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 для покупки/продажи товаров для творчества»

Выполнила

студентка 4 курса 3 группы

Котельникова Полина Анатольевна

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 просмотром и заказом товаров для творчества. Проект должен представлять из себя библиотеку в формате jar архива.

Необходимо создать сервис обзора и покупки товаров для творчества. 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 товар, набор для творчества, готовый арт-продукт, заказ и пользователь.

1. UML моделирование
   1. Диаграмма вариантов использования

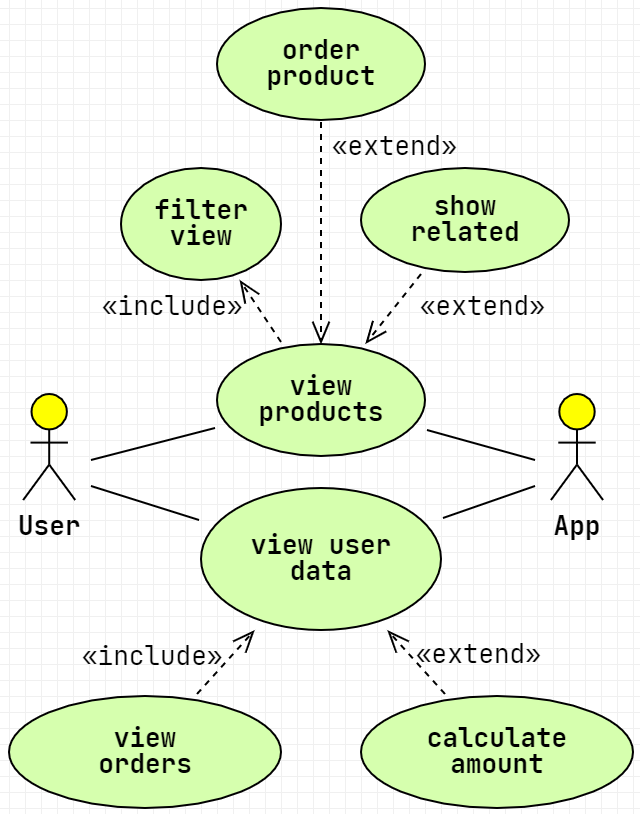


Схема 1. Диаграмма вариантов использования

Таблица 1. Детализация диаграммы вариантов использования.

|  |  |
| --- | --- |
| **Вариант использования** | **Описание** |
| **View products** | Просмотр товаров, доступных к покупке.  **Входные данные:** String filter, long productId;  **Выходные данные:** List<Product> products;  **Включающий метод** «Filter view» получает параметры из сигнатуры родительского метода;  **Расширяющиe методы** «Order product» и «Show related» получают параметры из сигнатуры родительского метода и выполняются при productId != 0 |
| Filter view | Фильтр товаров.  **Входные данные:** String filter;  **Выходные данные:** List <Product> filteredProducts. |
| Order product | Заказ товара.  **Входные данные:** long productId;  **Выходные данные:** Optional<Order> newOrder. |
| Show related | Просмотр рекомендуемых товаров (на основании заказов других пользователей).  **Входные данные:** long productId;  **Выходные данные:** Set<Product> relatedProducts. |
| **View user data** | Просмотр информации о пользователе.  **Входные данные:** long userId, boolean calculate  **Выходные данные:** Optional<User> requsetedUser;  **Включающий метод** «View orders» получает параметры из сигнатуры родительского метода;  **Расширяющий метод «**Calculate amount» получает параметры во время выполнения родительского метода и выполняется при calculate == true. |
| View user orders | Просмотр товаров, заказанных пользователем с указанным ID.  **Входные данные:** long userId;  **Выходные данные:** List<Order> userOrders. |
| Calculate amount | Подсчёт итоговой суммы, потраченной на все заказы выбранного пользователя.  **Входные данные:** long userId;  **Выходные данные:** double amount. |

* 1. Диаграмма классов

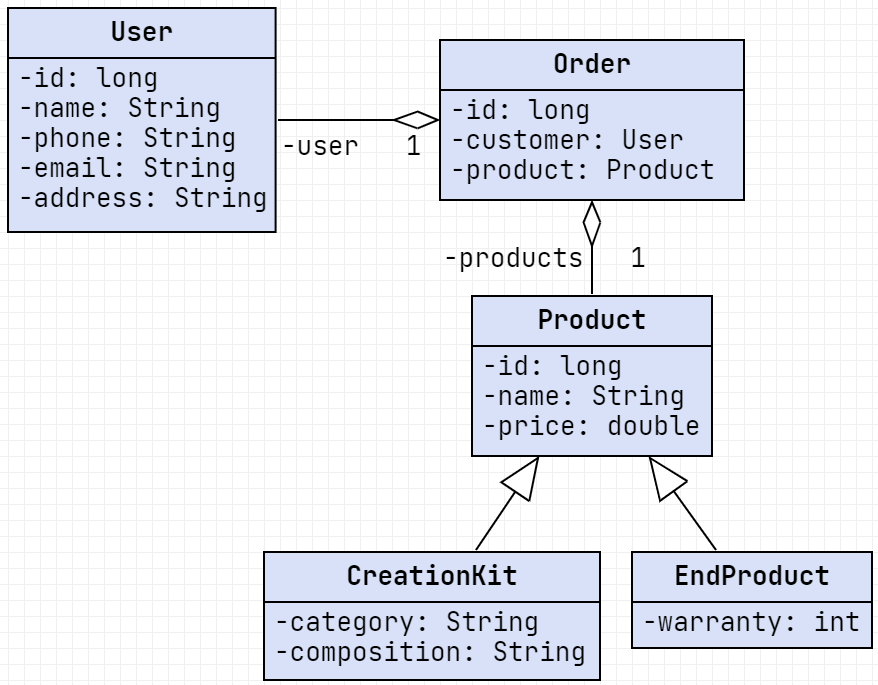


Схема 2. Диаграмма классов

* 1. Диаграммы активностей

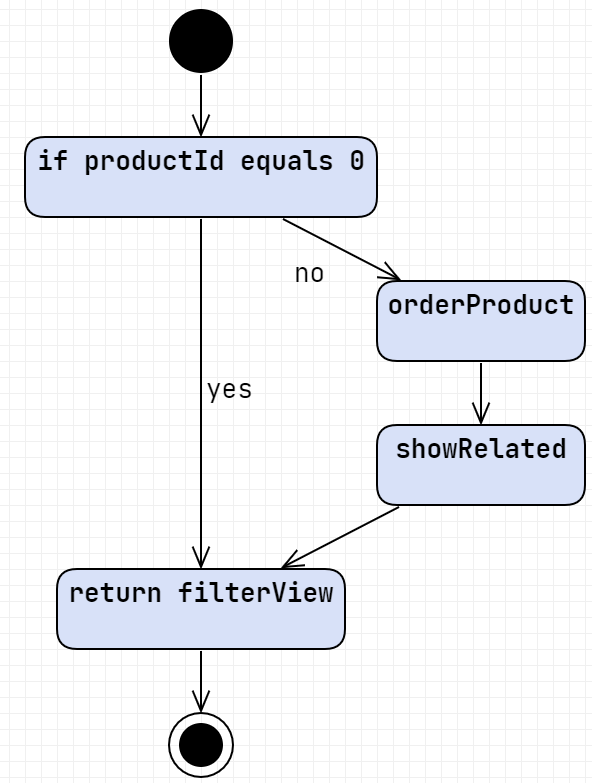


Схема 3. Диаграмма активностей метода **«View products»**

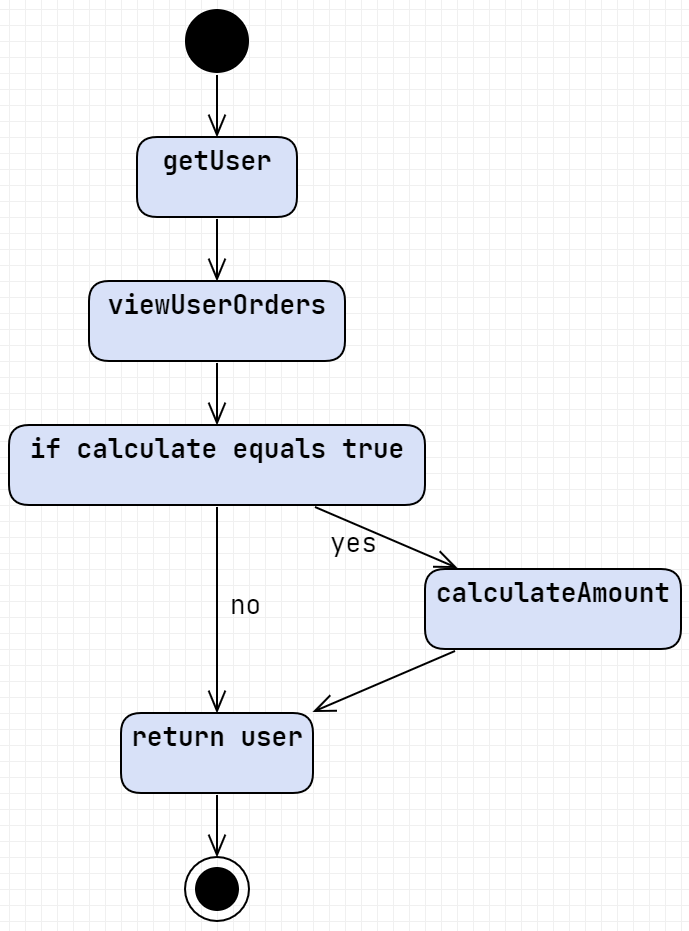


Схема 4. Диаграмма активностей метода **«View user data»**

* 1. Диаграмма компонентов

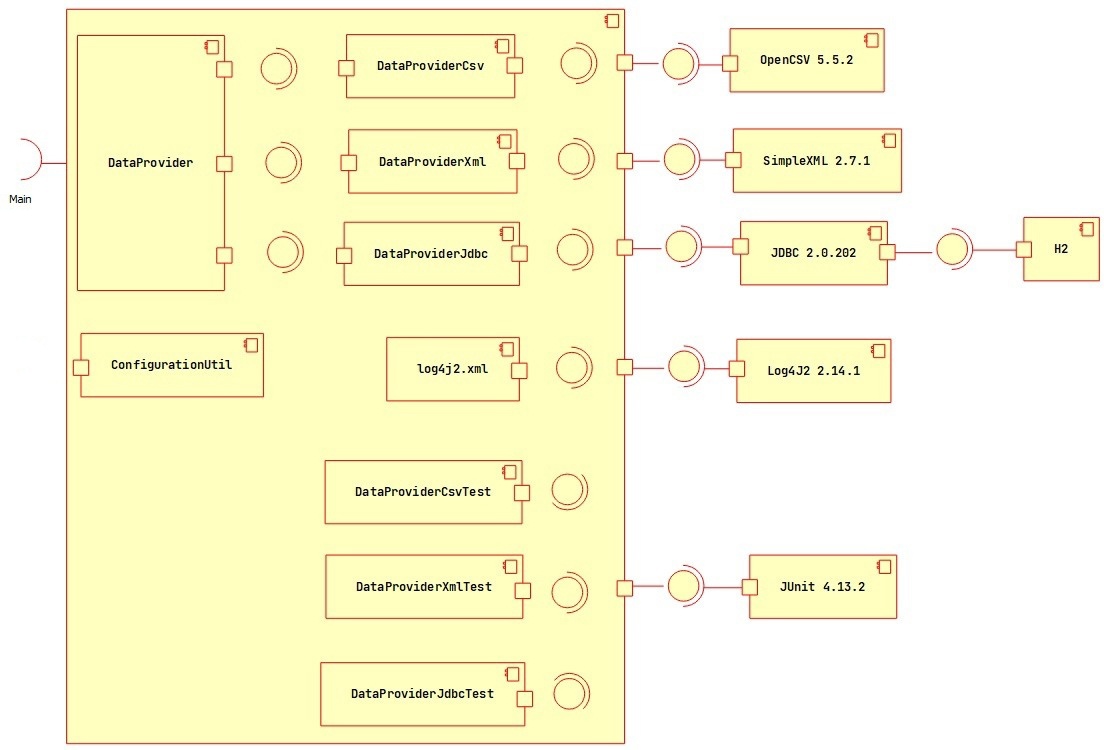


Схема 5. Диаграмма компонентов

1. Сценарии тестирования и результаты тестов

Таблица 2. Сценарии и результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Тестовый метод** | **Тип сценария** | **Тестируемый метод** | **Что контролируется?** |
| viewProductsPos | позитивный | viewProducts | Успешное выполнение: assertEquals(  expectedProducts, actualProducts); |
| viewProductsNeg | негативный | viewProducts | Успешное выполнение: assertNotEquals(  expectedProducts, actualProducts); |
| filterViewPos | позитивный | filterView | Успешное выполнение: assertEquals(  expectedProducts, actualProducts); |
| filterViewNeg | негативный | filterView | Успешное выполнение: assertNotEquals(  expectedProducts, actualProducts); |
| orderProductPos | позитивный | orderProduct | Успешное выполнение: assertEquals(  expectedOrder, actualOrder); |
| orderProductNeg | негативный | orderProduct | Успешное выполнение: assertNotEquals(  expectedOrder, actualOrder); |
| showRelatedPos | позитивный | showRelated | Успешное выполнение: assertNotEquals(  expectedSet, actualSet); |
| showRelatedNeg | негативный | showRelated | Успешное выполнение: assertNotEquals(  expectedSet, actualSet); |
| viewUserDataPos | позитивный | viewUserData | Успешное выполнение: assertEquals(  expectedUser, actualUser); |
| viewUserDataNeg | негативный | viewUserData | Успешное выполнение: assertNotEquals(  expectedUser, actualUser); |
| viewUserOrdersPos | позитивный | viewUserOrders | Успешное выполнение: assertEquals(  expectedOrders, actualOrders); |
| viewUserOrdersNeg | негативный | viewUserOrders | Успешное выполнение: assertNotEquals(  expectedOrders, actualOrders); |
| calculateAmountPos | позитивный | calculateAmount | Успешное выполнение: assertEquals(  expectedAmount, actualAmount); |
| calculateAmountNeg | негативный | calculateAmount | Успешное выполнение: assertNotEquals(  expectedAmount, actualAmount); |