

# Генерация диаграмм UML

Подготовили: Зинкин Станислав, Гордейко Никита, Усманов Игорь

# План доклада

- Цель и задачи
- PlantUML
- Диаграммы UML и нейросети
- Пример программной системы
- Диаграмма вариантов использования
- Диаграмма классов
- Результаты
- Вывод

# Цель и задачи

## Цель:

Проведение сравнительного анализа эффективности различных промптов и нейросетей для генерации диаграмм UML

## Задачи:

- Выбор диаграмм для дальнейшей генерации
- Разработка промптов различного уровня
- Выбор нейросетей для сравнения
- Генерация диаграмм UML, используя выбранные нейросети и созданные промпты
- Анализ полученных результатов

# PlantUML

PlantUML —  
инструмент с открытым  
исходным кодом,  
который позволяет  
создавать диаграммы  
на основе текста.

```
1 @startuml
2 left to right direction
3
4 * Действующие лица (акторы)
5 actor Читатель as reader
6 actor Библиотекарь as librarian
7 actor Администратор as admin
8
9 * Варианты использования (use cases)
10 usecase "Зарегистрироваться" as uc_register
11 usecase "Авторизоваться" as uc_login
12 usecase "Поиск книг" as uc_search
13 usecase "Просмотр деталей книги" as uc_view_details
14 usecase "Бронирование книги" as uc_reserve
15 usecase "Взять книгу" as uc_borrow
16 usecase "Вернуть книгу" as uc_return
17 usecase "Продлить срок" as uc_extend
18 usecase "Просмотр личных данных" as uc_viewprofile
19 usecase "Управление читателями" as uc_manage_readers
20 usecase "Управление книгами" as uc_manage_books
21 usecase "Добавление книги" as uc_add_book
22 usecase "Редактирование книги" as uc_edit_book
23 usecase "Удаление книги" as uc_delete_book
24 usecase "Формирование отчётов" as uc_reports
25
26 * Связи акторов с вариантами использования
27 reader --> uc_register
28 reader --> uc_login
29 reader --> uc_search
30 reader --> uc_view_details
31 reader --> uc_reserve
32 reader --> uc_borrow
33 reader --> uc_return
34 reader --> uc_extend
35 reader --> uc_viewprofile
36
37 librarian --> uc_login
38 librarian --> uc_search
39 librarian --> uc_view_details
40 librarian --> uc_borrow
41 librarian --> uc_return
42 librarian --> uc_extend
43 librarian --> uc_manage_readers
44 librarian --> uc_manage_books
45
46 admin --> uc_login
47 admin --> uc_manage_readers
48 admin --> uc_add_book
49 admin --> uc_edit_book
50 admin --> uc_delete_book
51 admin --> uc_reports
52
53 * Правильные связи <<include>>
54 uc_login .> uc_register <<include>>
55 uc_search .> uc_view_details <<include>>
56
57 * Графические настройки
58 skinparam packageStyle rectangle
59 skinparam useCase {
60     BackgroundColor<<admin>> LightBlue
61     BackgroundColor<<librarian>> LightGreen
62     BackgroundColor<<reader>> LightYellow
63 }
64
65 @enduml
```

# Диаграммы UML и нейросети

В качестве первого эксперимента мы будем строить диаграмму вариантов использования (**Use Case Diagram**).

Во втором эксперименте – диаграмму классов (**Class Diagram**).

Используемые нейросети:

1. Alice AI LLM
2. DeepSeek
3. GPT-5 mini

# Пример программной системы

- Система управления библиотекой
- 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:

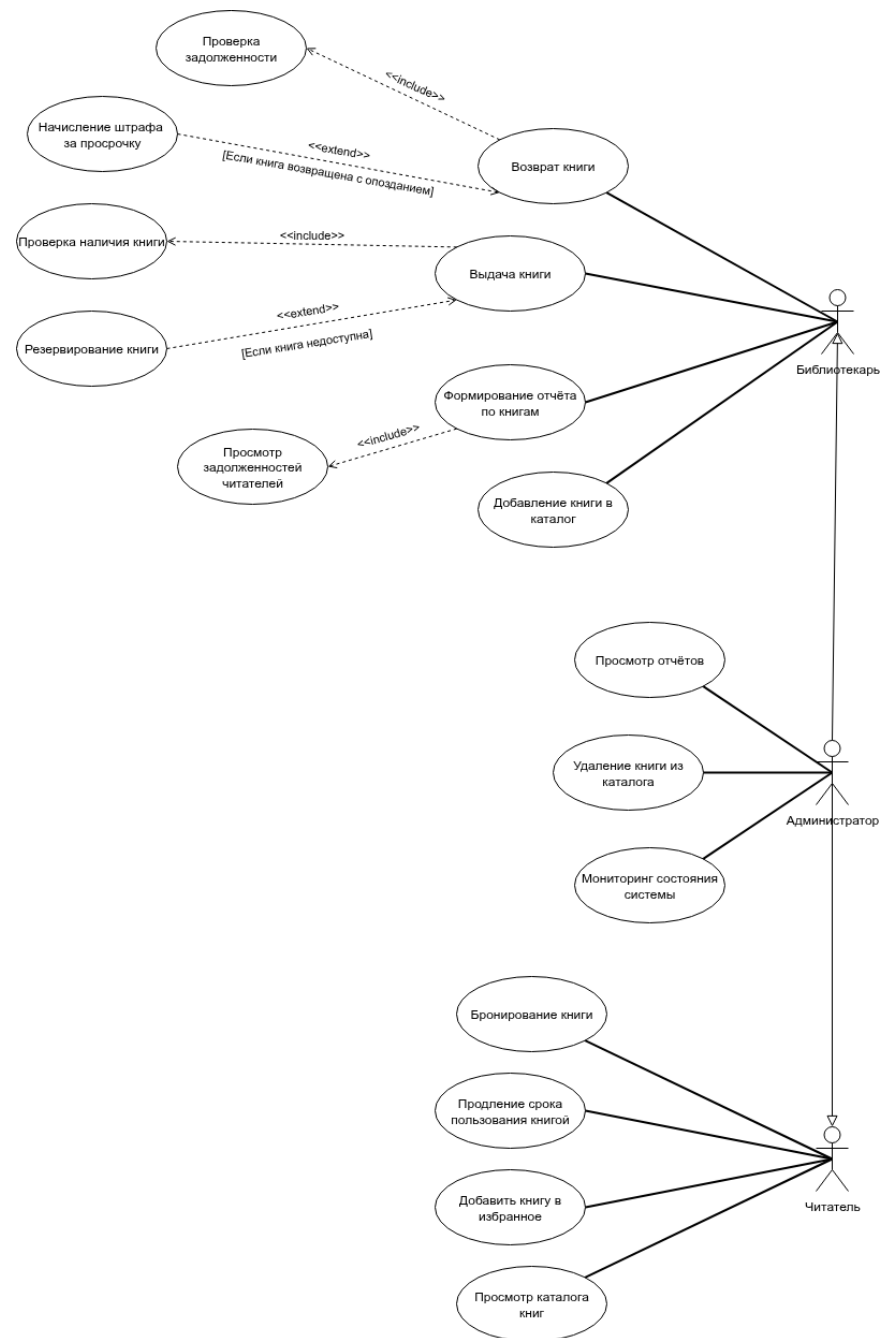
- 1) выдавать книги
- 2) возвращать книги
- 3) формировать отчёт по книгам
- 4) добавлять книги в каталог

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

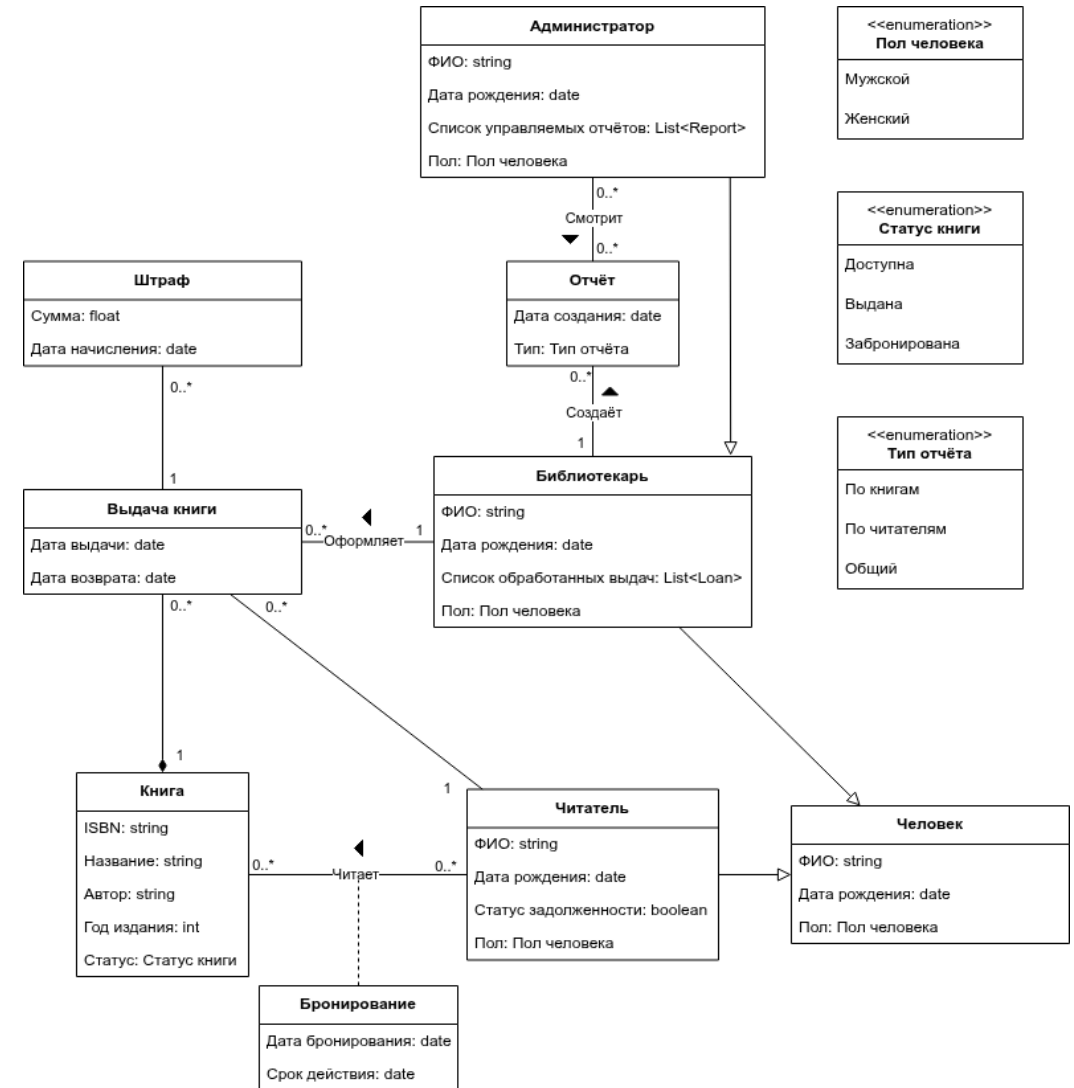
- 1) просматривать отчёты
- 2) удалять книги из каталога
- 3) мониторить состояние системы

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

# Пример диаграммы вариантов использования



# Пример диаграммы классов





# Диаграмма вариантов использования (1/4)

## **Промпт 1 (минимальный):**

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

# Диаграмма вариантов использования (1/4)

## Alice AI LLM

Первая попытка: `SyntaxError`

Уточнение:

“Получил ошибку `SyntaxError` на строке `uc_login .>. uc_register : <>`”

Вторая попытка:

**Работоспособность кода:**

Низкая (требуется 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.



# Диаграмма вариантов использования (1/4)

## DeepSeek

Первая попытка: `SyntaxError`

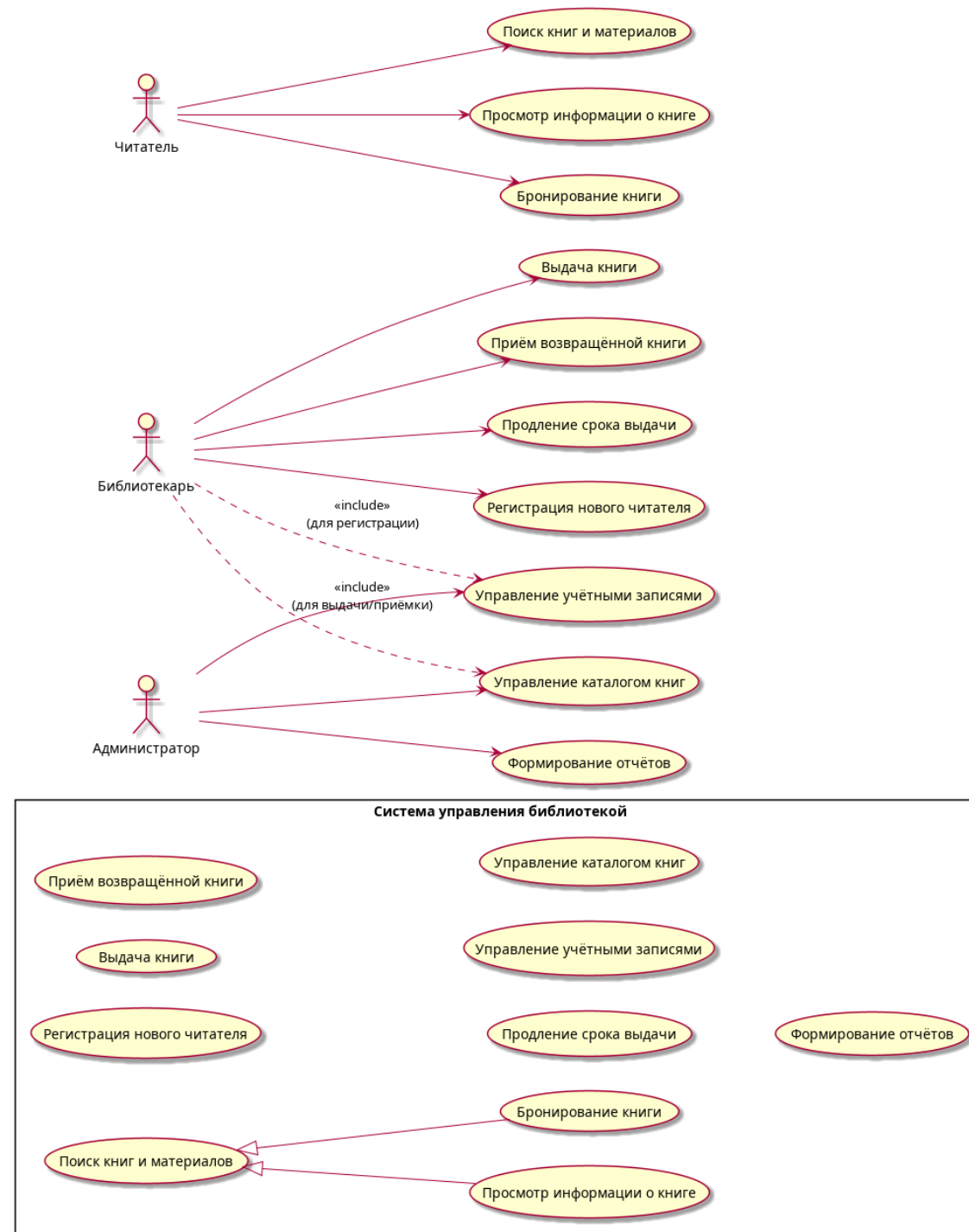
Уточнение: "Строка `!theme plain` вызвала `SyntaxError`, перепиши код"

Вторая попытка: далее

Уточнение: "На твоей диаграмме граница системы включает далеко не все варианты использования (некоторые находятся снаружи), а также ты применяешь наследование не к действующим лицам, а к вариантам использования, что очень странно«

Третья попытка: далее

# Результат второй попытки



# Результат третьей попытки

## 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.



# Диаграмма вариантов использования (1/4)

## GPT-5 mini

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

Соответствует.

**Сложность и стиль:**

Базовый (чистый код).

**Соответствие содержанию ТЗ:**

Соответствует.



# Диаграмма вариантов использования (2/4)

## Промпт 2 (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.



# Диаграмма вариантов использования (2/4)

## Alice AI LLM

Первая попытка: `SyntaxError`

Уточнение: "Получил `SyntaxError` на строке `use case` (Просмотр каталога) as UC1"

Вторая попытка: `SyntaxError`

Уточнение: "Получил `SyntaxError` на строке `actor` Администратор как Администратор"

Третья попытка: `SyntaxError`

Уточнение: "Получил `SyntaxError` на строке `use case` "Просмотр каталога"

Четвертая попытка: `SyntaxError`

# Диаграмма вариантов использования (2/4)

## DeerSeek

Первая попытка: далее

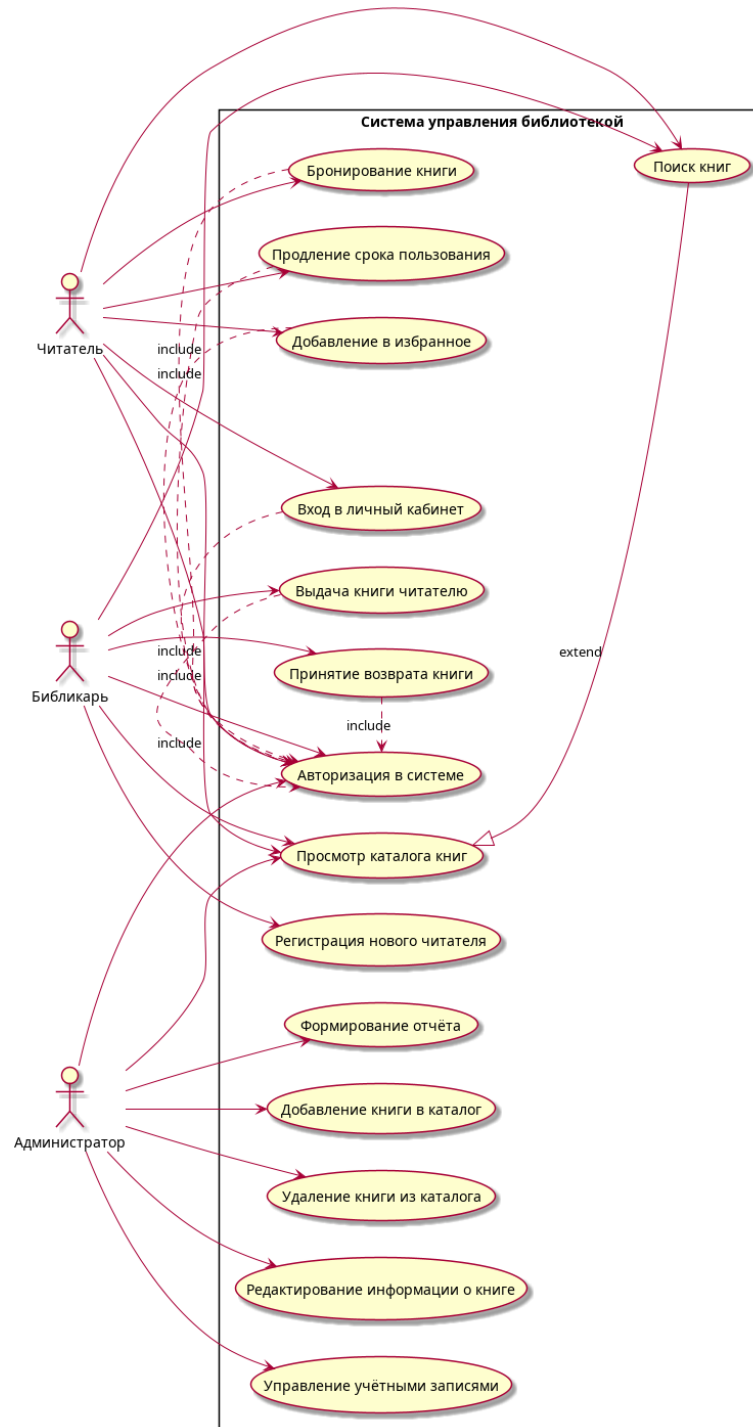
Уточнение: "Получилась очень хаотичная диаграмма, в которой сложно разобрать, куда и какие стрелочки ведут, нужно переписать"

Вторая попытка: далее

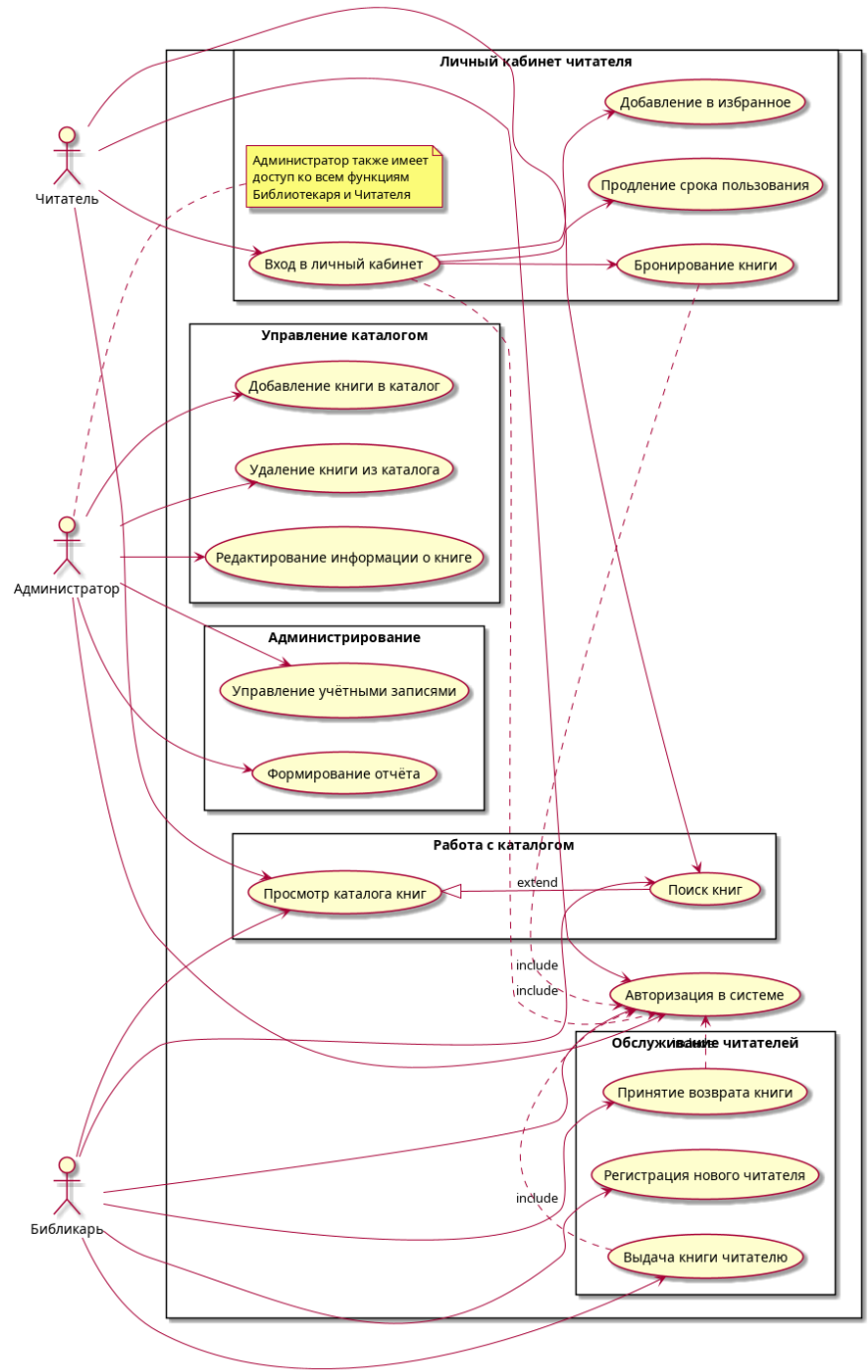
Уточнение: "Уже получше, но всё равно слишком хаотично. Давай оставим одну единственную границу системы. Также у тебя довольно странные include и extend«

Третья попытка: далее

# Результат первой попытки



# Результат второй попытки



# Результат третьей попытки

## Работоспособность кода:

Средняя (рабочий, но нечитаемый результат).

## Отсутствие ошибок:

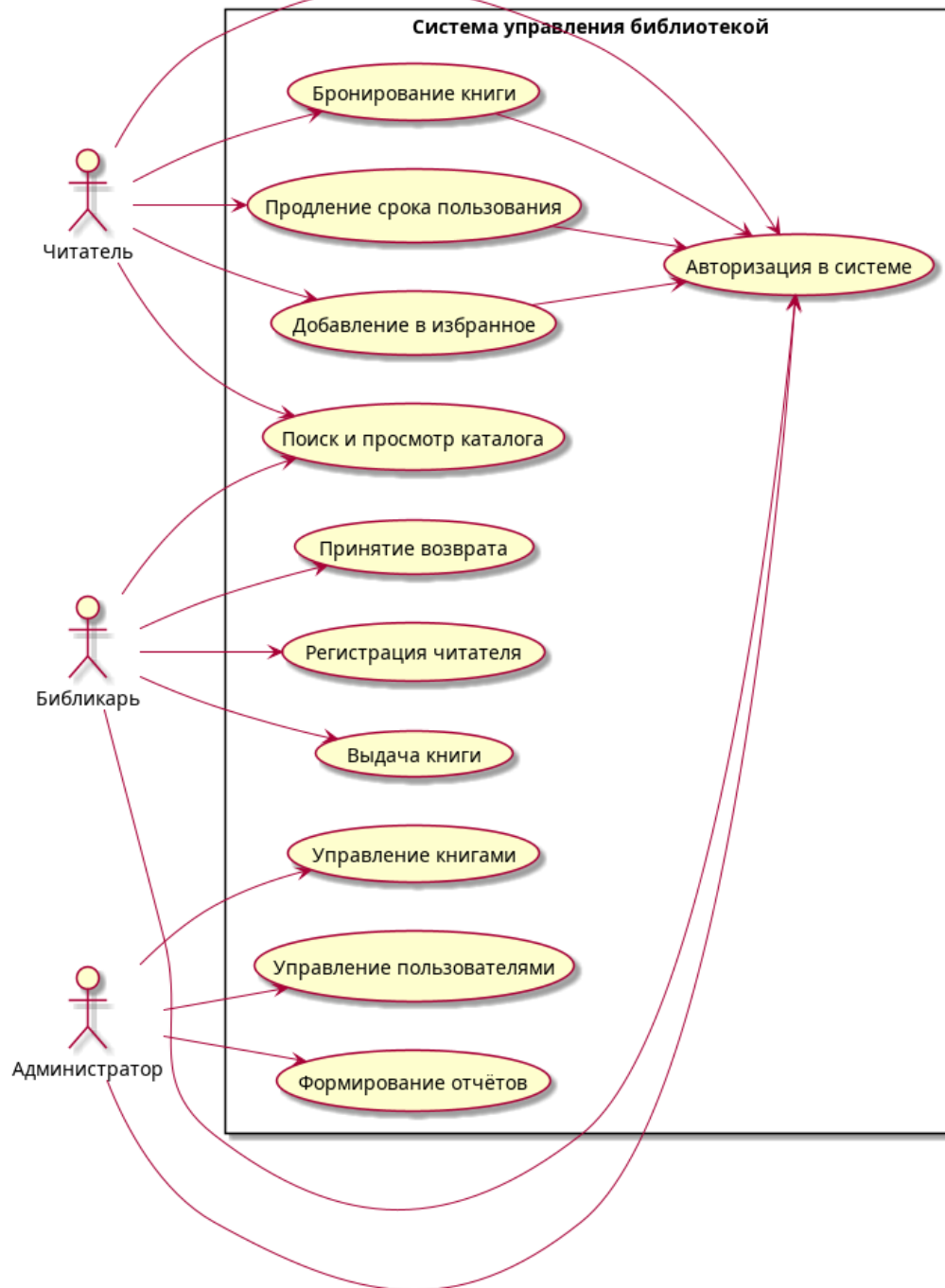
Соответствует.

**Соответствие логике UML:** Соответствует.

## Сложность и стиль:

Низкое качество визуализации.

**Соответствие содержанию ТЗ:** Полное.



# Диаграмма вариантов использования (2/4)

## GPT-5 mini

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

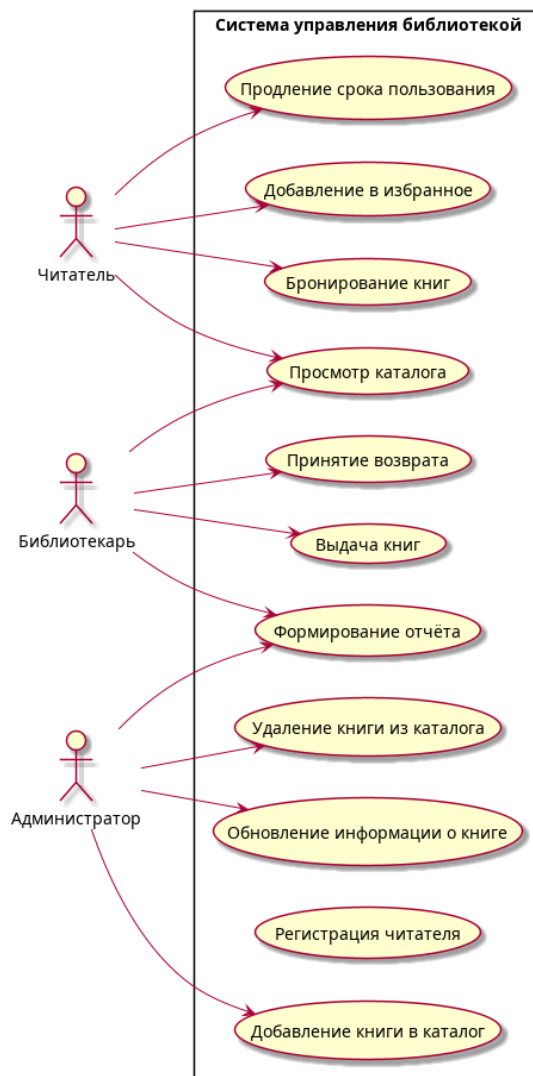
Частично соответствует (нет стрелочки к регистрации читателя).

**Сложность и стиль:**

Соответствует.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма вариантов использования (3/4)

## Промпт 3 (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.

# Диаграмма вариантов использования (3/4)

## Alice AI LLM

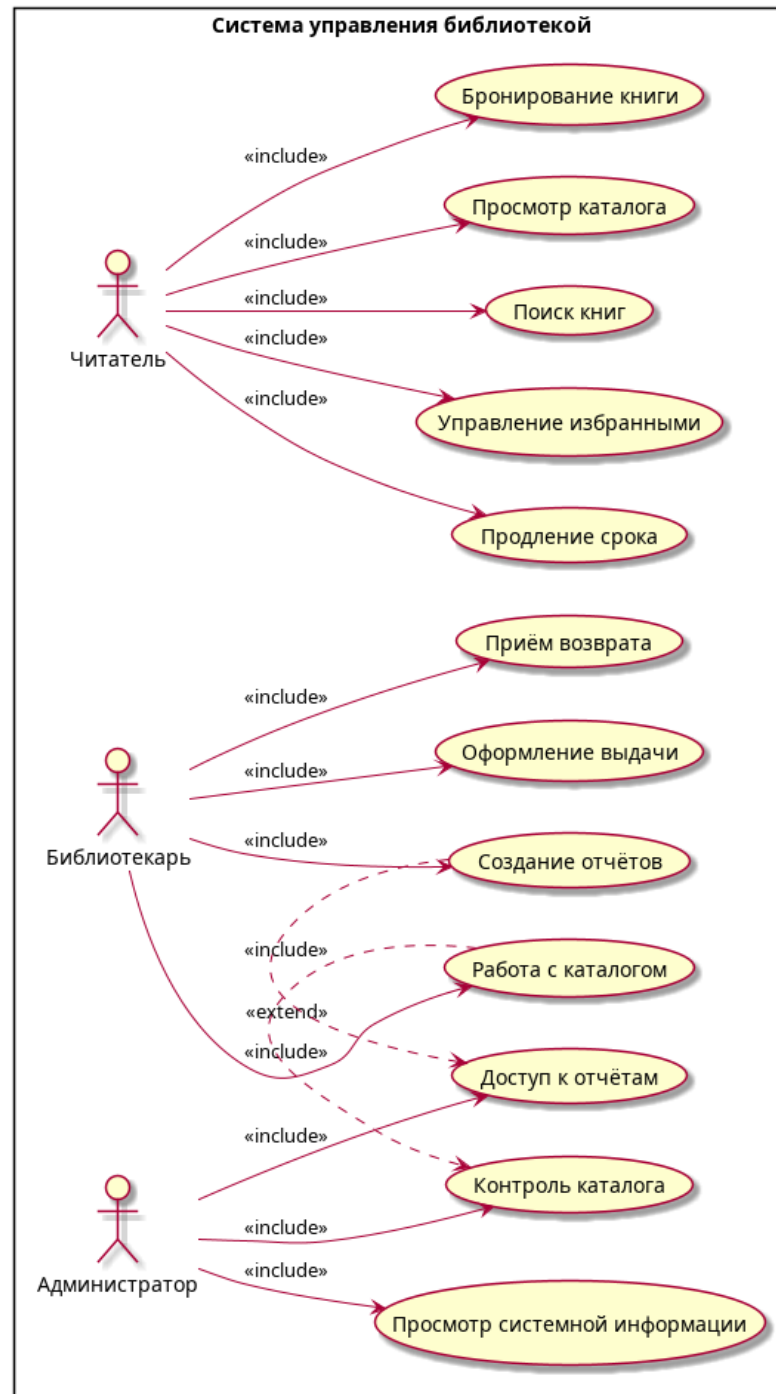
Первая попытка: неверными include, далее

Уточнение: "Ты понимаешь смысл использования include? Include показывает, что один вариант использования всегда включает в себя выполнение другого варианта использования. Include теряет свой смысл, когда ты его используешь от действующего лица к варианту использования. Нужно от варианта использования к другому варианту использования. Перепиши код"

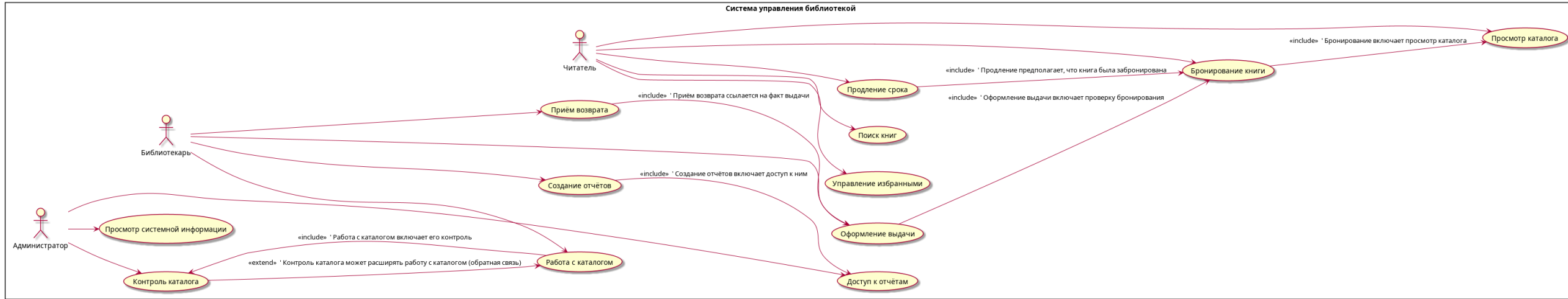
Вторая попытка: далее



# Результат первой попытки



# Результат второй попытки



**Работоспособность кода:** Средняя (требуется логическая правка).

**Отсутствие ошибок:** Соответствует.

**Соответствие логике UML:** Частичное (проблема с границами или группировкой).

**Сложность и стиль:** Соответствует.

**Соответствие содержанию ТЗ:** Неполное (на первом этапе).

# Диаграмма вариантов использования (3/4)

## DeerSeek

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

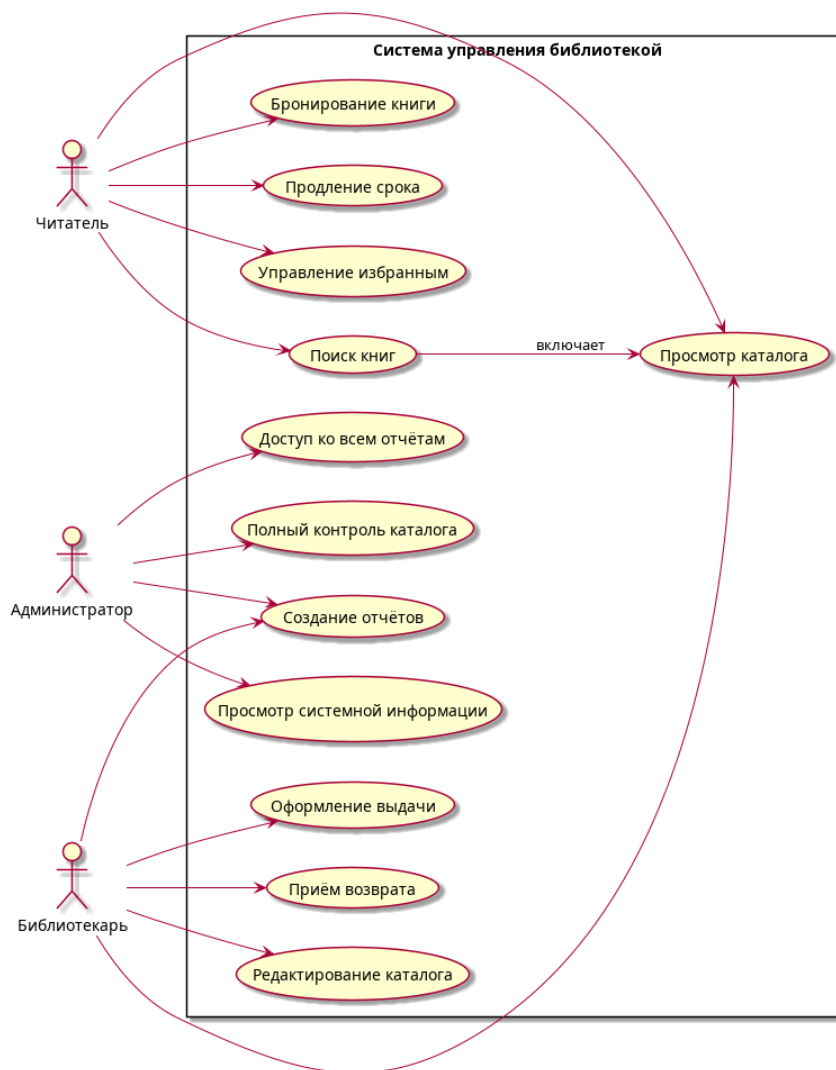
Частичное соответствие.

**Сложность и стиль:**

Соответствует.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма вариантов использования (3/4)

## GPT-5 mini

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

Соответствует.

**Сложность и стиль:**

Высокий.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма вариантов использования (4/4)

## Промпт 4 (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

# Диаграмма вариантов использования (4/4)

## Alice AI LLM

Первая попытка:

Работоспособность кода:  
Высокая.

Отсутствие ошибок:

Соответствует.

Соответствие логике UML:

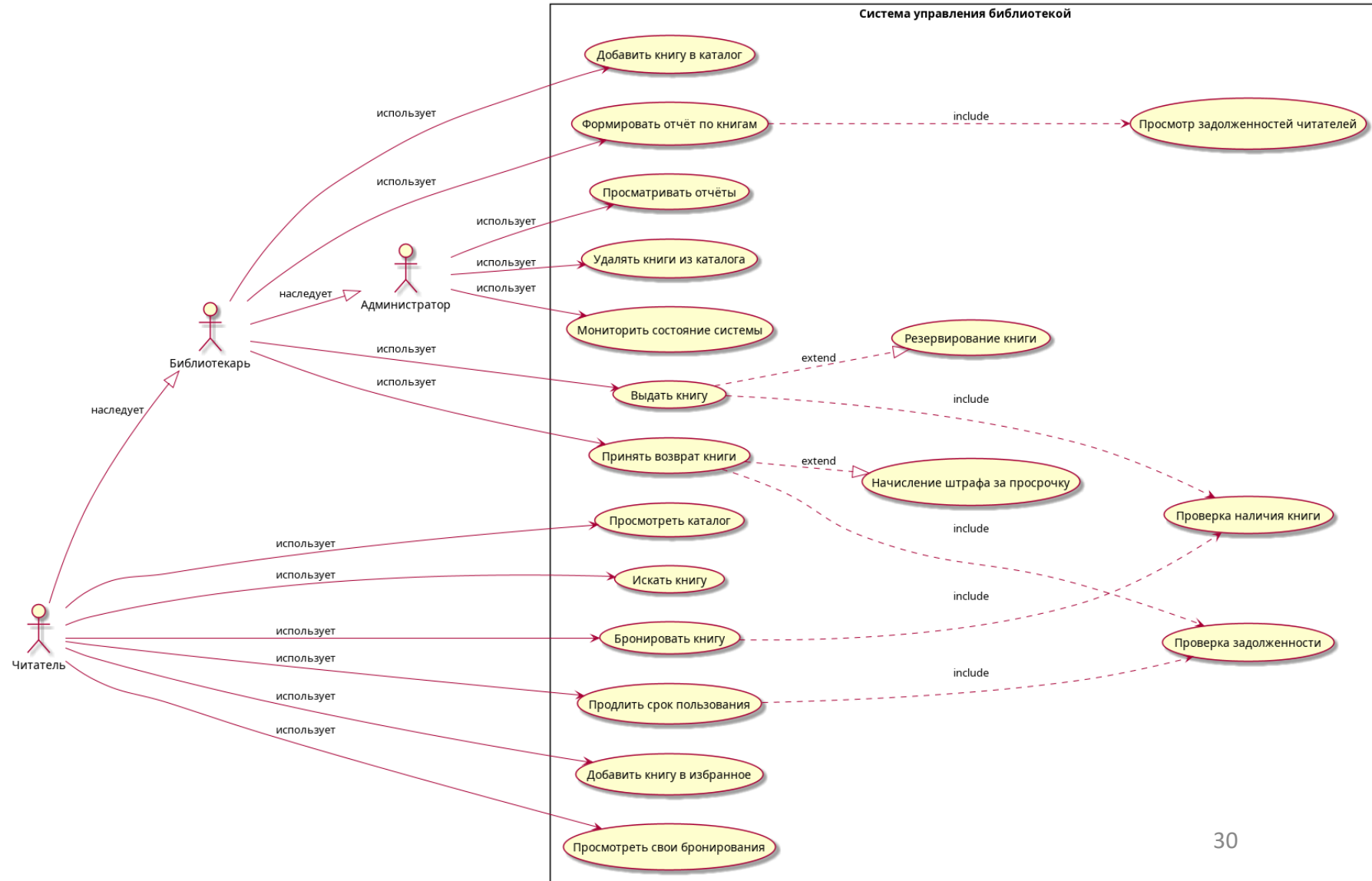
Частичное соответствие.

Сложность и стиль:

Соответствует.

Соответствие содержанию ТЗ:

Полное.



# Диаграмма вариантов использования (4/4)

## DeerSeek

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

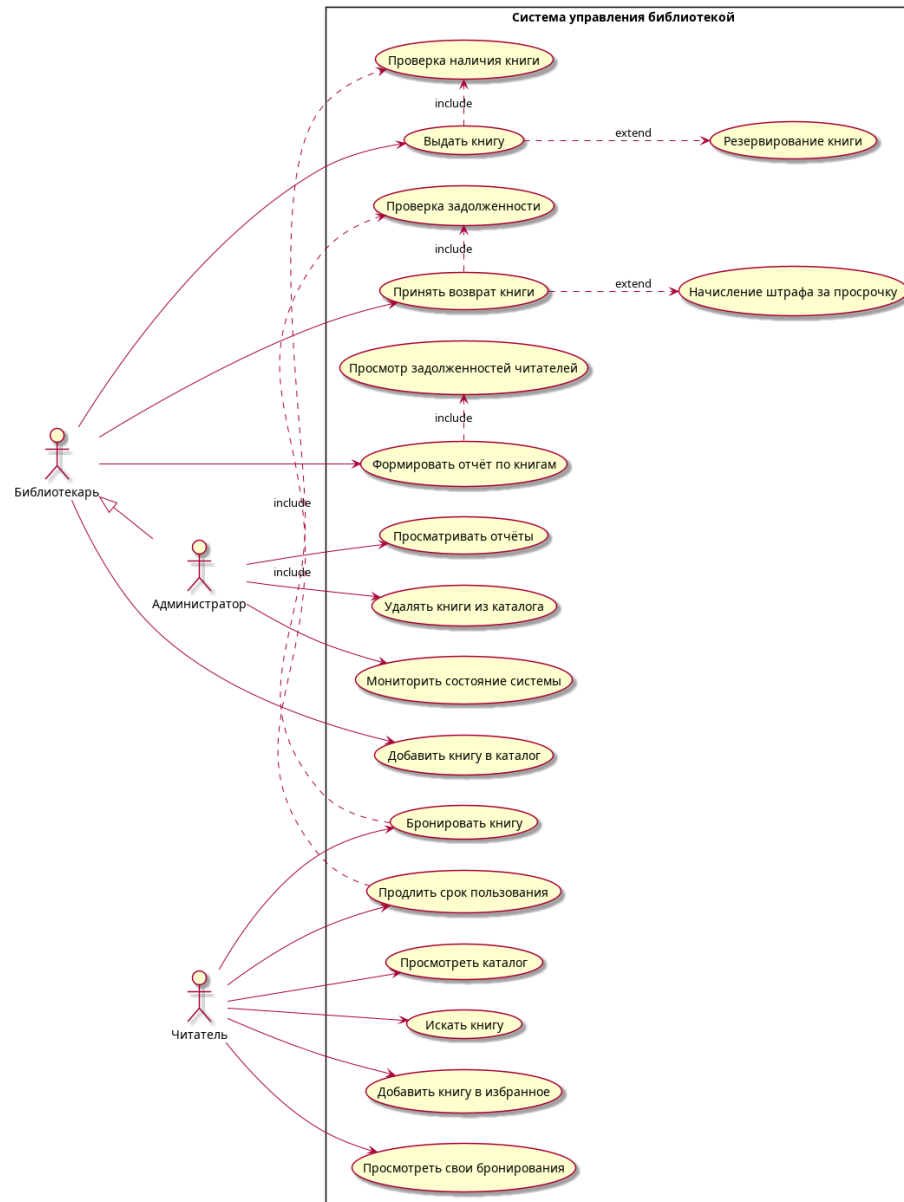
Частичное соответствие.

**Сложность и стиль:**

Высокий (продвинутый синтаксис).

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма вариантов использования (4/4)

## GPT-5 mini

Первая попытка:

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

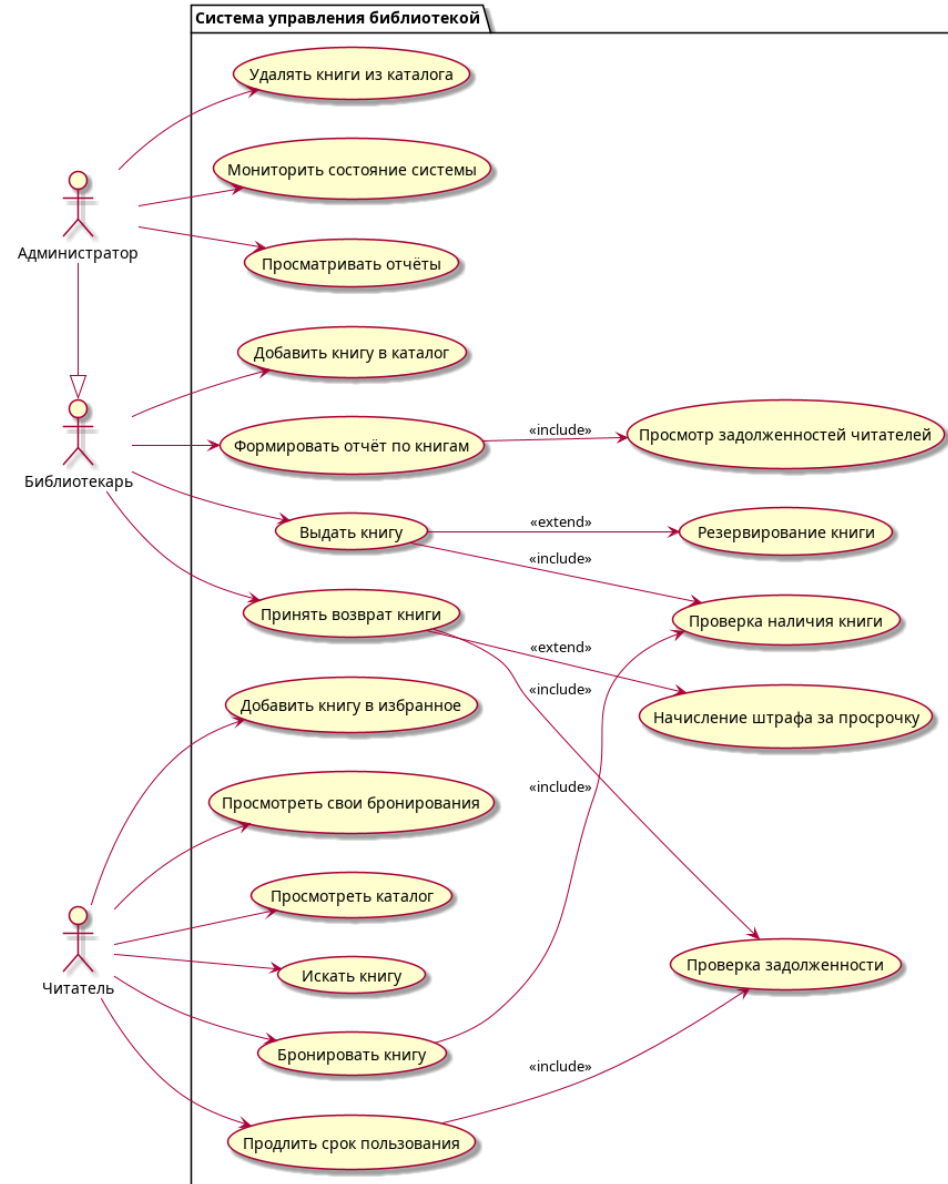
Частичное соответствие.

**Сложность и стиль:**

Высокий.

**Соответствие содержанию ТЗ:**

Полное.





# Диаграмма классов (1/4)

## **Промпт 1 (минимальный):**

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

# Диаграмма классов (1/4)

## Alice AI LLM

Работоспособность кода:

Высокая.

Отсутствие ошибок:

Соответствует.

Соответствие логике UML:

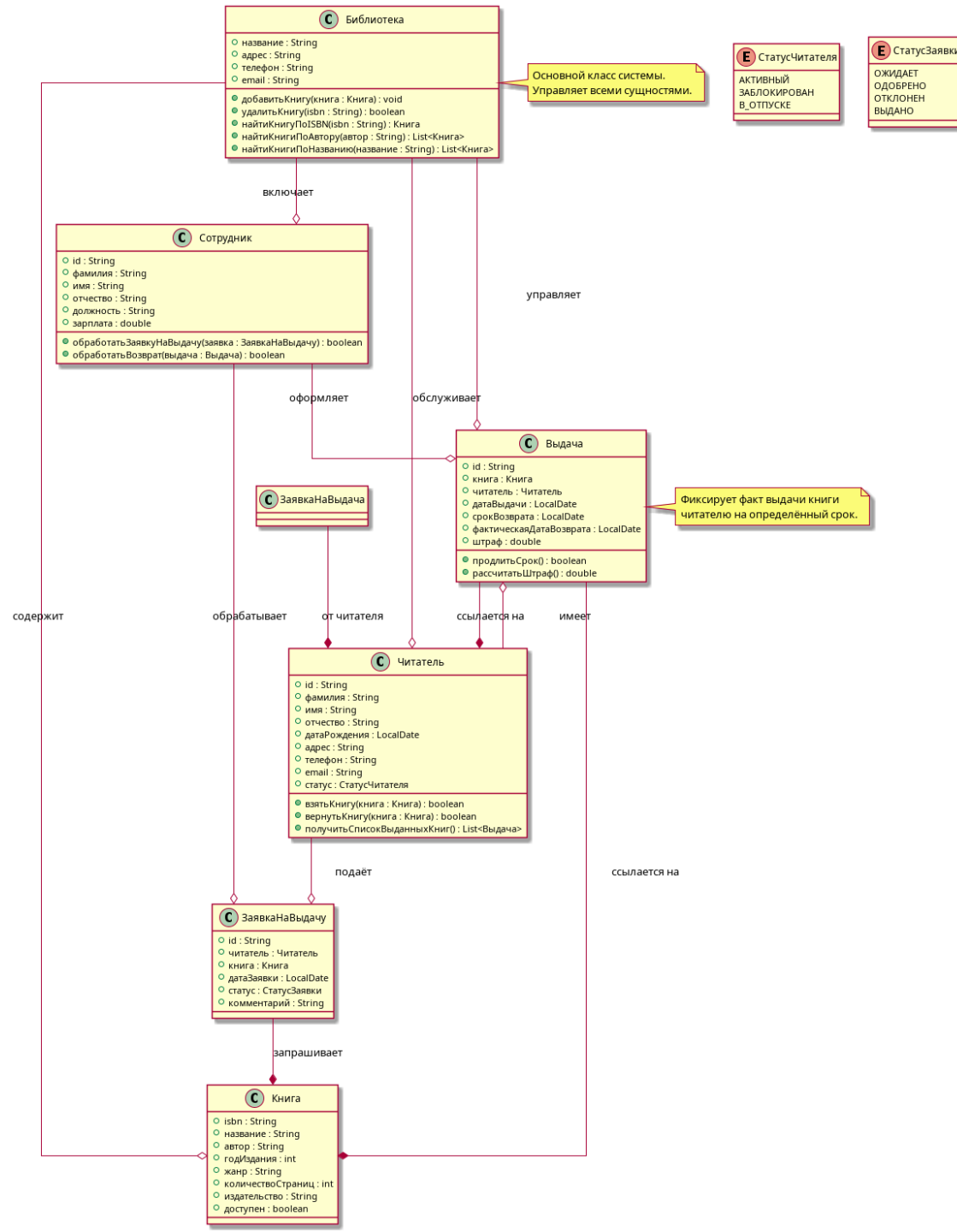
НЕ соответствует.

Сложность и стиль:

Избыточное усложнение.

Соответствие содержанию ТЗ:

Полное.



# Диаграмма классов (1/4)

## DeepSeek

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

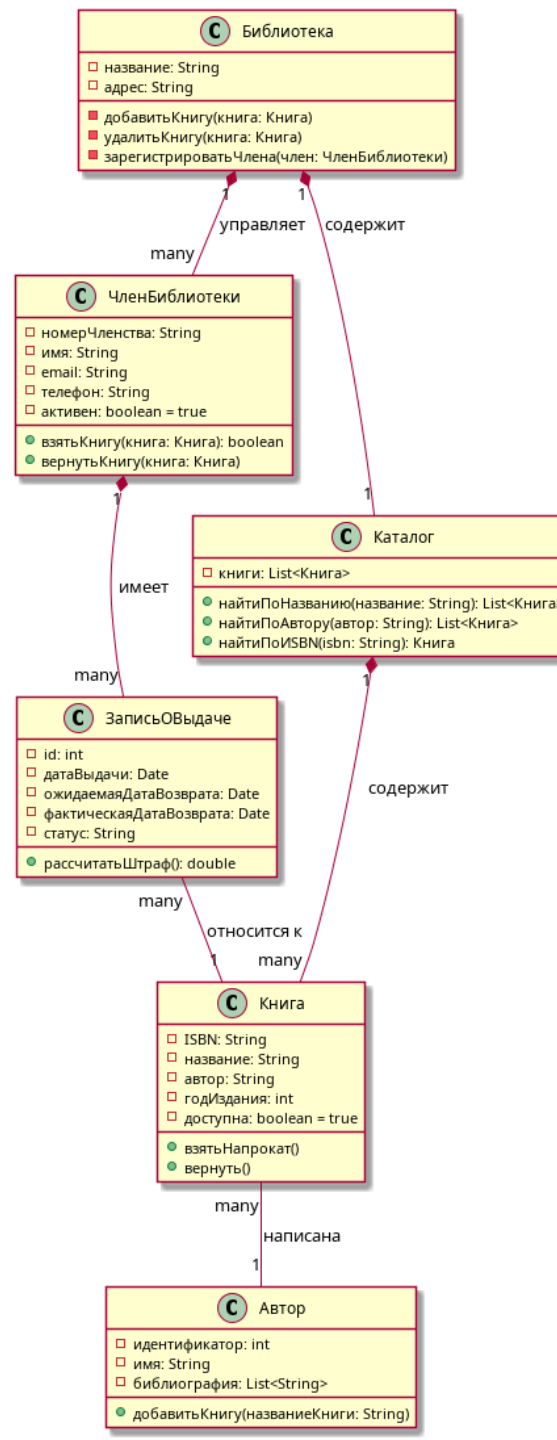
Частично соответствует.

**Сложность и стиль:**

Соответствует.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма классов (1/4)

## GPT-5 mini

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

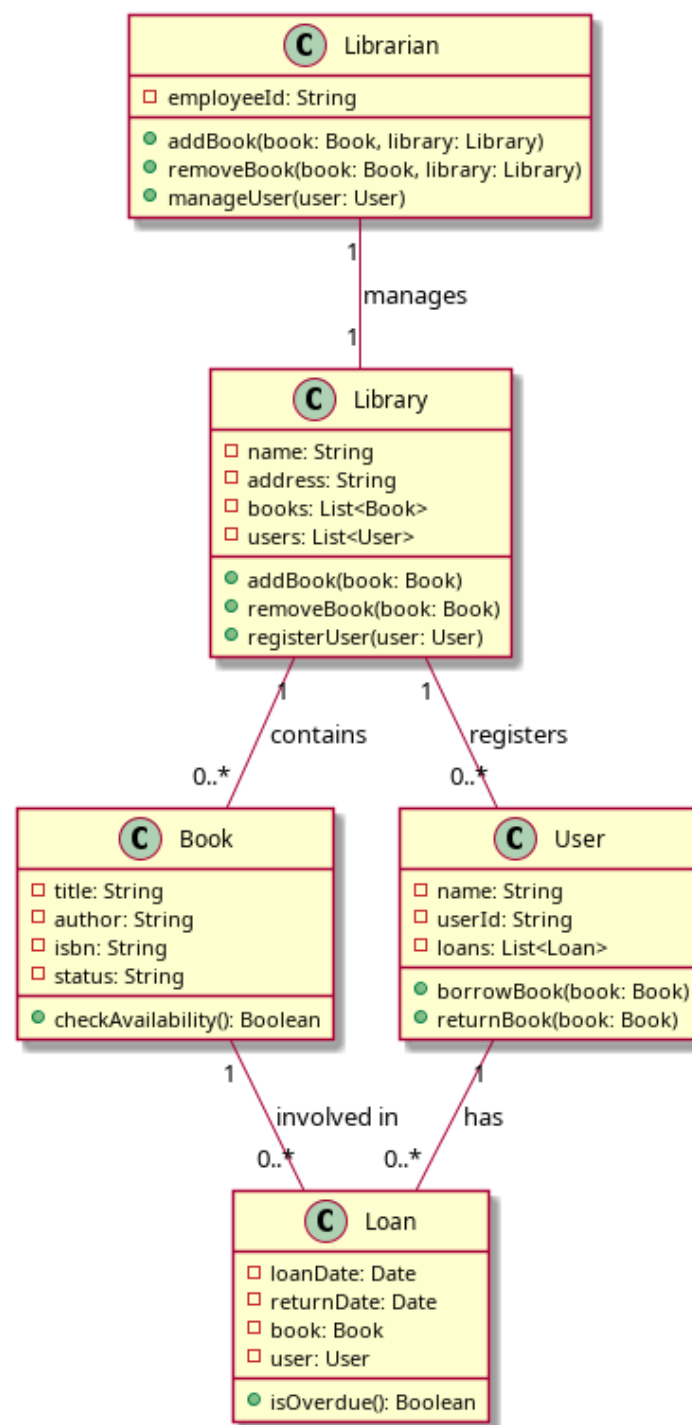
Соответствует.

**Сложность и стиль:**

Соответствует.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма классов (2/4)

## **Промпт 2 (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).

# Диаграмма классов (2/4)

## Alice AI LLM

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

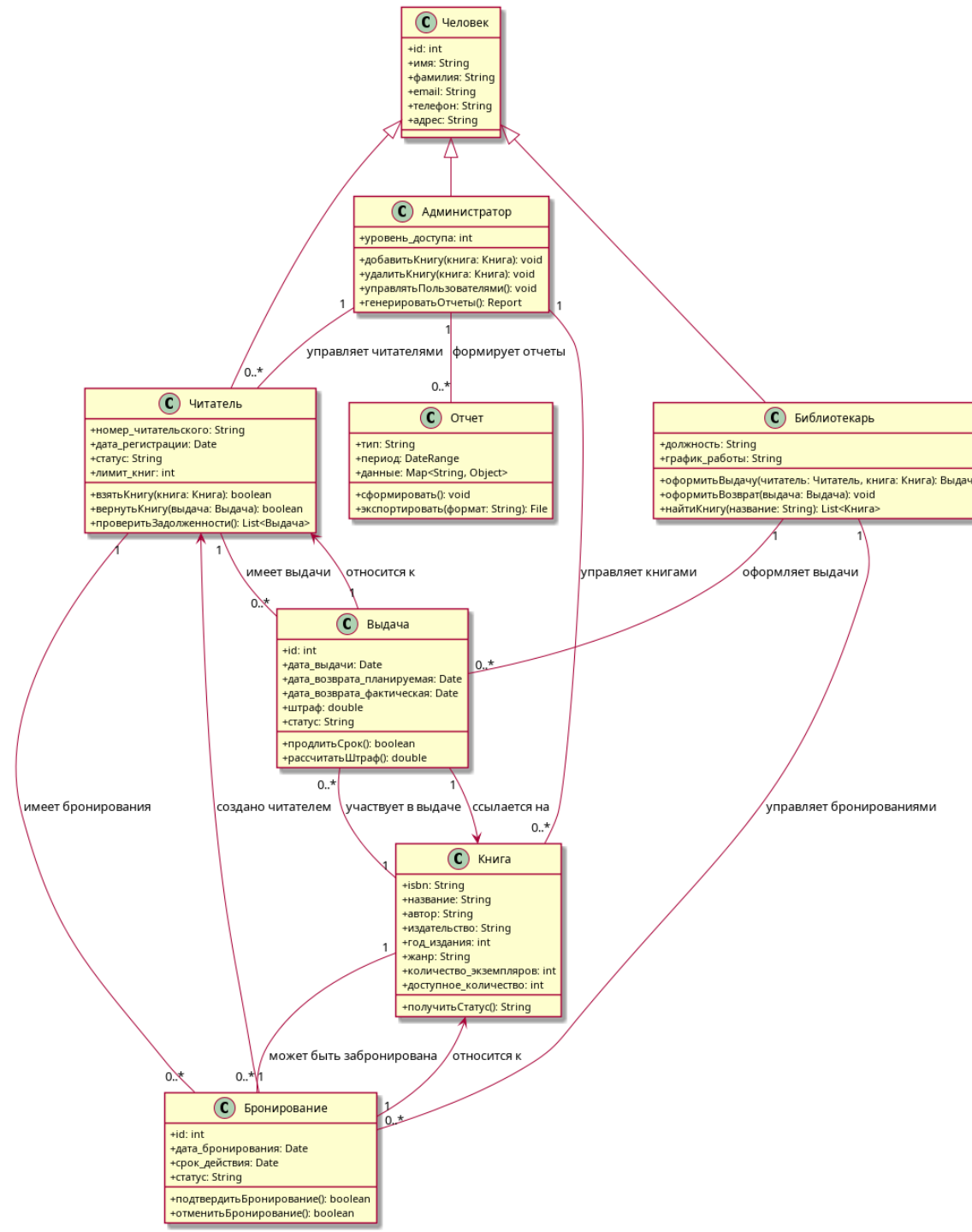
НЕ соответствует.

**Сложность и стиль:**

Базовый.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма классов (2/4)

## DeepSeek

Работоспособность кода:

Высокая.

Отсутствие ошибок:

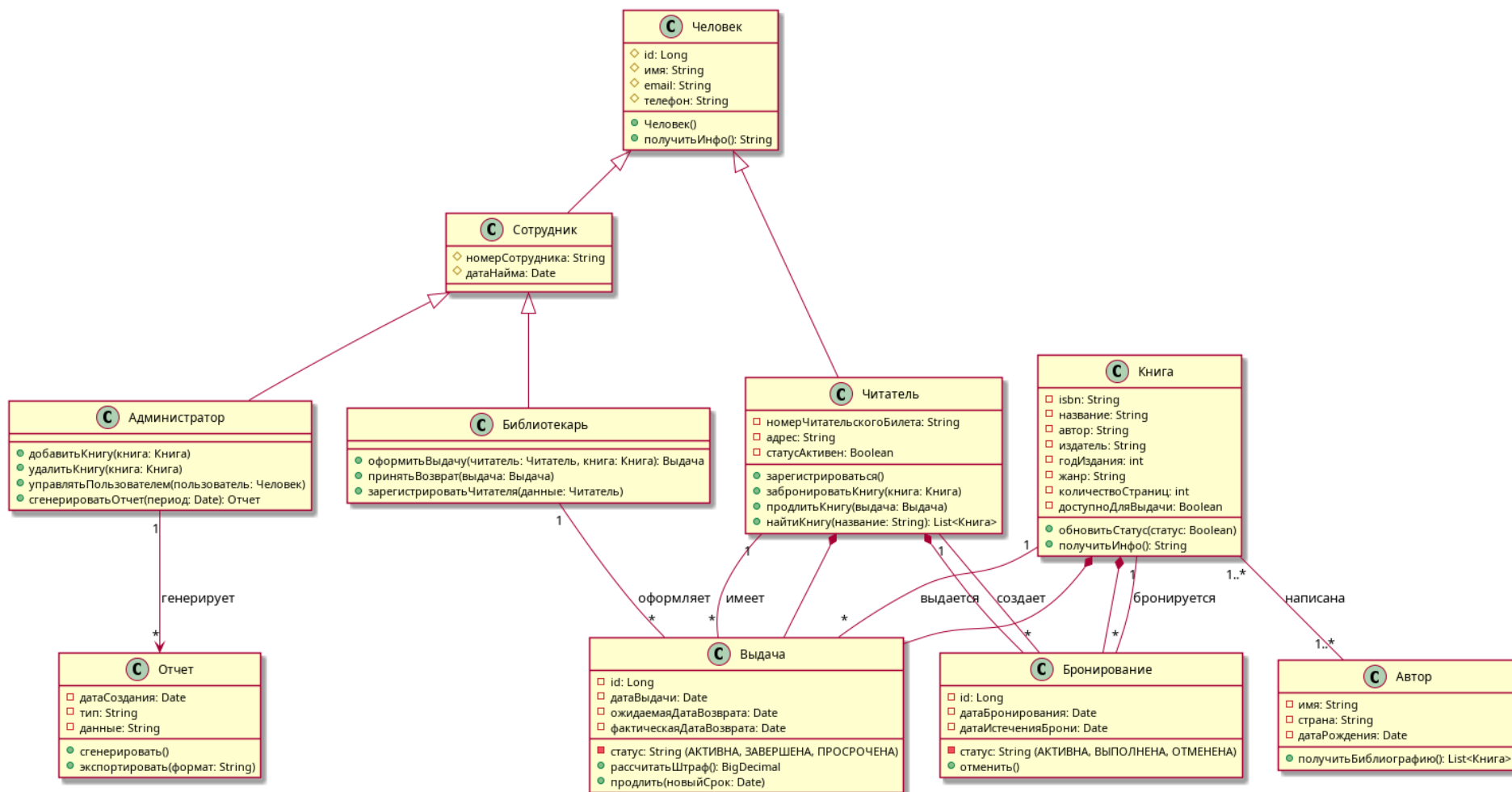
Соответствует.

Соответствие логике UML:

Частично соответствует.

Сложность и стиль:

Высокий (использование принципов ООП).



# Диаграмма классов (2/4)

## GPT-5 mini

Работоспособность кода:

Высокая.

Отсутствие ошибок:

Соответствует.

Соответствие логике UML:

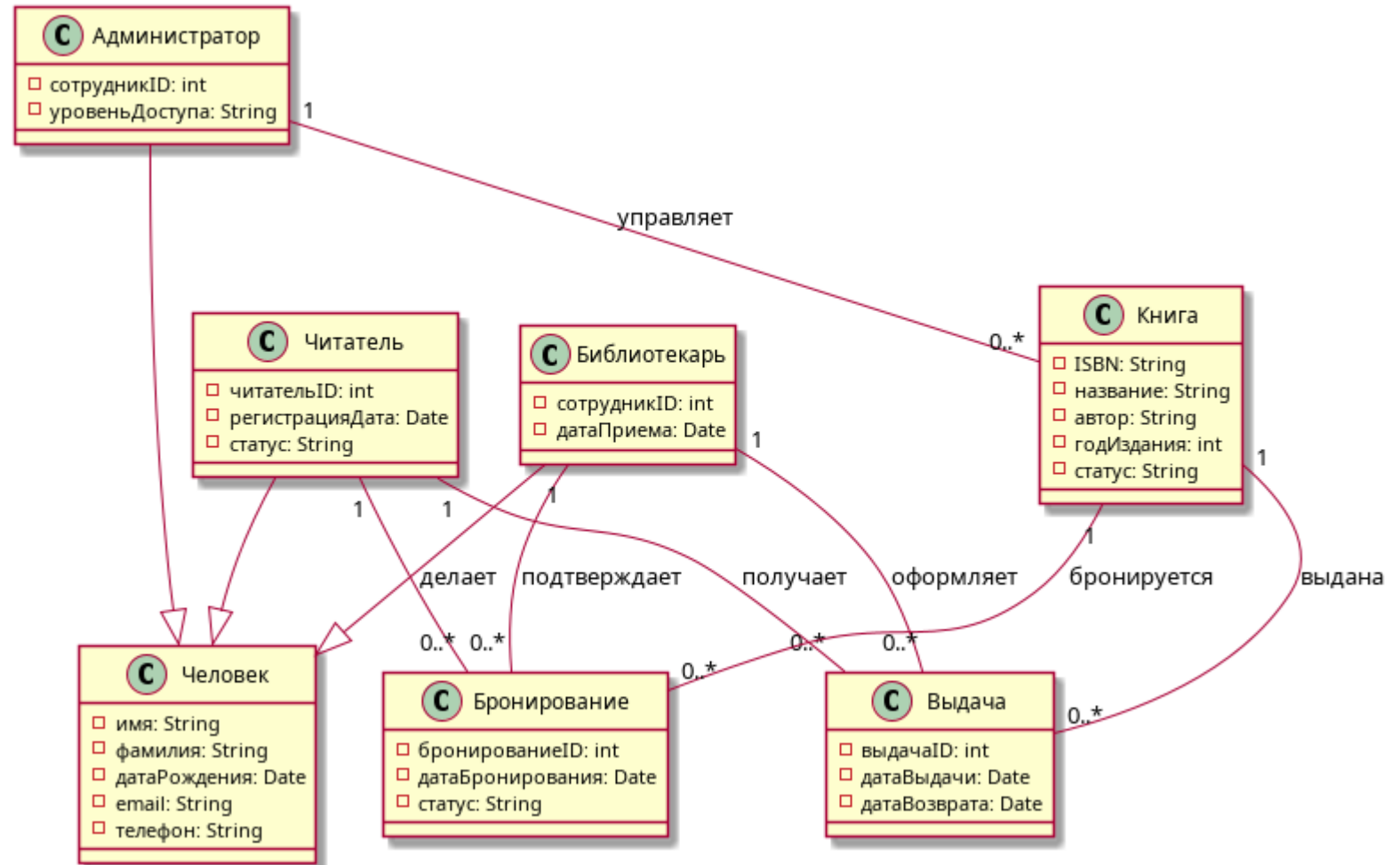
Соответствует.

Сложность и стиль:

Соответствует.

Соответствие содержанию ТЗ:

Полное.





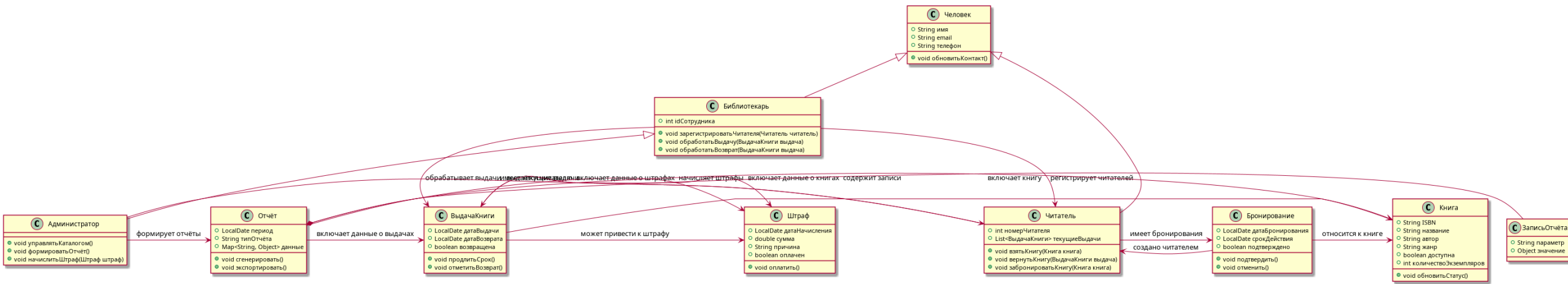
# Диаграмма классов (3/4)

## **Промпт 3 (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).

# Диаграмма классов (3/4)

## Alice AI LLM



**Работоспособность кода:** Высокая.

**Отсутствие ошибок:** Соответствует.

**Соответствие логике UML:** Полное.

**Сложность и стиль:** Визуально хаотичная.

**Соответствие содержанию ТЗ:** Полное.

# Диаграмма классов (3/4)

## DeerSeek

Работоспособность кода:

Высокая.

Отсутствие ошибок:

Соответствует.

Соответствие логике UML:

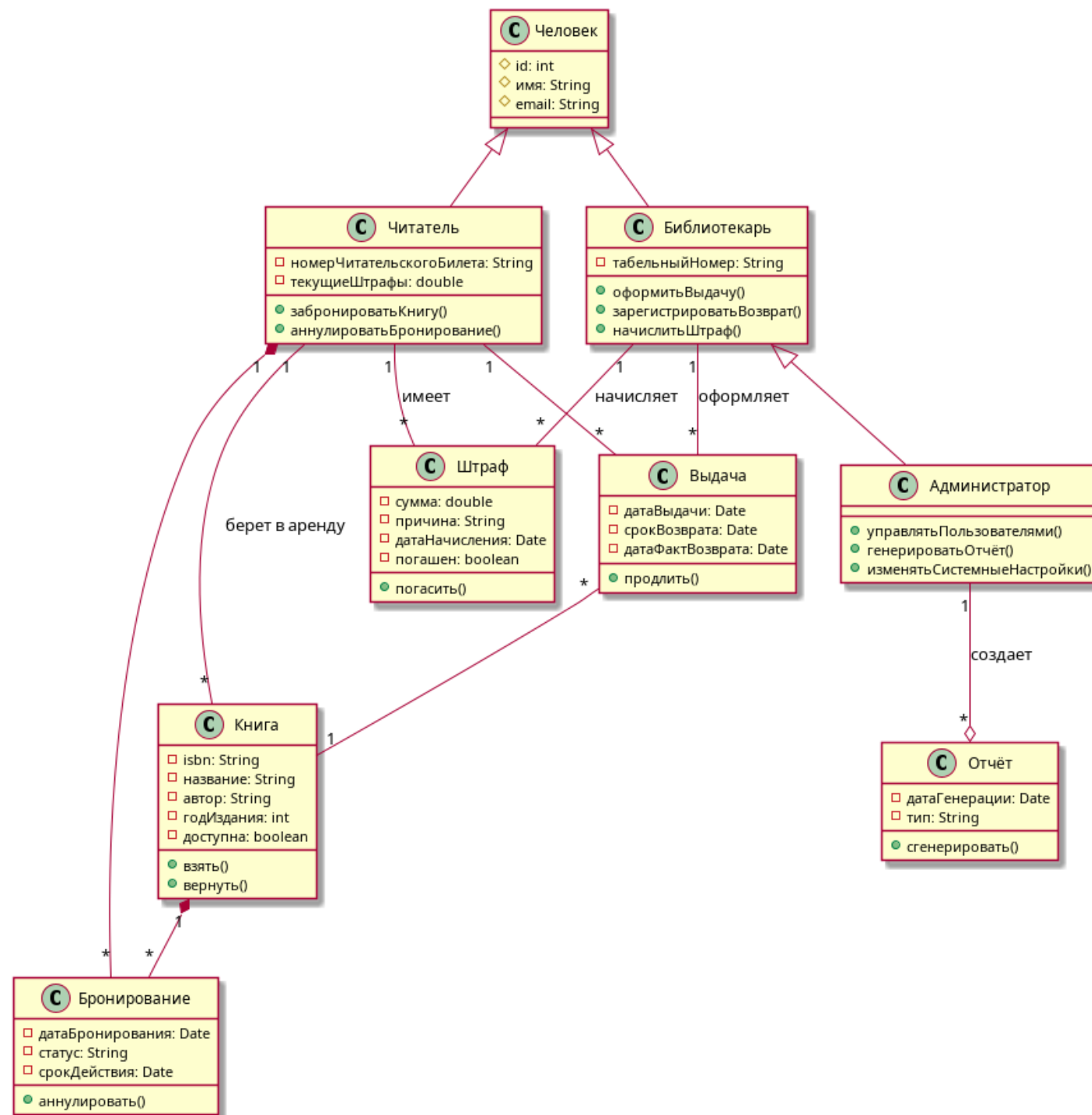
Частичное.

Сложность и стиль:

Высокий.

Соответствие содержанию ТЗ:

Полное.



# Диаграмма классов (3/4)

## GPT-5 mini

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

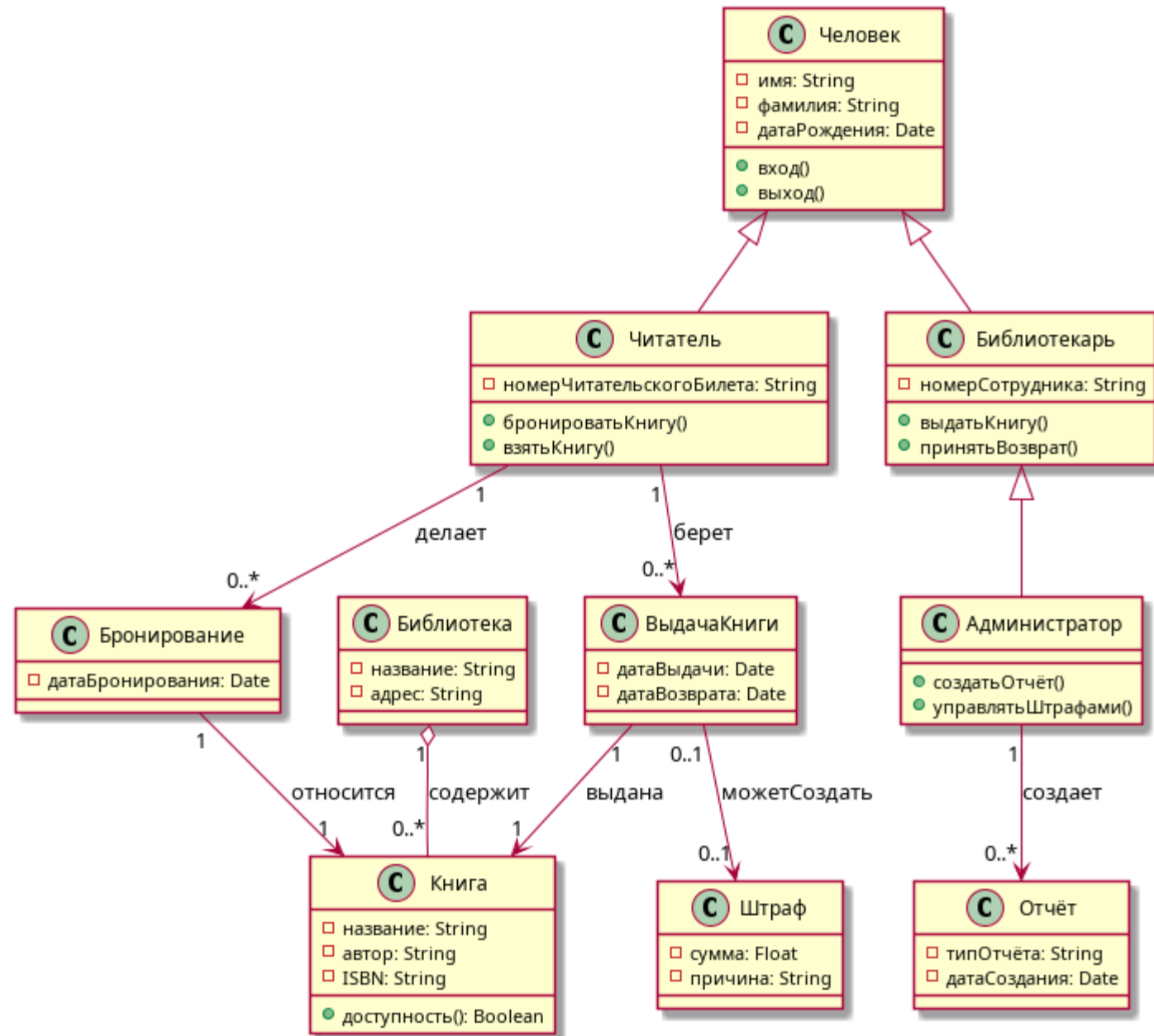
Почти полное.

**Сложность и стиль:**

Высокий.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма классов (4/4)

## Промпт 4 (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.)

# Диаграмма классов (4/4)

## Alice AI LLM

Первая попытка: `SyntaxError`

Уточнение: "SyntaxError на строке Читатель -- "1" --> "0.." ВыдачаКниги : имеет выдачи"

Вторая попытка: далее

# Результат второй попытки

**Работоспособность кода:**

Низкая (требуется отладка).

**Отсутствие ошибок:**

НЕ соответствует (Syntax Error в кратности).

**Соответствие логике UML:**

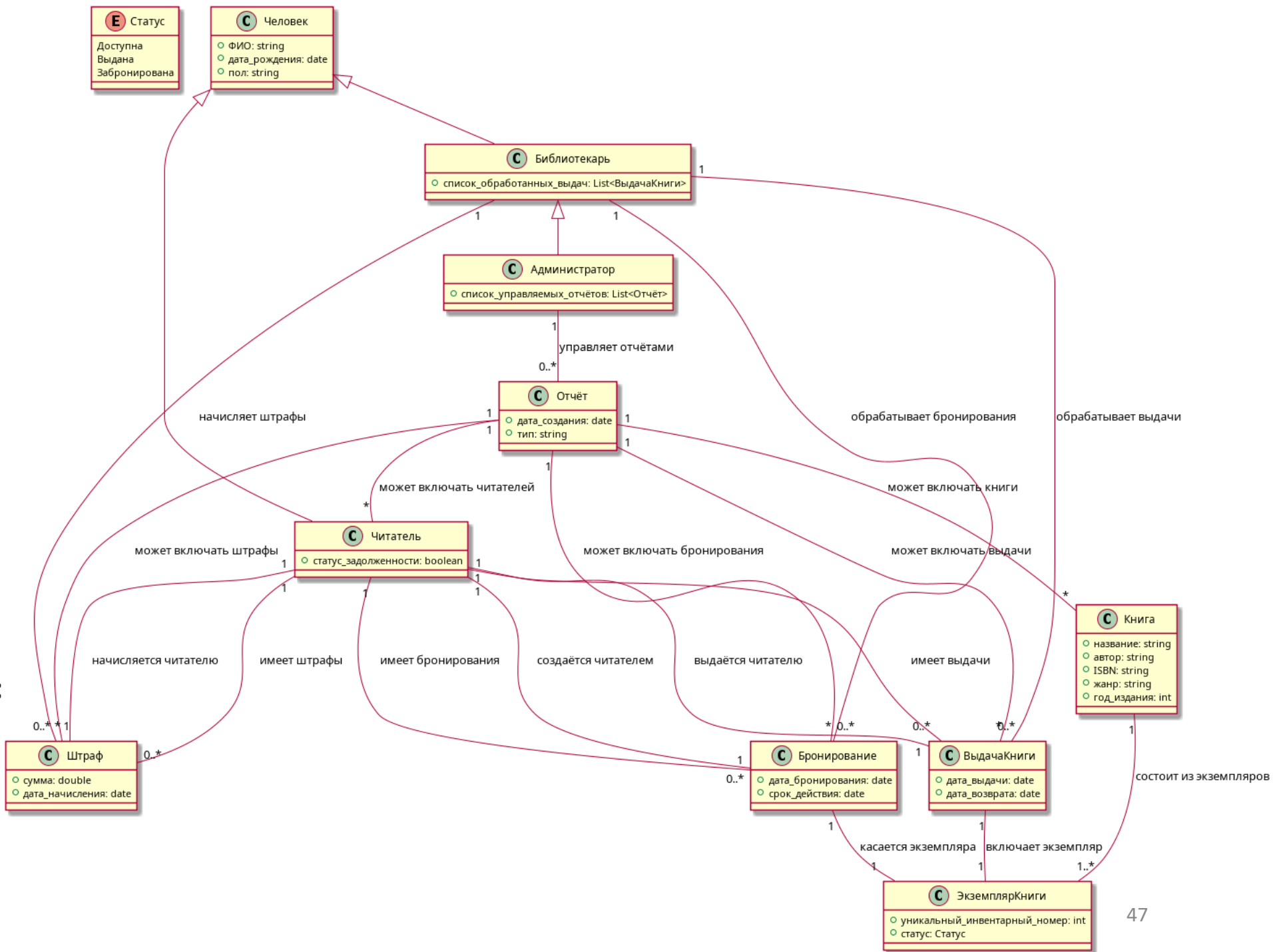
Соответствует.

**Сложность и стиль:**

Средний.

**Соответствие содержанию ТЗ:**

Полное.



# Диаграмма классов (4/4)

## DeepSeek

**Работоспособность кода:**

Высокая.

**Отсутствие ошибок:**

Соответствует.

**Соответствие логике UML:**

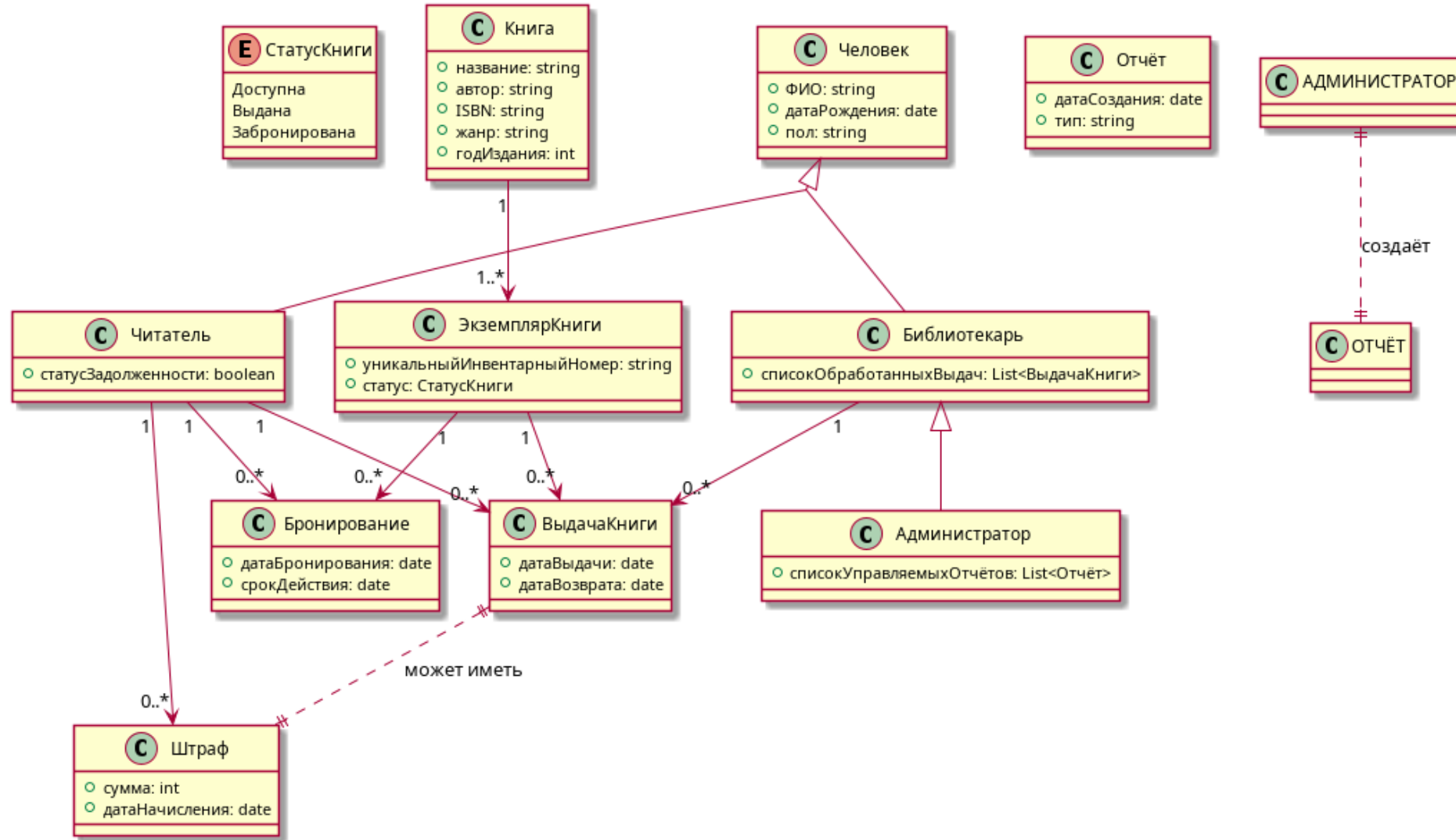
Частичное.

**Сложность и стиль:**

Очень высокий  
(профессиональные макросы).

**Соответствие содержанию ТЗ:**

Полное.





# Диаграмма классов (4/4)

## GPT-5 mini

Работоспособность кода:

Высокая.

Отсутствие ошибок:

Соответствует.

Соответствие логике UML:

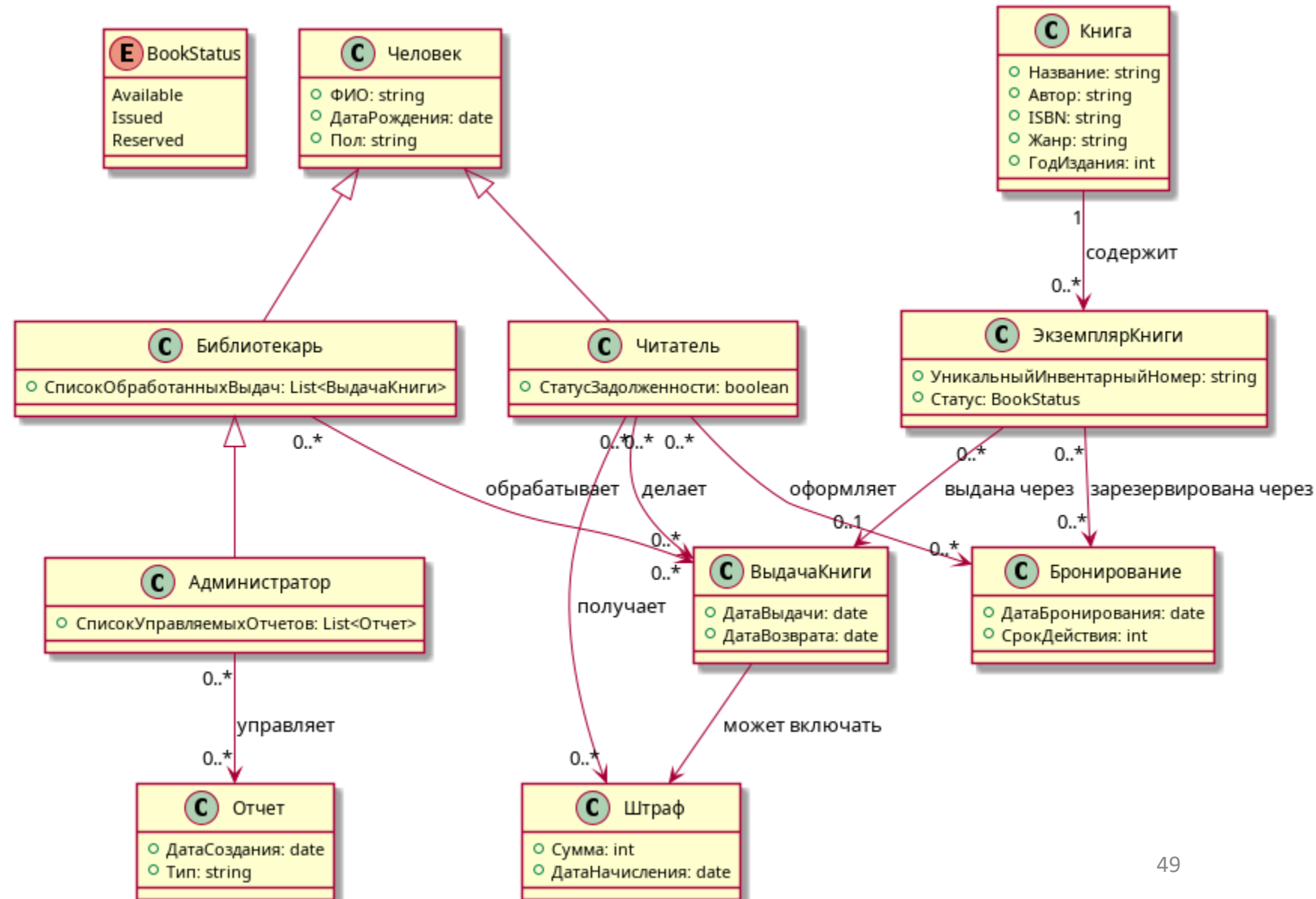
Почти полное.

Сложность и стиль:

Высокий.

Соответствие содержанию ТЗ:

Полное.



# Результаты

Часть	Промпт	Модель	Работоспособность	Ошибки	Соответствие UML	Соответствие ТЗ
A1	Минимальный	Алиса	Низкая	Syntax Error, неверное наследование	Нет	Полное
		DeepSeek	Низкая	Ошибка рендера	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
A2	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	Средняя	Ошибки в границах системы	Частично	Неполное
		DeepSeek	Высокая	Неверное слово вместо include	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
A4	Продвинутый	Алиса	Высокая	Ошибки в наследовании и extend	Частично	Полное
		DeepSeek	Высокая	Ошибка направления extend	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное

# Результаты

Часть	Промпт	Модель	Работоспособность	Ошибки	Соответствие UML	Соответствие ТЗ
B1	Минимальный	Алиса	Низкая	Нет	Нет	Полное
		DeepSeek	Средняя	Перегруженность, лишние связи	Частично	Частично
		GPT5-mini	Высокая	Нет	Полное	Полное
B2	Базовый	Алиса	Низкая	Ошибки в атрибутах	Нет	Полное
		DeepSeek	Средняя	Ошибки в связях	Частично	Частично
		GPT5-mini	Высокая	Мелкие недочёты	Почти полное	Полное
B3	Расширенный	Алиса	Средняя	Ошибки в наследовании	Частично	Частично
		DeepSeek	Высокая	Ошибки в типах связей	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное
B4	Продвинутый	Алиса	Средняя	Ошибки в типах данных, enum, кратностях	Частично	Частично
		DeepSeek	Высокая	Ошибки в кратностях	Частично	Полное
		GPT5-mini	Высокая	Нет	Полное	Полное

# Вывод

**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 ТЗ соответствуют полностью. Мелкие недочёты встречались редко и легко исправлялись.

# Вывод

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.