0..\*" --> "0..\*" ВыдачаКниги : обрабатывает

Администратор "0..\*" --> "0..\*" Отчет : управляет

Книга "1" --> "0..\*" ЭкземплярКниги : содержит

ЭкземплярКниги "0..\*" --> "0..1" ВыдачаКниги : выдана через  
ЭкземплярКниги "0..\*" --> "0..\*" Бронирование : зарезервирована через

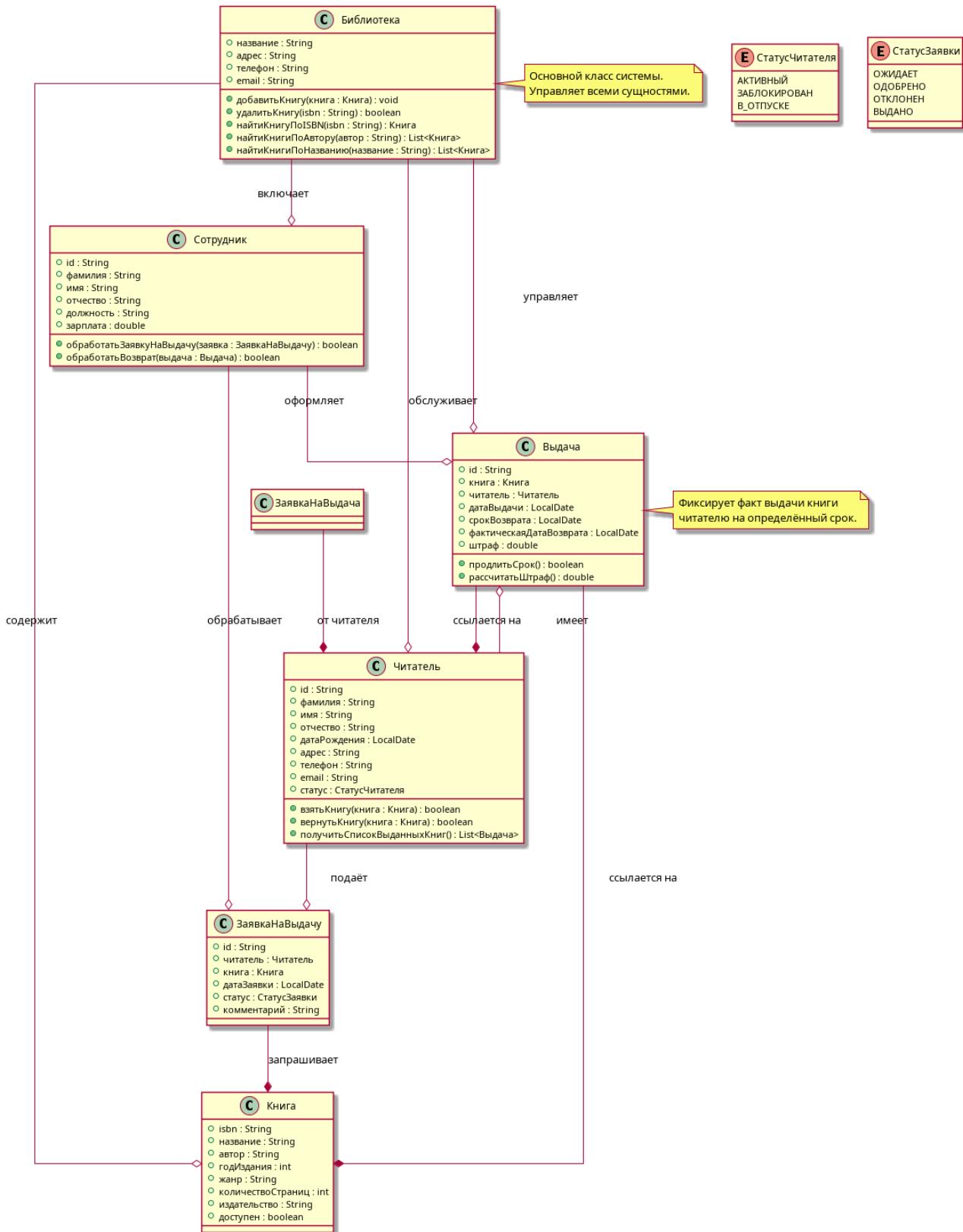
' --- Ассоциация-класс ---

ВыдачаКниги --> Штраф : может включать

@enduml

## **2.1. Эксперимент с промптом В1 (Минимальный)**

### **2.1.1. Модель: Алиса (Yandex)**



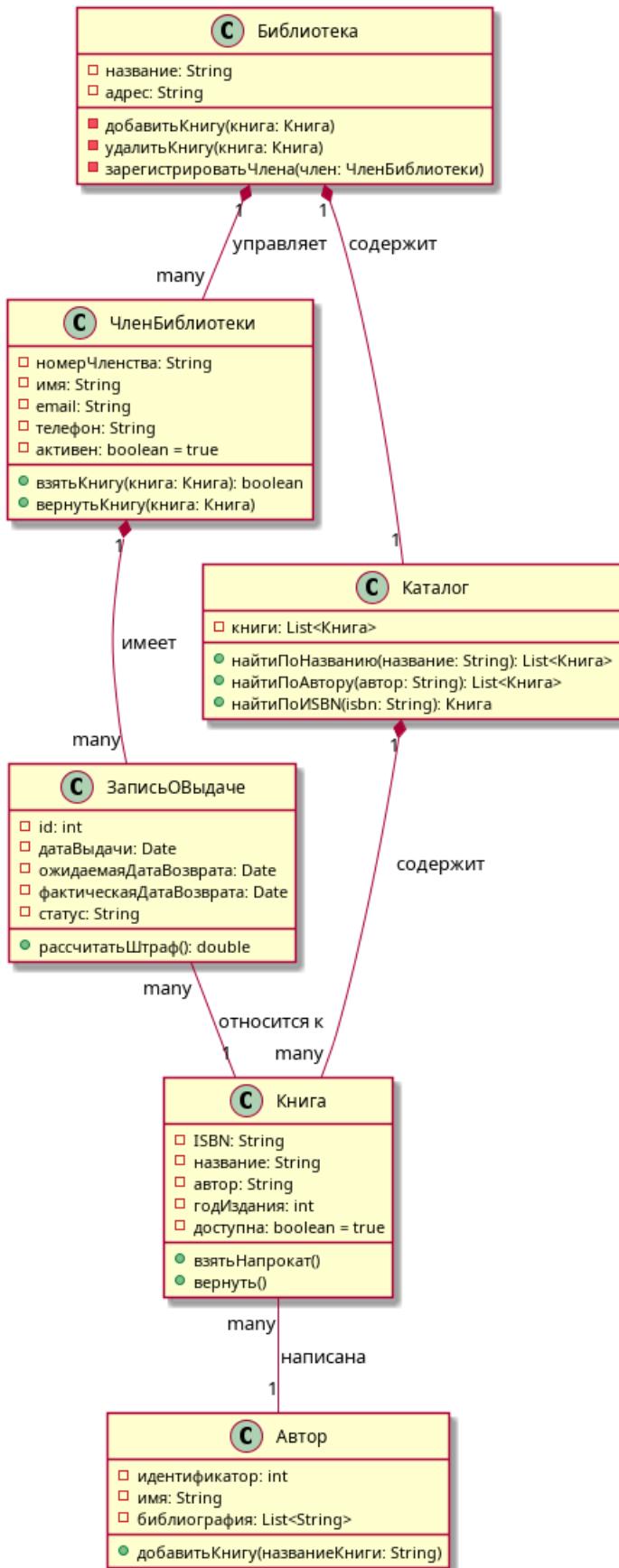
- Анализ:** Предложила избыточную структуру, самостоятельно добавив **методы и детализацию**, не запрошенные в минимальном ТЗ. Допустила такие ошибки: в отношении библиотеки и читателя белый ромб указан у читателя, хотя должен быть у

библиотеки, также в выдаче и читателя указано два отношения, хотя должно быть одно.

- **Выводы:**

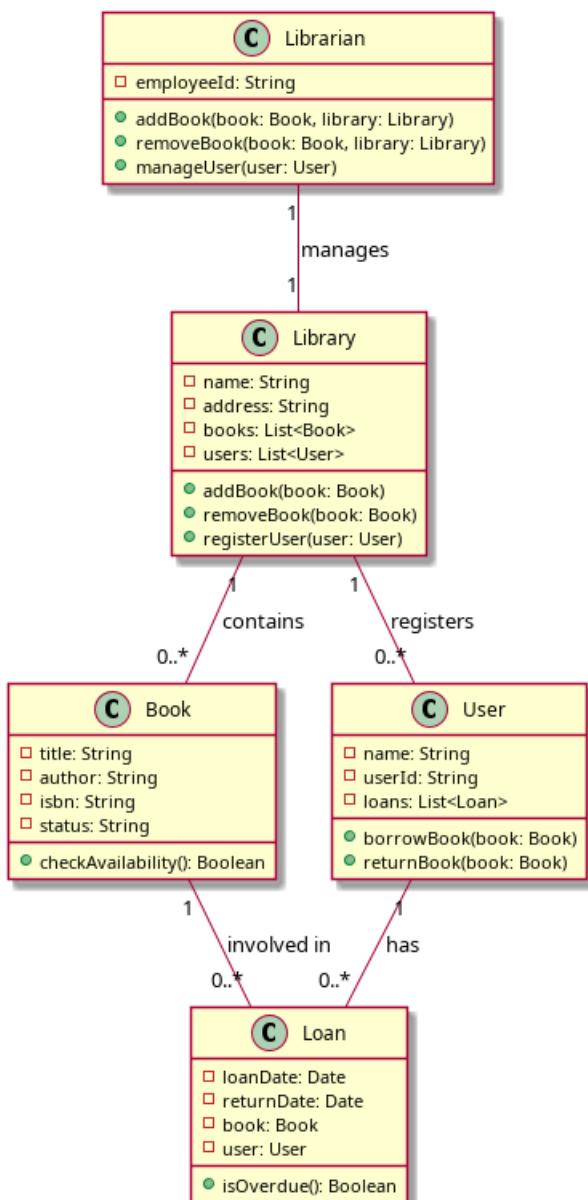
- . **Работоспособность кода:** Высокая.
- . **Отсутствие ошибок:** Соответствует.
- . **Соответствие логике UML:** НЕ соответствует.
- . **Сложность и стиль:** Избыточное усложнение.
- . **Соответствие содержанию ТЗ:** Полное.

### **2.1.2. Модель: DeepSeek**



- Анализ:** Использовала директиву hide empty members, выделив стандартные классы. Композиция должна идти в другую сторону.
- Выводы:**
  - Работоспособность кода:** Высокая.
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Частично соответствует.
  - Сложность и стиль:** Соответствует.
  - Соответствие содержанию ТЗ:** Полное.

### 2.1.3. Модель: GPT5-mini

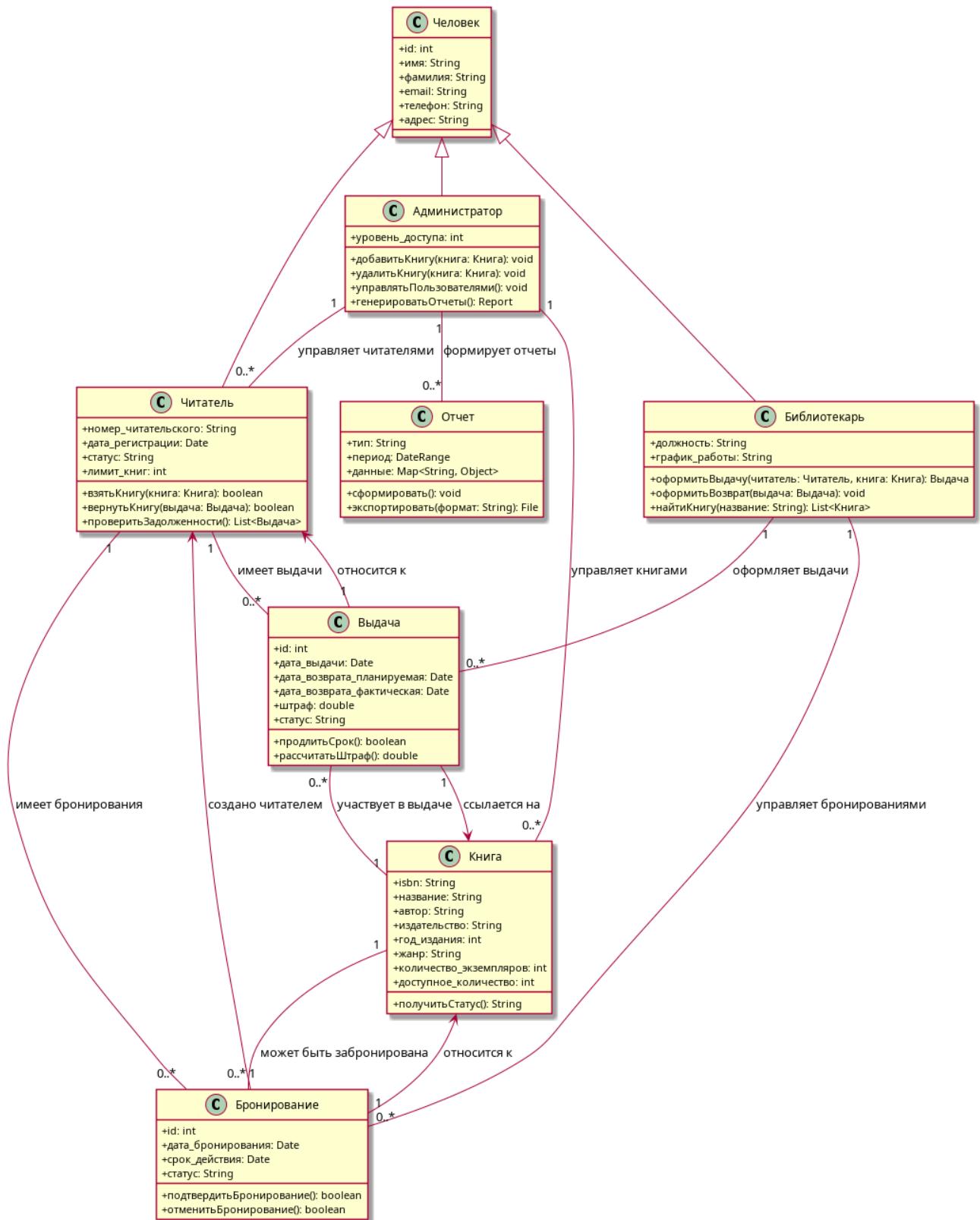


- Анализ:** Создала классическую, минимально необходимую структуру. У библиотекаря можно было бы указать, например, `1..*`, но это не критично.

- **Выводы:**
  - . **Работоспособность кода:** Высокая.
  - . **Отсутствие ошибок:** Соответствует.
  - . **Соответствие логике UML:** Соответствует.
  - . **Сложность и стиль:** Соответствует.
  - . **Соответствие содержанию ТЗ:** Полное.

## **2.2. Эксперимент с промптом В2 (Базовый)**

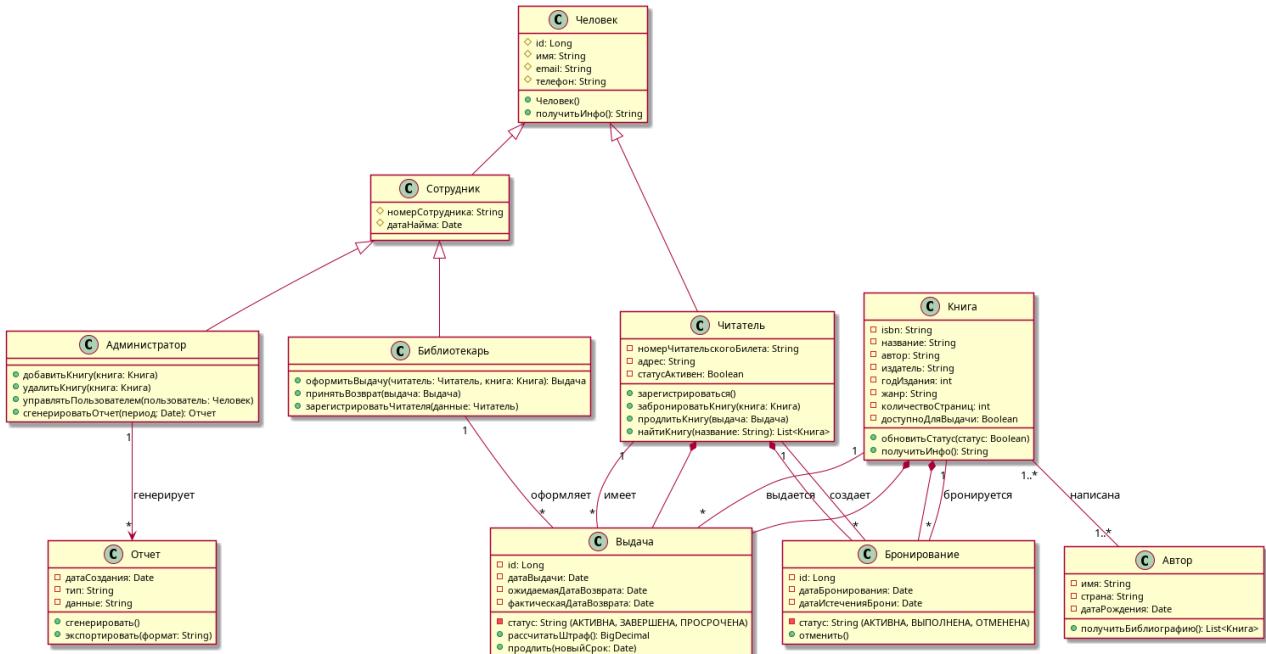
### **2.2.1. Модель: Алиса (Yandex)**



- Анализ:** Корректно добавила атрибуты, используя русский язык. Допустила такие ошибки: в диаграмме классов нет отношения со сплошной стрелкой.
- Выводы:**

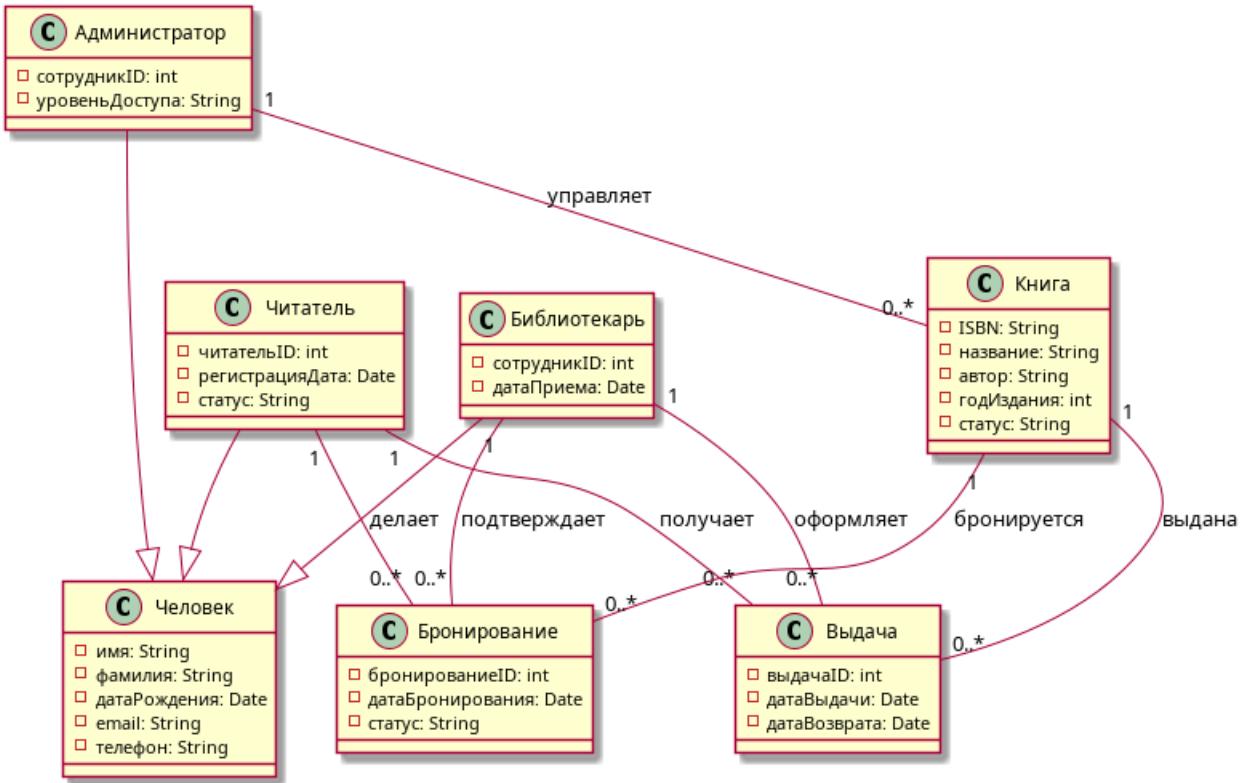
- Работоспособность кода:** Высокая.
- Отсутствие ошибок:** Соответствует.
- Соответствие логике UML:** НЕ соответствует.
- Сложность и стиль:** Базовый.
- Соответствие содержанию ТЗ:** Полное.

## 2.2.2. Модель: DeepSeek



- Анализ:** Определила атрибуты, сделав их **защищёнными (#)**, демонстрируя понимание принципов ООП. Вместо стрелки от администратора к отчёту должна быть ассоциация, также опять указаны два отношения у классов (лучше всего оставить композицию)
- Выводы:**
  - Работоспособность кода:** Высокая.
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Частично соответствует.
  - Сложность и стиль:** Высокий (использование принципов ООП).
  - Соответствие содержанию ТЗ:** Полное.

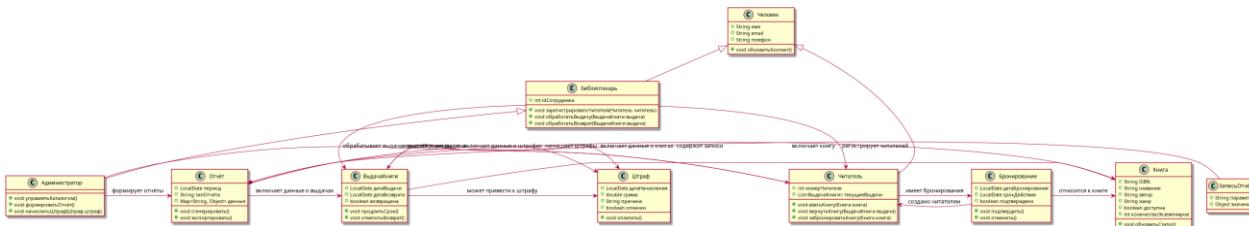
## 2.2.3. Модель: GPT5-mini



- **Анализ:** Четко следовала списку классов, добавила наследование Читатель --> Человек. Можно было бы объединить читателя, книгу и бронирование в "класс-ассоциация".
  - **Выводы:**
    - **Работоспособность кода:** Высокая.
    - **Отсутствие ошибок:** Соответствует.
    - **Соответствие логике UML:** Соответствует.
    - **Сложность и стиль:** Соответствует.
    - **Соответствие содержанию ТЗ:** Полное.

## 2.3. Эксперимент с промптом В3 (Расширенный)

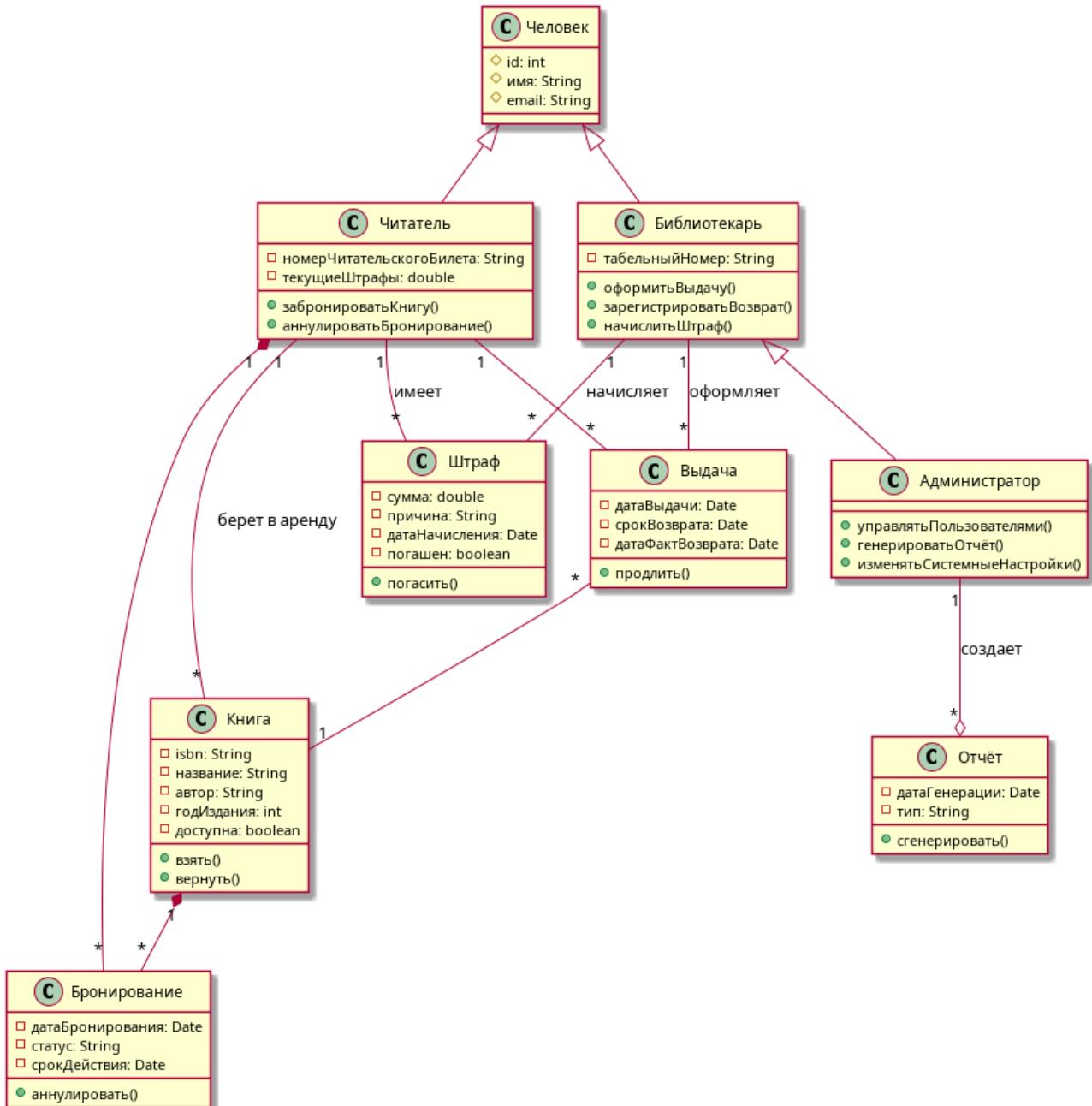
### 2.3.1. Модель: Алиса (Yandex)



- **Анализ:** Корректно реализовала наследование через extends и все типы связей. Диаграмма получилась хаотичной, отчего визуально она выглядит очень нагруженной.
  - **Выводы:**

- Работоспособность кода:** Высокая.
- Отсутствие ошибок:** Соответствует.
- Соответствие логике UML:** Полное.
- Сложность и стиль:** Визуально хаотичная.
- Соответствие содержанию ТЗ:** Полное.

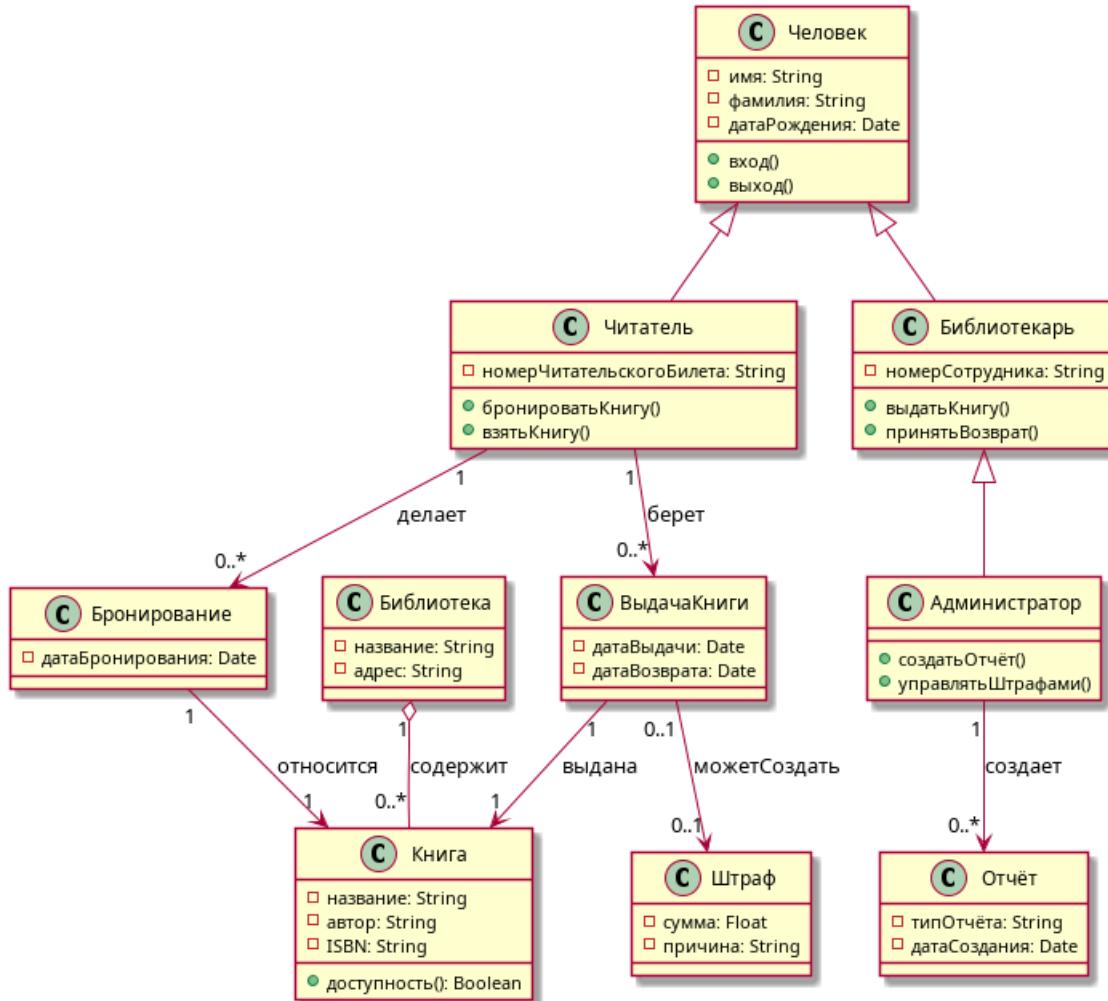
### 2.3.2. Модель: DeepSeek



- Анализ:** Успешно использовала агрегацию (o--) и композицию (\*--), реализовав все типы связей, но логичнее между читателем, книгой и бронированием использовать тернарное отношение "класс-ассоциация", также и в отношении администратора и отчёта должна быть обычная ассоциация.

- **Выводы:**
- **Работоспособность кода:** Высокая.
- **Отсутствие ошибок:** Соответствует.
- **Соответствие логике UML:** Частичное.
- **Сложность и стиль:** Высокий.
- **Соответствие содержанию ТЗ:** Полное.

### 2.3.3. Модель: GPT5-mini

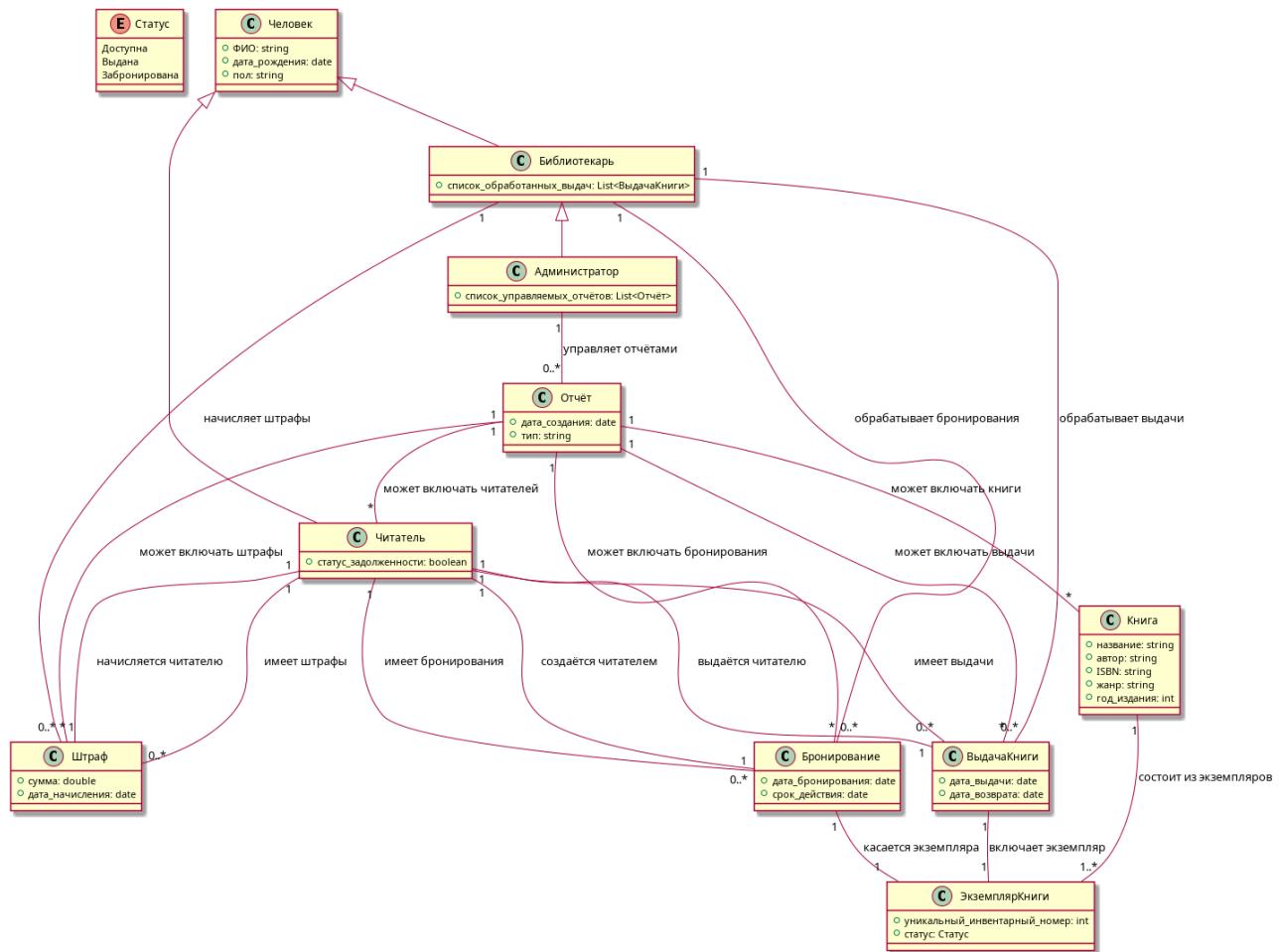


- **Анализ:** Корректно отобразила иерархию наследования и все типы связей. Использовала стрелки вместо линий, что является ошибкой.
- **Выводы:**
- **Работоспособность кода:** Высокая.
- **Отсутствие ошибок:** Соответствует.
- **Соответствие логике UML:** Почти полное.
- **Сложность и стиль:** Высокий.

. Соответствие содержанию ТЗ: Полное.

## 2.4. Эксперимент с промптом В4 (Продвинутый)

### 2.4.1. Модель: Алиса (Yandex)

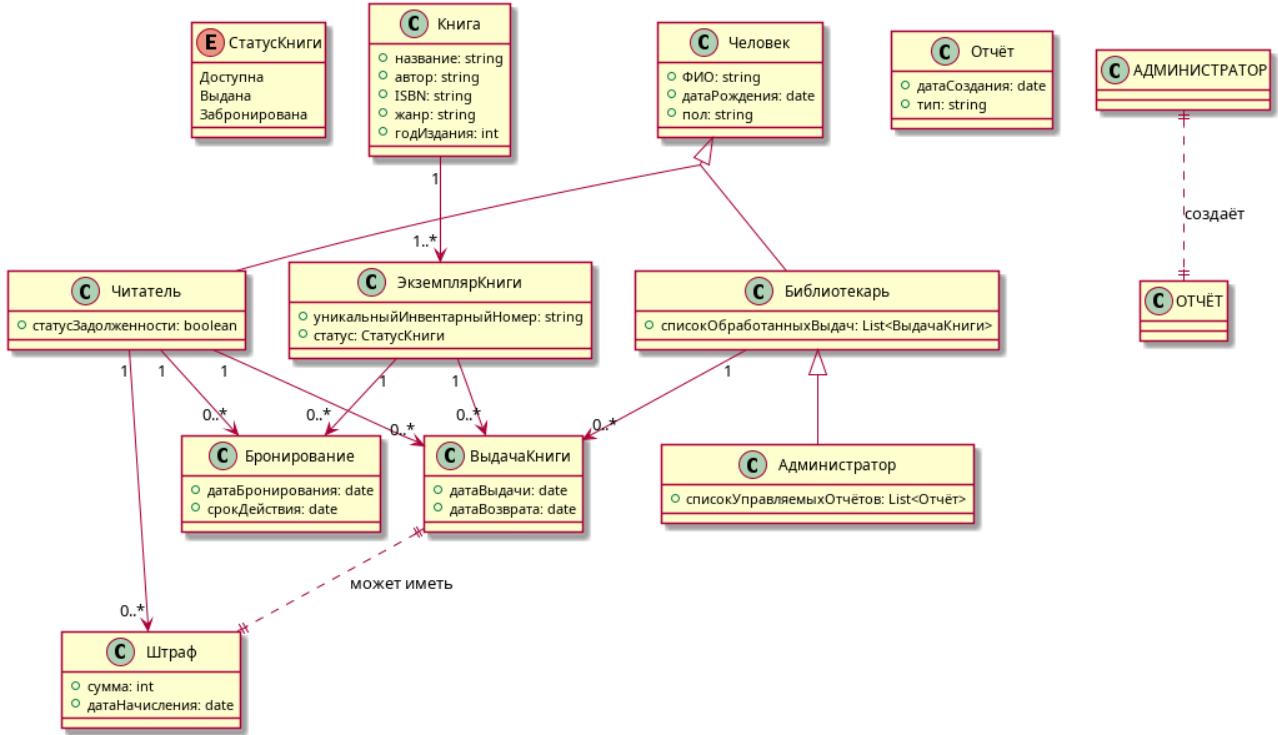


- Анализ:** Допустила **Syntax Error** в описании кратностей связей (смешение типов стрелок). Также немного неразборчиво сделала в целом всю диаграмму.

#### • Выводы:

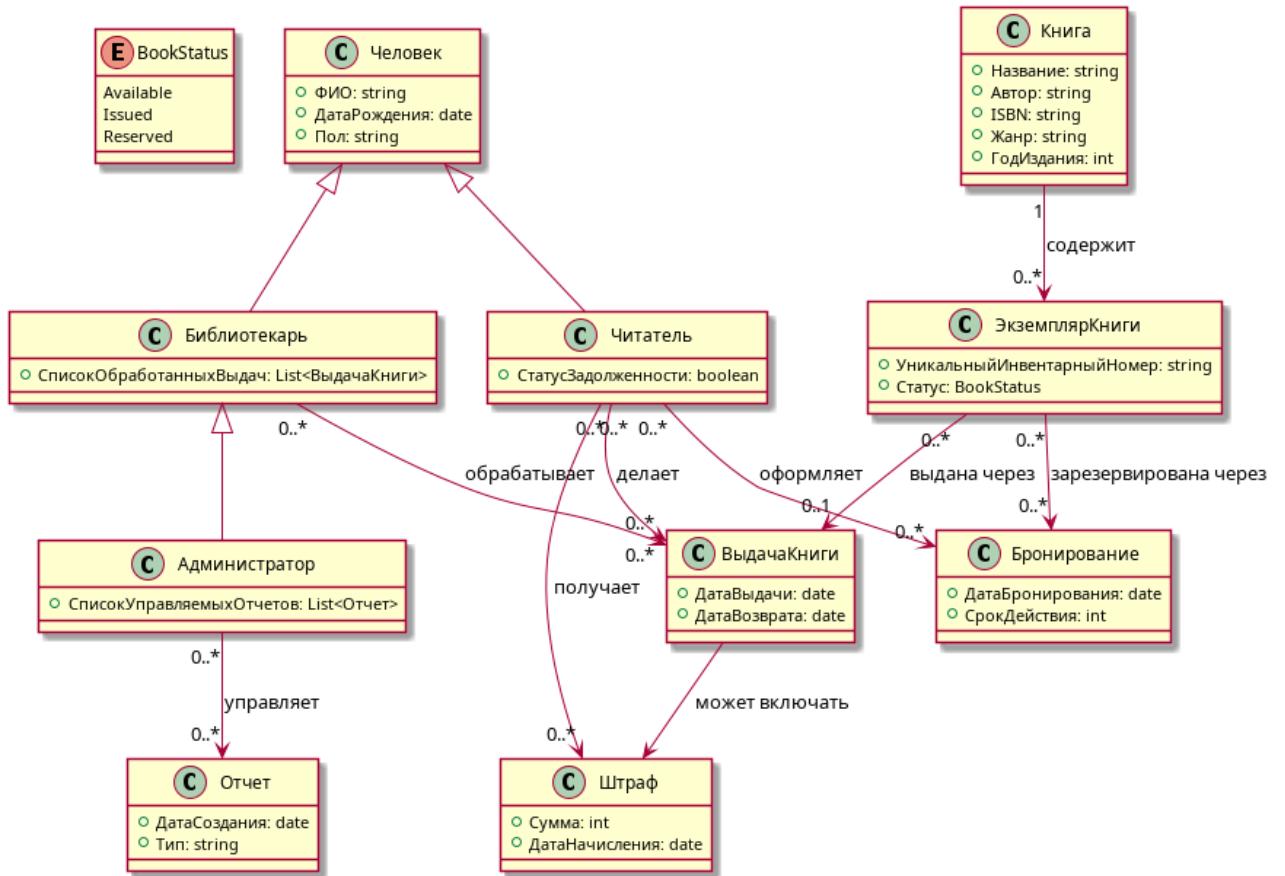
- Работоспособность кода:** Низкая (требует отладки).
- Отсутствие ошибок:** НЕ соответствует (**Syntax Error** в кратности).
- Соответствие логике UML:** Соответствует.
- Сложность и стиль:** Средний.
- Соответствие содержанию ТЗ:** Полное.

### 2.4.2. Модель: DeepSeek



- Анализ:** Продемонстрировала профессиональный уровень, использовав макросы !define и группировку наследования. Допустила такие ошибки: два класса администратора и отчёта, использовала непонятное отношение вместо обычной ассоциации (у штрафа и выдачи книг), применила стрелки вместо линий.
- Выводы:**
  - Работоспособность кода:** Высокая.
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Частичное.
  - Сложность и стиль:** Очень высокий (профессиональные макросы).
  - Соответствие содержанию ТЗ:** Полное.

#### 2.4.3. Модель: GPT5-mini



- Анализ:** Сгенерировала почти **идеальный код**. Верно определила enum BookStatus, прописала типы данных и расставила кратности. Опять же использовала стрелки вместо линий.
- Выводы:**
  - Работоспособность кода:** Высокая.
  - Отсутствие ошибок:** Соответствует.
  - Соответствие логике UML:** Почти полное.
  - Сложность и стиль:** Высокий.
  - Соответствие содержанию ТЗ:** Полное.

## Общий вывод и заключение по лабораторной работе

В ходе лабораторной работы были проведены эксперименты по генерации UML-диаграмм (варианты использования и диаграммы классов) с использованием различных моделей (Алиса, DeepSeek, GPT5-mini).



## Сводная таблица по всем экспериментам (A и B)

Часть	Промпт	Модель	Работоспособность	Ошибки	Соответствие UML	Соответствие ТЗ
A1	Минимальный Алиса	DeepSeek	Низкая	Syntax Error, неверное наследование	Нет	Полное
			Низкая	Ошибка рендера	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
	Базовый	Алиса	Низкая	Повторяющиеся Syntax Error	Нет	Нет
		DeepSeek	Средняя	Перегруженность, отсутствие include	Частично	Полное
		GPT5-mini	Высокая	Нет стрелки к регистрации	Частично	Полное
A3	Расширенный Алиса	DeepSeek	Средняя	Ошибки в границах системы	Частично	Неполное
			Высокая	Неверное слово вместо include	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
	Продвинутый	Алиса	Высокая	Ошибки в наследовании и extend	Частично	Полное
A4	Продвинутый	DeepSeek	Высокая	Ошибка направления extend	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
B1	Минимальный Алиса	Низкая	Нет	Нет	Нет	Полное

		DeepSe ek	Средняя	Перегруженность, лишние связи	Частично	Частично
		GPT5-mini	Высокая	Нет	Полное	Полное
B2	Базовый	Алиса	Низкая	Ошибки в атрибутах	Нет	Полное
		DeepSe ek	Средняя	Ошибки в связях	Частично	Частично
B3	Расширенный	Алиса	Высокая	Мелкие недочёты	Почти полное	Полное
		GPT5-mini	Средняя	Ошибки в наследовании	Частично	Частично
B4	Продвинутый	Алиса	Высокая	Ошибки в типах связей	Частично	Полное
		DeepSe ek	Средняя	Нет	Полное	Полное
		GPT5-mini	Высокая	Ошибки в типах данных, enum, кратностях	Частично	Частично
		DeepSe ek	Высокая	Ошибки в кратностях	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное

## Основные результаты:

- **Алиса (Yandex)** показала низкую надёжность: частые синтаксические ошибки, неверное наследование и некорректные связи. Работоспособность низкая или средняя, соответствие UML и ТЗ слабое.
- **DeepSeek** продемонстрировала среднюю–высокую работоспособность: код компилируется, но требует ручных правок. Основные ошибки связаны с перегруженностью диаграмм, неверными кратностями и путаницей в связях (include, extend, композиция/агрегация). UML частично соответствует, ТЗ чаще всего выполнено.

- **GPT5-mini** показала наилучшие результаты: чистый, рабочий код, корректное наследование, связи и атрибуты. Работоспособность высокая, UML и T3 соответствуют полностью. Мелкие недочёты встречались редко и легко исправлялись.

## Выводы:

1. Для генерации UML-диаграмм **наиболее надёжной моделью является GPT5-mini**, так как она обеспечивает стабильный и корректный результат.
2. **DeepSeek** может использоваться как дополнительный инструмент, но требует внимательной проверки и доработки.
3. **Алиса (Yandex)** в текущем виде непригодна для автоматической генерации UML-диаграмм без значительных ручных исправлений.

## Рекомендации:

- Для учебных и исследовательских целей рекомендуется использовать **GPT5-mini** как основную модель, но тщательно следить за корректностью кода.
- При работе с **DeepSeek** необходимо закладывать время на ручную корректировку диаграмм.
- Использование **Алисы** целесообразно только в качестве вспомогательного инструмента для идей, но не для готового кода.
- В дальнейшем стоит расширить эксперименты на другие типы UML-диаграмм (например, диаграммы последовательностей, активности), чтобы проверить устойчивость моделей в более сложных сценариях.

Таким образом, лабораторная работа достигла своей цели: проведено сравнение моделей, выявлены их сильные и слабые стороны, и сформированы практические рекомендации по выбору инструмента для генерации UML-диаграмм.

# **Лабораторная работа: Генерация диаграмм UML**

## **Цель работы**

Разработать и проанализировать диаграммы UML для информационной системы по выбору студента.

## **Задание**

### **1. Часть 1. Диаграмма вариантов использования**

- Выбрать предметную область для проектирования информационной системы
- Определить основных действующих лиц системы
- Выявить основные функциональные требования
- Построить диаграмму вариантов использования с учетом следующих требований:
  - Отобразить все действующие лица
  - Определить основные и вспомогательные варианты использования
  - Установить отношения между вариантами использования (include, extend)
  - Добавить границу системы
  - Обеспечить логическую целостность диаграммы

### **2. Часть 2. Диаграмма классов**

- На основе выбранной предметной области разработать диаграмму классов
- Определить основные классы системы
- Установить иерархию наследования
- Определить типы связей между классами
- Указать атрибуты классов с типами данных
- Добавить кратности ассоциаций
- Провести исследование для 1-2 моделей ИИ

## **Требования к выполнению**

- Использовать PlantUML для генерации диаграмм
- Соблюдать стандарты UML
- Обеспечить логическую целостность диаграмм
- Предоставить комментарии к ключевым элементам диаграмм
- Проверить работоспособность сгенерированного кода

## **Критерии оценки**

- Корректность синтаксиса PlantUML
- Соответствие стандартам UML
- Полнота отображения функциональных требований
- Качество визуализации
- Логическая целостность диаграмм
- Обоснованность выбора предметной области

## **Сроки выполнения**

- Дедлайн: 29.11

## **Форма отчёта**

1. Исходный код PlantUML для обеих диаграмм
2. Сгенерированные диаграммы
3. Пояснительная записка с описанием:
  1. Выбранной предметной области
  2. Обоснования выбора действующих лиц
  3. Функциональных требований
  4. Структуры классов
4. Анализ результатов проектирования