МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по дисциплине «Спецификация, проектирование и архитектура программных систем»

Студент гр. 9304 _____ Ковалёв П.Д.

Студент гр. 9304 _____ Прокофьев М.Д.

Студент гр. 9304 _____ Борисовский В.Ю.

Студент гр. 9304 _____ Боблаков Д.С.

Студент гр. 9304 _____ Силкин В.А.

Преподаватель _____ Романенко С.А.

Санкт-Петербург
2021

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Ковалёв П.Д., Прокофьев М.Д., Борисовский В.Ю., Боблаков Д.С., Силкин В.А.

Группа 9304

Тема работы: информационная система купли-продажи товаров Buy&Sell

Содержание пояснительной записки:

«Содержание»

«Описание предметной области»

«Архитектурные решения»

«Соответствие системы требованиям ТЗ»

Предполагаемый объем пояснительной записки: не менее 80 страниц.

Дата выдачи задания: 09.09.2021

Дата сдачи реферата: 20.12.2021

Дата защиты реферата: 24.12.2021

Студент гр. 9304	_____	Ковалёв П.Д.
Студент гр. 9304	_____	Прокофьев М.Д.
Студент гр. 9304	_____	Борисовский В.Ю.
Студент гр. 9304	_____	Боблаков Д.С.
Студент гр. 9304	_____	Силкин В.А.
Преподаватель	_____	Романенко С.А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Работа представляет из себя проект по автоматизации систему купли-продажи товаров Buy&Sell. Для системы изначально было разработано ТЗ, в соответствии с которым прорабатывалась архитектура системы и описывалась предметная область. Результат работы представлен в данной пояснительной записке.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Полное наименование системы

Информационная система безопасной купли-продажи товаров “Buy&Sell”. Условное обозначение: ИС “Buy&Sell”.

1.2. Цели, назначение и области использования АС

Система предназначена для автоматизации безопасной покупки и доставки товаров.

Цели создания системы:

1. Создать инструмент для обеспечения безопасной сделки между продавцом и покупателем.
2. Организовать отношения с сервисами почтовой доставки для осуществления с их помощью доставки товаров от продавца покупателю.
3. Организовать отношения с сервисами почтовой доставки для осуществления с их помощью доставки товаров от продавца покупателю.
4. Организовать систему приема платежей.
5. Создать инструмент для рассмотрения апелляций от покупателей, в случае отправки продавцом товара не соответствующего качества.
6. Реализовать систему рейтинга продавцов и покупателей.

2. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Описание предметной области в нотации IDEF0

2.1.1. Словарь терминов предметной области

1. **Товар** - Вещь, участвующая в свободном обмене на другие вещи, или продукт, произведённый для продажи.
2. **Безопасная сделка** - Обмен товара на товар (или товара на валюту), в котором гарантируется соблюдение обязательств обеих сторон обмена.
3. **Апелляция** - Жалоба, подаваемая одной из сторон обмена в случае необеспечения посредником (в данном случае, компанией "Buy&Sell") безопасной сделки.
4. **Сотрудник апелляционной комиссии** - сотрудник, который должен рассматривать апелляции пользователей и выносить по ним решение.
5. **Сотрудник технической поддержки** - сотрудник, который должен следить за стабильной работой веб-приложения и устранять неисправности в случае их появления.
6. **Рейтинг** - Оценка покупателя или продавца сервисом, которая может быть отредактирована сотрудником апелляционной комиссии в соответствии с условиями пользования.
7. **"Замороженные" деньги** - Деньги, находящиеся на счету компании, но принадлежащие стороннему лицу.
8. **Качество товара** - Соответствие свойств товара его обозначенным свойствам.
9. **Веб-модуль** - Один из интерфейсов веб-приложения, доступ к которому может быть выдан автоматически, или владельцем этого веб-приложения.
10. **Подсистема** - Самостоятельный функциональный отдел внутри системы, который состовляет её часть и взаимодействует с другими функциональными отделами.
11. **Пользователь** - Лицо, которое может выступать в роли покупателя или продавца в данной системе. Имеет раздельные рейтинги покупателя и продавца.

2.1.2. Контекстная диаграмма IDEF0

На рисунке 1 представлена контекстная диаграмма.

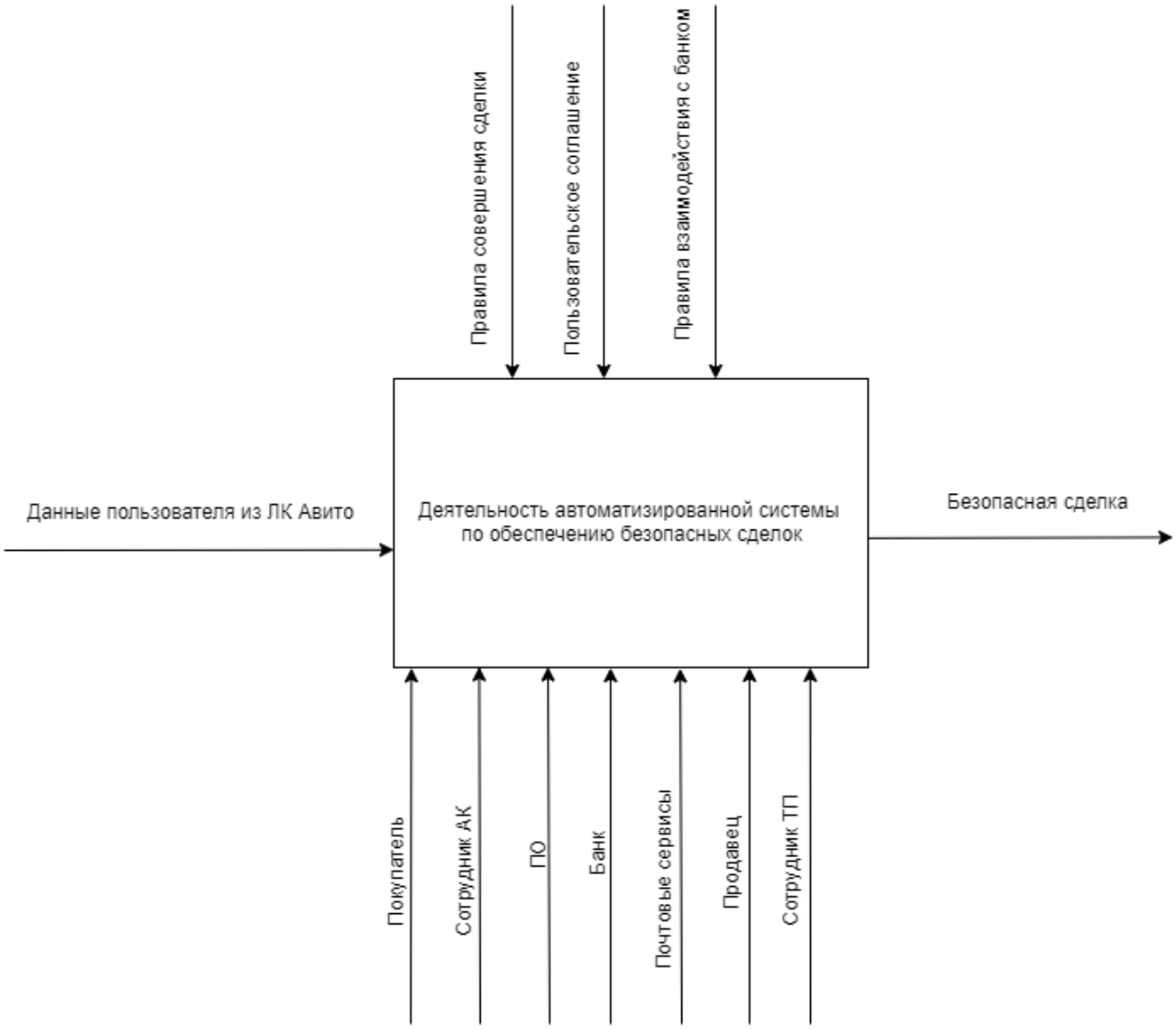

Рисунок 1 – Контекстная диаграмма

Данная диаграмма определяет границы системы. В соответствии с ТЗ (п.4.1.), система имеет следующие механизмы – Покупатель, Сотрудник АК, ПО, Банк, Почтовые сервисы, Продавец, Сотрудник ТП.

Т.к. система работает в связке с Авито, то для входа в систему необходим действующий аккаунт в Авито. Для взаимодействия с системой предусмотрены элементы управления: правила совершения сделки, пользовательское соглашение, правила взаимодействия с банком.

2.1.3. Диаграмма IDEF0 1-го уровня

На рисунке 2 отображена диаграмма IDEF0 1-го уровня.

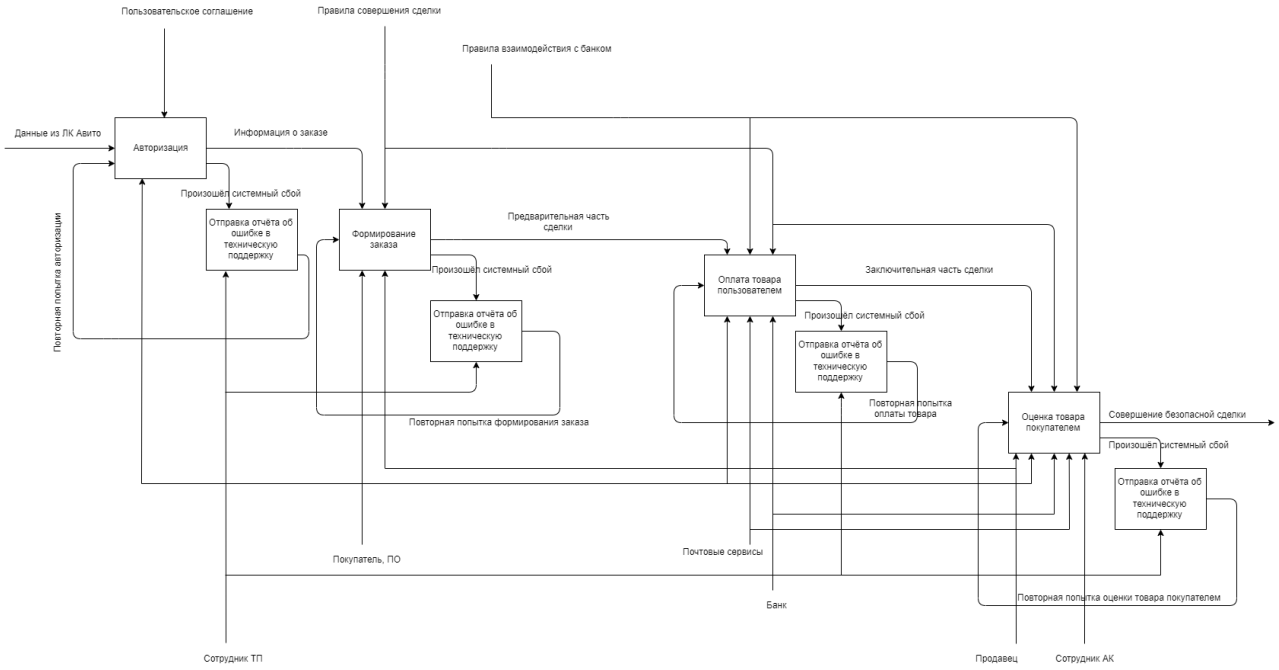

Рисунок 2 – Диаграмма IDEF0 1-го уровня A0

Блок “Авторизация” включает в себя функцию авторизации пользователей и сотрудников в системе. На вход блок получает данные о пользователе из ЛК Авито, а на выходе – информацию о заказе (она передается совместно с покупателем) или – в случае системного сбоя управление переходит к блоку “Отправка отчёта об ошибке в техническую поддержку”, после которого есть возможность вновь авторизоваться. В качестве управляющих элементов блок получает пользовательское соглашение, а в качестве механизмов – Покупатель, Продавец, Сотрудник ТП и ПО.

Блок “Формирование заказа” включает в себя функции выбора продавца и получение от него ответа. На вход блок получает информацию о заказе, а на выходе – предварительная часть сделки, или в случае системной ошибки – системный сбой, который в свою очередь подается на вход блоку “Отправка отчёта об ошибке в техническую поддержку”. В качестве управляющих элементов блок получает правила совершения сделки и правила взаимодействия с банком, а в качестве механизмов – Покупатель, ПО и Продавец.

Блок “Оплата товара пользователем” включает в себя функции оплаты товара и взаимодействия с финансовой частью системы. На вход система получает предварительную часть сделки, а на выходе – заключительную часть сделки. В случае системной ошибки, происходит отправка отчёта в техническую поддержку. В качестве управляющих элементов блок получает правила совершения сделки и правила взаимодействия с банком, а в качестве механизмов – Покупатель, ПО и Банк.

Блок “Оценка товара покупателем” включает в себя функции рейтинговой системы, функции работы с апелляциями, функции взаимодействия с почтовыми сервисами, а также функции финансовой системы. На вход блок получает заключительную часть сделки, а на выходе – безопасную сделку, которая является целью работы системы. В случае системного сбоя управление переходит к блоку “Отправка отчёта об ошибке в техническую поддержку”, после которого есть возможность вновь совершить безопасную сделку, используя блок “Оценка товара покупателем”. В качестве управляющих элементов блок получает правила совершения сделки и правила взаимодействия с банком, а в качестве механизмов – Покупатель, Сотрудник АК, ПО и Банк.

2.1.4. Диаграммы IDEF0 2-го уровня

На рисунке 3 отображена диаграмма IDEF0 2-го уровня A2.

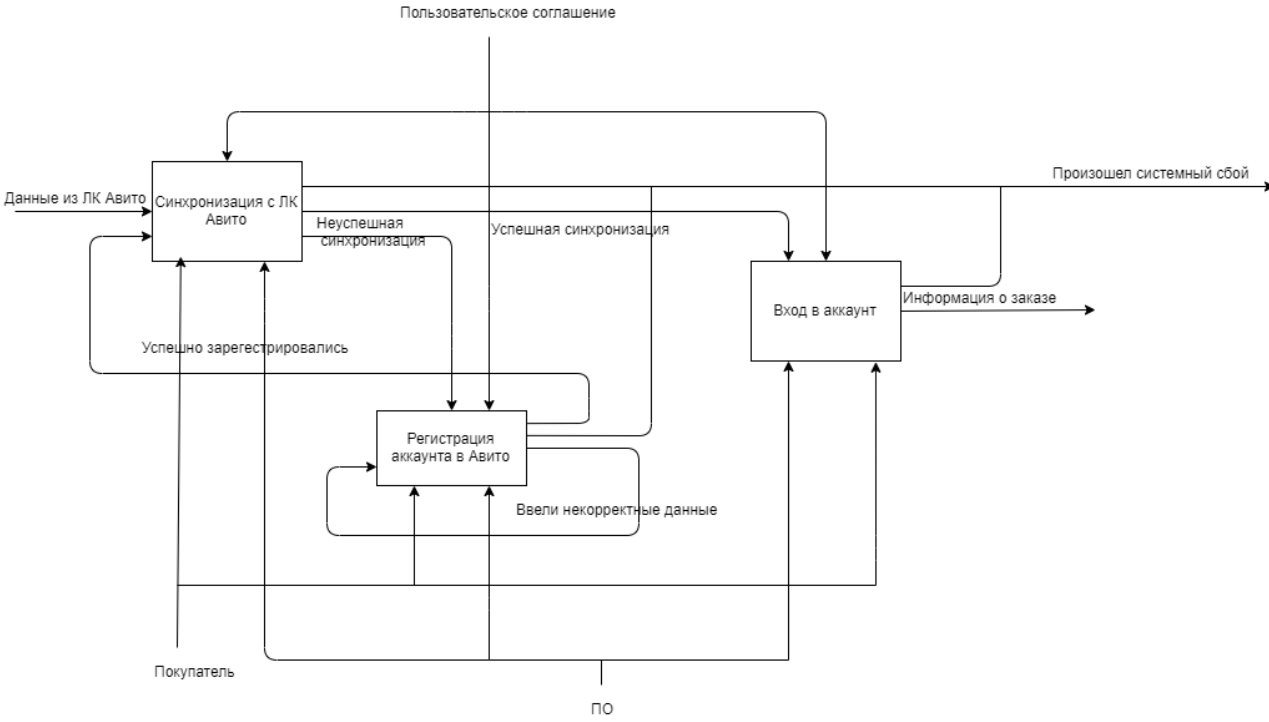

Рисунок 3 – Диаграмма IDEF0 2-го уровня A2

Данная диаграмма детализирует процесс авторизации в системе. Как можно видеть, т.к. система связана с Авито, то изначально происходит синхронизация с личным кабинетом Авито. В случае успешной синхронизации (такой ЛК существует), происходит вход в систему. В случае неуспешной синхронизации, система позволяет зарегистрироваться в Авито. В случае если пользователь введет некорректные данные, предусмотрена возможность повторной регистрации.

На рисунке 4 отображена диаграмма IDEF0 2-го уровня A2.

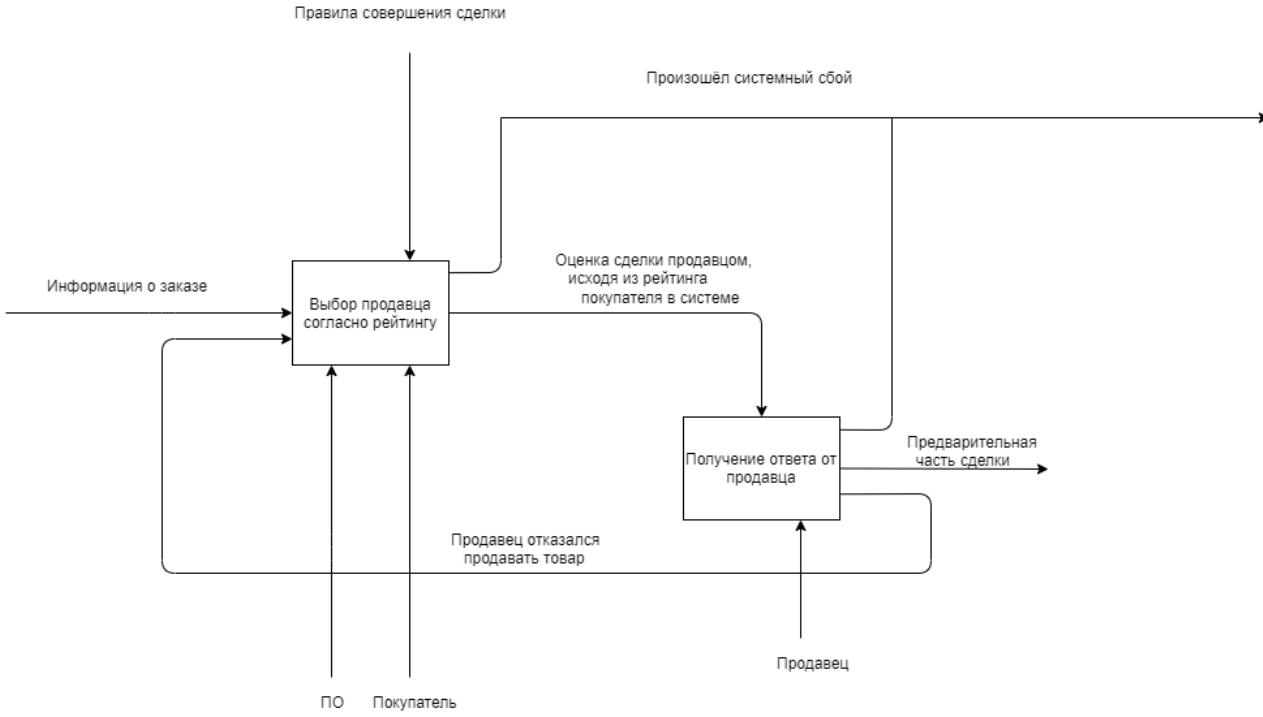

Рисунок 4 - Диаграмма IDEF0 2-го уровня A2

Данная диаграмма детализирует процесс формирования заказа. Согласно техническому заданию, покупатель имеет возможность выбрать продавца согласно его рейтингу, а продавец имеет возможность подтвердить согласие на сделку или отказаться от сделки исходя из рейтинга покупателя. Как можно видеть, сначала происходит блок “Выбор продавца согласно рейтингу”, управляющие элементы – правила совершения сделки, а механизмы – ПО и Покупатель. Далее управление передается к блоку “Получение ответа от продавца”, на выходе из которого получаем либо предварительную часть сделки, либо управление переходит к блоку “Выбор продавца согласно рейтингу”, в случае отказа продавца от сделки.

Механизмы блока “Получение ответа от продавца” – Продавец.

На рисунке 5 отображена диаграмма IDEF0 2-го уровня А3.

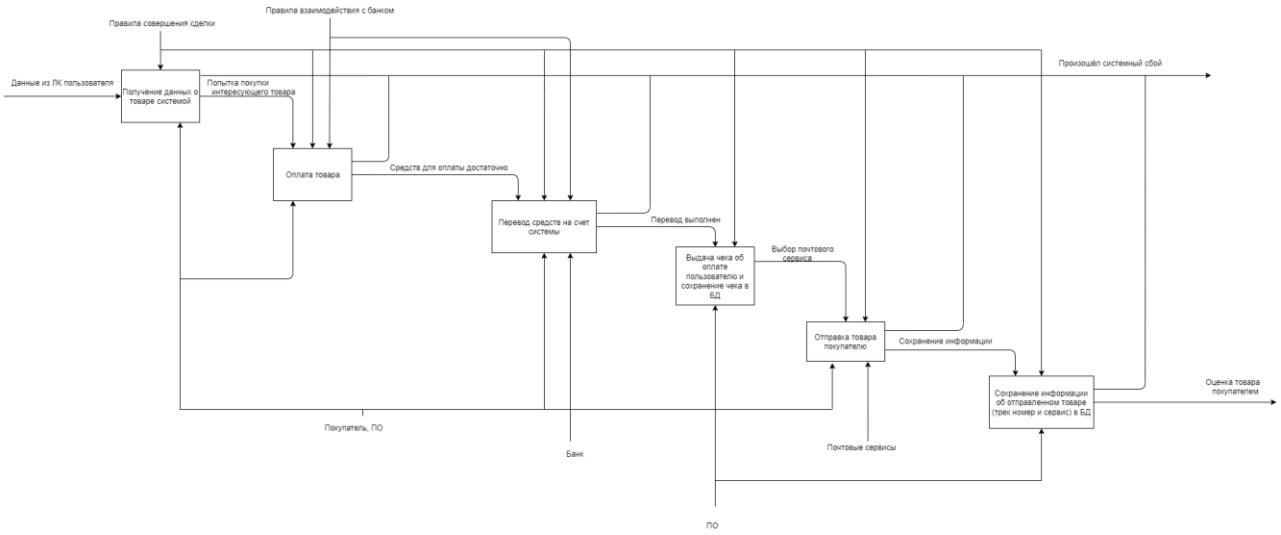

Рисунок 5 - Диаграмма IDEF0 2-го уровня А3

Данная диаграмма детализирует процесс оплаты товара пользователем. Как можно видеть, изначально управление передается блоку “Получение данных о товаре системой”, после чего, на выходе получается попытка покупки интересующего товара. Управляющие элементы – правила совершения сделки, механизмы – Покупатель и ПО.

Далее управление передается блоку “Оплата товара”, который детализирует процесс транзакции со стороны покупателя. Данный блок на выходе получает информацию о том, что средств для оплаты достаточно, после чего управление переходит к блоку перевод средств на счёт системы. Управляющие элементы блока “Оплата товара” – правила совершения сделки и правила взаимодействия с банком. Механизмы – Покупатель, ПО.

Блок “Перевод средств на счёт системы” включает в себя функции, которые осуществляют перевод средств со счета пользователя на счет системы для осуществления безопасной сделки. Управляющие элементы – правила совершения сделки и правила взаимодействия с банком. Механизмы – Покупатель, ПО, Банк.

После успешного перевода средств на счёт системы, сама система должна выдать покупателю чек о проведенном денежном переводе, что осуществляется

блоком “Выдача чека об оплате пользователю и сохранение чека в БД”. Управляющие элементы блока – правила взаимодействия с банком. Механизмы – ПО.

После того, как товар был оплачен покупателем, товар отправляется покупателю, что показано наличием блока “Отправка товара покупателю”. Управляющие элементы блока – правила совершения сделки. Механизмы блока – ПО.

Сама система после отправки товара сохраняет информацию об отправленном товаре в БД системы, что отражено блоком “Сохранение информации об отправленном товаре (трек номер и сервис) в БД”. Управляющие элементы блока – правила совершения сделки. Механизмы блока – ПО.

В случае системной ошибки в одном из блоков, этот блок на выходе вернет отчет об ошибке.

На рисунке 6 представлена отображена диаграмма IDEF0 2-го уровня A31.

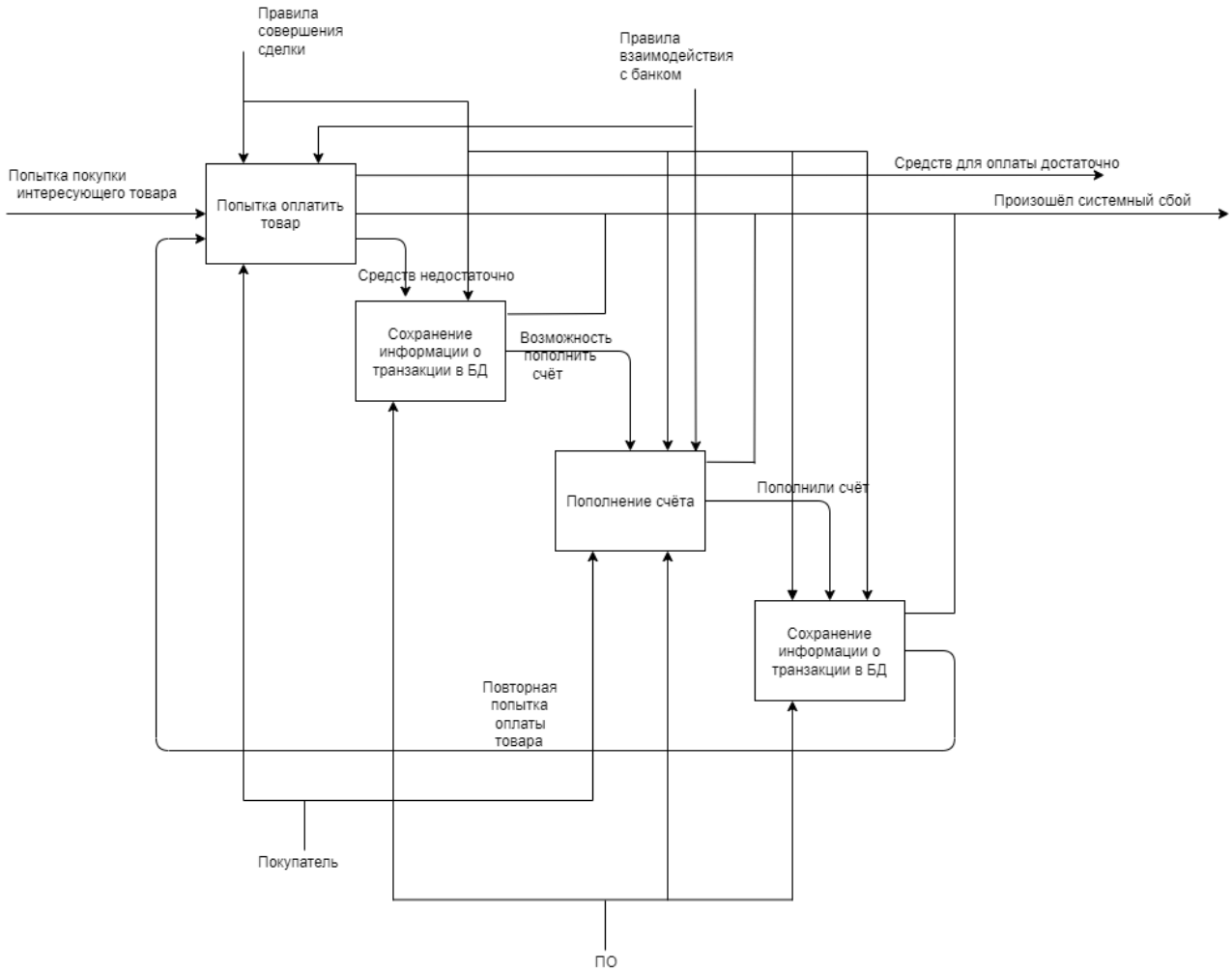

Рисунок 6 - Диаграмма IDEF0 2-го уровня A31

Данная диаграмма детализирует процесс оплаты товара из диаграммы A3, а конкретно рассматривает случаи, когда средств хватает и не хватает на счёте. На вход блоку “Попытка оплатить товар” приходит попытка покупки интересующего товара. Управляющие элементы – правила совершения сделки, механизмы – Покупатель. Далее, если средств для оплаты достаточно, то управление передается в блок из диаграммы A3, в противном случае – попытка неудачной оплаты сохранятся в системе, а управление передается сначала блоку “Сохранение информации о транзакции в БД”, а после – “Пополнение счёта”. Управляющие элементы блока “Сохранение информации о транзакции в БД” – правила совершения сделки, механизмы – ПО.

Управляющие элементы блока “Пополнение счёта” – правила взаимодействия с банком, механизмы – Покупатель и ПО.

После пополнения счёта управление передается блоку “Сохранение информации о транзакции в БД ”. Управляющие элементы блока - правила совершения сделки и правила взаимодействия с банком. Механизмы – ПО.

На рисунке 7 представлена диаграмма IDEF0 2-го уровня А4.

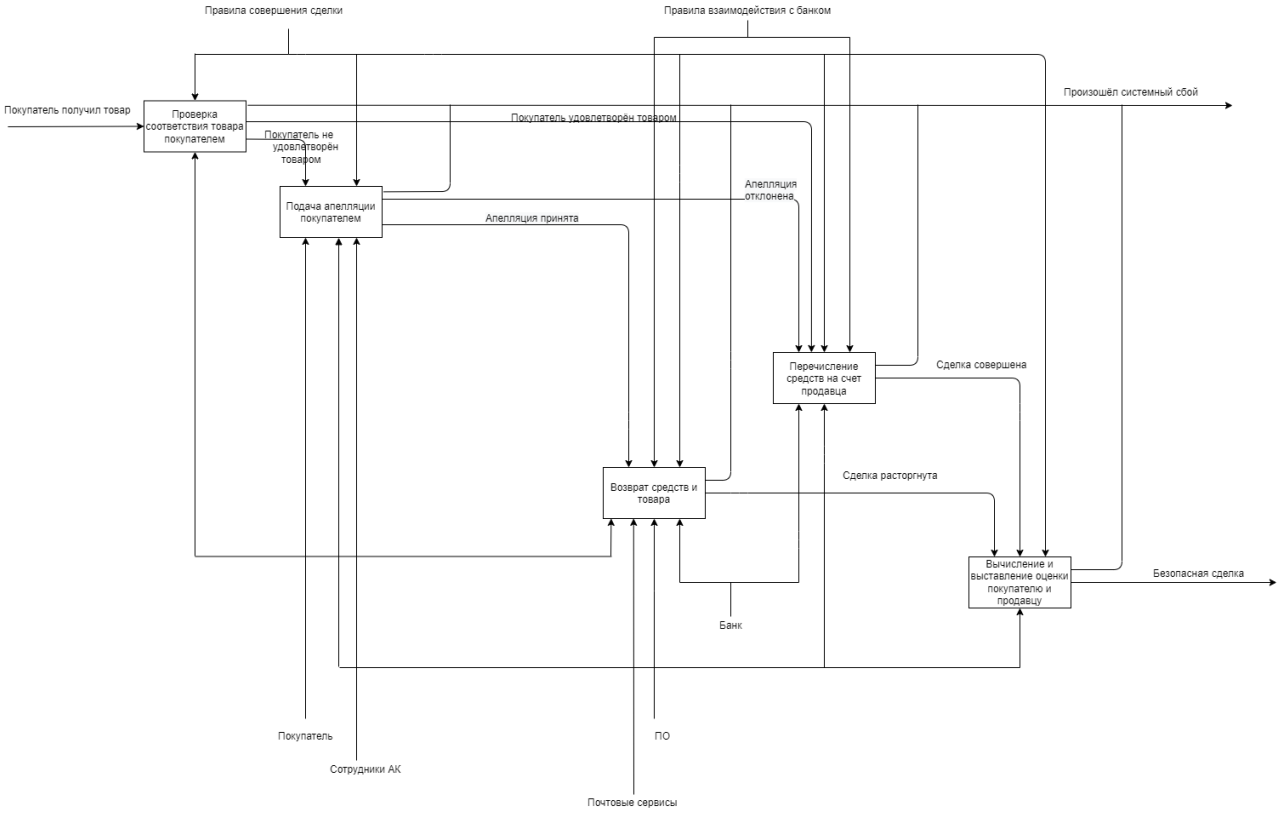

Рисунок 7 – Диаграмма IDEF0 2-го уровня А4

Данная диаграмма детализирует процесс оценки товара покупателем. Изначально управление передается блоку “Проверка соответствия товара покупателем”. Блок подразумевает под собой то, что покупатель проверит состояние товара и на основании этого решит вопрос о том: писать апелляцию по поводу товара или нет. Управляющие механизмы: правила совершения сделки, механизмы – Покупатель. Управление передается одному из двух блоков: “Подача апелляции покупателем” или “ Перечисление средств на счёт продавца ” в зависимости от того, писал ли пользователь апелляцию.

Далее управление передается блоку “Подача апелляции покупателем”. Результатом данного блока является принятие или отклонение апелляции сотрудником АК. Управляющие элементы блока – правила совершения сделки, механизмы – Покупатель, Сотрудник АК, ПО.

В зависимости от того, была апелляция принята или нет управление передается к одному из блоков: “Перечисление средств на счёт продавца” –

если апелляция отклонена или “Возврат средств и товара” – если апелляция была принята.

Блок “Возврат средств и товара” на выходе возвращает расторгнутую сделку. Управляющие элементы блока – правила взаимодействия с банком и правила совершения сделки. Механизмы блока – Покупатель, ПО, Почтовые сервисы, Банк.

Блок “Перечисление средств на счёт продавца” на выходе вызвращает совершенную сделку. Управляющие элементы блока – правила взаимодействия с банком и правила совершения сделки. Механизмы блока – ПО, Банк.

Управление передается блоку “Вычисление и выставление оценки покупателю и продавцу”. Блок включает в себя функции для работы с рейтингами покупателя и продавца, а также функции, реализующие алгоритм изменения рейтинга на основе данных о сделке. Управляющие элементы блока – правила совершения сделки. Механизмы блока – ПО.

На рисунке 8 представлена диаграмма IDEF0 2-го уровня A41.

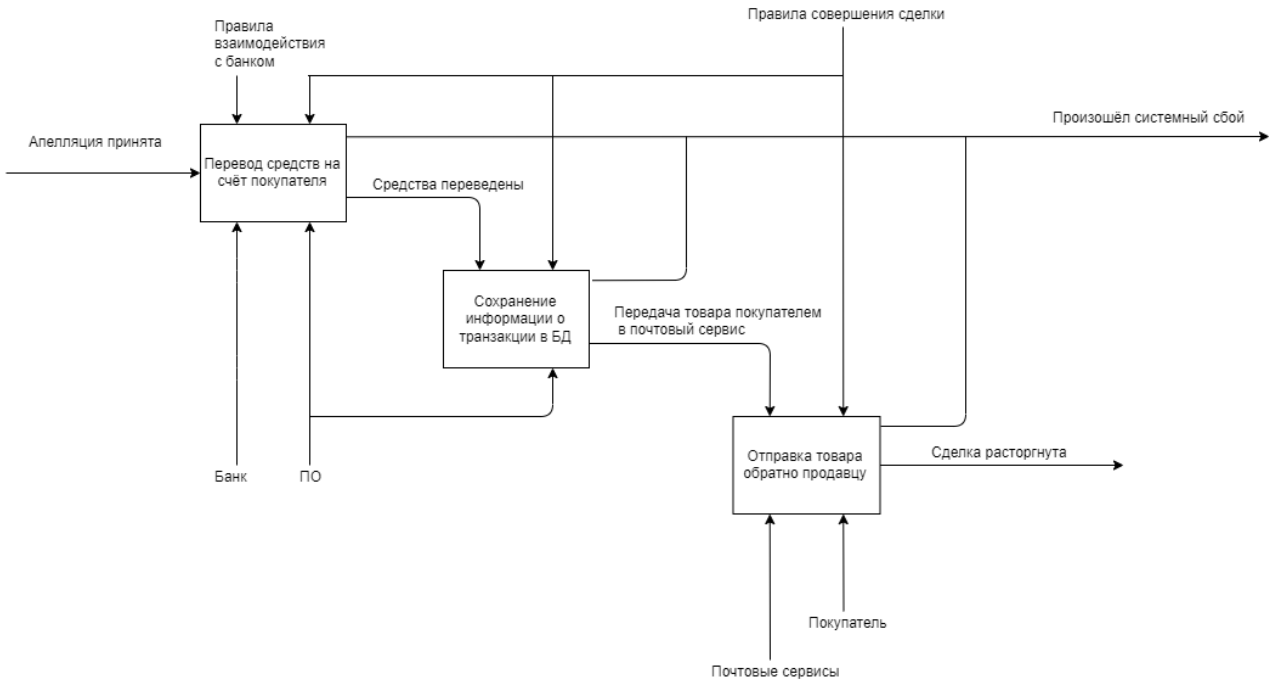

Рисунок 8 – Диаграмма IDEF0 2-го уровня A41

Диаграмма A41 детализирует процесс возврата средств и товара в случае, если пользователь написал апелляцию, и она была принята апелляционной комиссией.

Сперва управление передается блоку “Перевод средств на счёт покупателя”, в который включены функции, осуществляющие перевод средств со счёта системы на счёт покупателя. Управляющие элементы блока – правила взаимодействия с банком, механизмы блока – Банк и ПО.

Далее управление передается блоку “Сохранение информации о транзакции в БД”. На вход блок получает информацию о том, что средства переведены, а на выходе – передача товара покупателем в почтовый сервис. Управляющие элементы блока – правила совершения сделки, механизмы блока - ПО.

Далее управление передается блоку “отправка товара обратно продавцу”. Управляющими элементами блока являются правила совершения сделки, а механизмами – Покупатель и Почтовые сервисы. Блок на выходе выдает информацию о том, что сделка расторгнута.

В случае системного сбоя, каждый из блоков имеет возможность сообщить об этом.

На рисунке 9 представлена диаграмма IDEF0 2-го уровня A41.

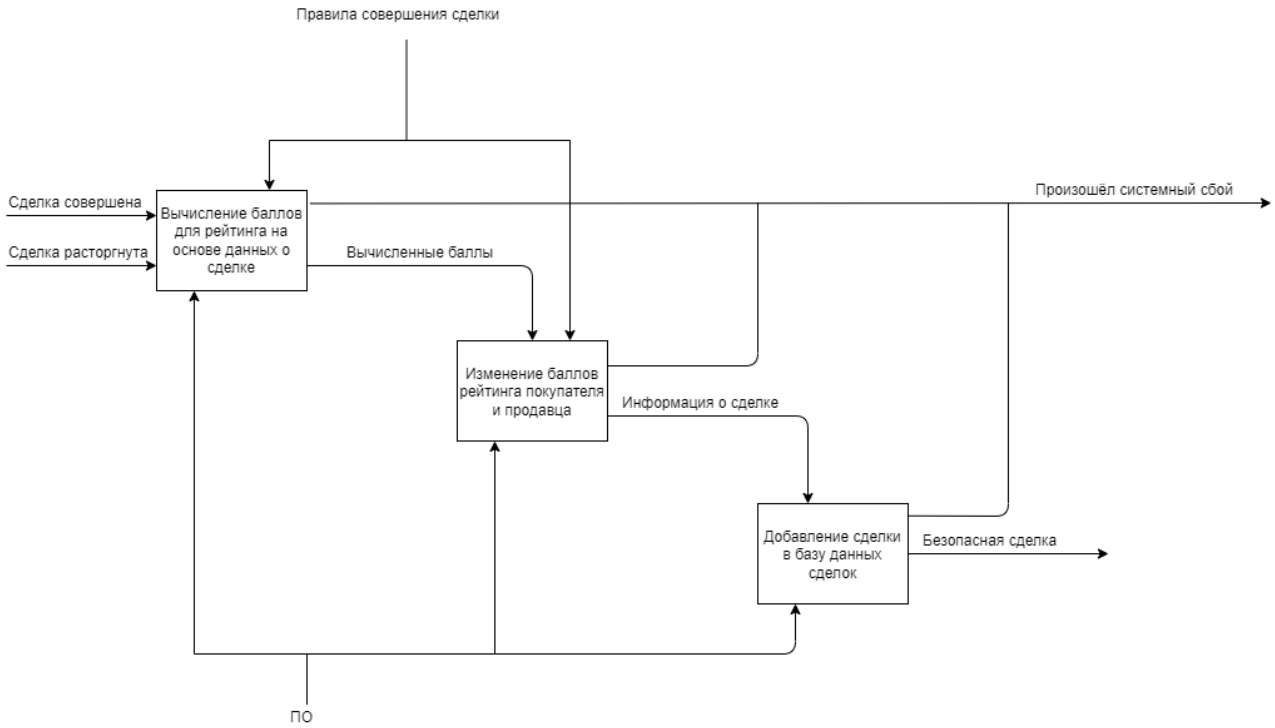

Рисунок 9 – Диаграмма IDEF0 2-го уровня A42

Данная диаграмма детализирует процесс вычисления и выставления оценки пользователям. На вход блоку “Выичсление баллов для рейтинга на основе данных о сделке” поступает информация о сделке: расторгнута ли она или совершена. В блок включены функции, реализующие вычисление баллов участников сделки. Управляющие элементы блока – правила совершения сделки. Механизмы блока – ПО.

Далее управление передается блоку “Изменение баллов рейтинга покупателя и продавца”. На вход блоку подаются вычисленные баллы, а на выходе – информация о сделке. Управляющие элементы блока – правила совершения сделки, механизмы блока – ПО.

Далее управление передается блоку “Добавление сделки в базу данных сделок”. На вход подается информация о сделке, на выходе получаем безопасную сделку. Механизм блока – ПО.

В случае системного сбоя, каждый из блоков имеет возможность сообщить об этом.

На рисунке 10 представлена диаграмма IDEF0 2-го уровня A43.

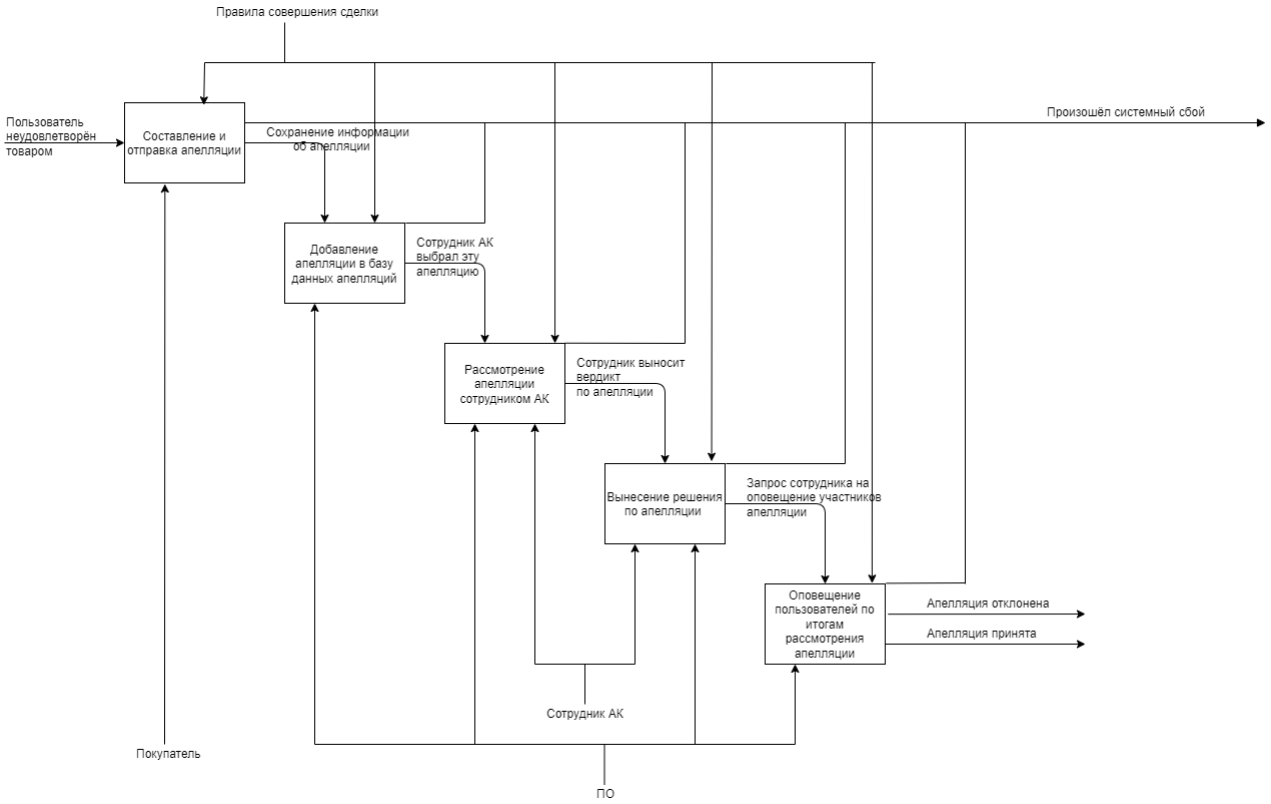

Рисунок 10 – Диаграмма IDEF0 2-го уровня A43

Диаграмма детализирует процесс подачи апелляции пользователем. Управление изначально передается блоку “Составление и отправка апелляции”. На выходе из блока выводится апелляция. Управляющие элементы блока - правила совершения сделки, механизмы – ПО.

Далее управление передается блоку “Добавление апелляции в базу данных апелляций”. Блок включает в себя функции, которые помещают полученную апелляцию в базу данных апелляций. Управляющие элементы блока - правила совершения сделки, механизмы – ПО.

Далее управление передается блоку “Рассмотрение апелляции сотрудником АК”. На вход блоку подается апелляция, которую выбрал сотрудник, на выходе из блока выводится вердикт сотрудника по апелляции. Сам блок содержит в себе функции для рассмотрения апелляции сотрудником АК а также отправки запроса на уведомление участников апелляции о решении. Управляющие элементы блока - правила совершения сделки, механизмы – Сотрудник АК и ПО.

Далее управление передается блоку “Оповещение пользователей по итогам рассмотрения апелляции”. Блок включает в себя функции, которые позволяют оповестить участников апелляции о принятом решении. На вход блок получает запрос об оповещении от сотрудника АК. На выходе из блока выводится решение по апелляции: апелляция принята или апелляция не принята. Управляющие элементы блока - правила совершения сделки, механизмы – ПО.

2.2. Описание потоков данных

2.2.1. Диаграмма потоков данных

На рисунке 11 представлена диаграмма потоков данных.

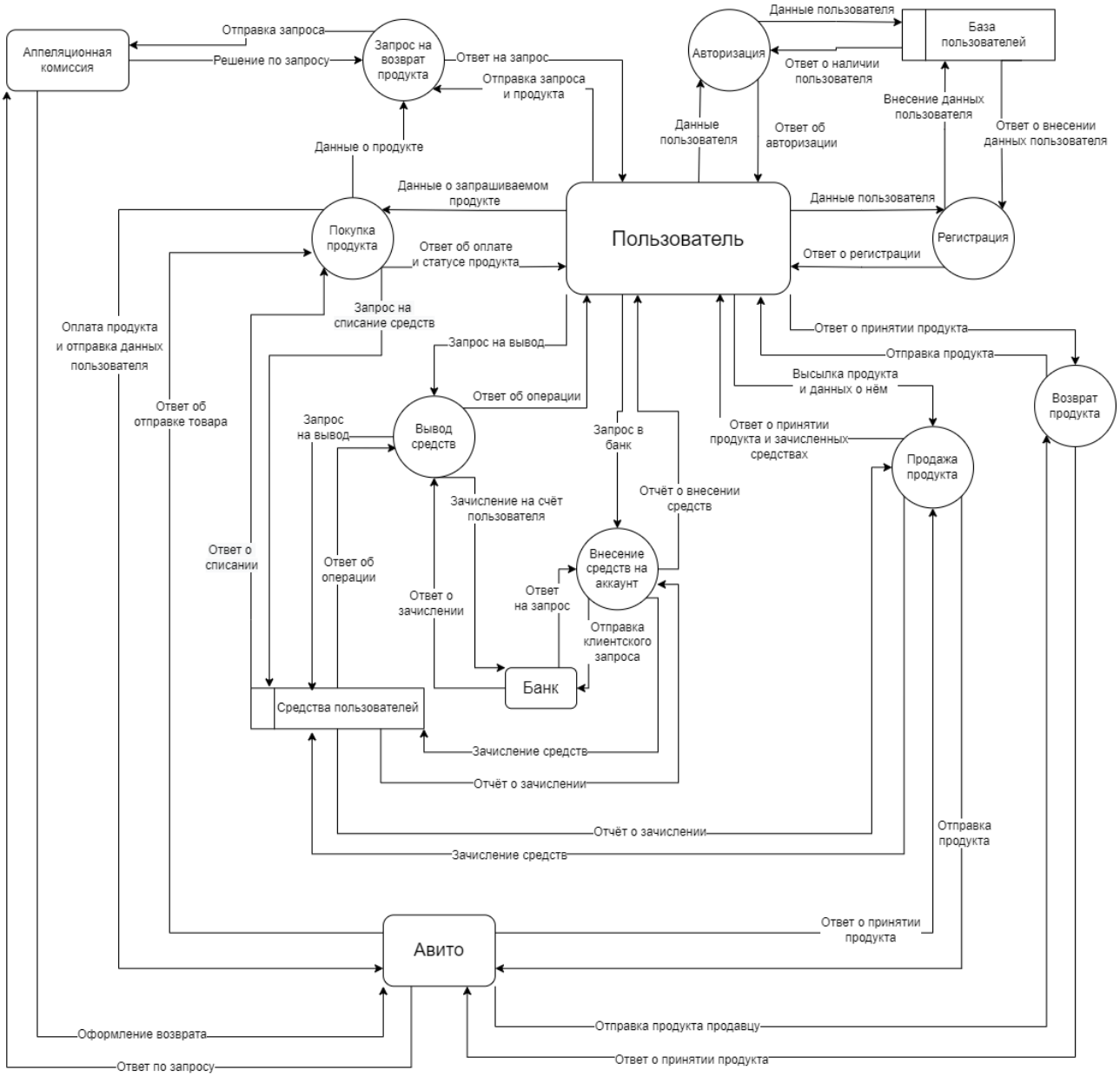

Рисунок 11 – Диаграмма потоков данных

Диаграмма потоков описывает все процессы и связанные с ними сущности в соответствии с техническим заданием, которые необходимы для корректной работы системы.

2.3. Модель предметной области

2.3.1. Диаграмма отношений сущностей

На рисунке 12 представлена диаграмма отношений сущностей.

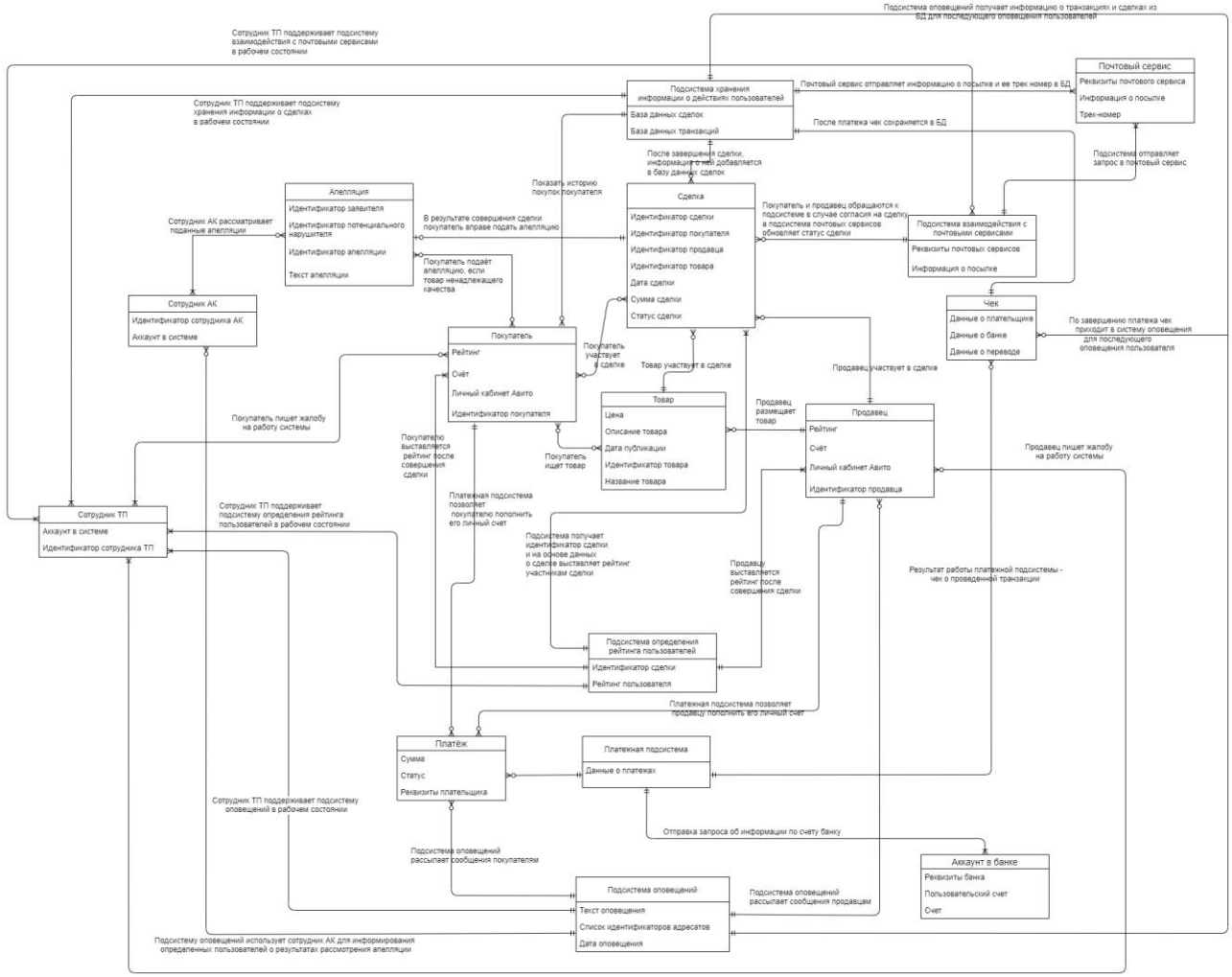

Рисунок 12 – Диаграмма отношений сущностей

Покупатель – хранит свой рейтинг, свой денежный счет, свой личный идентификатор в системе, а так же личные данные из ЛК Авито. Покупатель участвует в сделке, имеет возможность написать апелляцию по сделке, написать жалобу в службу технической поддержки, просмотреть свою историю покупок, обратившись к подсистеме хранения информации о сделках, оплатить товар используя платёжную подсистему, выбрать почтовый сервис в случае необходимости возврата товара (если сотрудниками АК была принята поданная апелляция), возможность выставить предварительную оценку продавцу, которая будет учитываться при вычислении рейтинга платежной подсистемой.

Продавец - хранит свой рейтинг, свой денежный счет, свой личный идентификатор в системе, а так же личные данные из ЛК Авито. Продавец имеет возможность просмотреть свою историю продаж, рейтинг покупателя (при совершении сделки). Также продавец имеет возможность выставить предварительную оценку покупателю по завершению сделки (если сделка завершилась успешно). В случае системных неполадок, продавец имеет возможность написать жалобу в службу технической поддержки.

Сделка – хранит 4 идентификатора: свой, продавца, покупателя, продукта, а также дату сделки, сумму сделки, статус сделки: сделка еще не начата, сделка на этапе до отправки товара, сделка на этапе отправки товара, сделка на этапе получения товара(в таком случае возможна апелляция по товару в течение одних календарных суток со дня получения товара), сделка завершена успешно, сделка завершена неуспешно. По завершению сделки, информация о ней добавляется в базу данных сделок.

Апелляция – хранит текст заявления об апелляции, и три идентификатора: идентификатор апелляции, заявителя и потенциального нарушителя. Апелляция подается покупателем, в случае если его не устраивает полученный товар. Апелляцию рассматривает сотрудник АК и выносит по ней свое решение, от которого зависит исход сделки.

Сотрудник АК – хранит личный идентификатор и аккаунт в системе, рассматривает апелляции пользователей, выносит по ним решения.

Сотрудник ТП - хранит личный идентификатор и аккаунт в системе, поддерживает систему в рабочем состоянии, а также рассматривает жалобы пользователей на работу системы, имеет возможность включения режима “песочницы” для безопасного редактирования страницы, на которой произошла ошибка.

Платежная подсистема – хранит реквизиты банков, позволяя покупателю выбрать интересующий его банк. Позволяет производить переводы денежных средств от пользователей на счет системы и обратно.

Подсистема оповещений – хранит текст оповещения, дату оповещения, а также идентификаторы адресатов, которым будет доставлено оповещение (покупатель и продавец, фигурирующие в одной апелляции). Суть подсистемы в том, чтобы оповещать пользователей о результатах рассмотрения апелляций.

Подсистема определения рейтинга пользователей – хранит идентификатор сделки и рейтинг этого пользователя. После завершения сделки вычисляет новые рейтинги покупателя и продавца исходя из успешности сделки: если сделка завершилась неуспешно по вине продавца(отправил некачественный товар), рейтинг продавца понижается. Если сделка завершилась успешно, рейтинги покупателя и продавца повышаются, с учетом выставленной покупателем оценки. В случае, если покупатель писал апелляцию, а она была отвергнута, то рейтинг покупателя уменьшится.

Подсистема взаимодействия с почтовыми сервисами – хранит реквизиты почтовых сервисов, которыми могут воспользоваться покупатели и продавцы для доставки продукта. При совершении сделки, продавец взаимодействует с данной подсистемой для отправки товара покупателю. Покупатель взаимодействует с данной подсистемой также при совершении сделки, а именно в случае, если ему необходимо отправить товар обратно продавцу при неуспешной сделке.

Подсистема хранения информации о сделках – содержит базу данных сделок, в которую добавляет новую сделку после того, как она состоится. Позволяет пользователям просматривать свои истории покупок и продаж.

Платёж - содержит в себе информацию о сумме денег, поступившей на оплату, статусе оплаты и реквизитах плательщика. Позволяет получать оплату от покупателей и переводить деньги продавцам с помощью платежной подсистемы. Информация о платеже поступает в подсистему оповещений.

Аккаунт в банке - информационный блок о пользователе, хранящий его банковские реквизиты с целью сохранения введённых данных для будущих покупок, его счёт в системе, и банковский счёт. Предоставляет платёжной подсистеме необходимую информацию для проведения платежей.

Чек - содержит данные о плательщике, банке, совершившем транзакцию и о переводе. Хранится в подсистеме хранения информации о действиях пользователей и использует информацию, выдаваемую платёжной подсистемой, а также направляется в подсистему оповещений для оповещения пользователей об успешности операции перевода.

Почтовый сервис - содержит реквизиты почтового сервиса, информацию о посылках и трек-номер. Информация о почтовых сервисах содержится в подсистеме хранения информации о действиях пользователей и используется подсистемой взаимодействия с почтовыми сервисами.

3. АРХИТЕКТУРНЫЕ РЕШЕНИЯ

3.1. Диаграмма Use Case

На рисунке 13 представлена диаграмма Use Case.

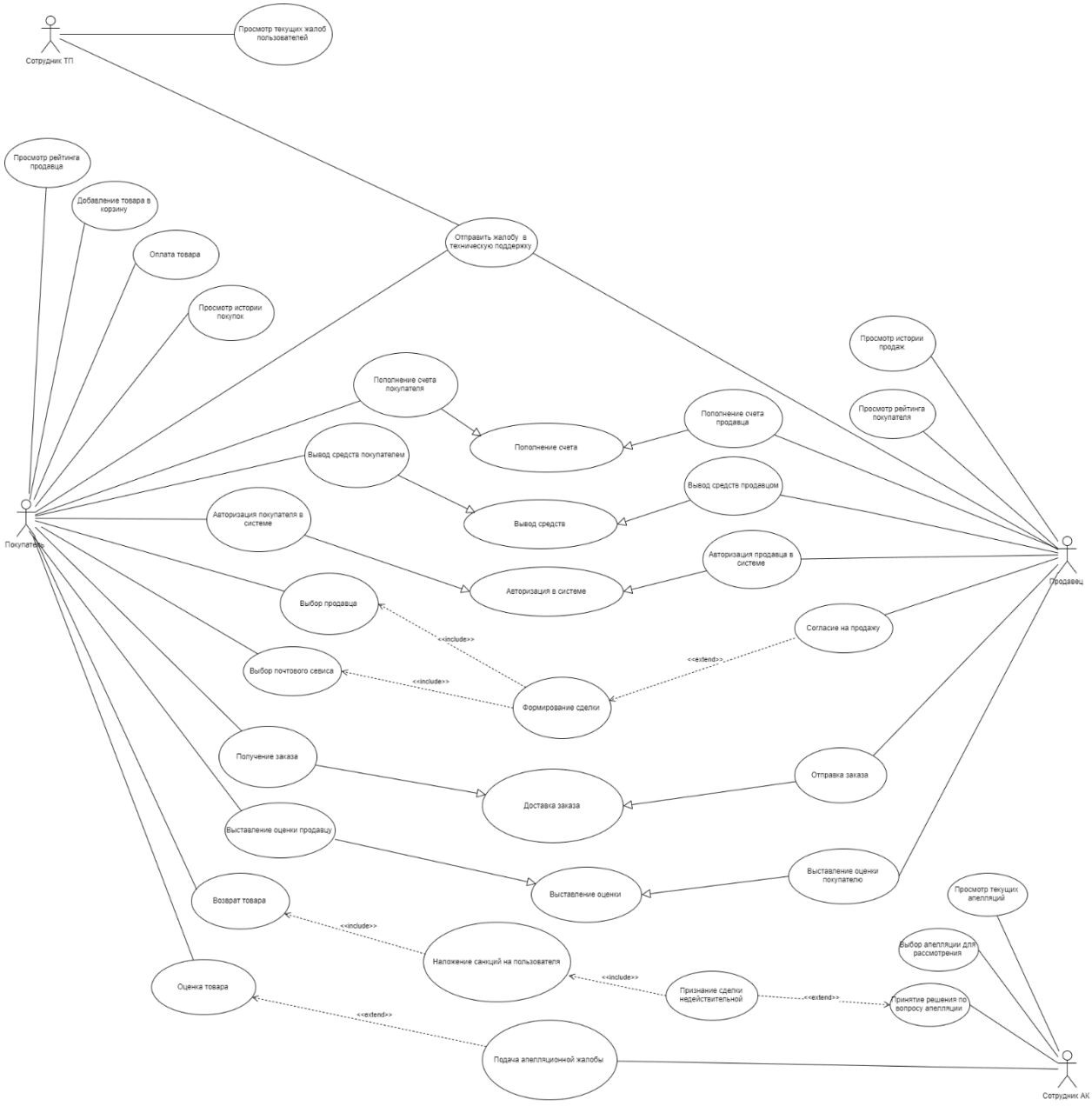

Рисунок 13 – Диаграмма Use Case

Покупатель имеет возможность просмотреть рейтинг продавца, добавить товар в корзину, оплатить товар, просмотреть историю своих покупок, пополнить свой счёт, вывести свои средства, авторизоваться в системе, выбрать продавца, у которого собирается купить товар, выбрать почтовый сервис, получить товар, выставить оценку продавцу, вернуть товар, оценить товар

(имеется в виду написать апелляцию, если товар не оправдал ожиданий), а также отправить жалобу в техническую поддержку.

Продавец имеет возможность просмотреть историю своих продаж, просмотреть рейтинг покупателя, пополнить свой счёт, вывести свои средства, авторизоваться в системе, подать согласие на продажу товара (если ему не нравится рейтинг покупателя, он имеет возможность отказаться от сделки), отправить товар, выставить оценку покупателю, а также отправить жалобу в техническую поддержку.

Сотрудник апелляционной комиссии имеет возможность просмотреть текущие апелляции, выбрать одну из апелляций и вынести решение по конкретной апелляции.

Сотрудник технической поддержки имеет возможность просмотреть список жалоб пользователей с целью их дальнейшего решения.

3.2. Диаграмма классов

На рисунке 14 представлена диаграмма классов.

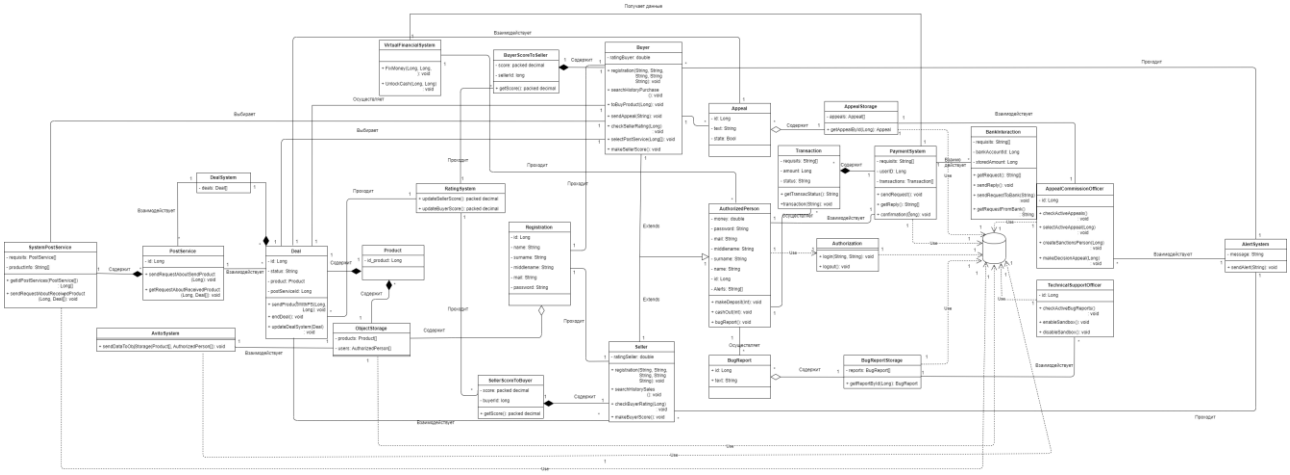

Рисунок 14 – Диаграмма классов

3.2.1. Описание классов

Класс Registration:

Класс представляет из себя регистрацию пользователя в системе.

Поля:

id – идентификатор пользователя

name – имя пользователя

surname – фамилия пользователя

middlename – отчество пользователя (опционально)

mail – почта пользователя

password – пароль

Класс AuthorizedPerson:

Класс представляет из себя шаблон класса пользователя (покупателя или продавца).

Поля:

money – деньги пользователя

password – пароль пользователя в системе

mail – почта пользователя

middlename – отчество пользователя

surname – фамилия пользователя

name – имя пользователя

id – уникальный идентификатор пользователя

alerts – оповещения

Методы:

makeDeposit() – пополнение счета

cashOut() – вывод средств

bugReport() – написание и отправка отчета об ошибке

Класс Seller:

Класс представляет из себя сущность продавца.

Поля:

ratingSeller – рейтинг продавца

Методы:

registration() – регистрация продавца

searchHistorySales() – просмотр истории опубликованных товаров

checkBuyerRating() – просмотр рейтинга покупателя

makeBuyerScore() – выставление оценки покупателю(от 1 до 5, вещественное)

Класс Buyer:

Класс представляет из себя сущность покупателя.

Поля:

ratingBuyer – рейтинг покупателя

Методы:

registration() – регистрация

searchHistoryPurchase() – просмотр истории покупок

toBuyProduct() – покупка товар

toSendAppeal() – написание и отправка апелляции о товаре

checkSellerRating() – просмотр рейтинга продавца

selectPostService() –выбор почтового сервиса

makeSellerScore() – выставление оценки продавцу(от 1 до 5, вещественное)

Класс Appeal:

Класс представляет из себя апелляцию.

Поля:

id – идентификатор апелляции

text – текст апелляции

state – состояние апелляции (принята/не принята)

Класс AppealStorage:

Класс представляет из себя хранилище апелляций.

Поля:

appeals – апелляции

Методы:

getAppealById() – функция получения апелляции по идентификатору.

Класс BugReport:

Класс представляет из себя отчет о неисправностях АС Buy&Sell

Поля:

id – идентификатор отчета об ошибке

text – текст отчета об ошибке

Класс BugReportStorage:

Класс представляет из себя хранилище отчётов об ошибках АС Buy&Sell.

Поля:

reports – отчеты об ошибке

Методы:

getReportById() – функция получения отчета об ошибке по идентификатору

Класс AlertSystem:

Класс представляет из себя систему оповещений.

Поля:

message – текст сообщения

Методы:

sendAlert() – отправка сообщения каждому из объектов из массива объектов, указанного в аргументе

Класс DealSystem:

Класс представляет из себя хранилище сделок.

Поля:

deals[] – массив сделок

Класс AppealCommissionOfficer:

Класс представляет из себя сотрудника апеляционной комиссии.

Поля:

id – идентификатор сотрудника апеляционной комиссии

Методы:

checkActiveAppeals() - просмотр актуальных апелляций покупателей

selectActiveAppeal() – выбор актуальной апелляции покупателя

createSanctionPerson() – наложение санкций пользователю.

makeDecisionAppeal() – вынесение решения по апелляции

Класс TechnicalSupportOfficer:

Класс представляет из себя сотрудника технической поддержки.

Поля:

Id – идентификатор сотрудника технической поддержки

Методы:

checkActiveBugReports() – просмотр актуальных отчетов об ошибке.

enableSandbox() – включение режима “песочницы”

Класс Authorization:

Класс представляет из себя авторизацию.

Методы:

login() – вход на сайт

logout() – выход из сайта

Класс Transaction:

Класс представляет из себя транзакцию.

Поля:

requisuts – реквизиты транзакции

amount – сумма транзакции

status – статус транзакции

Методы:

transaction() – выполнение транзакции

Класс PaymentSystem:

Класс представляет из себя систему пополнения кошелька. Взаимодействует с классом BankInteraction, содержит Transaction. Взаимодействует с классом AuthorizedPerson.

Поля:

requisits – реквизиты транзакции (из класса Transaction)

userId – идентификатор пользователя, который совершает транзакцию

transactions – массив платежей/выводов средств пользователя

Методы:

sendRequest() – отправка запроса в банк о соблюдении условий для совершения корректной транзакции.

getReply() – получение ответа от банка на запрос о соблюдении условий для совершения корректной транзакции.

confirmation() – отправка пользователю(после getReply()) сообщения о подтверждении/отклонении транзакции

Класс BankInterraction:

Класс представляет из себя связь с программной системой банка. Взаимодействует с классом PaymentSystem.

Поля:

requisits – реквизиты транзакции (из класса PaymentSystem)

bankAccountId – идентификатор счета в банке

storedAmount – количество средств на счете

Методы:

getRequest() – получение запроса от платёжной системы о корректности транзакции

sendReply() – отправка ответа банка о корректности транзакции в платёжную систему.

sendRequestToBank() – отправка запроса банку о транзакции.

getRequestFromBank() – получение запроса банка о транзакции.

Класс VirtualFinancialSystem:

Класс представляет из себя систему оплаты товара покупателем.

Методы:

fixMoney() – изменение счета указанного пользователя на указанное число

unlockCash() – размораживание средств продавца на определенную сумму в конце сделки.

Класс ObjectStorage:

Класс представляет из себя хранилище пользователей и товаров.

Поля:

products – массив товаров

users – массив пользователей

Класс AvitoSystem:

Класс представляет из себя связь системы с программной системой Avito. Взаимодействует с классом ObjectStorage.

Методы:

sendDataToObjStorage() – импорт данных из Авито и переназначение, соответственно импортированных пользователей и товаров на массивы products и users из ObjectStorage.

Класс BuyerScoreToSeller:

Класс представляет из себя оценку покупателя.

Поля:

score – оценка покупателя

Методы:

getScore – получение оценки продавца покупателем

Класс SellerScoreToBuyer:

Класс представляет из себя оценку продавца.

Поля:

score – оценка продавца

Методы:

getScore – получение оценки покупателя продавцом

Класс RatingSystem:

Класс представляет из себя рейтинговую систему.

Методы:

updateSellerScore() – обновление оценки продавца

updateBuyerScore() – обновление оценки покупателя

Класс Product:

Класс представляет из себя товар.

Поля:

id – идентификатор продукта

Класс Deal:

Класс представляет из себя сделку, в которой участвую товар, покупатель и продавец.

Поля:

id –идентификатор сделки

status – статус сделки

product – приобретаемый товар

Методы:

sendProductPS() – отправка запроса почтовому сервису с входными данными соответствующего идентификатора сделки и идентификатора почтового сервиса.

endDeal() – завершение сделки.

updateDealSystem() – обновление данных DealSystem. При создании сделки, данная сделка добавляется в DealSystem. При её изменении, соответствующая сделка из DealSystem меняется, в соответствии с данной.

Класс PostService:

Класс представляет из себя почтовый сервис.

Поля:

deal – объект класса сделка

Методы:

sendRequestAboutSendProduct() – отправка запроса почтовому сервису об отправке товара с последующим изменением статуса сделки на “Сделка на этапе после отправки товара”

getRequestAboutReceivedProduct() – получение от почтового сервиса ответа о том что товар доставлен покупателю с последующим изменением статуса сделки на “Сделка на этапе получения товара”

Класс SystemPostService:

Класс представляет из себя систему взаимодействия с почтовыми сервисами.

Поля:

requisits – реквизиты почтового сервиса

productInfo – информация о товаре.

Методы:

getIdPostServices() – получение массива идентификаторов почтовых сервисов.

sendRequestAboutReceivedProduct() – отправка запроса почтовому сервису о том, что товар доставлен

3.2.2. Описание процессов

3.2.2.1. Заказ товара

При выборе товара на Авито, покупатель переадресовывается на сайт Buy&Sell, где происходит автоматическая регистрация с помощью класса Registration, в случае, если покупатель впервые входит на Buy&Sell. Регистрация происходит с учётом данных, полученных с аккаунта покупателя, зарегистрированного на Авито. Далее, происходит автоматическая авторизация пользователя с помощью класса Authorization. Таким образом, вошедший пользователь идентифицируется объектом класса Buyer, который наследуется от класса AuthorizedPerson.

Вместе с переадресацией конкретного покупателя, система получает информацию о интересующем покупателя товаре. Далее, формируется сделка (создается экземпляр класса Deal), которая содержит информацию о покупателе, товаре и продавце, который разместил товар. Пользователю отображается данная сделка и её текущий статус “Ещё не начата”(Поле: status). Также, пользователь в этот момент времени может посмотреть текущий рейтинг продавца (метод checkSellerRating), а также выбрать почтовый сервис (участвуют методы PostServiceSystem и PostService).

Для заказа товара покупатель должен изначально пополнить счет (метод makeDeposit() класса AuthorizedPerson). Далее покупатель с помощью метода transaction() посылает экземпляру класса PaymentSystem сообщение о желании совершить оплату товара по определенной цене (хранит поле amount) с реквизитами (хранит поле requisits). В свою очередь класс PaymentSystem с помощью метода sendRequest() посылает запрос на проведение операции пользователя (хранит поле UserID) с определенными реквизитами (хранит поле requisits) в класс BankInteraction, который в свою очередь отправляет запрос на проведение платежа в банк с помощью метода sendRequestToBank().

После пополнения счета покупатель может нажать кнопку “Оплатить”, после чего вызовется метод toBuyProduct(). При вызове метода происходит обращение к VirtualFinancialSystem(), где происходит изменение счета пользователя:

- из текущего баланса покупателя(money) вычитается сумма сделки
- к замороженному балансу продавца (moneyReserved) прибавляется сумма сделки

, а также изменение создавшегося экземпляра класса Deal:

- статус сделки Deal (поле: status) изменяется на “Сделка на этапе до отправки товара”.

После оплаты покупателем товара на сайте Buy&Sell происходит отправка товара, статус сделки(поле: status) меняется на “Сделка на этапе отправки товара”, PostService вызывает метод

sendRequestAboutSendProduct(посылается запрос почтовому сервису об отправке товара), при этом, массив сделок запрос получает из DealSystem. Почтовый сервис обрабатывает запрос и высылает товар.

После доставки товара покупателю, PostService сообщает о успешной отправке методом getRequestAboutReceivedProduct(), при этом, массив сделок запрос получает из DealSystem. После чего статус сделки, статус экземпляра класса Deal (поле: status) изменяется на “Сделка на этапе получения товара.”.

Если покупатель удовлетворен полученным товаром (он не подавал апелляцию в течении одних календарных суток) сделка, экземпляр класса Deal, меняет статус (поле: status) на “Сделка завершена успешно”, а также вызывает метод endDeal(), предназначенный для завершения сделки и удаления экземпляра класса, который вызвал эту функцию. Во время вызова endDeal() также изменяется счёт продавца (экземпляра класса Seller, участвующего в сделке): к полю money прибавляется значение поля moneyReserved, а поле moneyReserved обнуляется.

3.2.2.2. Рейтинговая система

Расчет оценок

После завершения сделки рассчитывается рейтинг продавца и покупателя. В классе RatingSystem вызываются методы обновления рейтинга продавца и покупателя (updateBuyerScore() и updateSellerScore()). Для обоих пользователей рейтинг рассчитывается, как среднее арифметическое оценок последних 20 сделок.

Оценка за прошедшую сделку

Для получения оценки за прошедшую сделку используются методы getScore() классов SellerScore и BuyerScore. Методы getScore() классов SellerScore и BuyerScore формируют оценки покупателя и продавца по следующему принципу: оценка складывается с учетом знаков из двух оценок – оценки пользователя и оценки системы.

Оценка покупателя (метод getScore() класса SellerScore)

Продавец оценивает покупателя по шкале от -5 до 5 (метод makeBuyerScore()). Система смотрит на наличие апелляций и решение по ним. Если апелляция была подана и удовлетворена, то система оценит покупателя в этой сделке по шкале от -5 до 5 на максимальный балл. Если апелляция была отклонена, система учтет количество апелляций отклоненных до этого, в случае если таковых было меньше трёх, то оценка системы будет равна 3, если отклоненных апелляций было три или больше, то оценка системы будет считаться, как «оценка= количество отклоненных апелляций * (-1)», но оценка по модулю не может быть больше 5.

В конце оценки выставленные продавцом и системой складываются с учетом знаков, результат суммы по шкале от -10 до 10 будет оценкой покупателя за прошедшую сделку.

Оценка продавца (метод getScore() класса BuyerScore)

Продавец оценивает покупателя по шкале от -5 до 5 (метод makeSellerScore()). Система смотрит на наличие апелляций и решение по ним. Если апелляция была подана и отклонена, то система оценит продавца в этой сделке по шкале от -5 до 5 на максимальный балл. Если апелляция была удовлетворена, система учтет количество апелляций удовлетворенных до этого, в случае если таковых было меньше трёх, то оценка системы будет равна 1, если удовлетворенных апелляций было три или больше, то оценка системы будет считаться, как «оценка= количество удовлетворенных апелляций * (-2)», но оценка по модулю не может быть больше 5.

В концFe оценки выставленные покупателем и системой складываются с учетом знаков, результат суммы по шкале от -10 до 10 будет оценкой продавца за прошедшую сделку.

3.2.2.3. Техническая поддержка и апелляционная комиссия

Если при получении товара покупатель не удовлетворен им, он имеет право написать апелляцию по сделке. При написании апелляции создается новый экземпляр класса Appeal, где поле text соответствует содержанию

апелляции покупателя. Экземпляр класса Appeal содержится в AppealStorage, хранилище апелляций, который доступен сотруднику апелляционной комиссии (экземпляр класса: AppealCommisionOfficer). Алгоритм работы сотрудника апелляционной комиссии следующий: изначально он получает список всех актуальных апелляций (метод: checkActiveAppeals()), далее, выбирает одну из них (метод: selectActiveAppeal()), после чего выносит решение по апелляции(метод: makeDecisionAppeal()), в зависимости от которого накладывает/не накладывает санкции на пользователей(createSanctionsPerson()). При вынесении решения об апелляции высылаются уведомление покупателю и продавцу, участвующим в сделке, о соответствующем решении с помощью AlertSystem (метод: sendAlert).

При взаимодействии покупателя/продавца с сайтом могут возникнуть неполадки. В таком случае авторизованный пользователь (любой из потомков класса AuthorizedPerson или покупатель/продавец) может написать отчет об ошибке. При написании отчета об ошибке создается новый экземпляр класса BugReport, где поле text соответствует содержанию отчета об ошибке авторизованного пользователя (любой из потомков класса AuthorizedPerson или покупатель/продавец). Экземпляр класса BugReport содержится в BugReportStorage, хранилище отчетов об ошибке, который доступен сотруднику технической поддержки (экземпляр класса: TechnicalSupportOfficer). Алгоритм работы сотрудника технической поддержи следующий: изначально он получает список всех актуальных отчетов об ошибке (метод: checkActiveBugReports()), затем, он включает режим песочницы(метод: enableSandbox()), который позволяет редактировать страницу сайта. Для того, чтобы выйти из режима песочницы, сотрудник технической поддержки (экземпляр класса: TechnicalSupportOfficer) использует метод disableSandbox().

3.2.2.4. Импорт данных из Авито

Импорт данных о продавцах, их товарах и пользователей происходит посредством экземпляров класса ObjectStorage и класса AvitoSystem. Для получения соответствующих данных используется метод sendDataToObjStorage() экземпляром класса AvitoSystem. Соответствующие данные (о пользователях и товарах) приходят в класс ObjectStorage и записываются в соответствующие поля: данные о пользователях записываются в поле users, данные о товарах записываются в поле products.

3.2.3. Отношения классов

Класс Buyer связан с классом Registration ассоциацией один к одному, потому что Buyer проходит регистрацию в системе. Переходят следующие поля: name, surname, middlename, id, password, mail.

Класс Seller связан с классом Registration ассоциацией один к одному, потому что Buyer проходит регистрацию в системе. Переходят следующие поля: name, surname, middlename, id, password, mail.

Seller и Buyer расширяют класс AuthorizedPerson, поскольку продавец и покупатель являются авторизованными пользователями.

Класс AuthorizedPerson зависит от Authorization связью один к одному. Чтобы реализовать авторизованного пользователя, нужно использовать авторизацию.

Класс AuthorizedPerson с помощью метода makeDeposit() осуществляет Transaction(пополняет счет), связан ассоциацией один к одному.

Класс PaymentSystem с помощью метода confirmation() подтверждает транзакцию пользователя, поэтому PaymentSystem связан с AuthorizedPerson ассоциацией один ко многим.

Buyer содержит BuyerScoreToSeller, покупатель содержит информацию о выставленных оценках разным продавцам.

Seller содержит SellerScoreToBuyer, продавец содержит информацию о выставленных оценках разным покупателям.

Класс RatingSystem связан с классом BuyerScoreToSeller ассоциацией, поскольку BuyerScoreToSeller оказывает влияние на RatingSystem при формировании рейтинга(метод: updateSellerScore()) продавца (поле ratingSeller экземпляра класса Seller)

Класс RatingSystem связан с классом SellerScoreToBuyer ассоциацией, поскольку SellerScoreToBuyer оказывает влияние на RatingSystem при формировании рейтинга(метод: updateBuyerScore()) продавца (поле ratingBuyer экземпляра класса Buyer)

Класс Product связан с классом Deal композицией, поскольку товар является частью сделки, без которой сделка невозможна.

Класс Product связан с классом ObjectStorage, поскольку хранилище пользователей и товаров содержит товары, импортированные из Авито.

Класс Registration связан с классом ObjectStorage, поскольку хранилище пользователей и товаров содержит данные о пользователях, которые импортируются из Авито.

Класс AvitoSystem связан с классом ObjectStorage ассоциацией, поскольку AvitoSystem передает данные, пользователей и товаров из Авито, в ObjectStorage, соответственно с помощью метода sendDataToObjStorage().

Класс Buyer связан с классом Deal ассоциацией.

Класс SystemPostService связан с классом Buyer ассоциацией, SystemPostService отправляет массив идентификаторов почтовых сервисов покупателю, который участвует у покупателя при выборе почтового сервиса.

Класс DealSystem содержит массив Deal.

Класс PostService связан с классом Deal ассоциацией один ко многим. При выполнении методов sendRequestAboutSendProduct() и getRequestAboutReceivedProduct() изменяются параметры сделки Deal.

Класс VirtualFinancialSystem связан с классом AuthorizedPerson ассоциацией один ко многим. Данная связь необходима для реализации покупок пользователя внутри системы.

Класс PaymentSystem связан с классом VirtualFinancialSystem ассоциацией один к одному. Необходимо для отправки от PaymentSystem к VirtualFinancialSystem данных и обратно: от VirtualFinancialSystem к PaymentSystem.

Класс PaymentSystem содержит Transaction в массиве transactions[].

Класс PaymentSystem связан с классом AuthorizedPerson ассоциацией один ко многим. Это нужно для отправки пользователю подтверждения/отказы совершения оплаты с помощью метода confirmation()

Класс Transaction связан с классом AuthorizedPerson ассоциацией “осуществляет” многие к одному. Это нужно для того, чтобы пользователь сообщил в PaymentSystem о желании совершить оплату товара.

Класс BankInteraction связан с классом PaymentSystem ассоциацией многие к одному. Данная связь предназначена для передачи данных о совершаемой оплате (реализуемой экземпляром класса Transaction) от PaymentSystem в BankInteraction.

Класс AuthorizedPerson связан с классом BugReport ассоциацией “осуществляет” один ко многим. Эта связь отражает возможность создания экземпляра класса BugReport, отчета об ошибке, пользователем (покупателем/продавцом)

Класс BugReportStorage содержит экземпляры класса BugReport.

Класс TechnicalSupportOfficer связан с классом BugReportStorage ассоциацией один к одному. Сотрудник технической поддержки (TechnicalSupportOfficer) получает данные отчетов об ошибке из хранилища отчетов об ошибке(поле reports из экземпляра класса BugReportStorage).

Класс Buyer связан с классом Appeal ассоциацией “осуществляет” один ко многим. Эта связь отражает возможность создания экземпляра класса Appeal, апелляции по товару, покупателем.

Класс AppealStorage содержит экземпляры класса Appeal.

Класс AppealCommissionOfficer связан с классом AppealStorage ассоциацией один к одному. Сотрудник апелляционной комиссии

(AppealCommissionOfficer) получает данные апелляций из хранилища апелляций(поле appeals из экземпляра класса AppealStorage).

Класс AlertSystem связан с классом AppealCommissionOfficer связью один ко многим. Данная связь нужна для реализации возможности отправки оповещения(поле message в экземпляре класса AlertSystem) сотрудником апелляционной комиссии(экземпляр класса AppealCommissionOfficer)

Класс AlertSystem связан с классом Buyer ассоциацией один ко многим. Это необходимо для отправки соответствующего оповещения(поле message в экземпляре класса AlertSystem) покупателю(экземпляр класса Buyer), который участвовал в соответствующей сделке.

Класс AlertSystem связан с классом Seller ассоциацией один ко многим. Это необходимо для отправки соответствующего оповещения(поле message в экземпляре класса AlertSystem) продавцу(экземпляр класса Seller), который участвовал в соответствующей сделке.

3.3. Диаграмма объектов

На рисунке 15 показана диаграмма объектов процесса заказа товара.

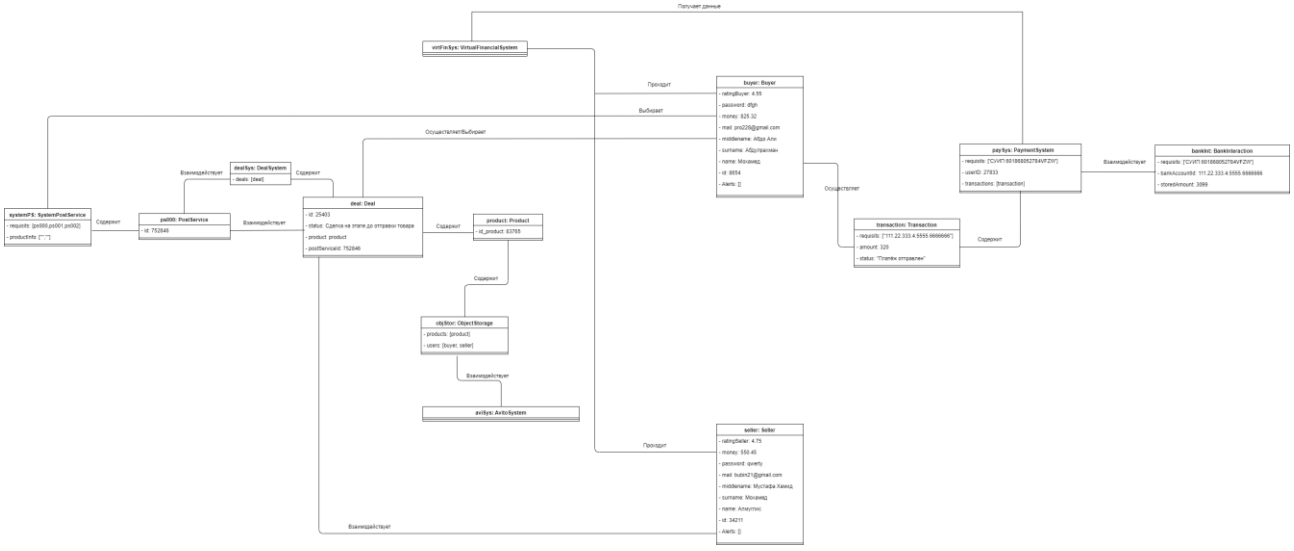

Рисунок 15 – Диаграмма объектов процесса заказа товара

Объект buyer связан ассоциацией с объектом deal, т.к. покупатель участвует в сделке и взаимодействует с ней.

Объект product связан с объектом deal композицией, т.к. товар является частью сделки.

Объект deal содержится в объекте dealSys, т.к. dealSys выполняет функционал хранилища всех сделок, поэтому объекты связаны композицией.

Объект ps000 связан с объектом dealSys ассоциацией, т.к. объект ps000 должен получить массив сделок при взаимодействии с определенной сделкой.

Объект ps000 связан с объектом deal ассоциацией, т.к. почтовый сервис, представленный объектом ps000 связан со сделкой (объект deal) напрямую.

Объект systemPS связан с объектом ps000 композицией, т.к. объект systemPS является хранилищем почтовых сервисов (объектов класса PostService).

Объект seller связан ассоциацией с объектом deal, т.к. продавец участвует в сделке.

Объект objStor содержит информацию о товарах (экземпляры класса Product, например – объект product) и пользователях – объекты классов Seller и Buyer.

Объект aviSys связан с объектом objStor ассоциацией, т.к. aviSys передает данные о товарах и пользователях из Авито в objStor.

Объекты buyer и seller связаны с объектом virtFinSys ассоциацией, т.к. объект virtFinSys взаимодействует с buyer и seller: изменяет значение денежных средств.

Объект virtFinSys связан с объектом paySys ассоциацией, т.к. он взаимодействует с paySys при совершении транзакции пользователем.

Объект buyer связан с объектом transaction ассоциацией, т.к. покупатель осуществляет транзакции в системе.

Объект paySys содержит объект transaction, т.к. одна из его функций – хранение транзакций, произведенных на сайте.

Объект bankInt связан с объектом paySys ассоциацией, т.к. объект paySys при совершении платежа связывается с банком для произведения оплаты.

На рисунке 16 показана диаграмма объектов процесса написания апелляции.

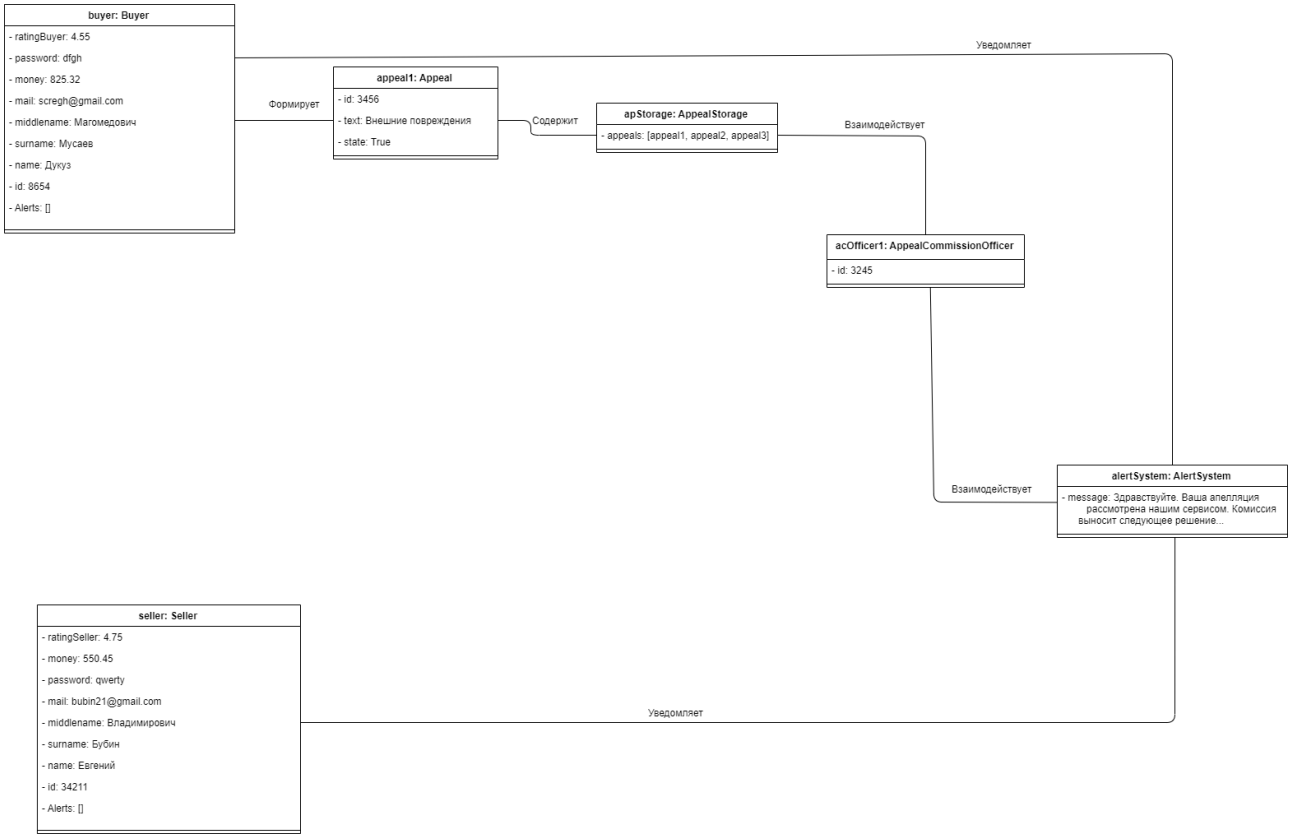

Рисунок 16 - Диаграмма объектов процесса написания апелляции

Объект buyer связан ассоциацией с объектом appeal1, т.к. право написания апелляции есть у покупателя.

Объект appeal1 содержится в apStorage, т.к. apStorage выполняет функции хранилища апелляций.

Объект apStorage связан с объектом acOfficer1 ассоциацией, т.к. объект acOfficer1 взаимодействует с объектом apStorage: получает апелляцию и просматривает список апелляций.

Объект acOfficer1 связан с объектом alertSystem ассоциацией, т.к. он взаимодействует с объектом alertSystem: с помощью системы оповещения (представленной в виде объекта alertSystem) уведомляет пользователей о результатах по апелляции.

Объект buyer связан ассоциацией с объектом alertSystem, т.к. alertSystem отправляет уведомление покупателю (объект buyer).

Объект seller связан ассоциацией с объектом alertSystem, т.к. alertSystem отправляет уведомление продавцу (объект seller).

На рисунке 17 представлена диаграмма объектов процесса написания отчёта об ошибке.

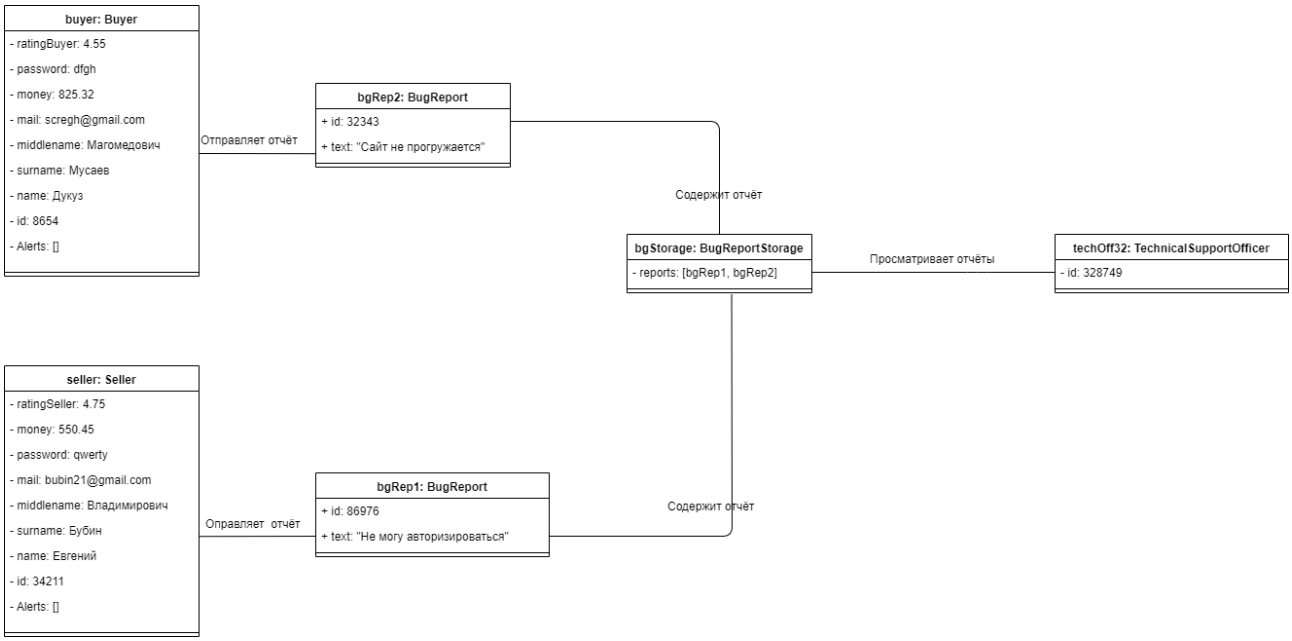

Рисунок 17 - диаграмма объектов процесса написания отчёта об ошибке

Объект buyer связан с объектом bgRep2 ассоциацией, т.к. buyer создает отчёт об ошибках в работе системы.

Объект bgRep2 связан с объектом bgStorage агрегацией, т.к. объект bgRep2 содержится в bgStorage.

Объект bgStorage связан с объектом techOff32 ассоциацией, т.к. объект techOff32 взаимодействует с объектом bgStorage: получает отчёт об ошибке и просматривает список отчётов.

Объект seller связан с объектом bgRep1 ассоциацией, т.к. seller создает отчёт об ошибках в работе системы.

Объект bgRep1 связан с объектом bgStorage агрегацией, т.к. объект bgRep1 содержится в bgStorage.

3.4. Диаграмма состояний

На рисунке 18 представлена диаграмма состояния сделки.

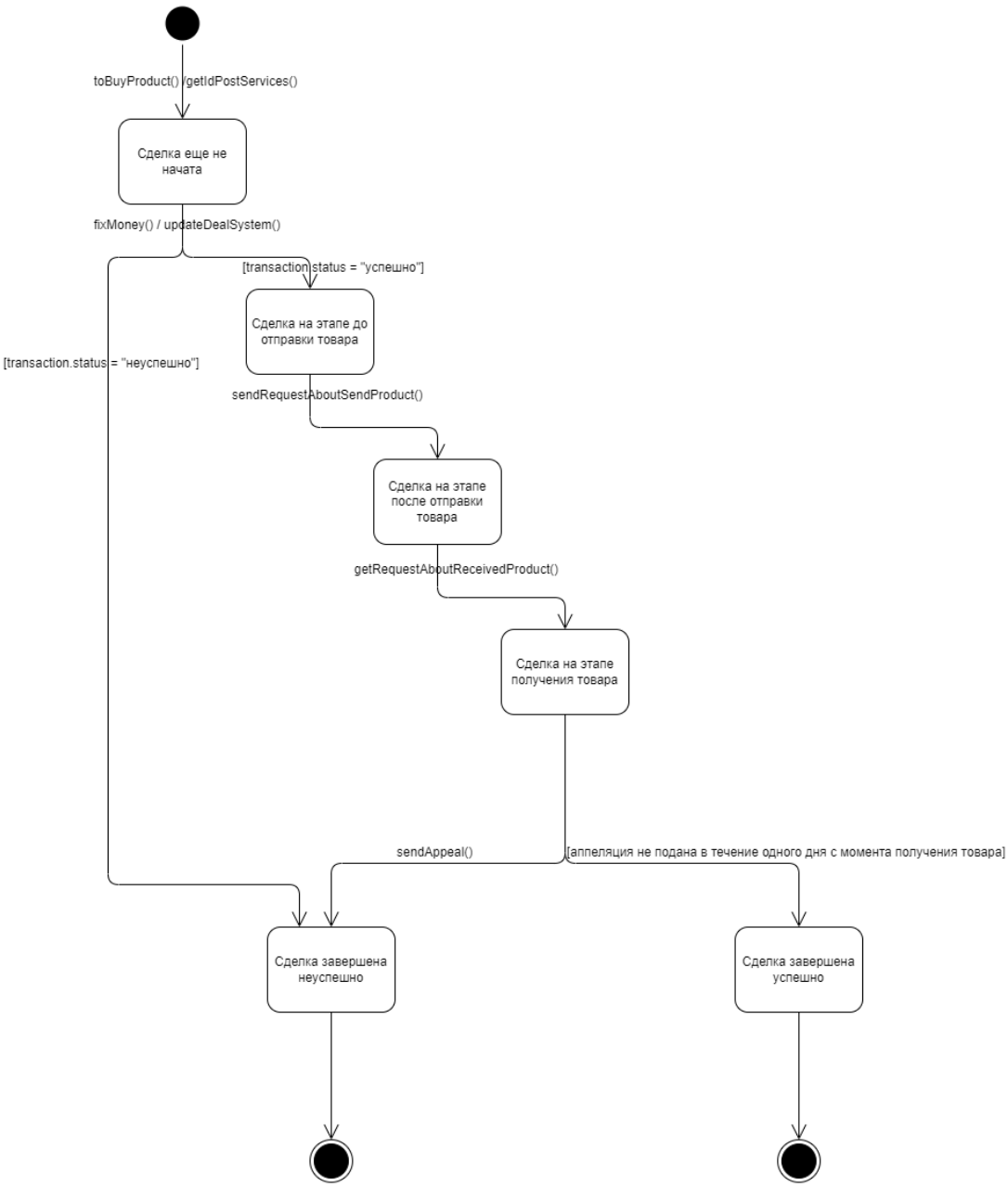

Рисунок 18 – Диаграмма состояния сделки

На рисунке 18 представлена диаграмма состояний сделки. В начальный момент времени сделка в состоянии “Сделка ещё не начата”. После того, как покупатель сделает перевод средств на счет системы и выберет почтовый сервис, сделка будет иметь статус “Сделка на этапе до отправки товара”. Далее, после обработки заказа и последующей отправке товара почтовым сервисом

соответствующей сделке устанавливается статус “Сделка на этапе после отправки товара”. После доставки товара покупателю и последующего запроса от почтового сервиса об успешной доставке соответствующей сделке устанавливается статус “Сделка на этапе получения товара”. Если в течение дня покупатель не писал апелляцию сделке устанавливается статус: “Сделка завершена успешно”, после чего переход в заключительное состояние, иначе сделке устанавливается статус “Сделка завершена неуспешно”, после чего также осуществляется переход в заключительное состояние.

Если в начальный момент времени, когда сделка находится в состоянии “Сделка ещё не начата”, пополнение средств со стороны покупателя прошло неуспешно, то сделке присваивается статус “Сделка завершена неуспешно” с последующим переходом в заключительное состояние.

На рисунке 19 представлена диаграмма состояния апелляции.

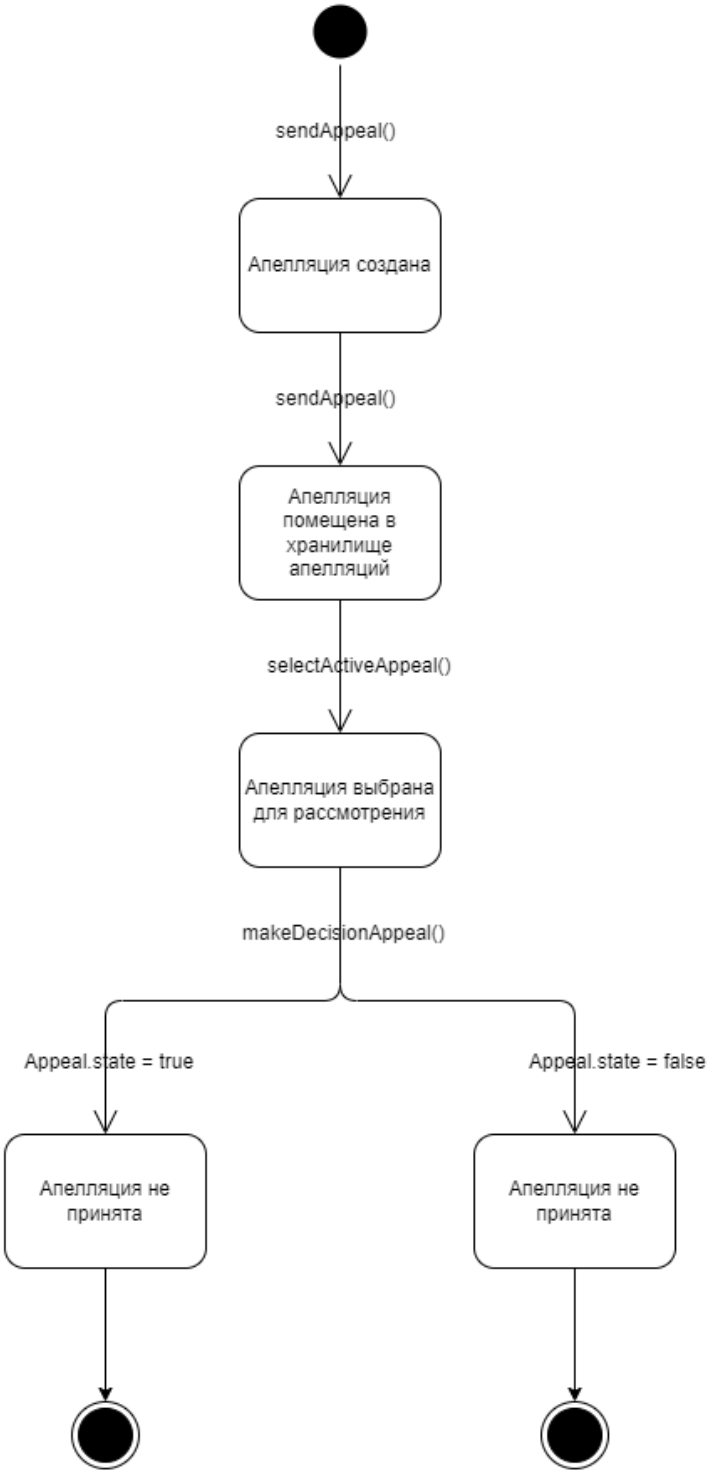

Рисунок 19 – Диаграмма состояния апелляции

На рисунке 19 представлена диаграмма состояний апелляции. В начальный момент времени, после отправки апелляции покупателем, апелляция имеет статус “Апелляция создана”. Когда апелляция будет создана, она будет

помещена в хранилище апелляций, а также иметь статус “Апелляция помещена в хранилище апелляций”. После того, как сотрудник апелляционной комиссии выберет соответствующую апелляцию, данной апелляции будет установлен статус “Апелляция выбрана для рассмотрения”. Затем, после того, как сотрудник апелляционной комиссии выносит решение по поводу апелляции, сделке устанавливается один из двух статусов: “Апелляция не принята” – в случае, если апелляция отклонена, после чего, перевод в заключительное состояние, или “Апелляция принята” – в случае, если апелляция одобрена, после чего, также перевод в заключительное состояние.

На рисунке 20 представлена диаграмма состояния платежа.

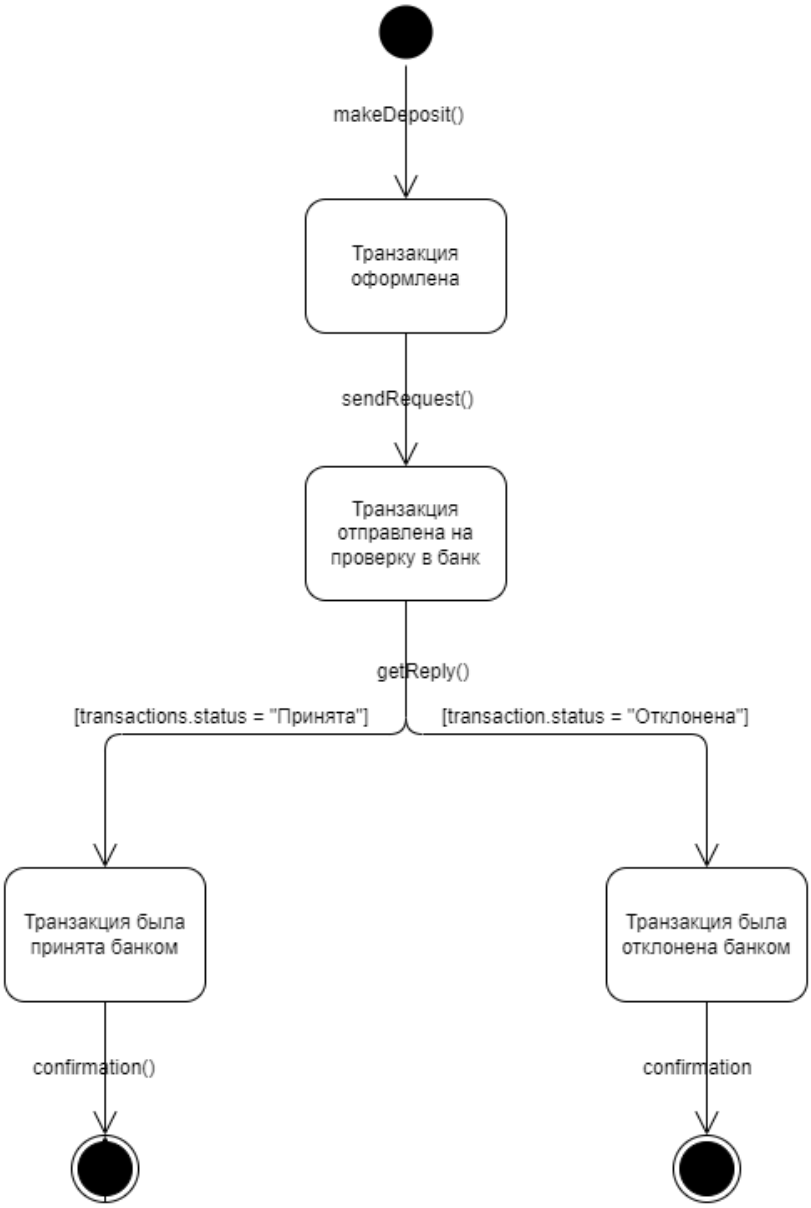

Рисунок 20 – Диаграмма состояния платежа

На рисунке 20 представлена диаграмма состояний платежа. В начальный момент времени, платеж имеет статус “Транзакция оформлена”. После отправки платежной системой запроса в банк, платежу устанавливается статус “Транзакция отправлена на проверку в банк”. После чего банк посылает ответ с помощью метода getReply(). В этот момент, если транзакция была принята банком, то платежу присваивается статус “Транзакция была принята банком”,

иначе платежу устанавливается статус “Транзакция была отклонена банком”. После чего осуществляется переход в заключительное состояние с последующим вызовом метода confirmation().

3.5. Диаграмма деятельности

На рисунке 21 представлена диаграмма деятельности.

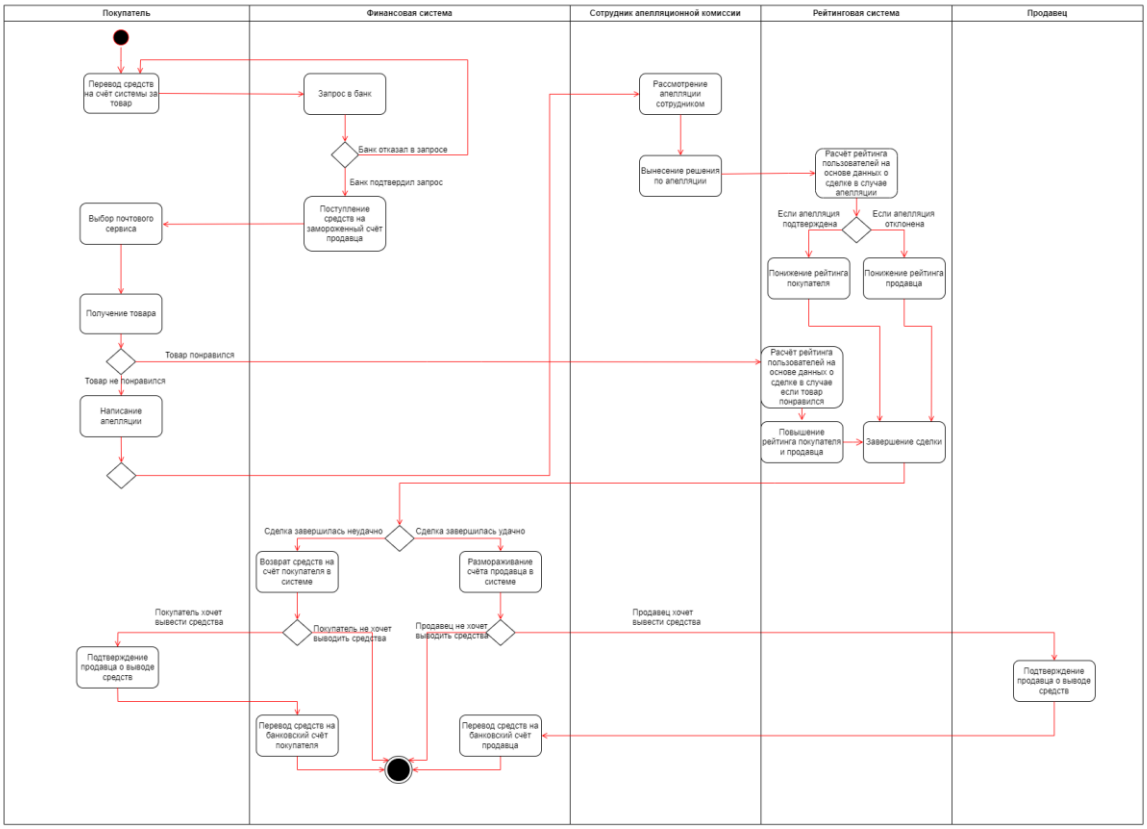

Рисунок 21 – Диаграмма деятельности

На рисунке 1 представлена диаграмма деятельности для оформления и выполнения заказа, которая иллюстрирует переход управления в частях подсистемы при совершении сделки.

Изначально, при авторизации на сайте, покупатель осуществляет перевод средств на счет системы за товар, путем отправки запроса в банк (Финансовая система). После отправки запроса ожидается ответ от банка: отказ или подтверждение. В случае отказа происходит возврат к действию “Перевод средств на счет системы за товар”, в случае подтверждения необходимая сумма за товар поступает на замороженный счет продавца, после чего покупателю предоставляется выбор почтового сервиса. После выбора почтового сервиса следует отправка товара и его получение. После получения товара, следуют две ситуации: либо покупателю понравился товар, либо не понравился. В случае,

если товар понравился, то происходит расчёт рейтинга пользователей: рейтинг покупателя и продавца повышаются, после чего завершение сделки. В случае, если товар не понравился, покупатель пишет апелляцию, которую рассматривает сотрудник апелляционной комиссии. После рассмотрения апелляции, сотрудник апелляционной комиссии выносит решение и накладывает определенные санкции на пользователей: При подтверждении апелляции понижается рейтинг продавца, при отклонении апелляции понижается рейтинг покупателя. После этого также происходит завершение сделки.

При завершении сделки могут быть две ситуации: либо сделка завершилась успешно, либо сделка завершилась неуспешно. В случае, если сделка завершилась неуспешно, то покупателю предлагается возврат средств, где он может подтвердить его или отклонить, после чего осуществляется переход в заключительное состояние. В случае, если сделка завершилась успешно, то сумма, заплаченная покупателем за товар, размораживается и переходит на счет продавца, также, продавцу предлагается вывести средства, где он может, после чего осуществляется переход в заключительное состояние.

3.6. Диаграмма последовательности

На рисунке 22 представлена диаграмма последовательности заказа товара.

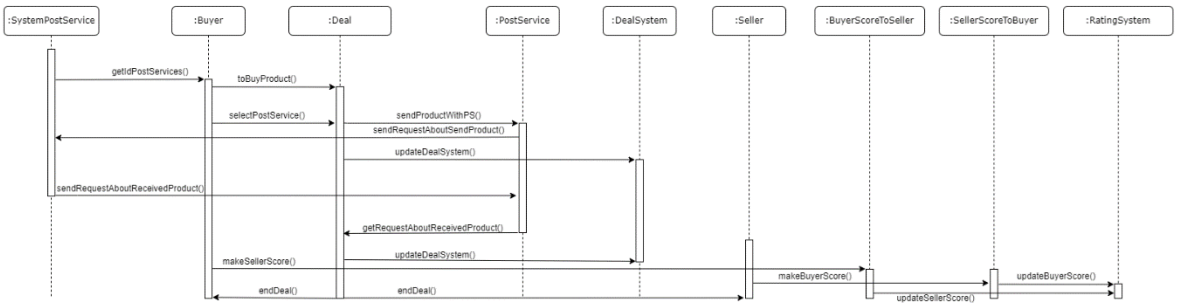

Рисунок 22 – Диаграмма последовательности заказа товара

На данной диаграмме представлена последовательность действий для осуществления заказа товара на сайте. Изначально, после авторизации покупателя (экземпляра класса Buyer) система почтовых сервисов (экземпляр класса SystemPostService) посылает список идентификаторов (с помощью метода getIdPostService) покупателю. Затем покупатель создает экземпляр класса Deal(Сделку) с помощью метода toBuyProduct(), после чего вызывается метод selectPostService, с помощью которого в экземпляр класса Deal записывается идентификатор выбранного покупателем почтового сервиса. Далее, после подтверждения оплаты экземпляр класса Deal(Сделка) посылает запрос на отправку товара почтовому сервису (экземпляру класса PostService) с помощью метода sendProductWithPS(), вместе с тем обновляется хранилище сделок (экземпляр класса DealSystem). После того, как запрос от экземпляра класса Deal на отправку товара дошел до почтового сервиса (экземпляра класса PostService), почтовый сервис, в свою очередь, посылает запрос об отправке товара системе почтовых сервисов с помощью метода sendRequestAboutSendProduct(). Спустя некоторое время, когда заказ дошел до покупателя, система почтовых сервисов (экземпляр класса SystemPostService) посылает запрос почтовому сервису(экземпляру класса PostService) о том, что товар доставлен с помощью метода sendRequestAboutReceivedProduct(). Когда

соответствующий запрос от экземпляра класса SystemPostService дошел экземпляру класса PostService, почтовый сервис (экземпляр класса PostService) посылает запрос экземпляру класса Deal(Сделка) на изменение статуса сделки, в это же время обновляется хранилище сделок(экземпляр класса DealSystem). Далее, покупатель(экземпляр класса Buyer) ставит оценку продавцу методом makeSellerScore() через экземпляр класса BuyerScoreToSeller. В этот же момент экземпляр класса BuyerScoreToSeller посылает экземпляру класса RatingSystem запрос о обновлении оценки продавца. После чего, продавец (экземпляр класса Seller) ставит оценку покупателю методом makeBuyerScore() через экземпляр класса SellerScoreToBuyer. В этот же момент экземпляр класса SellerScoreToBuyer посылает экземпляру класса RatingSystem запрос о обновлении оценки покупателя. Наконец, экземпляр класса Deal методом endDeal() завершает сделку.

На рисунке 23 представлена диаграмма последовательности апелляции.

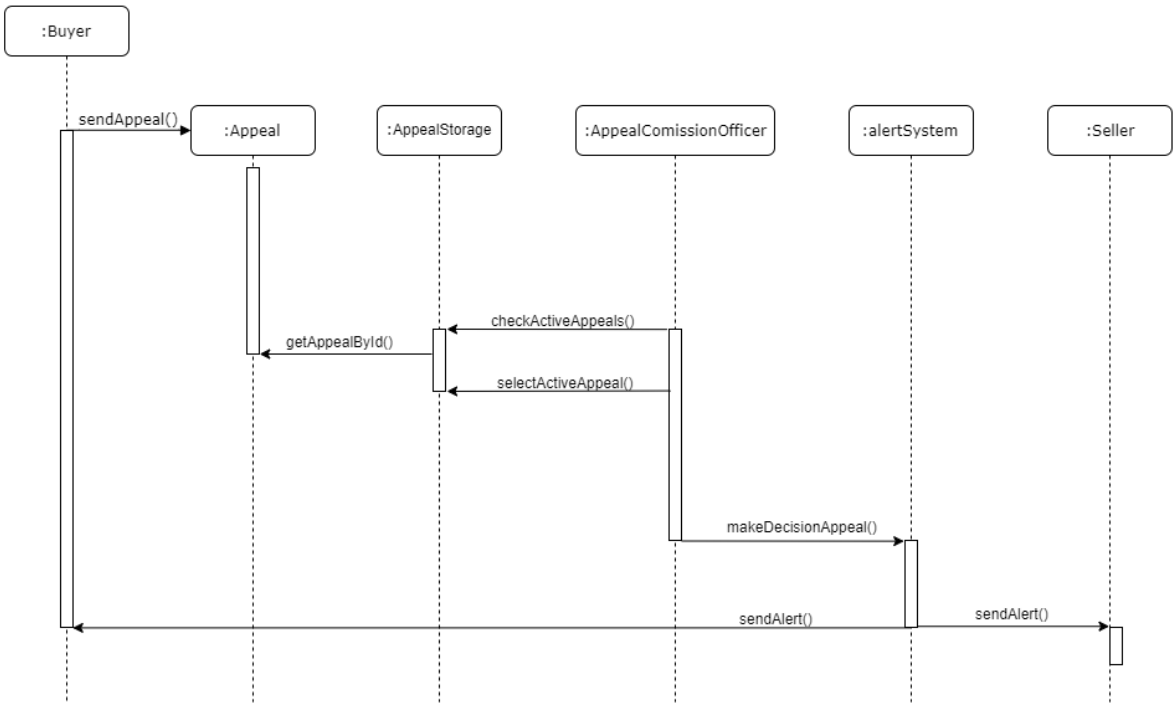

Рисунок 23 – Диаграмма последовательности апелляции

На данной диаграмме представлена последовательность действий для написания апелляции покупателем на сайте. Изначально покупатель (экземпляр класса Buyer) создает апелляцию (экземпляр класса Appeal) с помощью метода sendAppeal(), при этом апелляция автоматически помещается в экземпляр класса AppealStorage. Далее сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) просматривает список апелляций с помощью метода checkActiveAppeals, обращаясь к экземпляру класса AppealStorage, который в свою очередь получает ссылки на апелляции с помощью метода getAppealById(). Далее, с помощью метода selectActiveAppeal сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) выбирает нужную апелляцию. После чего, сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) выносит решение по выбранной апелляции с помощью метода makeDecisionAppeal(), вместе с тем посылается запрос в подсистему оповещений на отправку уведомления

покупателю(экземпляру класса Buyer) и продавцу(экземпляру класса Seller) о принятом сотрудником апелляции решении.

На рисунке 24 представлена диаграмма последовательности отчёта об ошибке.

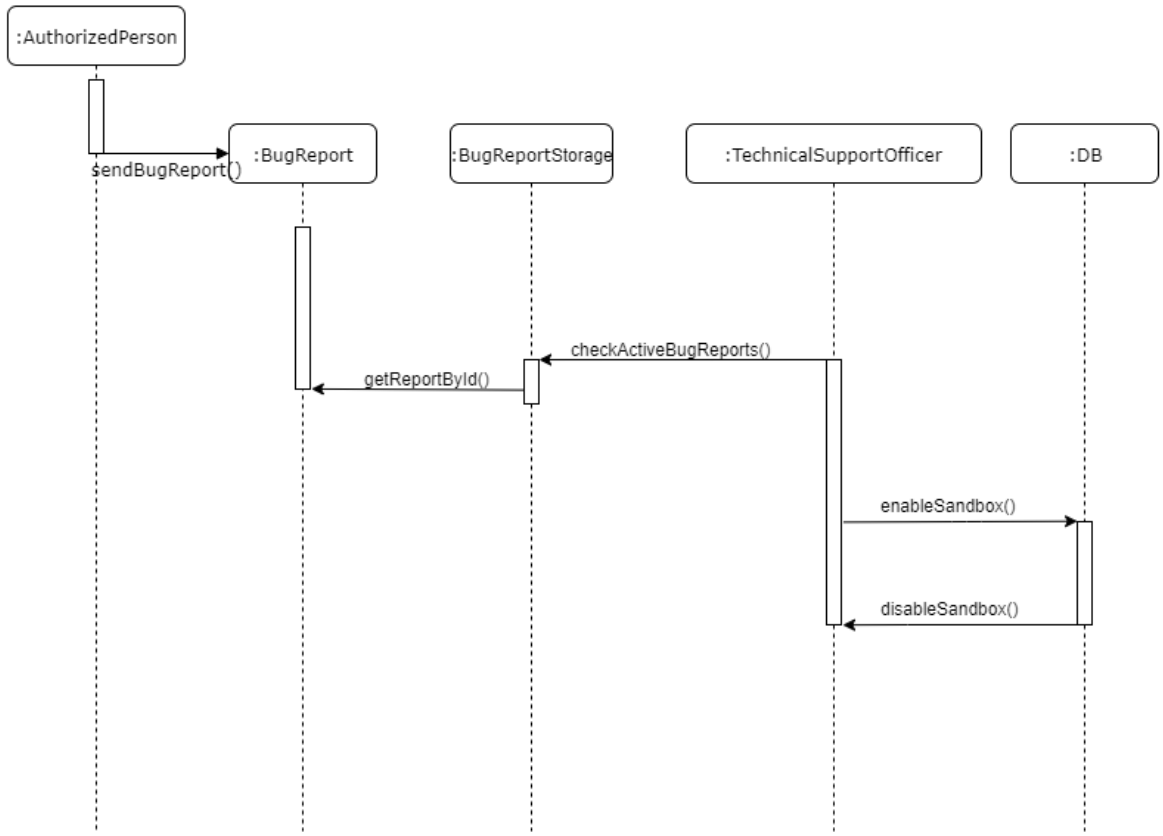

Рисунок 24 – Диаграмма последовательности отчёта об ошибке

На данной диаграмме представлена последовательность действий для написания отчета об ошибке пользователем на сайте. Изначально пользователь (экземпляр класса AuthorizedPerson) создает отчет об ошибке (экземпляр класса BugReport) с помощью метода sendBugReport(), при этом отчет о ошибке автоматически помещается в экземпляр класса BugReportStorage. Далее сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) просматривает список отчетов об ошибке с помощью метода checkActiveBugReports, обращаясь к экземпляру класса BugReportStorage, который в свою очередь получает ссылки на отчеты об ошибке с помощью

метода getReportById(). Далее, с помощью метода enableSandbox() сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) включает режим “песочницы” (для редактирования сайта). Когда редактирование сайта будет завершено, сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) выключает режим “песочницы” с помощью метода disableSandbox().

На рисунке 25 представлена диаграмма последовательности взаимодействия с финансовой системой.

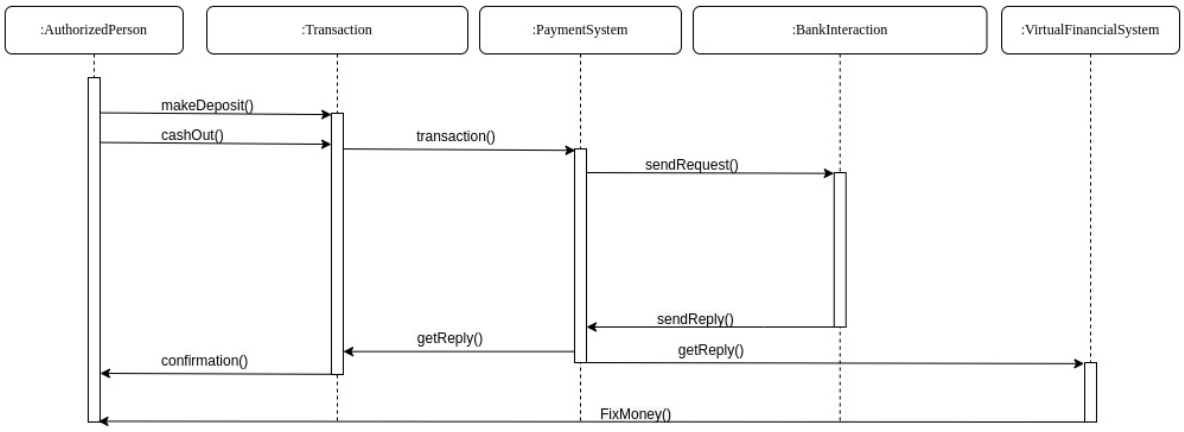

Рисунок 25 – Диаграмма последовательности взаимодействия с финансовой системой

На данной диаграмме представлена последовательность действий для осуществления пополнения счета/снятия средств пользователем на сайте. Изначально пользователь (экземпляр класса AuthorizedPerson) вызывает функцию makeDeposit() или cashOut(в зависимости от того что хочет пользователь: пополнить счет или снять средства, для пополнения счета – метод makeDeposit(), для снятия – cashOut()). С помощью метода makeDeposit()/cashOut() создается транзакция(экземпляр класса Transaction), которая с помощью метода transaction() посылает запрос платёжной системе(экземпляру класса PaymentSystem) о желаемой транзакции. После того, как данный запрос дошел до платежной системы (экземпляру класса

PaymentSystem), экземпляр класса PaymentSystem посылает запрос экземпляру класса BankInteraction(класс для взаимодействия с банком) с помощью функции sendRequest(). Когда экземпляр класса BankInteraction получает запрос от банка, он отправляет запрос экземпляру класса PaymentSystem о успешности/неуспешности транзакции с помощью метода sendReply(). В свою очередь, платежная система (экземпляр класса PaymentSystem), после получения ответа, использует метод getReply() для отправки запроса о корректности транзакции экземпляру класса Transaction и внутренней финансовой системе(экземпляр класса VirtualFinancialSystem). После чего экземпляр класса Transaction подтвержает корректность транзакции с помощью метода confirmation() для экземпляра класса AuthorizedPerson, а внутренняя финансовая система(экземпляр класса VirtualFinancialSystem) изменяет количество денег на счете у пользователя(экземпляр класса AuthorizedPerson).

3.7. Диаграмма коммуникаций

На рисунке 26 представлена диаграмма коммуникаций заказа товара.

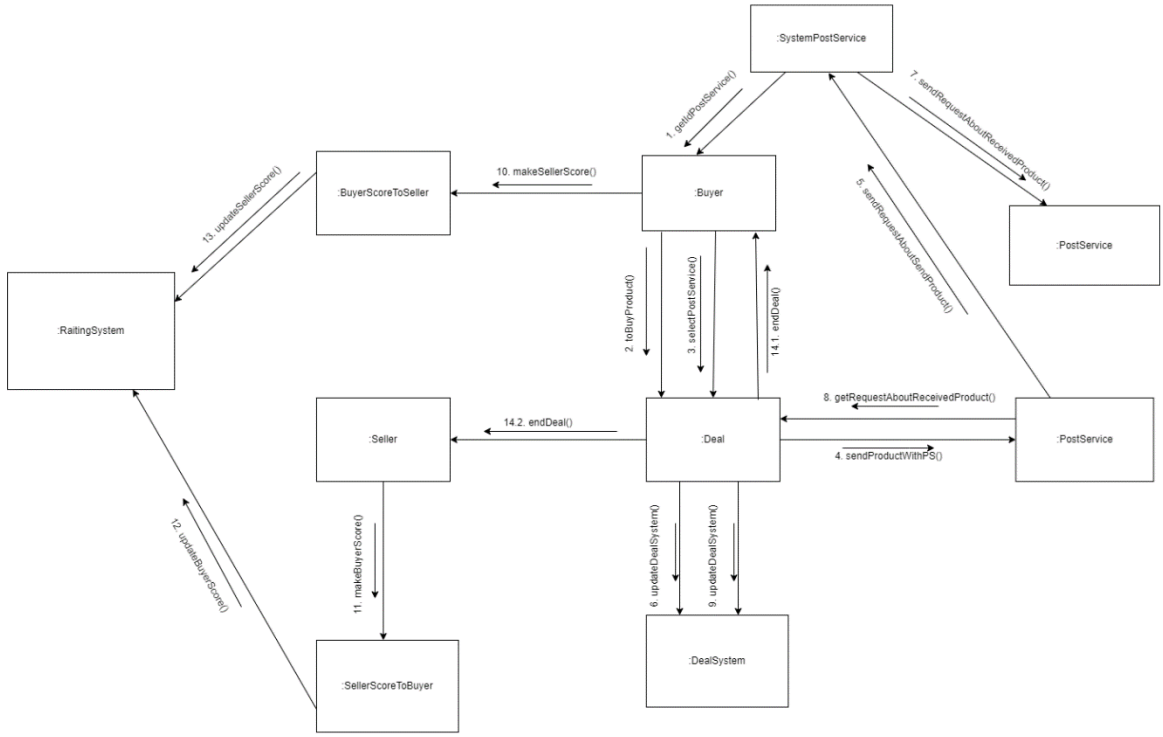

Рисунок 26 – Диаграмма коммуникаций заказа товара

На данной диаграмме представлена последовательность действий для осуществления заказа товара на сайте. Изначально, после авторизации покупателя (экземпляра класса Buyer) система почтовых сервисов (экземпляр класса SystemPostService) посылает список идентификаторов (с помощью метода getIdPostService) покупателю. Затем покупатель создает экземпляр класса Deal(Сделку) с помощью метода toBuyProduct(), после чего вызывается метод selectPostService, с помощью которого в экземпляр класса Deal записывается идентификатор выбранного покупателем почтового сервиса. Далее, после подтверждения оплаты экземпляр класса Deal(Сделка) посылает запрос на отправку товара почтовому сервису (экземпляру класса PostService) с помощью метода sendProductWithPS(), вместе с тем обновляется хранилище сделок (экземпляр класса DealSystem). После того, как запрос от экземпляра класса Deal на отправку товара дошел до почтового сервиса (экземпляра класса

PostService), почтовый сервис, в свою очередь, посылает запрос об отправке товара системе почтовых сервисов с помощью метода sendRequestAboutSendProduct(). Спустя некоторое время, когда заказ дошел до покупателя, система почтовых сервисов (экземпляр класса SystemPostService) посылает запрос почтовому сервису (экземпляру класса PostService) о том, что товар доставлен с помощью метода sendRequestAboutReceivedProduct(). Когда соответствующий запрос от экземпляра класса SystemPostService дошел экземпляру класса PostService, почтовый сервис (экземпляр класса PostService) посылает запрос экземпляру класса Deal(Сделка) на изменение статуса сделки, в это же время обновляется хранилище сделок (экземпляр класса DealSystem). Далее, покупатель (экземпляр класса Buyer) ставит оценку продавцу методом makeSellerScore() через экземпляр класса BuyerScoreToSeller. В этот же момент экземпляр класса BuyerScoreToSeller посылает экземпляру класса RatingSystem запрос о обновлении оценки продавца. После чего, продавец (экземпляр класса Seller) ставит оценку покупателю методом makeBuyerScore() через экземпляр класса SellerScoreToBuyer. В этот же момент экземпляр класса SellerScoreToBuyer посылает экземпляру класса RatingSystem запрос о обновлении оценки покупателя. Наконец, экземпляр класса Deal методом endDeal() завершает сделку.

На рисунке 27 представлена диаграмма коммуникаций апелляции.

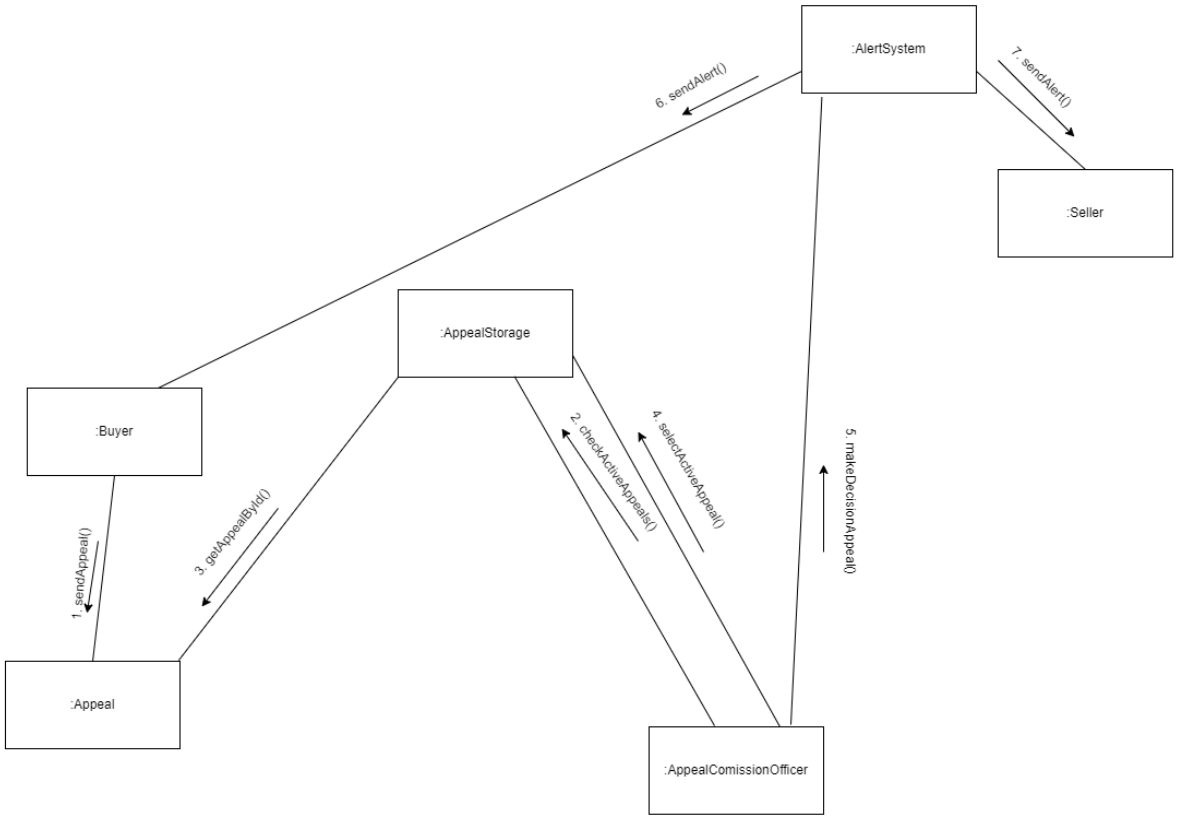

Рисунок 27 – Диаграмма коммуникаций апелляции

На данной диаграмме представлена последовательность действий для написания апелляции покупателем на сайте. Изначально покупатель (экземпляр класса Buyer) создает апелляцию (экземпляр класса Appeal) с помощью метода sendAppeal(), при этом апелляция автоматически помещается в экземпляр класса AppealStorage. Далее сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) просматривает список апелляций с помощью метода checkActiveAppeals, обращаясь к экземпляру класса AppealStorage, который в свою очередь получает ссылки на апелляции с помощью метода getAppealById(). Далее, с помощью метода selectActiveAppeal сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) выбирает нужную апелляцию. После чего, сотрудник апелляционной комиссии (экземпляр класса AppealCommisionOfficer) выносит решение по выбранной апелляции с помощью метода makeDecisionAppeal(), вместе с тем посылается запрос в подсистему оповещений на отправку уведомления

покупателю(экземпляру класса Buyer) и продавцу(экземпляру класса Seller) о принятом сотрудником апелляции решении.

На рисунке 28 представлена диаграмма коммуникаций отчёта об ошибке.

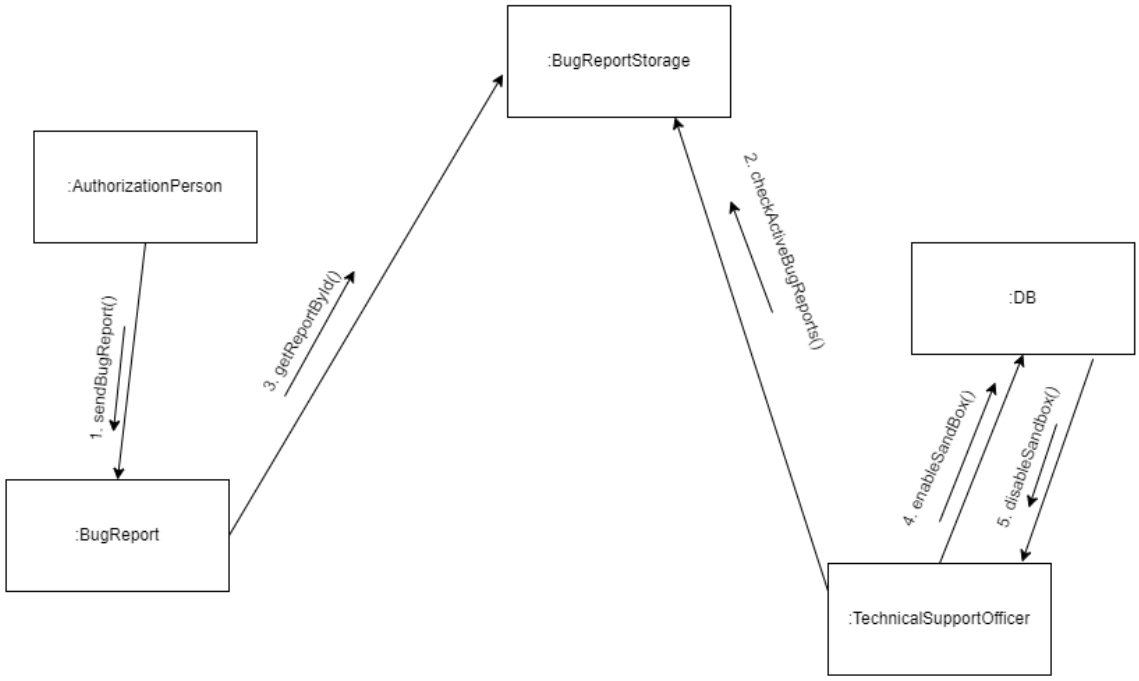

Рисунок 28 – Диаграмма коммуникаций об ошибке.

На данной диаграмме представлена последовательность действий для написания отчета об ошибке пользователем на сайте. Изначально пользователь (экземпляр класса AuthorizedPerson) создает отчет об ошибке (экземпляр класса BugReport) с помощью метода sendBugReport(), при этом отчет о ошибке автоматически помещается в экземпляр класса BugReportStorage. Далее сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) просматривает список отчетов об ошибке с помощью метода checkActiveBugReports, обращаясь к экземпляру класса BugReportStorage, который в свою очередь получает ссылки на отчеты об ошибке с помощью метода getReportById(). Далее, с помощью метода enableSandbox() сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) включает режим “песочницы” (для редактирования сайта). Когда редактирование сайта

будет завершено, сотрудник технической поддержки (экземпляр класса TechnicalSupportOfficer) выключает режим “песочницы” с помощью метода disableSandbox().

На рисунке 29 представлена диаграмма коммуникаций взаимодействия с финансовой системой.

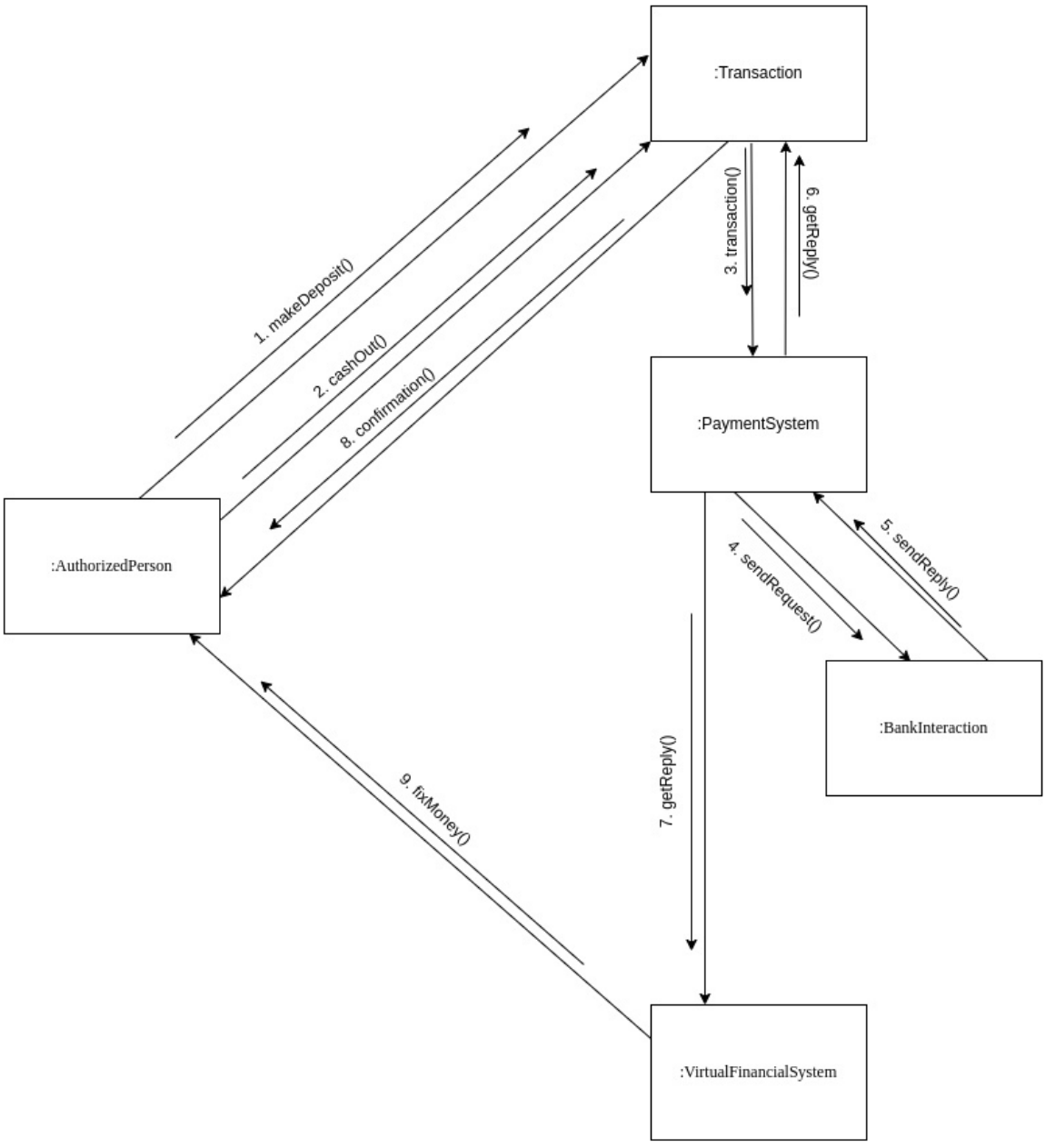

Рисунок 29 - Диаграмма коммуникаций взаимодействия с финансовой системой

На данной диаграмме представлена последовательность действий для осуществления пополнения счета/снятия средств пользователем на сайте. Изначально пользователь (экземпляр класса AuthorizedPerson) вызывает функцию makeDeposit() или cashOut(в зависимости от того что хочет пользователь: пополнить счет или снять средства, для пополнения счета – метод makeDeposit(), для снятия – cashOut()). С помощью метода makeDeposit()/cashOut() создается транзакция(экземпляр класса Transaction), которая с помощью метода transaction() посылает запрос платёжной системе(экземпляру класса PaymentSystem) о желаемой транзакции. После того, как данный запрос дошел до платежной системы (экземпляру класса PaymentSystem), экземпляр класса PaymentSystem посылает запрос экземпляру класса BankInteraction(класс для взаимодействия с банком) с помощью функции sendRequest(). Когда экземпляр класса BankInteraction получает запрос от банка, он отправляет запрос экземпляру класса PaymentSystem о успешности/неуспешности транзакции с помощью метода sendReply(). В свою очередь, платежная система (экземпляр класса PaymentSystem), после получения ответа, использует метод getReply() для отправки запроса о корректности транзакции экземпляру класса Transaction и внутренней финансовой системе(экземпляр класса VirtualFinancialSystem). После чего экземпляр класса Transaction подтвержает корректность транзакции с помощью метода confirmation() для экземпляра класса AuthorizedPerson, а внутренняя финансовая система(экземпляр класса VirtualFinancialSystem) изменяет количество денег на счете у пользователя(экземпляр класса AuthorizedPerson).

3.8. Диаграмма компонентов

На рисунке 30 представлена диаграмма компонентов.

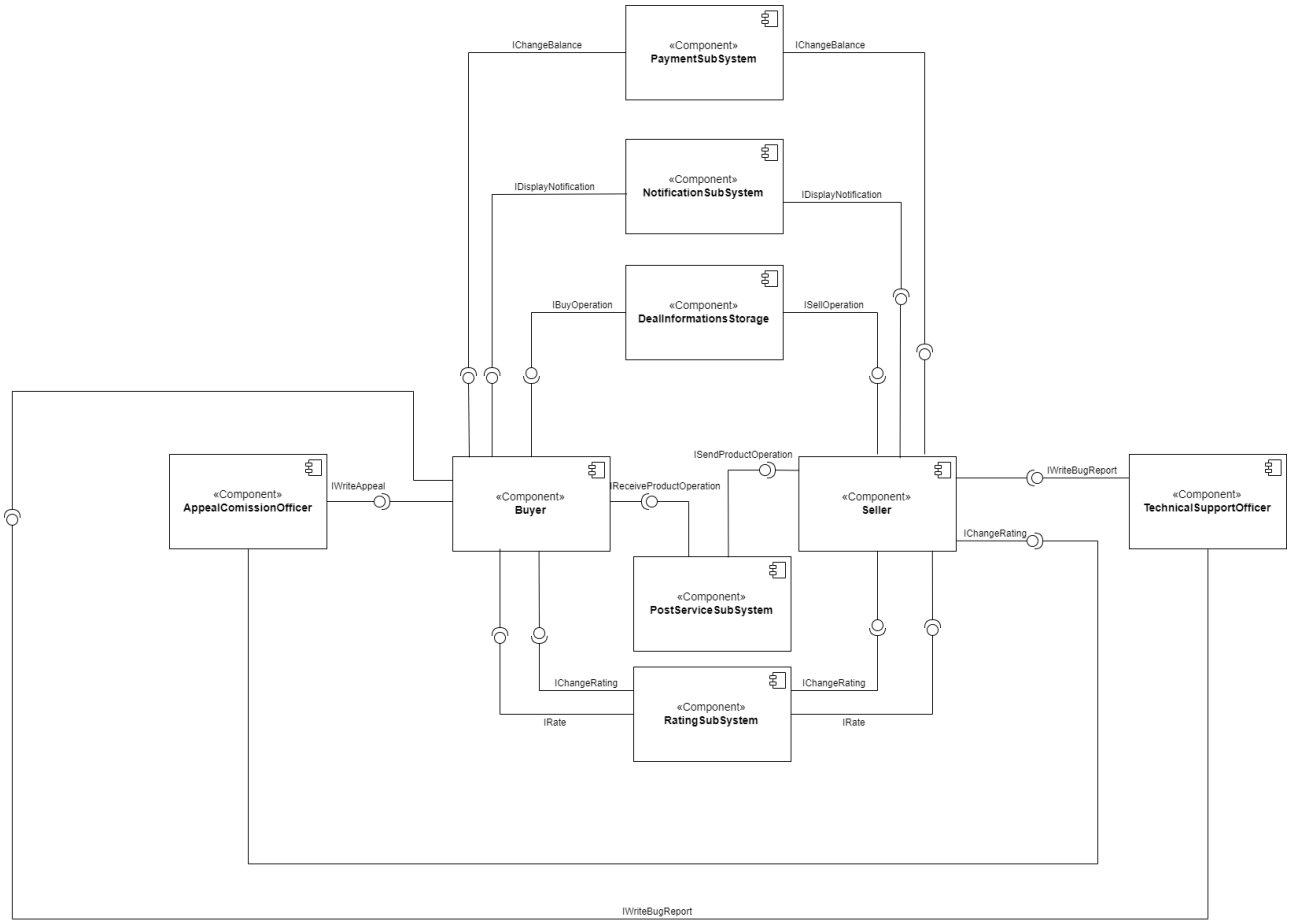

Рисунок 30 – Диаграмма компонентов

Компоненты системы:

PaymentSubSystem:

Компонент, отвечающий за работу платёжной системы.

Состав: VirtualFinancialSystem, PaymentSystem, Transaction, BankInterraction, Cheque.

AlertSubSystem:

Компонент, отвечающий за работы системы оповещения.

Состав: AlertSystem

DealInformationStorage:

Компонент, отвечающий за хранение и взаимодействие со сделками.

Состав: Product, Deal, DealSystem

AppealCommissionOfficer:

Компонент, отвечающий за работу сотрудника апелляционной комиссии.

Состав: Appeal, AppealStorage, AppealComissionOfficer

Buyer:

Компонент, отвечающий за покупателя.

Состав: Buyer.

Seller:

Компонент, отвечающий за продавца.

Состав: Seller.

TechnicalSupportOfficer:

Компонент, отвечающий за работу сотрудника технической поддержки.

Состав: BugReport, BugReportStorage, TechnicalSupportOfficer.

PostServiceSubSystem:

Компонент, отвечающий за работу почтового сервиса.

Состав: PostService, SystemPostService

RatingSubSystem:

Компонент, отвечающий за рейтинговую систему.

Состав: BuyerScoreToSeller, SellerScoreToBuyer, RatingSystem

Разделение на компоненты обусловлено требованием 4.1.1 “Требование к структуре и функционированию системы”. Требуемым подсистемам соответствуют разработанные компоненты, которые инкапсулируют в себе логику требуемых подсистем.

Интерфейсы компонентов:

PaymentSubSystem:

Данный компонент предоставляет интерфейс IChangeBalance для компонентов Buyer и Seller.

AlertSubSystem:

Данный компонент предоставляет интерфейс IDisplayAlert для компонентов Buyer и Seller.

AppealCommissionOfficer:

Данный компонент предоставляет интерфейс IChangeRating для компонента Seller.

Buyer:

Данный компонент предоставляет интерфейсы IReceiveProductOperation для компонента PostServiceSubSystem, IRate для компонента RatingSubSystem, IWriteAppeal для компонента AppealComissionOfficer, IBuyOperation для компонента DealInformationStorage, IWriteBugReport для компонента TechnicalSupportOfficer.

Seller:

Данный компонент предоставляет интерфейсы ISendProductOperation для компонента PostServiceSubSystem, IRate для компонента RatingSubSystem, ISellOperation для компонента DealInformationStorage, IWriteBugReport для компонента TechnicalSupportOfficer.

RatingSubSystem:

Данный компонент предоставляет интерфейс IChangeRating для компонентов Buyer и Seller.

3.9. Диаграмма развертывания

На рисунке 31 представлена диаграмма развертывания.

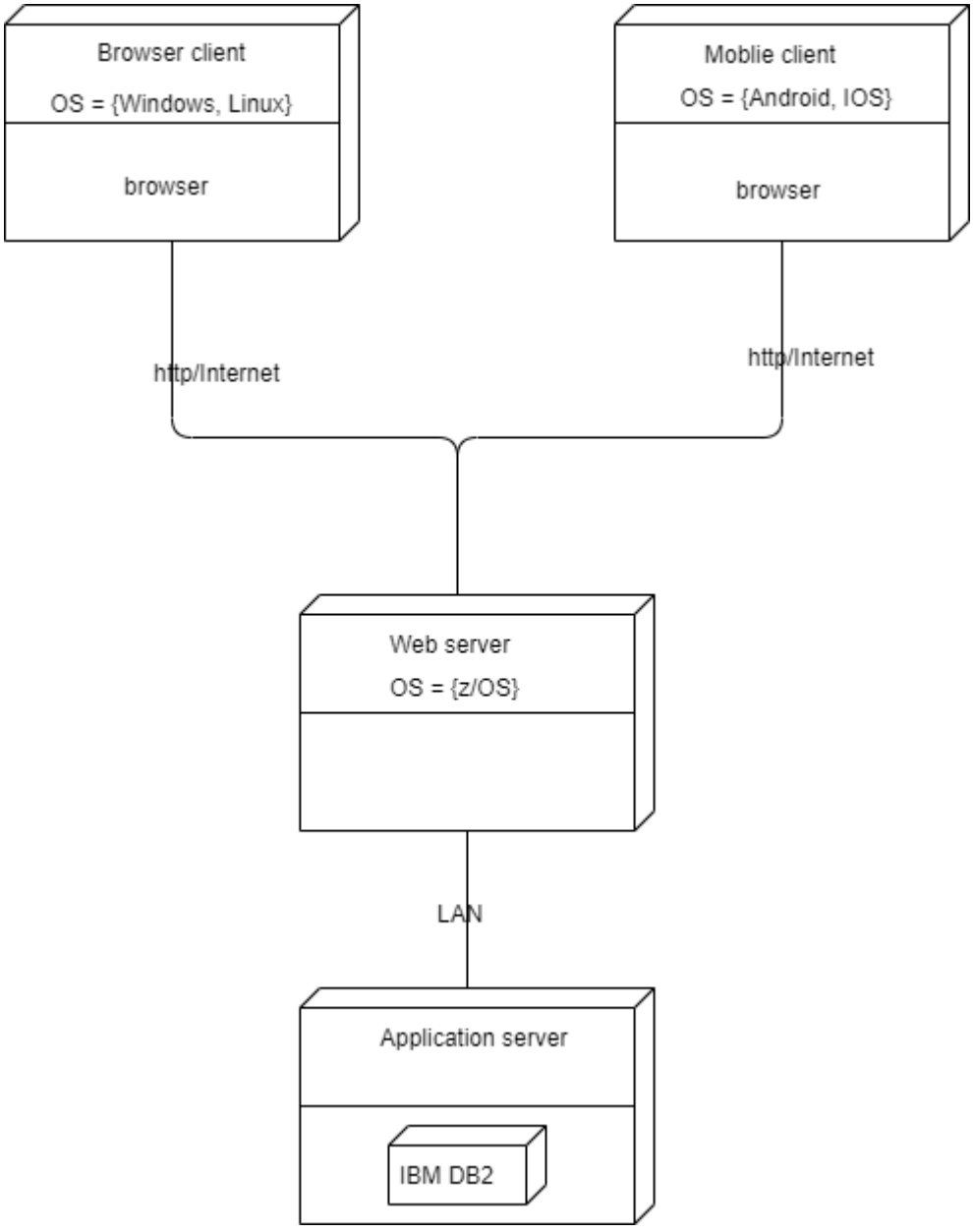

Рисунок 31 - диаграмма развертывания.

Browser Client и Mobile Client- клиенты, которые могут являться как пользователями, так и сотрудниками системы. Такое разделение отображает возможность входа в систему с разных типов устройств. Пользователи взаимодействуют с Web Server через интернет, в качестве самого сервера используется IBM z/OS Communication Server из-за повышенной отказоустойчивости и недорогой эксплуатации. Web Server поддерживает интерфейсы программы для связи с основной логикой в Application Server, в

котором также хранится база данных системы. Application Server в качестве базы данных использует IBM DB2.

3.10. Диаграмма пакетов

На рисунке 32 представлена диаграмма пакетов.

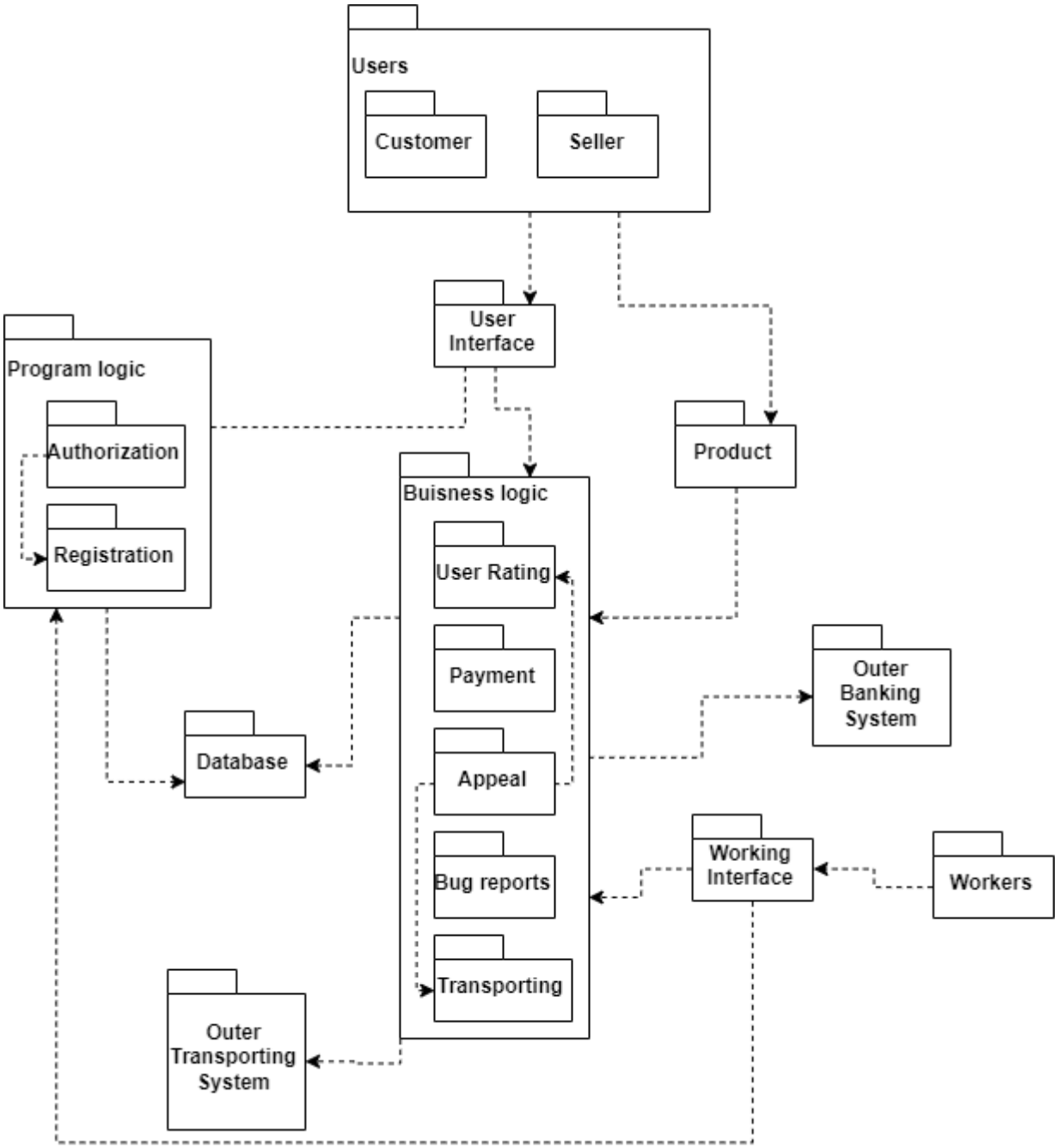

Рисунок 32 - Диаграмма пакетов

Users - Характеризует пользователей данного сервиса. Customer и Seller имеют разные взаимодействия с User Interface и Product - Seller поставляет его, Customer получает, Customer может указать, что он получил продукт в интерфейсе, и т.д.

User Interface - Служит для отправки информации от пользователей к бизнес логике и наоборот. Также осуществляет связь с программной логикой приложения.

Product - Поставляемый продукт от Seller к Customer

Database - База данных, хранящая всю нужную системе информацию

Outer Banking System - Независимая внешняя банковская система, обслуживает денежные транзакции

Outer Transporting System - Независимая система почтовых сервисов, обеспечивает доставку товаров.

Workers - Обслуживающий персонал системы, сотрудники апелляционной комиссии и технической поддержки по-разному взаимодействуют с рабочим интерфейсом

Working Interface - Интерфейс сотрудников, который обеспечивает их взаимодействие с программной и бизнес логикой.

Program logic - Техническая реализация пользовательских интерфейсов, также осуществляет прямое взаимодействие с базой данных через авторизацию и регистрацию.

Business logic - Обрабатывает поступающую от пользователей информацию, и с помощью полученных результатов, обращается к другим частям системы

User Rating - Рейтинг пользователей

Payment - Взаимодействует с информацией о количестве денег пользователя в системе

Appeal - Взаимодействует с апелляциями пользователей, также в случае принятия апелляции, взаимодействует с поставкой товара и рейтингом пользователей.

Bug reports - Взаимодействует с отчётами о проблемах в технической части системы, отсылая их тех поддержке

Transporting - Отвечает за взаимодействие с почтовыми сервисами

4. СООТВЕТСТВИЕ СИСТЕМЫ ТРЕБОВАНИЯМ ТЗ

4.1.Соответствие требованиям

В таблице №1 представлены требования ТЗ и решения, реализующие эти требования.

ТРЕБОВАНИЯ ТЗ	РЕШЕНИЯ
Модуль покупателя должен предоставлять следующие функции: 1. Авторизация в системе; 2. Функция пополнения личного счета; 3. Функция вывода средств; 4. Функция просмотра истории покупок; 5. Функция оплаты; 6. Функция подачи апелляции; 7. Функция просмотра рейтинга продавца; 8. Функция выбора почтового сервиса; 9. Функция выставления оценки продавцу, по окончании сделки; 10. Функция подачи жалобы на функционал сайта;	Весь необходимый функционал обеспечен классом Buyer, который наследуется от класса AuthorizedPerson, а именно: 1. registration() 2. makeDeposit() 3. cashOut() 4. searchHistoryPurchase() 5. toBuyProduct() 6. sendAppeal() 7. checkSellerRating() 8. selectPostService() 9. makeSellerScore() 10.bugReport()
Модуль продавца должен предоставлять следующие функции: 1. Авторизация в системе; 2. Функция пополнения личного счета; 3. Функция вывода средств; 4. Функция просмотра истории	Весь необходимый функционал обеспечен классом Seller, который наследуется от класса AuthorizedPerson, а именно: 1. registration() 2. makeDeposit() 3. cashOut()

продаж; 5. Функция просмотра рейтинга пользователя; 6. Функция выставления оценки покупателю, по окончании сделки; 7. Функция подачи жалобы на функционал сайта;	4. searchHistoryPurchase() 5. checkBuyerRating() 6. makeBuyerScore () 7. bugReport() .
Модуль сотрудника апелляционной комиссии должен предоставлять следующие функции: 1. Функция просмотра текущих апелляций; 2. Функция выбора конкретной апелляции для рассмотрения; 3. Функция наложения санкций на пользователей (блокировка); 4. Функция вынесения решения по вопросу аппеляции;	Весь необходимый функционал обеспечен классом AppealComissionOfficer, а именно: 1. checkActiveAppeals() 2. selectActiveAppeal() 3. createSanctionsPerson() 4. makeDecisionAppeal()
Данный модуль предоставляет функционал для сотрудника технической поддержки. Модуль должен предоставлять следующие функции: 1. Функция просмотра текущих жалоб от пользователей; 2. Функция включения режима «песочницы» (режим безопасного редактирования страницы);	Весь необходимый функционал обеспечен классом TechnicalSupportOfficer, а именно: 1. checkActiveBugReports() 2. enableSandbox()/disableSandbox()

ЗАКЛЮЧЕНИЕ

Был написан проект, содержащий в себе описание архитектуры автоматизированной системы Buy&Sell.