МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ВЫСОКИХ ТЕХНОЛОГИЙ И ПЬЕЗОТЕХНИКИ

Кафедра прикладной информатики и инноватики

КУРСОВАЯ РАБОТА

по дисциплине

«Архитектура информационных систем»

на тему

«Сервис управления библиотекой»

Выполнил

студент 4 курса 3 группы

Пантелеев Михаил Евгеньевич

Принял

профессор, д.т.н.

Жмайлов Борис Борисович

Ростов-на-Дону

2022

1. Техническое задание

Предметная область курсовой работы связана с пользователями и информационными ресурсами, связанными с управлением библиотекой. Проект должен представлять из себя библиотеку в формате jar архива.

Необходимо создать сервис выдачи книг, а также продления их аренды и читательских билетов. Проект должен представлять собой Java приложение, предоставляющее API (application programming interface).

Предусмотреть в системе базовые операции: создание, редактирование, удаление и получение основных сущностей, таких как книга, читательский билет (временный и бессрочный), запись об аренде.

1. UML моделирование
   1. Диаграмма вариантов использования

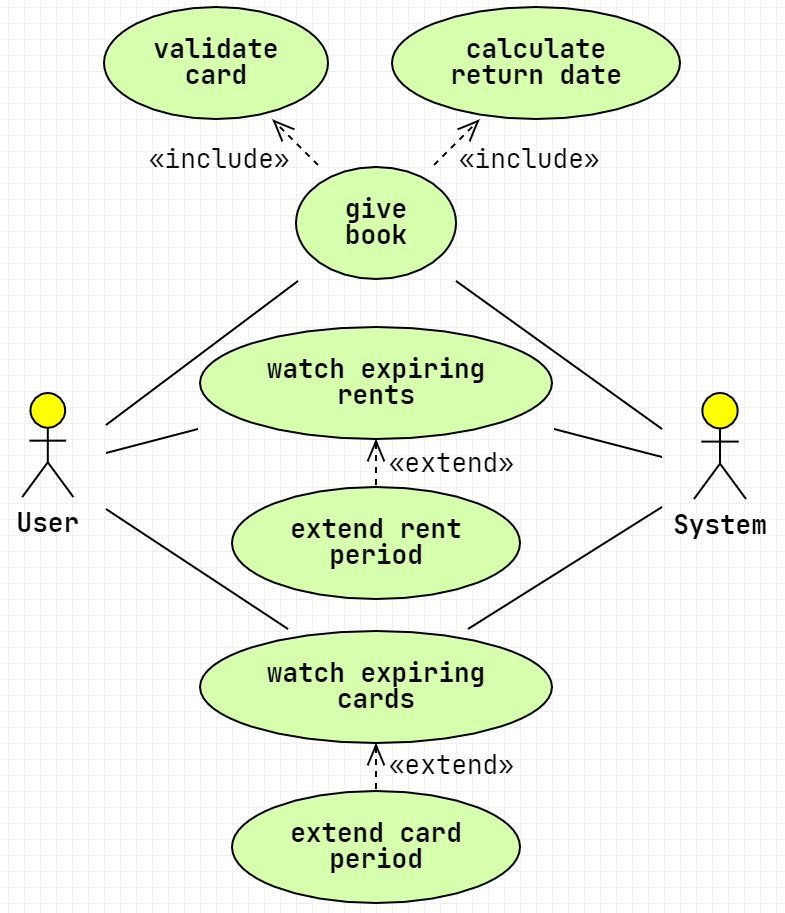


Схема 1. Диаграмма вариантов использования

Таблица 1. Детализация диаграммы вариантов использования.

|  |  |
| --- | --- |
| **Вариант использования** | **Описание** |
| **Give book** | Выдать книгу читателю.  **Входные данные:** long bookId, long cardId;  **Выходные данные:** Optional<Rent> rent;  **Включающие методы** «Validate card» получает параметры из сигнатуры родительского метода; «Calculate return date» получает параметры во время выполнения. |
| Validate card | Проверка актуальности срока читательского билета. Возвращает false, если срок истёк.  **Входные данные:** long cardId;  **Выходные данные:** bool isActual. |
| Calculate return date | Получение даты возврата книги (срок — 2 недели).  **Входные данные:** int startYear, int startMonth, int startDay;  **Выходные данные:** Optional<LocalDate> returnDate. |
| **Watch expiring rents** | Просмотр записей об аренде с истекающим сроком возврата.  **Входные данные:** long rentId;  **Выходные данные:** List<Rent> rents;  **Расширяющий метод** «Expire rent period» получает параметры из сигнатуры родительского метода и выполняется при rentId != 0. |
| Expire rent period | Продлить срок аренды книги.  **Входные данные:** long rentId;  **Выходные данные:** Optional<Rent> rent; |
| **Watch expiring cards** | Просмотр читательских билетов с истекающим сроком действия.  **Входные данные:** long rentId;  **Выходные данные:** List< TemporaryCard> cards;  **Расширяющий метод** «Expire rent period» получает параметры из сигнатуры родительского метода и выполняется при cardId!= 0. |
| Expire card period | Продлить срок действия читательского билета.  **Входные данные:** long cardId;  **Выходные данные:** Optional< TemporaryCard> card; |

* 1. Диаграмма классов

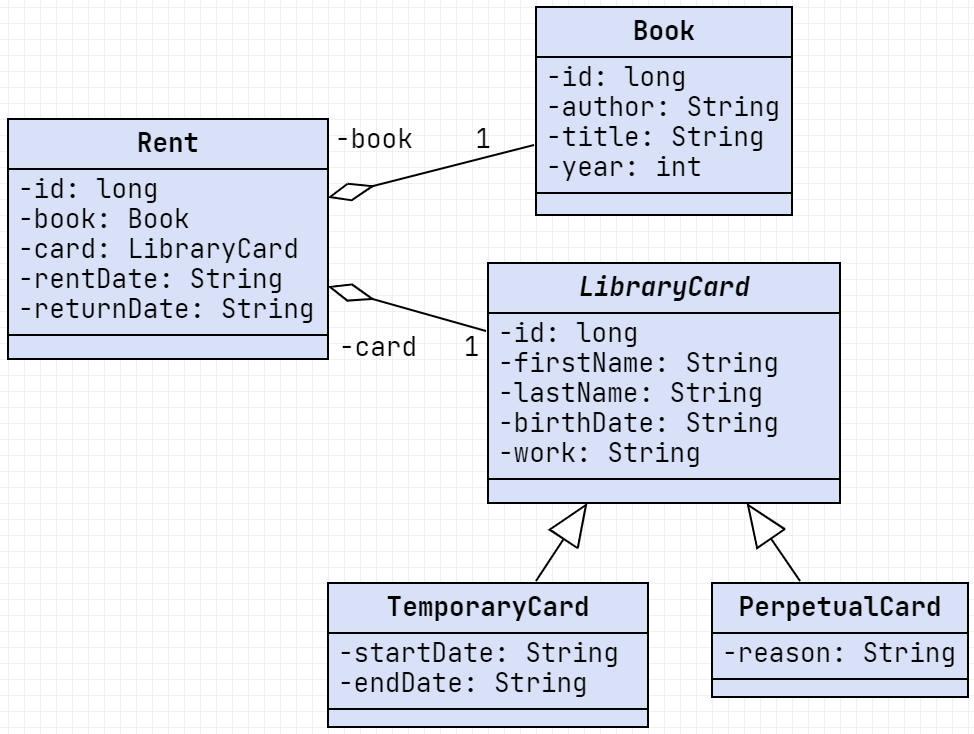


Схема 2. Диаграмма классов

* 1. Диаграммы активностей

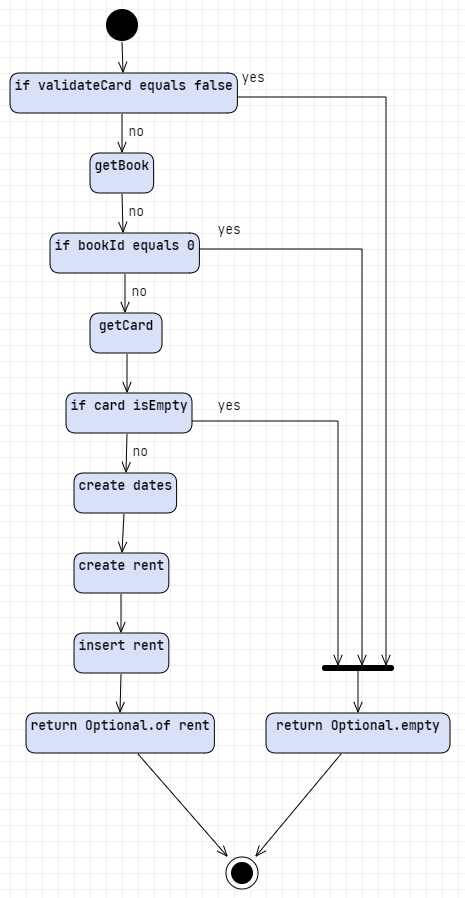


Схема 3. Диаграмма активности метода giveBook

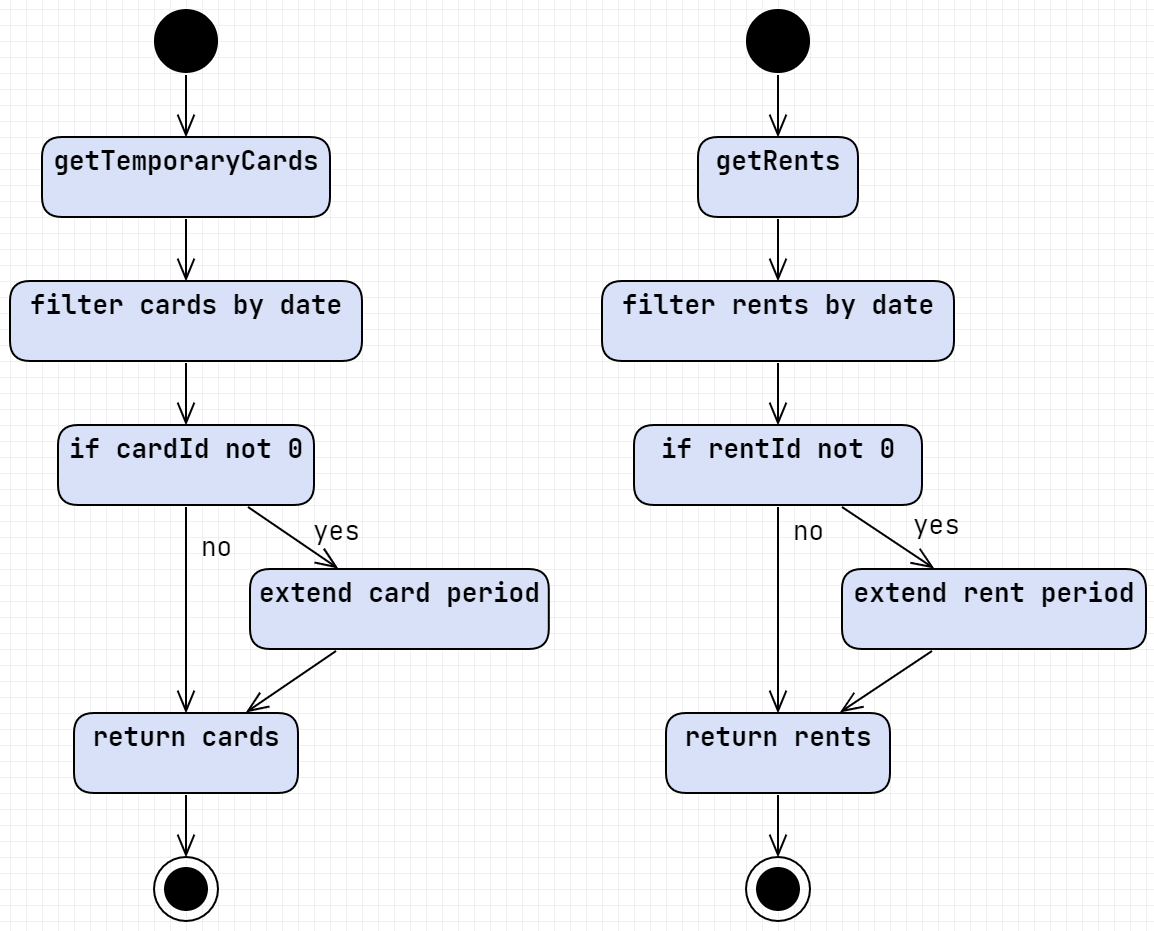


Схема 4. Диаграммы активностей методов getExpiringCards и getExpiringRents

* 1. Диаграмма компонентов

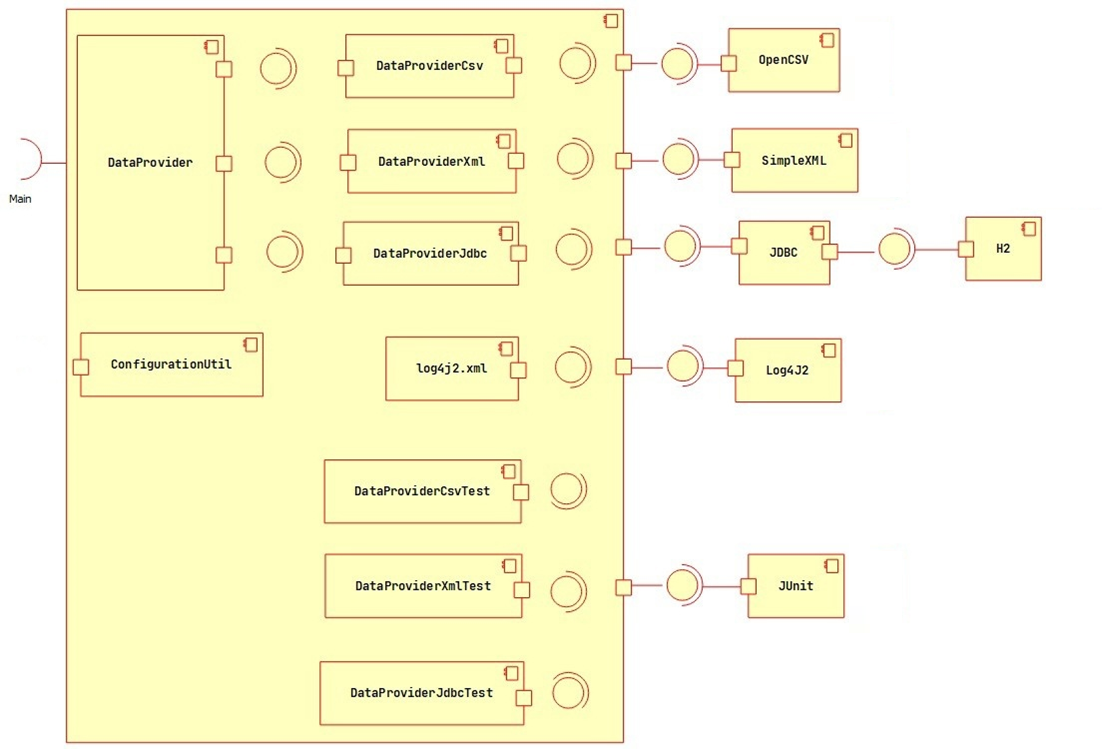


Схема 5. Диаграмма компонентов

1. Сценарии тестирования и результаты тестов

Таблица 2. Сценарии и результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестовый метод** | **Тип сценария** | **Тестируемый метод** | **Что контролируется?** |
| giveBookPos | позитивный | DataProvider. giveBook | Assertions.assertEquals( expectedRent, actualRent) |
| giveBookNeg | негативный | DataProvider. giveBook | Assertions.assertEquals( expectedRent, actualRent) |
| validateCardPos | позитивный | DataProvider. validateCard | Assertions.assertTrue(dp.validateCard(t1.getId())) |
| validateCardNeg | негативный | DataProvider. validateCard | Assertions.assertFalse(dp.validateCard(123)) |
| calculateReturnDatePos | позитивный | DataProvider. calculateReturnDate | Assertions.assertEquals(expectedDate, actualDate) |
| calculateReturnDateNeg | негативный | DataProvider. calculateReturnDate | Assertions.assertEquals(expectedDate, actualDate) |
| watchExpiringRentsPos | позитивный | DataProvider. watchExpiringRents | Assertions.assertEquals( expectedRents, actualRents); |
| watchExpiringRentsNeg | негативный | DataProvider. watchExpiringRents | Assertions.assertEquals(expectedRents, actualRents) |
| expireRentPeriodPos | позитивный | DataProvider. expireRentPeriod | Assertions.assertEquals( expectedRent, actualRent); |
| expireRentPeriodNeg | негативный | DataProvider. expireRentPeriod | Assertions.assertEquals( expectedRent, actualRent); |
| watchExpiringCardsPos | позитивный | DataProvider. watchExpiringCards | Assertions.assertEquals( expectedCards, actualCards); |
| watchExpiringCardsNeg | негативный | DataProvider. watchExpiringCards | Assertions.assertEquals( expectedCards, actualCards); |
| expireCardPeriodPos | позитивный | DataProvider. expireCardPeriod | Assertions.assertEquals( expectedCard, actualCard); |
| expireCardPeriodNeg | негативный | DataProvider. expireCardPeriod | Assertions.assertEquals( expectedCard, actualCard); |