ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«СИСТЕМА УПРАВЛЕНИЯ ЗАКАЗАМИ ОХРАННЫХ СИСТЕМ ДЛЯ ТРАНСПОРТНЫХ СРЕДСТВ НА ОСНОВЕ ТЕХНОЛОГИИ BLOCKCHAIN»

Шубин Данил Витальевич

2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнева»**

Институт информатики и телекоммуникаций
Кафедра информационно-управляющих систем
Направление: 09.04.04 Программная инженерия
Магистерская программа: Разработка и сопровождение автоматизированных систем и web-приложений

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Вид ВКР: магистерская диссертация

СИСТЕМА УПРАВЛЕНИЯ ЗАКАЗАМИ ОХРАННЫХ СИСТЕМ ДЛЯ ТРАНСПОРТНЫХ СРЕДСТВ НА ОСНОВЕ ТЕХНОЛОГИИ BLOCKCHAIN

Обучающийся _____ (подпись) Д.В. Шубин

Руководитель _____ (подпись) М.Г. Доррер

Рецензент _____ (подпись) М.С. Медведев

Ответственный за нормоконтроль _____ (подпись) С.С. Москалева

Допускается к защите

Заведующий кафедрой _____ (подпись) А.В. Мурыгин

«_____» _____ 2023 г.

Красноярск 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет науки и технологий имени академика М.Ф. Решетнева»
Институт информатики и телекоммуникаций
Кафедра информационно-управляющих систем

УТВЕРЖДАЮ
Заведующий кафедрой ИУС
_____ А.В. Мурыгин
«_____» _____ 20___г.

ЗАДАНИЕ
на выпускную квалификационную работу (ВКР)
в форме магистерской диссертации

Обучающийся_Шубин Данил Витальевич_____ группы_МПА320-01_____
направления (специальности)_09.04.04 «Программная инженерия»_____
направленность (магистерская программа)_«Разработка и сопровождение автоматизированных систем и web-приложений»_____
1. Тема ВКР: _«Система управления заказами охранных систем для транспортных средств на основе технологии BlockChain»_____
утверждена приказом по университету № _1762_ от «14»___11____ 2022 г.
2. Руководитель ВКР_М. Г. Доррер, кандидат технических наук, доцент кафедры ИУС СИБГУ им М. Ф. Решетнева_____
3. Срок сдачи студентом первого варианта ВКР _10.01.2023_____
4. Срок сдачи студентом окончательного варианта ВКР _21.01.2023_____
5. Исходные данные: _техническая документация из интернет-источников, научные статьи, учебные пособия, книги_____
6. Содержание (перечень вопросов подлежащих разработке в ВКР): проанализировать процессы информационного обеспечения процессов управления цепочками поставок; проанализировать технологические решения, разрешающее проблемы; рассмотреть блокчейн технологию и его возможности в процессах управления заказами. спроектировать архитектуру и функциональные составляющие проблеморазрешающей системы; разработать распределенное приложение на базе блокчейна и смарт-контрактов._____
7. Перечень графического материала_____
_____
8. Перечень разделов ВКР: введение, анализ предметной области, технология блокчейн, проектирование информационной системы, разработка информационной системы, заключение, список использованных источников_____

Дата выдачи задания __14___ ___ноября_____ 2022_г.

Подпись руководителя ВКР _____ /_____М. Г. Доррер_____/

Задание принял к исполнению дата __14__ ___ноября_____ 2022_г.

Подпись студента _____ /_____Д. В. Шубин_____/

АННОТАЦИЯ

к магистерской диссертации

«Система управления заказами охранных систем для транспортных средств на основе технологии Blockchain»

Шубин Данил Витальевич

Цель диссертационной работы заключается в совершенствовании процессов управления заказами при помощи распределенной информационной системы, которая исполняет роль независимого контролирующего органа, обеспечивающую доверительные взаимоотношения между заинтересованными сторонами процессов.

На основе проанализированных процессов управления заказов и информационных источников спроектирована архитектура системы управления заказами, которая в своей концепции должна включать в себя основные компоненты: графический интерфейса, с которым взаимодействует пользователь; blockchain GoQuorum, использующий язык программирования Solidity для смарт-контрактов; сервер запросов к распределенной базе данных; распределенная база данных Citus, которая кеширует все данные, связанные с процессами управления заказами; сервер IoT-устройств, принимающий и обрабатывающий сигналы и фиксирующий их в blockchain; IPFS для организации децентрализованной сети обмена файлами.

В результате выполнения работы был реализован смарт-контракты – наиболее значимая, в контексте исследования, часть системы. Смарт-контракт имеет следующие функциональные особенности:

– обеспечивают управление доступа к системе управления заказами и разграничивают доступ к функциям распределенного приложения;

– обеспечивают полный учет всех имеющихся активов организаций, подключенных к системе;

– обеспечивает процессы передачи активов между организациями;

– обеспечивают процессы создания заказов, сохраняют историю передвижений;

– обеспечивают привилегированных пользователей в системе возможностями производства продукции, ее восстановление и выход из эксплуатации;

– смарт-контракт спроектирован таким образом, чтобы обеспечить его масштабирование при развертывании в производственной среде.

Диссертационная работа содержит: 86 страниц, 2 таблицы, 38 рисунка, 2 приложения, использованных источников – 55.

Ключевые слова: УПРАВЛЕНИЕ ЗАКАЗАМИ, УЧЕТ ЗАПАСОВ, РАСПРЕДЕЛЕННАЯ СИСТЕМА, BLOCKCHAIN, СМАРТ-КОНТРАКТЫ, АРХИТЕКТУРА.

ABSTRACT
for the master's thesis
«Order management system for security systems for vehicles based on Blockchain technology»
SHubin Danil Vitalevich

The purpose of the dissertation is to improve order management processes using a distributed information system that acts as an independent regulatory body that ensures trusting relationships between the stakeholders of the processes.

Based on the analyzed order management processes and information sources, the architecture of the order management system was designed, which in its concept should include the main components: a graphical interface with which the user interacts; the GoQuorum blockchain, which uses the Solidity programming language for smart contracts; query server to a distributed database; distributed database Citus, which caches all data related to order management processes; an IoT device server that receives and processes signals and fixes them in the blockchain; IPFS for organizing a decentralized file sharing network.

As a result of the work, smart contracts were implemented - the most significant, in the context of the study, part of the system. A smart contract has the following functional features:

– provide access control to the order management system and delimit access to the functions of a distributed application;

– provide full accounting of all available assets of organizations connected to the system;

– provides processes for the transfer of assets between organizations;

– provide processes for creating orders, save the history of movements;

– provide privileged users in the system with the possibilities of production, its restoration and decommissioning;

– the smart contract is designed in such a way as to ensure its scalability when deployed in a production environment.

The dissertation work contains: 86 pages, 2 tables, 38 figures, 2 appendices, 55 sources used.

Keywords: ORDER MANAGEMENT, INVENTORY ACCOUNT, DISTRIBUTED SYSTEM, BLOCKCHAIN, SMART CONTRACTS, ARCHITECTURE.

Содержание

СПИСОК СОКРАЩЕНИЙ

БД – база данных.
СУБД – Система управления базами данных.
API – Application Programming Interface (интерфейс прикладного программирования).
CID – Content Identifier.
ERD – Entity-Relationship diagram (диаграмма отношений сущностей).
EVM – Ethereum Virtual Machine.
IoT – Internet of Things (интернет вещей).
IPFS – InterPlanetary File System (межпланетная файловая система).
JSON-RPC – JavaScript Object Notation Remote Procedure Call (JSON-вызов удалённых процедур).
MVC – Model-View-Controller.
PoW – Proof-of-work (доказательство выполнения работы).
PoS – Proof-of-Stake (доказательство доли владения).
P2P – peer-to-peer.
QBFT – Quorum Byzantine Fault Tolerant.
RFID – Radio Frequency Identification (радиочастотная идентификация).
REA – Resources, events, agents (ресурсы, события, агенты).
UML – Unified Modeling Language (унифицированный язык моделирования).
UI – User Interface (пользовательский интерфейс).

ВВЕДЕНИЕ

В последние годы наблюдается существенный интерес на изменение взаимоотношений между различными экономическими образованиями. Такое стало возможно в условиях, когда интернет доступен для каждого, за счет чего появилось множество возможностей не только для обмена информаций развлекательного характера, но и для ведения бизнеса. Такого рода системы имеют централизацию хранения информации, а значит существует в некотором смысле монополия на контроль данных в системе.

Появление технологии blockchain, обеспечивающее децентрализацию, безопасность и прозрачность хранения информации положило начало для экономических взаимоотношений между пользователями за счет криптовалют. После появления смарт-контрактов у разработчиков появилась возможность для создания децентрализованных прикладных приложений, которые могут формировать доверительную среду, исключая организацию-посредника, которая могла бы вмешаться в ход экономических взаимоотношений между участниками.

Крупные организации инвестируют в blockchain технологии, а также развивают на его базе решения B2B в самых различных направлениях, которые в последствии предлагают, как услуги для других организаций, которые желают оптимизировать собственные бизнес-процессы. Несмотря на это, blockchain и смарт-контракты все еще считается новыми технологиями, поскольку имеют особенности применения, а архитектура приложений, которые используют blockchain имеет существенные отличия в сравнении с обычными централизованными технологиями и подходами к разработке.

Одним из востребованных направлений для внедрения blockchain технологии являются различного рода учетные процессы, функционирующие в цепочках поставок и, в частности, процессы управления заказами. Современные децентрализованные blockchain технологии позволяют решить проблемы, вызванные отсутствием согласованности между данными и предоставить пользователям прозрачность, организовать объективный учет и сформировать доверительную среду для экономических взаимоотношений.

Актуальность работы заключается в применении blockchain технологии и смарт-контрактов для обеспечения доверия в цепочках поставок, за счет выполнения blockchain роли независимого контролирующего органа. Для достижения большей эффективности от эксплуатации системы использование кроссплатформенного подхода к разработке клиентского прикладного приложения позволяет увеличить скорость разработки, облегчить последующую поддержку и унифицировать приложение для разных операционных систем.

Объектом исследования в данной работе являются процессы информационного обеспечения в управлении цепочками поставок.

Предметом исследования является применение методов программной инженерии для совершенствования информационного обеспечения в процессах управлении заказами.

Цель диссертационной работы заключается в совершенствовании процессов управления заказами при помощи децентрализованной информационной системы, которая исполняет роль независимого контролирующего органа, обеспечивающую доверительные взаимоотношения между заинтересованными сторонами процессов.

Для достижения поставленной цели потребовалось решить следующие задачи:

1) проанализировать процессы информационного обеспечения процессов управления заказами в цепочке поставки;

2) проанализировать технологические решения, разрешающее проблемы;

3) рассмотреть blockchain технологию и его возможности в процессах управления заказами;

4) спроектировать архитектуру и функциональные составляющие проблеморазрешающей системы;

5) разработать распределенное приложение на базе blockchain и смарт-контрактов.

Научная новизна данного исследования заключается в том, что использование модели для обеспечения учетных функций при взаимодействиях между собой контрагентов ранее не было реализовано в системах управления заказами на основе blockchain и смарт-контрактов.

Обоснованность научных положений и выводов определяется всесторонним исследованием описанной проблемы. Достоверность положений и выводов диссертации подтверждается фактом работы информационной системы, совершенствованием информационного обеспечения в управлении цепочками поставок.

Теоретическая значимость работы заключается в том, что предложенная концепция использования технологии blockchain может представлять интерес для исследователей, занимающихся вопросами информационного обеспечения учетных процессов в различных областях.

Практическая значимость работы заключается в том, что программная реализация системы управления заказами может быть применена в различных процессах, как учетная децентрализованная система, реализованная на базе blockchain. Кроме того, разработанное программное решение является открытым и может быть изменено, улучшено или адаптировано для специфического использования.

Основные результаты диссертационной работы:

– сформулирована концепция распределенного программного обеспечения на базе смарт-контрактов для информационного обеспечения цепочек поставок;

– осуществлено проектирование программного решения;

– разработаны смарт-контракты.

ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Разработка любой проблеморазрешающей системы начинается с моделирования предметной деятельности для выявления важных информационных и физических составляющих в процессах, рассмотрении процессов детализировано и формировании требований.

Рассматриваются цепочка создания ценности и ее детализированные процессы в условиях современной экономике, и проблемы в этих процессах, которые вызваны устаревшей информационной инфраструктурой, не удовлетворяющей актуальному уровню потребностей.

Моделирование бизнес-процессов позволяет наглядно и с достаточной детализированностью отобразить события в процессах и информационный обмен между участниками. Моделирование бизнес-процессов позволяет перейти от анализирования предметной деятельности к проектированию при помощи выявленных недостатков или ограничений существующих процессов к сформированным требованиям разрабатываемой информационной системе.

1.1 Процессы цепочки поставок в современном мире

В современном мире, любая организация, которая участвует в потреблении, продажи, перевозки, распределении товарно-материальных ценностей, является звеном цепочки поставки, от которой зависят все участники.

Цепочка поставок является подмножеством цепочки создания стоимости. Сеть устроена таким образом, что включает в себя деятельность различного рода, информационные потоки, материальные ресурсы и людей, которые связаны с протекающими процессами [43, 47].

Процесс управления цепочками поставок включает в себя все функции, с момента получения заказа от конечного потребителя и до удовлетворения этого спроса.

Цепочки поставок исторически представляли собой прямолинейные модели. На рисунке 1 представлена обобщённая схема традиционной цепочки поставки.

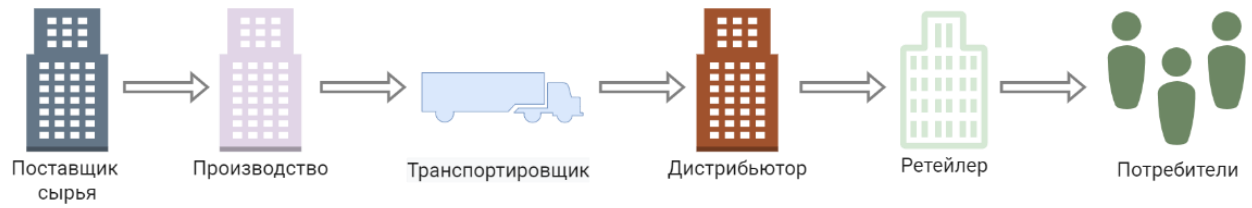

Рисунок 1 – Обобщенная схема цепочки поставок

Традиционная цепочка поставок включается в себя таких участников как:

– поставщики сырья – организации ответственные за поставку сырья для его последующего преобразования в продукт;

– производители товаров – организации, разрабатывающие и производящие товары, также организуют маркетинговые кампании, формирующие спрос на товар;

– транспортировщики – организации, планирующие, контролирующие движение и хранения товаров и услуг от пункта происхождения продукта до пункта назначения, и ответственные за транспортировку товара. В цепочках поставок может быть различное количество такого рода участников, где возможны передачи товара от одного транспортировщика к другому;

– дистрибьюторы – компания, закупающая у производителя товар в крупных объемах и распределяющая его по торговым точкам;

– ретейлеры – организации, реализующие товар конечному потребителю;

– конечные потребители – физические или юридические лица, осуществляющие покупку и потребление товаров [2].

Стоит отметить, что фокусная организация, которой может выступать промышленная или торговая фирма, определяет структуру цепочки поставок и управление взаимоотношениями с другими участниками экономических взаимоотношений.

В современной глобальной экономике цепочка поставок больше похожа на паутину, в центре которой находится производитель, а значит связей между конечным потребителем, который приобретет конкретный товар, могут быть и другие промежуточные звенья.

Цепочки поставок также могут отличаться и это зависит от разных ситуаций, таких как специфика деятельности производства, интенсивности конкуренции между производителями, время жизненного цикла товара.

Существует 6 основных моделей цепочек поставок:

– модель быстрой цепочки – лучше всего подходит для реагирования и быстрого выхода на рынок для компаний, которые часто меняют свои продукты;

– модель с непрерывным потоком: обеспечивает стабильную поставку продуктов и ресурсов в зонах повышенного спроса с небольшим отклонением;

– эффективная модель – сквозная эффективность оптимизирована в этой модели для рынков с высокой конкуренцией;

– отзывчивая модель – подходит производства специализированных товаров, где продукты могут требовать особого внимания в цепочке поставок и наличие опыта. Эта модель обычно точно настраивается для продукта, для которого она используется;

– гибкая модель – оптимальна для продуктов и услуг, когда есть данные о продажах за прошедшие периоды, а также когда поведение потребителя предсказуемо, модель хорошо себя показывает в пики высокого спроса и длительные периоды низкого спроса;

– модель с индивидуальной конфигурацией – гибрид гибкой модели и модели с непрерывным потоком, эта модель позволяет создавать пользовательские конфигурации во время производства и сборки, подходит для производства небольших партий [33].

Указанные модели цепочек поставок имеют разные преимущества для определенных ситуаций, которые определяются спецификой бизнеса и реализуемого продукта, а процессы управления цепочками поставок могут иметь существенные отличия друг от друга, но несмотря на отличия между собой, большинство организованных цепочек поставок имеют в себе сложные вспомогательные процессы информационного обеспечения, благодаря которым возможен информационный обмен между участниками цепочек поставок.

1.2 Процессы информационного обеспечения процессов управления заказами

Процессы информационного обеспечения в цепочке поставок – это информационная инфраструктура, которая является совокупностью процессов сбора, обработки хранения, анализа и выдачи информации, которая необходима для обеспечения процессов цепочки поставок.

Для выяснения того, какая информационная база и процессы составляют процесс информационного обеспечения необходимо знать специфику производимого товара, потребителя, законодательную базу, и другие условия сбыта.

Процессы управления заказами неразрывно связаны с цепочкой поставок и формируются в результате взаимодействий контрагентов между собой.

Цепочка поставок формируется при появлении потребности на товар или окончании запасов на складе у компании-дистрибьютора, которая формирует заказ на поставку поставщику или производителю – что может встречаться особенно часто в случаях, когда компания сотрудничает с фабрикой напрямую. После производства товара компания отгружает товар для транспортировки посреднику, который обязуется поставить товар. Транспортировкой может заниматься как частная компания, сотрудничающая с производителем и дистрибьютором, так и быть частью одной из организаций, участвующих в процессах покупки и продажи, кроме того, организаций транспортировщиков в одной цепочке поставок может участвовать несколько.

Организация-транспортировщик обычно имеет собственные системы и методы работы для управления логистикой. Поставленный товар распределяется по складам или торговым точкам дистрибутора для последующего доведения его до конечного потребителя.

В жизненном цикле товара также встречается процессы вывода из эксплуатации и возврат по разным причинам, например в случаях бракованного изделия или если покупатель вернул товар в установленное время после продажи. В таких ситуациях производитель с дистрибьютором может договориться об отправке товара производителю или придании товарного вида и восстановлении для следующей продажи.

На рисунке 2 представлена IDEF0 диаграмма, которая обобщенно отражает связь между подпроцессами в цепочке поставки.

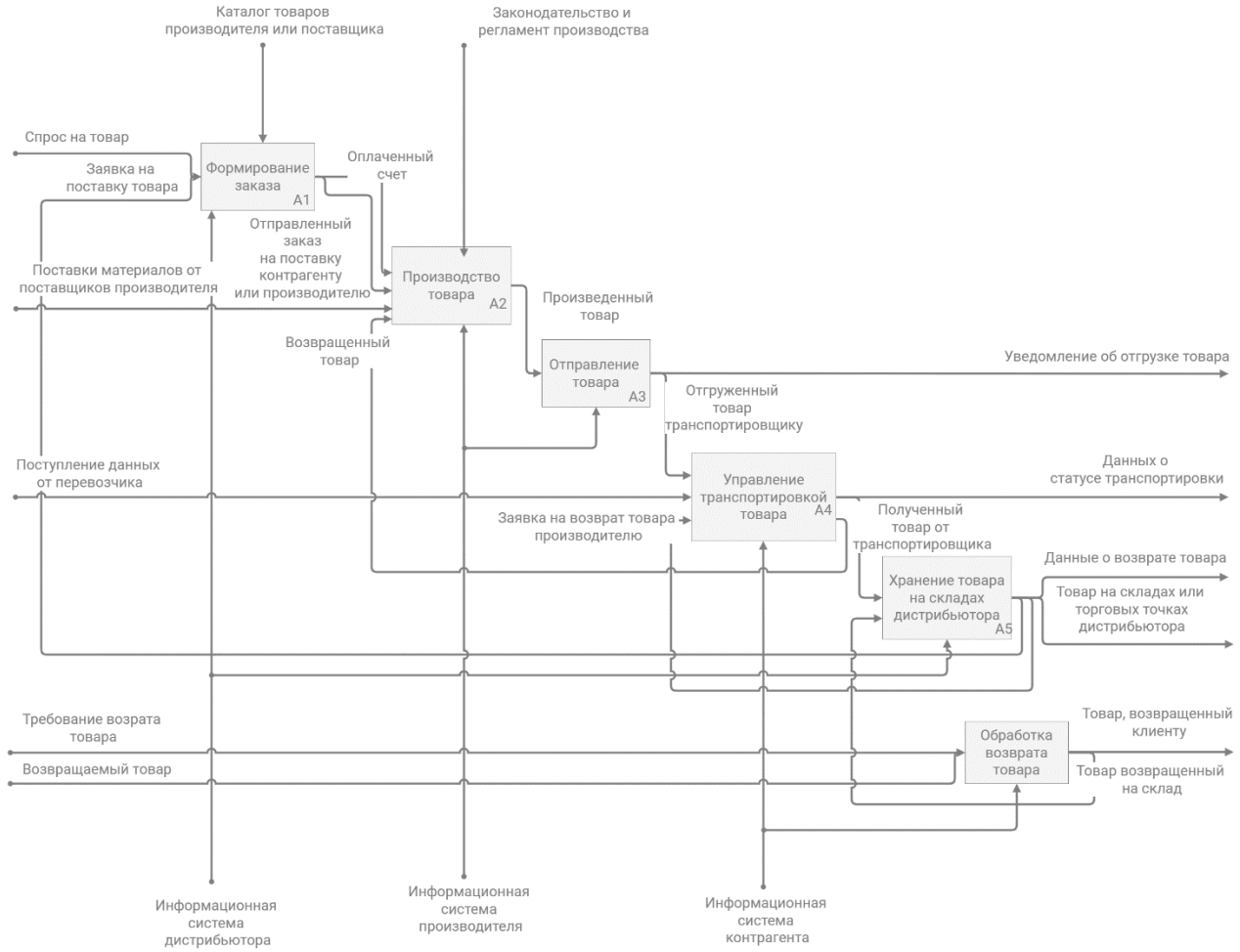

Рисунок 2 – Диаграмма процессов в цепочке поставок в нотации IDEF0

Исходя из информации, представленной на диаграмме, очевидно, что все процессы и участники связаны посредством создаваемой информации в различных процессах нижнего уровня, а также ее обменом между участниками.

Активно используемая парадигма взаимодействия в большинстве отраслях заключается в интеграции планирования и управления цепочкой поставок в организационные процессы, что позволяет оперативно настраивать производственные процессы для удовлетворения спроса потребителя, то для существенного качественного изменения процессов, большей автоматизации и достижения наибольшей эффективности необходимо интегрировать цепочку поставок во внешние глобальные взаимоотношения между всеми ее участниками. Предпосылками этого являются появление новых форм взаимоотношений, которые основываются на тесном партнёрском взаимодействии между участниками цепочки поставки.

Для наглядного представления детализированных бизнес-процессов в данной работе применяется методология графического моделирования процессов BPMN 2.0, которая отличается простотой восприятия, как для аналитиков, так и для обычных участников моделируемых процессов.

Для цепочек поставок такими являются следующие процессы: заказ товара ретейлером – процесс, который связывает производителя с конечным потребителем, необходимый для поставки товара и его последующим

распределением, транспортировка товара, хранение товара, покупка товара потребителем, возврат товара, передача товара производителю, производство товара, сборка партии товаров для отгрузки.

На рисунке 3 представлена диаграмма процесса «заказ товара ретейлером», который связывает производителя с конечным потребителем, необходимый для поставки товара и его последующим распределением.

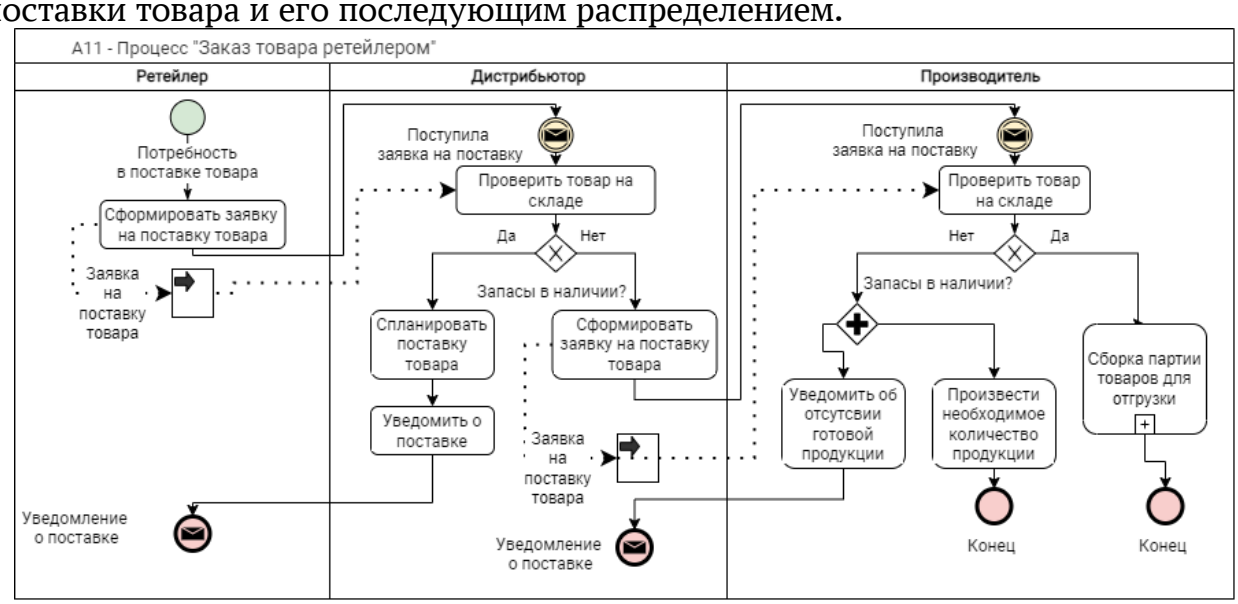

Рисунок 3 – Процесс «заказ товара ретейлером»

Заказ товара может осуществляться без помощи специализированных систем, то есть при непосредственным взаимодействие ретейлера с дистрибьютором при помощи звонков или электронной почты. Дистрибьютор может иметь такую систему, поскольку он должен быть в состоянии обрабатывать большие объемы поставок. В данном процессе каждый из участников может иметь собственную систему для учета, при этом история товара теряется при передаче товара от одного участника к другому.

На рисунке 4 представлена диаграмма процесса «транспортировка товара», необходимый для перемещения товара от одного пункта к другому.

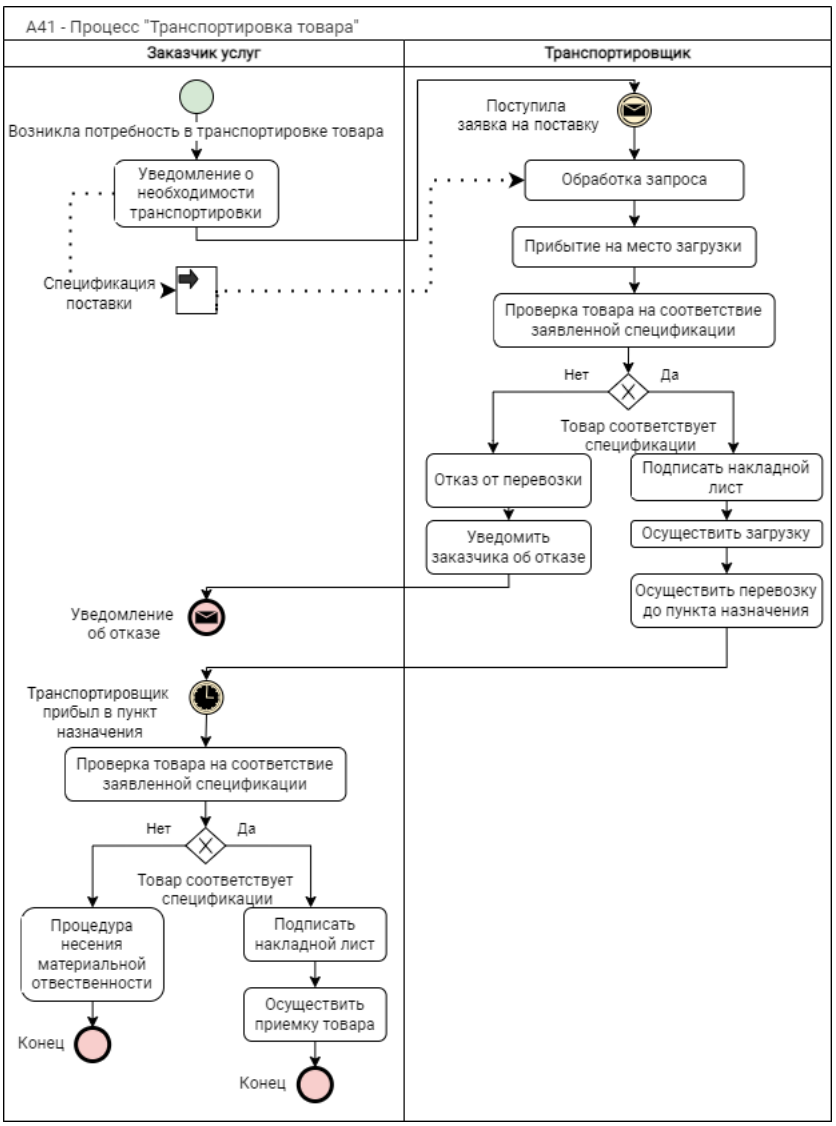

Рисунок 4 – Процесс «Транспортировка товара»

Транспортировщик выступает посредником, но несет полную материальную ответственность за перевозимый груз. Каждый участник при приемке и получении товара имеет копию спецификации товара и накладной лист, в котором принявший товар расписывается и именно в этот момент переходит право передачи собственности и ответственности за товар. На данном этапе также история товара может теряться, поскольку для некоторых участников процесса важным является только соответствии груза заявленной спецификации заказчиком транспортировки.

На рисунке 5 представлена диаграмма процесса «хранение товара», в котором организация, владеющая складом, ответственна за хранение товара в складских помещениях. Организация здесь несет полную материальную ответственность за сохранность товара.

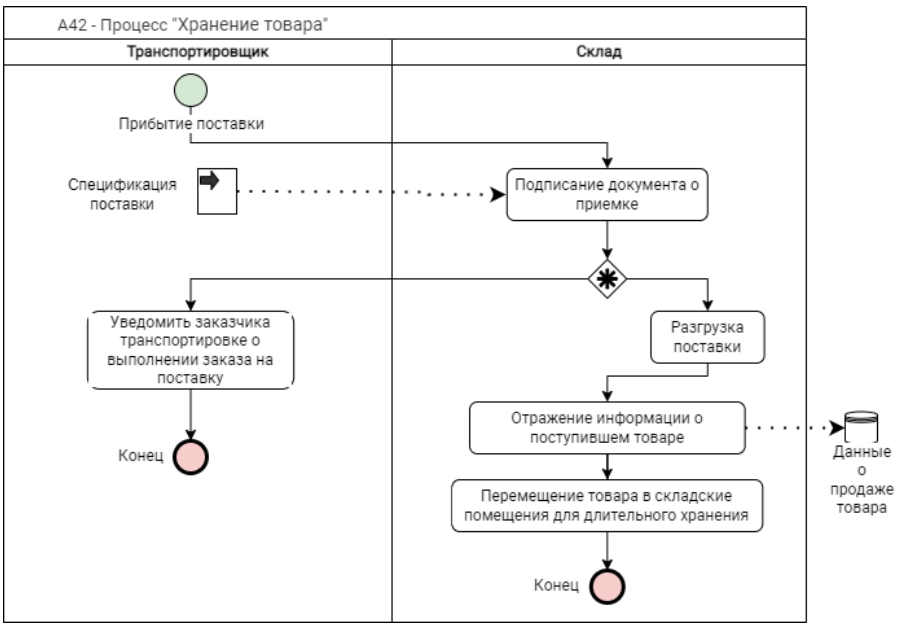

Рисунок 5 – Процесс «Хранение товара»

Смысл данного процесса заключается в том, что условия хранения товара могут соответствовать допустимым, а товар может быть испорчен. При последующей транспортировке или продаже этот факт может быть не замечен или скрыт, а конечный потребитель обнаружить дефекты после гарантийного срока.

На рисунке 6 представлена диаграмма процесса «покупка товара потребителем», в котором товар распределяется по конечным покупателям.

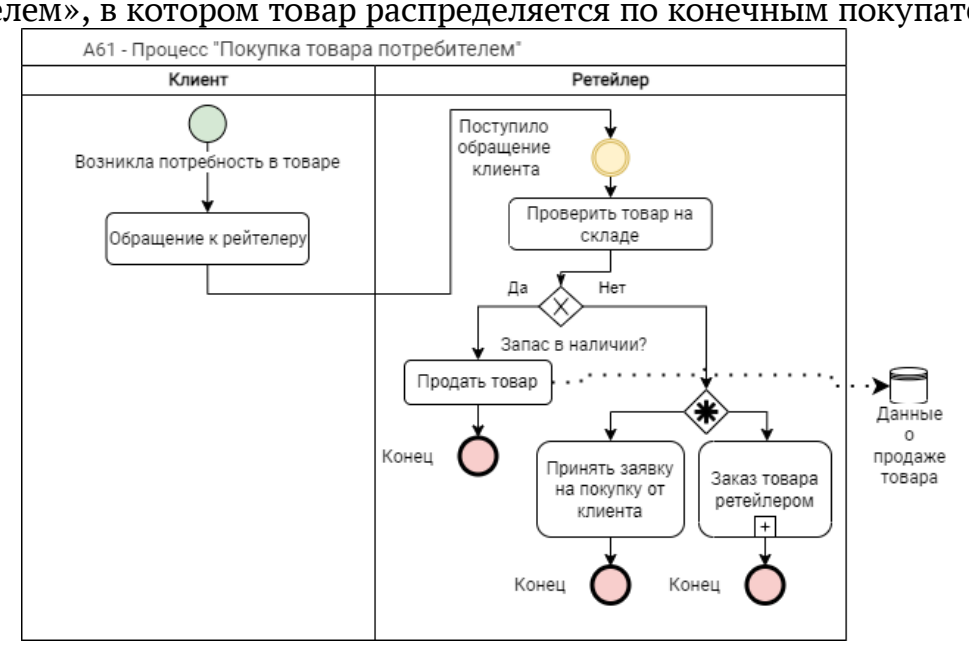

Рисунок 6 – Процесс «Покупка товара потребителем»

На данный момент потребители товаров большинства организаций, не могут видеть всю цепочку поставки товара, что обусловлено технологическим несовершенством процессов в цепочках поставок, в результате чего доверие к товару, бренду или организации может быть ниже, учитывая возможные частные случаи, в которых компании может быть нанесен имиджевый урон.

На рисунке 7 представлена диаграмма процесса «возврат товара», в котором потребитель по какой-либо причине возвращает купленный ранее товар.

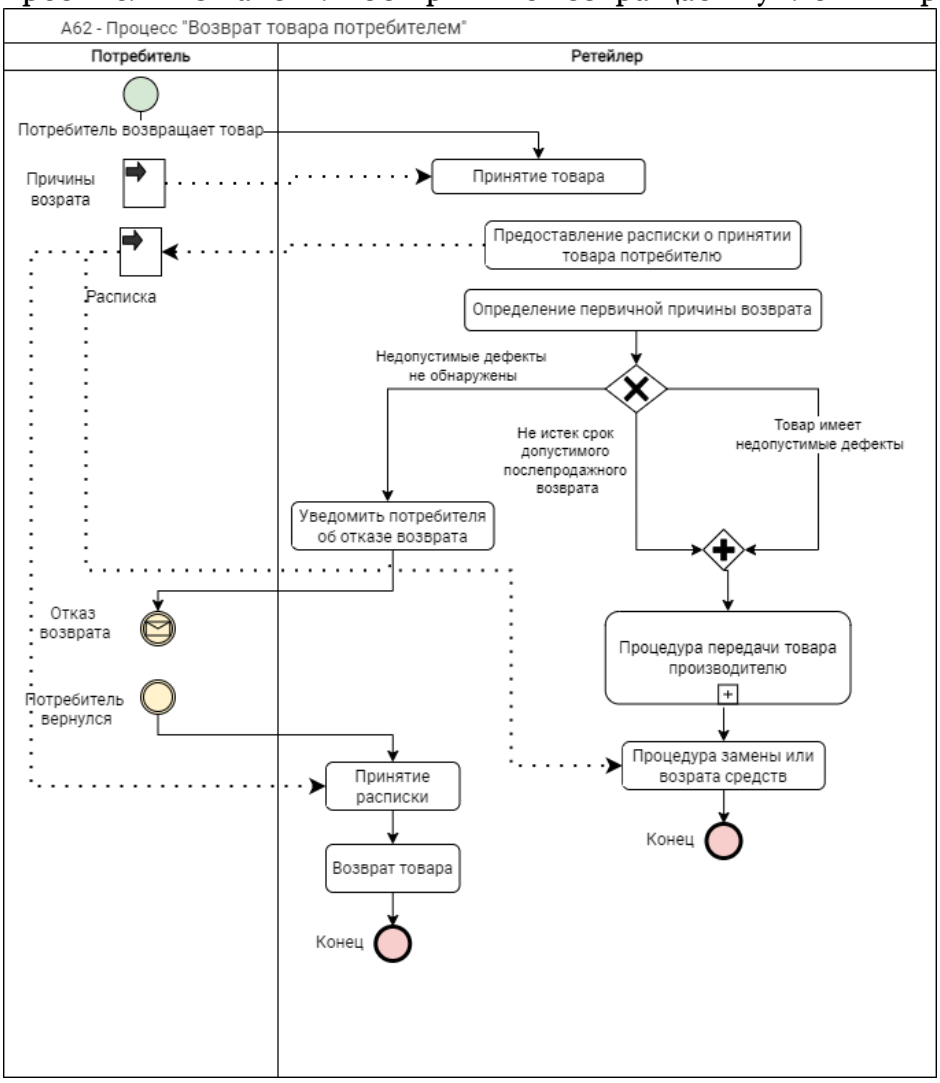

Рисунок 7 – Процесс «Возврат товара»

Процесс интересен с точки зрения того, что возвращаемый товар может быть восстановлен производителем, после чего теряется история о факте возврата, а товар может быть продан как новый. Кроме того, существуют случаи, когда серия одного и того же товара может иметь дефекты, которые носят системный характер. При этом дефекты могут проявить себя после гарантийного срока, в результате чего никто из участников в цепочке поставок не несет ответственности за вред, причиненный потребителю, так как обнаружить проблемы становится практически невозможным. Данная проблема особенно актуальна для технически сложного товара, а также пищевой промышленности.

На рисунке 8 представлена диаграмма процесса «передача товара производителю», в котором через распространяющие сегменты цепи поставок происходит возврат товара производителю для его последующего восстановления или изъятии остаточной полезности.

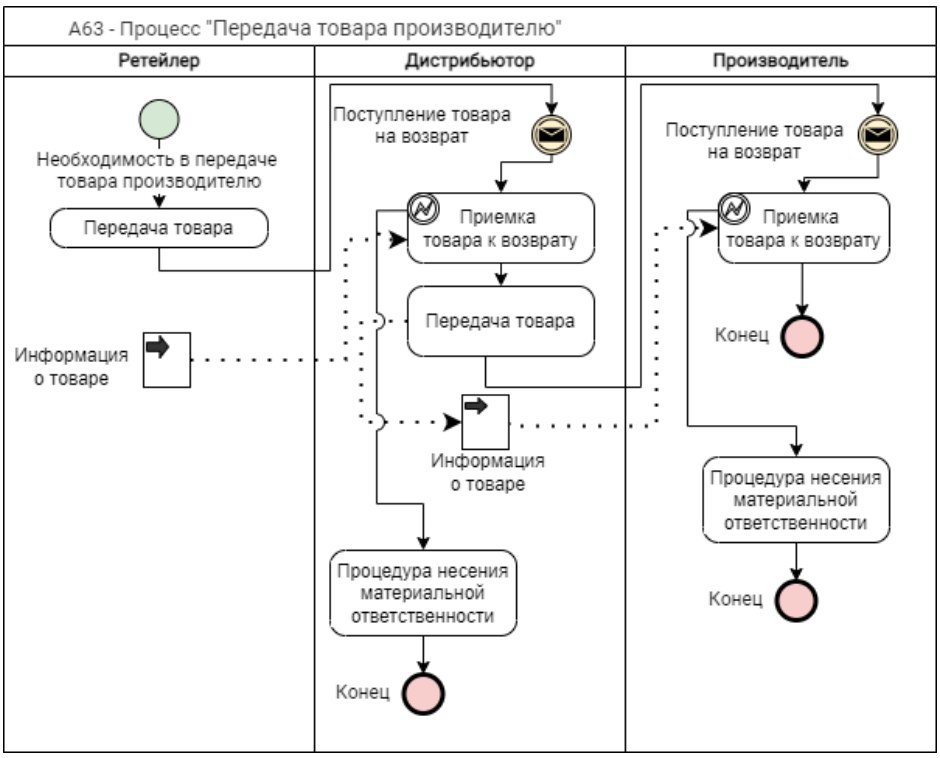

Рисунок 8 – Процесс «Передача товара производителю»

В данном процессе информация о дефектном или возвращенным по другим причинам товаре передается от инициирующего возврате звена. После восстановления товара история о нем теряется, но при этом существует возможность того, что товар может быть восстановлен частично, а сам факт его восставновления будет утерян, и в таком случае потребитель может принять ущерб на себя.

На рисунке 9 представлена диаграмма процесса «производство товара», в котором производитель производит товар и наделяет каждую единицу уникальным идентификатором. Такая идентификация может содержать в системе информацию о используемых компонентах в составе товара. При этом доступ к такой информации может быть только у производителя.

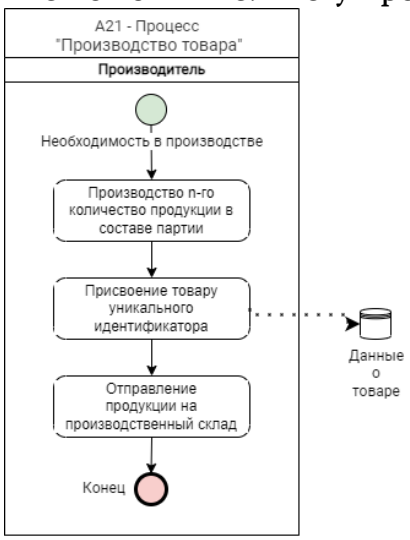

Рисунок 9 – Процесс «Производство товара»

На рисунке 10 представлена диаграмма процесса «сборка партии товаров для отгрузки», в котором дистрибьютор осуществляет заказ на поставку большой партии товара у завода для его последующего распределения по ретейлерам.

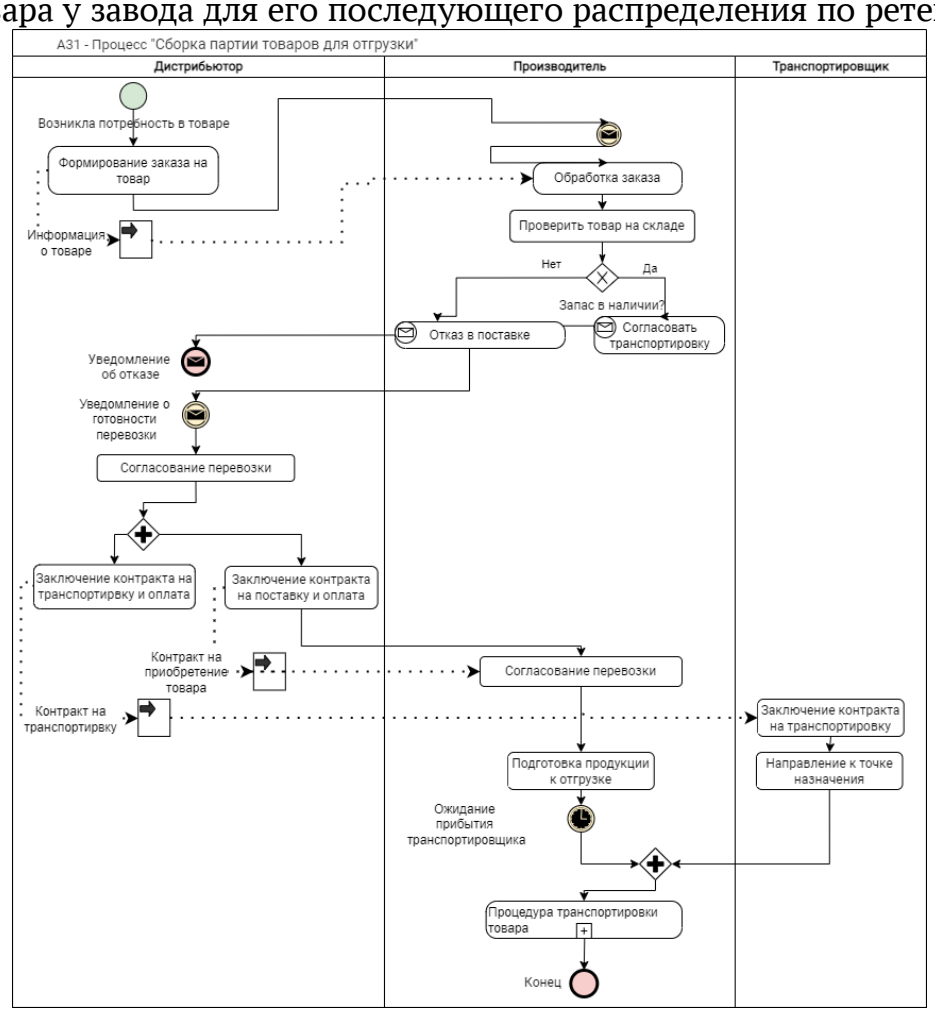

Рисунок 10 – Процесс «Сборка партии товаров для отгрузки»

В процессе, представленном на рисунке 10, существуют возможности для скрытия объемов товара, выпущенными производителем для создания искусственного дефицита и завышения цен дистрибьюторами и ретейлерами для повышения больших выгод с продажи конечному потребителю. Кроме того, это процесс, с которого теряется история о товаре.

Таким образом, проанализированные процессы позволяют сделать выводы о том, что в реализации цепочек поставок традиционным способом, существует множество недостатков и несовершенств, прежде всего вызванных информационной несогласованностью между всеми его узлами и распределенным характером цепочек поставок.

Процесс заказа товара, является наиважнейшим в цепочке создания ценности, поскольку именно этот процесс связывает между собой все партнёрские организационные структуры, которые могут быть никак не связанными между собой. В данной работе акцент исследования смещен в сторону повышения эффективности управления заказами, что позволит устранить большинство недостатков в устоявшихся способах взаимодействия

контрагентов друг с другом, за счет совершенствования электронного документооборота.

1.3 Несовершенство информационного обеспечения в цепочках поставок

Традиционная организация цепочек поставок и бизнеса потеряла свою актуальность из-за множества проблем, которые понижают эффективность бизнеса, несут различного рода риски и оказывают влияние на участников экономических взаимоотношений:

– проблемы масштабирования цепочки поставок, появляющиеся при усложнении цепочки по мере расширения производственных мощностей, увеличения количества потребителей, изменении контрагентов, появлении новых торговых точек;

– территориальный разброс подразделений предприятия, что может быть проблемой в случаях проблем на производстве;

– проблемы прозрачности, общей и справедливой системы контроля и доверия между покупателем и продавцом, которые могут выражаться в манипулировании рыночной стоимостью посредниками, что позволяет получать дополнительную прибыль за счет конечного потребителя, поскольку в созданных классических способах управления цепочками поставок информационная и ценовая прозрачность отсутствует;

– следует отметить то, что имеется вероятность мошенничества в сделках, когда оплаченный товар будет иметь низкое качество или поставщик вовсе не ответит по своим обязательствам перед организацией покупателем;

– возможная ненамеренная порча товара на участках цепочки поставок, что особенно критично для пищевой промышленности, где продукт может быть заражен или испорчен из-за условий хранения;

– возможность внедрения контрафактной продукции, замене или хищении на звеньях цепочки поставок, что порождает дальнейшие экономические риски для организации, угрозы здоровью потребителю и материальные потери;

– проблемы качества могут выражаться в том, что зачастую товар движется в одном направлении и в случае, если товар имеет дефекты, то последствия будут перенесены на потребителя;

– ошибки, возникающие из-за того, что в процессах транспортировки и доставки участвуют люди, которые подвержены человеческому фактору, который может выражаться в хищении продукции, неверно-заполненными документами, кроме того, за мониторинг в логистических процессах также ответственны люди, которые могут ошибаться [34].

Таким образом, вышеперечисленные проблемы оказывают сильное влияние на цепочку поставок и ее участников. Современные системы главным образом полагаются на доверительные отношения между людьми, участвующих в функционировании цепочек поставок и такая зависимость должна быть устранена, что позволит снизить риски в процессах, повысит их качество и

сформировать доверительные взаимоотношения между участниками экономических взаимоотношений.

1.4 Технологии информационного обеспечении цепочек поставок

Для решения вышеуказанных проблем в информационном обеспечении цепочек поставок на данный момент существует ряд перспективных технологий: blockchain, интернет вещей (IoT), искусственный интеллект и аналитика, роботизированная автоматизация процессов.

Интернет вещей (IoT) – концепция сети физических объектов, в которые встроены датчики, программное обеспечение и другие технологии обмена данными между другими подключенными устройствами [52].

На промышленном уровне IoT позволяет связать информационные системы с датчиками и устройствами контроля для отслеживания товаров, измерения в режиме реального времени состояния товара и окружающей среды, сигнализировать о недопустимых условиях хранения.

Искусственный интеллект и аналитика – экспертные системы, способные решать различные прикладные задачи и проблемы, принимать решения в контексте промышленных производств.

Наличие большего объема данных, а также поступающих данных в реальном времени может быть хорошей возможностью для применения искусственного интеллекта в аналитических целях, чтобы спрогнозировать тенденции будущего спроса или состояния цепочки поставок, подбирать и оценивать заменяемых участников цепочки поставки, а также обеспечивать управление всей цепочкой поставки [5].

Роботизированная автоматизация процессов – технологии, упрощающие создание продукции, управление системами, имитирующие сложные действия человека, но обладающие большей трудовой производительностью.

Роботизированные технологии как автоматизированные транспортные средства, системы доставки до конечного потребителя, системы хранения и поиска также позволяют решить множество задач в традиционных цепочках поставок [42].

Blockchain – позволяет обеспечить контроль для всей цепочки поставок, в отдельных случаях жизненный цикл продукта можно отследить при помощи информации, которая записана в blockchain при достижении определенных условий заранее [48, 53].

Таким образом, современные технологии способны предложить способы решения проблем в цепочках поставок, но указанные технологии на данный момент все еще являются развивающимися, существующие решения являются дорогостоящими, весьма ограничены и доступны только для высокотехнологичных компаний. Важным также является то, что зачастую в конкретных примерах реализации эти технологии только в своей комбинации позволяют организации получить преимущество от владения и использования.

В данной работе акцент смещен на технологию blockchain и ее практическое использовании в информационном обеспечении цепочек поставок для удовлетворения современных требований.

1.5 Модель оцифровки процессов

Для автоматизации процессов обмена информацией в распределенной среде необходимо рассмотреть процессы с точки зрения информационного обмена и событий, которые порождают объективную информацию для участников.

В данной работе для анализа бизнес-процессов, которые тесно связаны в цепочке поставки применятся онтология REA для моделирования учетных систем для процессов или организации деятельности внутри организации. Онтология главным образом позволяет перейти от классического бухгалтерского учета к представлению данных, которые характерны для компьютерных информационных систем. Такого рода модели в графическом исполнении представляют собой модифицированную версию ER-диаграммы, а соглашения описания связей соответствует соглашениям в классических ER-диаграммах. В отличии от ER-диаграмм для REA-диаграмм принято использовать 3 разновидности сущностей: ресурсы, события и агенты [28].

В модели REA имеются следующие элементы:

– ресурсы – дефицитные объекты, находящиеся под контролем организации, представляющие экономическую ценность и использующиеся в экономическом обмене с партнерами;

– события – экономические явления, вызывающиеся изменения ресурсов и представляющие из себя отношениями потоков запасов. В контексте хозяйственного учета эти события важны, поскольку при их возникновении происходит перераспределение ресурсов, что должно быть задокументировано и учитываться;

– события поддержки – события, которые не вызывают изменения ресурсов напрямую и служат обслуживающими для основных;

– агенты – участники, участвующие в хозяйственной деятельности, рассматриваемой ситуации в моделировании, и имеющие право распоряжаться ресурсами. Каждое событие связано хотя бы одним агентом [28].

Таким образом, событиями являются:

– оплата товара (потребителем) – возможное событие, в котором потребитель приобретает товар, отдавая деньги;

– покупка товара – событие, вызываемое при оплате товара, в котором потребитель получает товар;

– возврат товара – возможное событие, которое может произойти в том случае, если купленный товар будет возвращен потребителем;

– компенсация – событие, которое может возникнуть при возврате товара, в котором клиенту взамен возвращенного ранее товара предлагается компенсация с выбором вида получаемого ресурса;

– оплата – вызываемое событие, когда ретейлер намеревается закупить товар для его последующей перепродажи и в котором ретейлер оплачивает поставку товара;

– распределение товара – событие, в котором дистрибьютор распределяет заказанное количество товара по ретейлерам;

– оплата (услуг транспортировщика) – событие, в котором заказчик перевозки (ретейлер, дистрибьютор) оплачивает услуги транспортировщика для доставки груза от точки до точки;

– транспортировка – событие, в котором транспортировщик доставляет товар до пункта назначения;

– отправка товара – событие, в котором производитель осуществляет отправку товара дистрибьюторам. Является причиной для события транспортировки товара;

– оплата товара (дистрибьютором) – событие, вызываемое при реализации намерения дистрибьютора на приобретении товара для его дальнейшего распределения по ретейлерам.

Ресурсами являются:

– деньги – объекты, представляющие ценность для всех агентов;

– товар – объекты, представляющие ценность для конечных потребителей.

Агентами являются:

– покупатель – участник, который взамен денег намеревается получить товар;

– ретейлер – участник, который распределяет товар по конечным потребителям взамен денег и получающий товар у дистрибьютора в обмен на деньги;

– дистрибьютор – участник, который закупает товар у производителя и распределяет по ретейлерам взамен на деньги;

– транспортировщик – промежуточный участник, который перевозит груз от точки до точки и получает взамен своих услуг деньги;

– производитель – участник, организовывающий производство товара и распределяющий его по дистрибьюторам за деньги.

На рисунке 11 представлена диаграмма, описывающая REA-модель для процессов цепочки поставок.

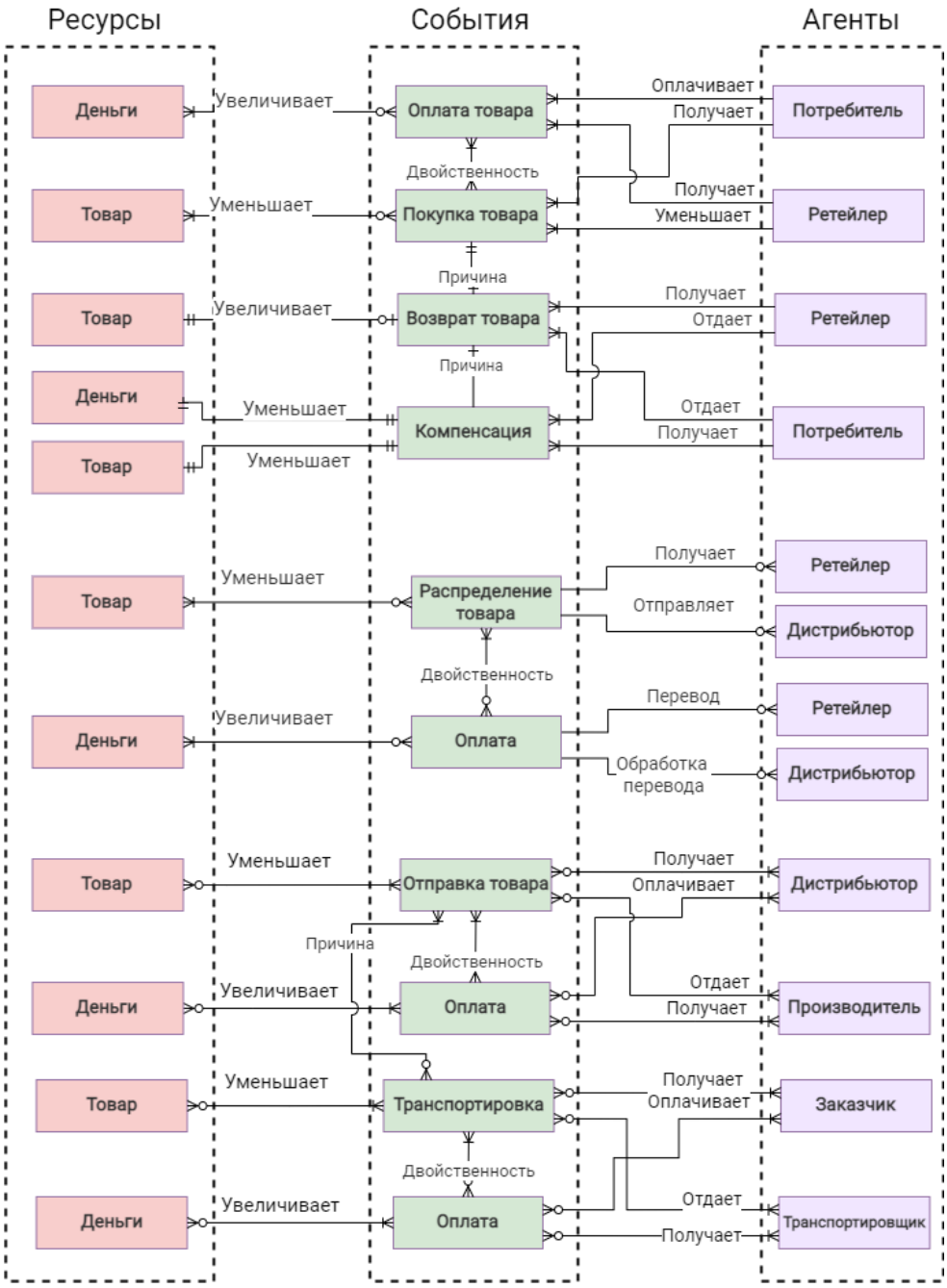

Рисунок 11 – REA-диаграмма для процессов цепочки поставок

Таким образом, представленная модель позволяет наглядно увидеть основные экономические события и взаимоотношения внутри цепочек поставок, которые имеют значение в контексте учета ресурсов.

1.6 Основные требования к информационной системе

Перед проектированием и разработкой информационной системы на основе рассмотренных в достаточной степени, детализированных процессах в цепочках поставок, необходимо сформировать требования.

Формирование требований возможно при достаточном понимании сценариев, при которых программный продукт будет использоваться. В данной работе для графического наглядного представления вариантов использования

применяется Use Case диаграмма из языка моделирования UML, позволяющая описать сценарий использования системы при взаимодействии с ней действующих пользователей.

На рисунке 12 представлена диаграмма Use Case для проблеморазрешающей информационной системы, описывающая требования пользователей к информационной системе.

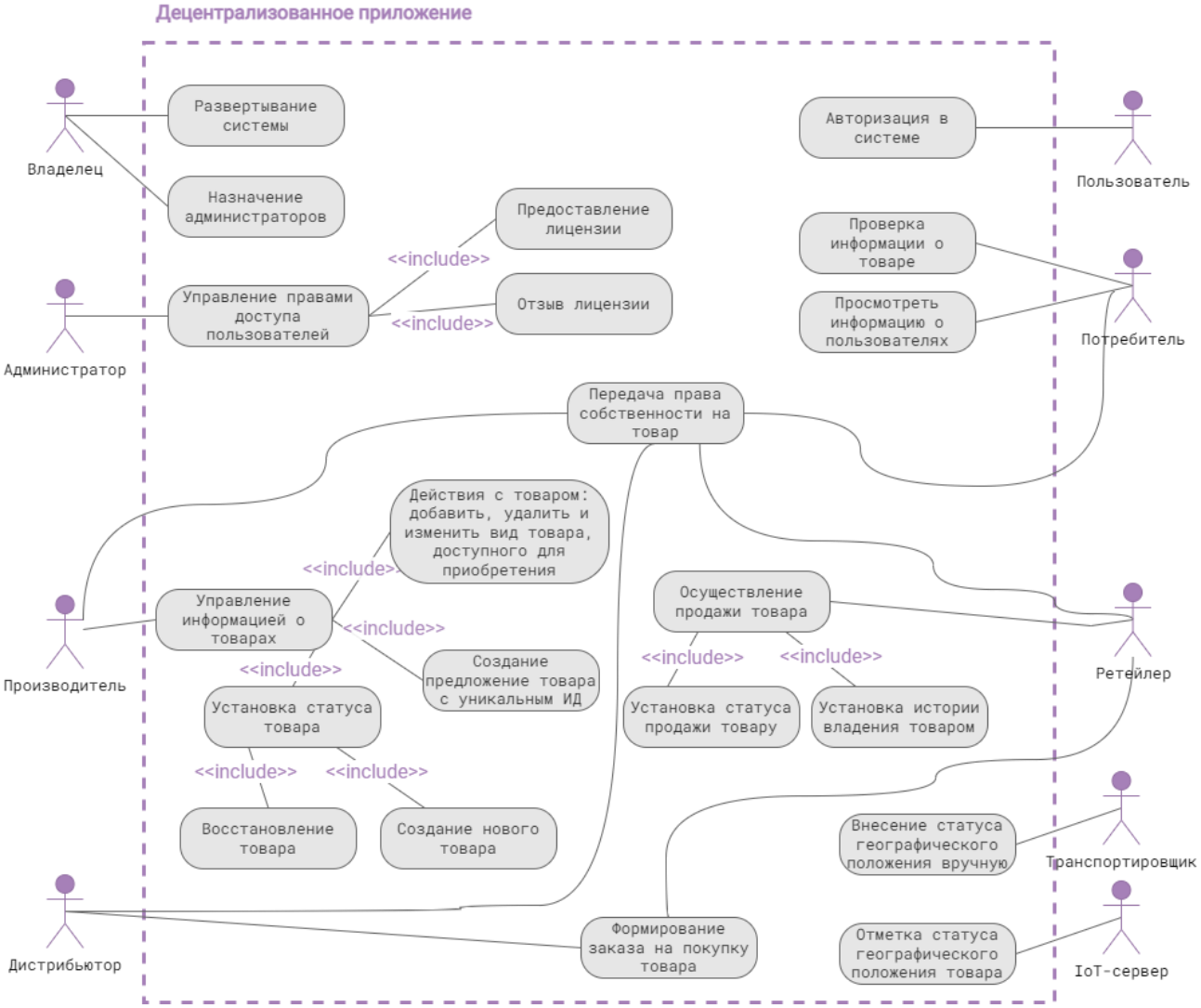

Рисунок 12 – Диаграмма Use Case для проблеморазрешающей информационной системы

Таким образом, на основе проанализированных процессов и ожидаемых вариантов использования системы, можно выделить следующие функциональные требования для разрабатываемой проблеморазрешающей системы:

– необходим учет каждого товара и отслеживание его в цепочке поставок;

– возможность добавления перечня доступных разновидностей товаров для их заказа у производителя;

– управление правами доступа к системе: регистрация вида пользователя, обслуживающего процессы в цепочках поставок.

– система должна предусматривать следующие основные роли или права доступа: владелец (осуществляет только развертывание системы, назначает

администратора и осуществляет возможную миграцию данных), администратор (предоставляет и отменяет доступ для низших уровней доступа к системе), производитель (выпуск продукции, управление перечнем разновидностей товаров, редактирование информации о товарах, осуществление продажи, восстановление товара), дистрибьютор (формировать заказ на поставку партии товара), ретейлер (осуществлять продажу товара, устанавливать статусы товару о продаже, изъятии, восстановлении), транспортировщик (вносить статус географического положения товару), IoT-сервер (регистрирование данных о состоянии товара и его перевозке с заданной периодичностью), потребитель (осуществлять проверку информации о товаре, просматривать информацию о обслуживающих процессы пользователях);

– возможность внесение данных о местоположении товара и его условиях хранения: при помощи IoT-устройств или вручную;

– система должна предусматривать концепцию владения товаром и передавать товар от одного участника к другому до момента его продажи, то есть отражение транспортировки товара между участниками;

– система должна предусматривать возможность обеспечения защиты товара от его копирования или незаконное присвоение;

– возможность хранения истории о владельцах товара и его состоянии (новый, в употреблении, восстановлен, обслуживается).

При разработке информационной системы необходимо учитывать нефункциональные требования, которые представляют собой спецификации и качества системы, описывают ее ограничения и возможности.

Были сформированы следующие нефункциональные требования:

– удобство использования: смарт-контракты и документация API должны быть понятны для реализации оракула, отслеживающего IoT устройства;

– производительность: система должна достаточно быстро синхронизировать данные, а также осуществлять транзакции в сравнении с децентрализованными публичными системами, но допускается более медленная работа, чем в централизованных;

– точность и правильность вводимых данных: в blockchain, отслеживаемые данные должны записываться так, чтобы их нельзя было изменить;

– надежность и доступность: система должна быть отказоустойчивой в случаях, если один или несколько узлов выходят из строя;

– масштабируемость: возможность использования системы сотнями компаний;

– ремонтопригодность и портативность: готовая система должна функционировать на базе платформы открытого программного обеспечения Hyperledger или GoQuorum, допускается реализация оракула с использованием Node.js или Golang. Развертывание новых узлов системы должно быть простым и описанным в итоговой документации. Смарт-контракты должны быть развернуты на базе blockchain-консорциума, чтобы каждый участник, обеспечивающий процессы в цепочках поставок, мог развернуть узел blockchain.

Клиентская часть системы должна быть реализована с использованием кроссплатформенных технологий создания приложений;

– безопасность: все проводимые операции должны быть безопасными, а пользователи аутентифицированными, транзакции в системы должны быть видны для пользователей, но допускается наличие конфиденциальных данных;

– неизменяемость: данные, являющиеся фактами сделок, не должны иметь возможность к изменению;

– авторизация: транзакции в процессах, где участвую несколько сторон должны быть одобрены всеми заинтересованными сторонами;

– прозрачность: цепочки поставок должны быть прослеживаемыми, обеспечение хранения неизменяемой истории товара;

Таким образом, для разрабатываемой информационной системы были определены требования.

ГЛАВА 2 ТЕХНОЛОГИЯ BLOCKCHAIN

В данной главе рассматривается технология blockchain, описывается необходимая для понимания терминология, как работает эта технология, ее место в процессах реального мира и управлении заказами, а также рассматриваются смарт-контракты, на основе которых работают существующие децентрализованные blockchain-приложения.

2.1 Общие сведения о работе blockchain технологии

Технология blockchain определена как одноранговая сеть или децентрализованный реестр данных, записывающий и определенным образом зашифровывающий и хранящий данные в блоках, записанные при помощи транзакций и обменивающийся хранимой информацией со всеми устройствами в одной сети [53].

Транзакция в blockchain – это криптографически подписанная учетной записью и одобренная в blockchain инструкция, содержащая какие-либо данные.

Каждая транзакция в сети blockchain имеет следующий жизненный цикл:

– инициация – пользователь совершает действие с blockchain, например передача информации;

– подписание транзакции – при помощи закрытого ключа пользователя подписывается и шифруется содержимое транзакции, после этого отправляется на узел сети blockchain;

– передача в локальный пул памяти узла – транзакция попадает в локальный пул памяти узла. Узел передает транзакцию соседним узлам сети, а они распространяют дальше, пока все узлы не получат эту транзакцию;

– подтверждение – транзакция ожидает своей проверки при помощи алгоритма консенсуса на всех узлах сети. В случае подтверждения транзакция группируется в блок с уникальным хэш-значением, а новый блок связывается с предыдущим криптографической связью;

– завершение – блок становится частью сети [26, 49].

На рисунке 13 схематично отражен жизненный цикл транзакции в blockchain.

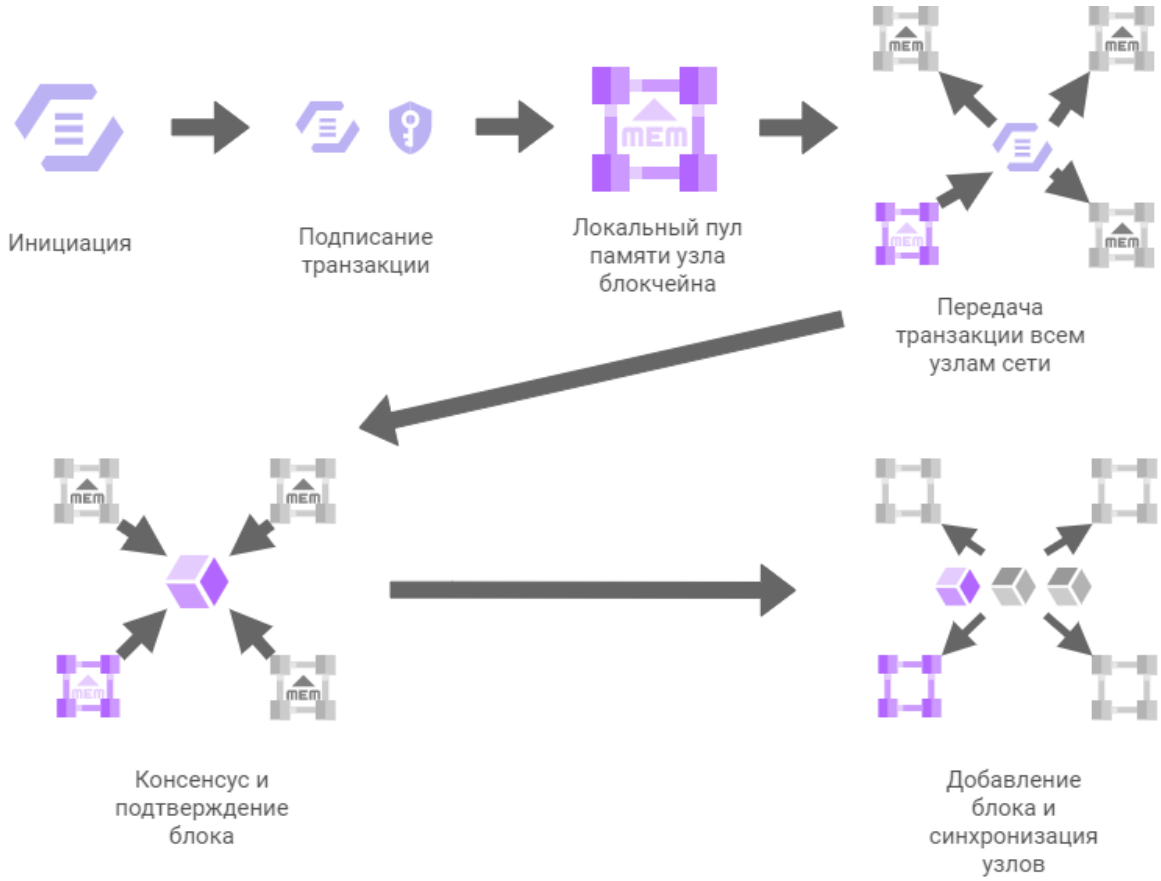

Рисунок 13 – Жизненный цикл транзакции в blockchain

Одно из главных качеств blockchain заключается в его децентрализации, а все блоки blockchain хранятся на его распределенных узлах и синхронизируются после добавления новых блоков. За счет такой децентрализации blockchain имеет высокую надежность и может гарантировать объективность данных.

Блок – пакет транзакций с хэшем предыдущего блока в blockchain, после добавлении блока в blockchain состояние его изменяется необратимо.

Подтверждения добавления транзакции в блок внутри децентрализованной blockchain сети происходит при помощи механизмов консенсуса. Достижение консенсуса в blockchain означает соглашение в подтверждении транзакции на большинстве узлов сети. Механизмы консенсуса представляют из себя набор правил, определенных в его стандарте, которые регламентирует то, как работает система и ее взаимодействие между ее узлами.

Существуют различные механизмы достижения консенсуса внутри децентрализованной сети. Среди них можно выделить следующие:

– доказательство работы (Proof of Work) – определяет какой майнер будет выбран для добычи следующего блока. Каждый узел сети решает математические вычисление, требующих больших вычислительных мощностей и следовательно электричества. Майнер, решивший задачу быстрее всех получает вознаграждение. Пример blockchain-платформ: Bitcoin, Ethereum;

– доказательство доли (Proof of Stake) – определяет какой блок будет добавлен исходя из ставок валидаторов, которые сосредоточивают валюту

системы и блокируют ее во время голосования. Пример blockchain-платформ: Cardano, Solana;

– доказательство сжигания (Proof of Burn) – валидаторы «сжигают» монеты, отправляя их на адрес, откуда их невозможно вернуть, но взамен получают привилегию быть выбранными в системе на основе процесса случайного выбора. От количества сожженных монет зависят шансы на вознаграждение при добыче блока. Пример blockchain-платформ: Slimcoin Blockchain;

– доказательство дееспособности (Proof of Capacity) – валидаторы делятся пространством на жестком диске и чем больше этот объем, тем больше шансы быть выбранным для добычи блока. Пример blockchain-платформ: Burst;

– доказательство прошедшего времени (Proof of Elapsed Time) – каждый валидатор сети имеет шанс на добавлении блока, но до этого происходит ожидание произвольного количества времени. Победителем становится валидатор с наименьшим значением таймера для доказательства. В механизме предусматриваются и другие проверки для снижения вероятности мошенничества. Пример blockchain-платформ: Hyperledger Fabiric;

– и другие [13].

Таким образом, существуют множество различных механизмов консенсуса, каждый из которых имеет свою специфику и ограничения на использование.

На рисунке 1 представлено сравнение версий blockchain-платформ, как технологии.

Таблица 1 – Сравнение версий blockchain-платформ [14]

	Blockchain 1.0 (Bitcoin)	Blockchain 2.0 (Ethereum, Solana)	Blockchain 3.0, (Hyperledger, GoQuorum)	Blockchain 4.0
Основан на	Распределенный реестр	Механизм консенсуса PoW, PoS, смарт-контракты,	Смарт контракты, децентрализованные приложения различные механизмы консенсуса	Внедренные ИИ в blockchain, совершенствованные технологии
Консенсус	PoW	PoS, PoW	PoS, PoA	Proof of integrity
Скорость, количество транзакций в сек	7	30	1000	1 миллион транзакций

Продолжение таблицы 1

	Blockchain 1.0 (Bitcoin)	Blockchain 2.0 (Ethereum, Solana)	Blockchain 3.0, (Hyperledger, GoQuorum)	Blockchain 4.0
Достоинства	Надежность, эффективность, независимость, безопасность,	Уменьшение количество проверок расходов, предотвращение мошенничества, повышение прозрачности и эффективности	Более высокая универсальность и модульность, дополнительные механизмы проверок, более эффективная масштабируемость и функциональная совместимость, повышенная скорость, нет единого контролирующего органа власти, устранение зависимости от майнинга	Высокая масштабируемость, гибкость технологии для разработки, устраненная сложность в разработке приложений, гибкость в запросах к данным
Недостатки	Ограничена функциональность, неспособность поддержки смарт контрактов, не масштабируемый	Сложность разработки приложений, критичность ошибок, плохая масштабируемость	Сложность для разработки приложений, множество ошибок из-за децентрализованности	На данный момент не выявлено, поскольку нет реализаций
Затраты	Высокие	Средние	Низкие	Сверхнизкие
Применение	Финансы	Электронное голосование, торговля, недвижимость	Бизнес-среда, промышленность	Промышленные приложения 4.0

Текущее развитие технологии blockchain является не конечным и потенциал ее дальнейшего развития большой, поскольку ее упрощение для разработки приложений и лучшая производительность в сочетании с другими технологиями будет хорошим драйвером для ее повсеместного принятия и использования.

На данный момент blockchain активно применяется в таких направлениях, как

– финансовое взаимодействие, учет и платежи – безопасность и скорость работы позволяют делать оперативные действия в этих направлениях;

– безопасность денежного оборота от мошенничества – защищенность и прозрачность транзакций может позволить отследить злоумышленника;

– аудит и нормативное и налоговое регулирование – неизменность уже записанной информации позволяет устранить необходимость в аудиторах, а

налоговая система при принятии и интеграции может иметь доступ ко всем торговым операциям организации;

– страхование – за счет возможностей современных blockchain-платформ возможна автоматизации заранее утвержденных действий и инструкции при достижении которых будут предоставлены определенные действия для клиента;

– управление цепочками поставок – преимущества blockchain в части неизменности, безопасности, распределённой и обновлении своего состояния в режиме реального времени позволяют существенно улучшить деятельность такого рода, поскольку она связана с информационной обеспеченностью. В сочетании с IoT-технологиями достигается большая автоматизация и скорость обработки информации, за счет чего системы, использующие сочетания этих технологий, отличаются своей простотой и эффективностью;

– здравоохранение – возможности хранения информации в зашифрованном виде о пациентах, интеграции с носимой электроникой распределено позволяет устранить разрозненность истории пациента;

– недвижимость – за счет сохранения истории о продажах недвижимости возможно уменьшить мошенничество и обеспечить прозрачность для торговли;

– защита интеллектуальной собственности – современный blockchain платформы и приложения при помощи смарт-контрактов позволяют внести концепции владения, за счет чего объектом интеллектуальной собственности можно по-настоящему владеть и определять, как его использовать;

– управление идентификацией личности – в случае, если blockchain используется для идентификации личности можно устранить необходимость в наличии дополнительных документов, подтверждающих личность;

– конфиденциальность – возможность скрытия зашифрованной информации [50].

Таким образом, технология blockchain имеет множество возможностей для своего применения в процессах реального мира и обобщая ее возможности можно выделить следующие:

– безопасность и конфиденциальность – исключение мошенничества и подделки данных, а также скрытие данных в случае их конфиденциальной значимости в бизнес-процессах;

– автоматизация юридических взаимоотношений и отказ от посредников – смарт-контракты позволяют перенести на себя доверие и гарантировать, что стороны соглашения выполнят свои обязательства, распределенность и шифрование системы исключает изменения из вне;

– прозрачность – реализация blockchain предполагает публичность производимых транзакций в системе, а реализация смарт-контрактов позволяет подтверждать право владения активом и предоставлять историю движения актива или денежных средств на счетах;

– цифровизация активов – автоматизация операций и юридическая ценность активов достигается за счет использования криптовалют, поскольку в таком случае контрагенты могут имеют доступ к быстрым переводам средств на счета, а смарт-контракты имеют экономическую ценность, ведь исполнение

обязательств позволяет участникам экономических взаимоотношений получить средства, которые удерживает смарт-контракт, если такие условия были в нем указаны;

– снижение издержек на операции, связанные с финансами – исключения организации посредника позволяет снизить комиссии за переводы и увеличить скорость движения финансов [53].

2.2 Типы blockchain

В зависимости от того какую задачу решает blockchain принято выделять следующие: публичный, частный, консорциум и гибридный.

Публичный blockchain – система, не требующая разрешения для своего использования, в которой каждый человек, имеющий доступ к интернету может стать авторизованным узлом и частью сети. Узел имеет полный доступ ко всей информации в blockchain, имеет право на участие в процессах консенсуса. Основное использование таких blockchain-платформ заключается в использовании криптовалют, децентрализованных приложений, игр, торговых площадок Публичный blockchain имеет преимущества в надежности, безопасности, открытости и прозрачности, но имеет проблемы низкой скорости транзакций и проблемы в масштабируемости;

Частный blockchain – blockchain, доступ к которому ограничен намеренно, а контролирующая организации ответственна за уровень безопасности, способы авторизации, разрешения на использования сети и ее доступность. Такие системы крайне полезны для контроля производственных процессов и определенных изолированных способов использования. Такие системы используются для голосования, управления цепочками поставок, идентификации владельцев активов и другие способы применения в промышленности. Частный blockchain имеет преимущества в скорости и масштабируемости сети, но централизованность понижает уровень доверия к сети и порождает низкую безопасность.

Blockchain консорциума – blockchain, имеющий частично децентрализованную структуру, в которой управляют blockchain несколько организаций, каждая из которых может выступать узлом системы. Такого рода blockchain используются несколькими организациями для гарантии равенства в пределах одной сети для каждого участника, например банками, государственными организациями.

Гибридный blockchain – blockchain, который может совмещать функции частного и публичного blockchain. В таком случае может быть система, подключение узла в которой может требовать особых разрешений для участия в процессах достижения консенсуса и добавления новых блоков или разрешения для получения к различным уровням доступ информации, а сами участники могут быть конфиденциальны друг для друга и раскрыты в случае принятия участия в транзакции.

Blockchain консорциума и гибридные увеличивают скорость сети и используются в сочетании с IoT, финансами и торговлей, цепочки поставок, государственном управлении и других областях, а также отличаются повышенной скоростью работы и безопасностью, то есть сочетают лучшие качества от частных и публичных blockchain-платформ.

Таким образом, существуют различные типы blockchain-платформ, каждый из которых имеет собственную специфику использования, отличается скоростью проводимых транзакций. Выбор типа blockchain зависит от специфики бизнес-процессов, для которых он используется [51].

2.3 Применение blockchain технологии в процессах цепочки поставки

Как было описано выше, blockchain технологии обладают свойствами, которые могут быть полезны для бизнеса самого различного рода. Применяя эти технологии в управление заказами или связанными с ними, организация, внедряющая систему такого рода, может получить следующие преимущества:

– уменьшение работы с физическими носителями информации – blockchain технологии позволят создать распределенную и надежную систему электронного документооборота, что позволит существенно сократить время на обработку информации, которая поступает от контрагентов и позволит сотрудничающим организациям установить более надежные и доверительные отношения;

– упрощение контроля над происхождением товаров – оригинальность поставляемой продукции является важным аспектом в деятельности организаций, поскольку некачественная подделка или бракованный товар несут не только репутационный и финансовые риски для организаций, которые ведут честный бизнес, а также могут иметь критическое значение для здоровья самого потребителя, что крайне важно для пищевой промышленности и фармацевтической индустрии. Системы, использующие blockchain для хранения информации, связанной с цепочкой поставки товара позволяют отследить в истории звено, на котором произошла подмена товара, его хищение или определить организацию, которая занимается созданием подделок, игнорируя законы или возможно выяснить на каком производственном или логистическом этапе существуют проблемы;

– автоматизация логистических процессов и расчетов за счет использования смарт-контрактов позволяет снизить расходы на электронный документооборот и операции, связанные с финансовыми транзакциями между организацией-заказчиком и потребителем, поскольку выполнение контрактных обязательств сторонами при такой инфраструктуре контролируется программным кодом, который в распределенном blockchain невозможно контролировать или изменять из вне.

На данный момент существуют разные решения на базе blockchain, которые учитывают цепочку поставок. Среди таких организаций можно выделить следующие наиболее известные и крупные:

– Walmart – Американская розничная компания, успешно внедрила и решила описанные в данной работе проблемы в процессах управления цепочками поставок. Используемый частный blockchain, ориентирован на партнеров организации, и построен на открытой платформе Hyperledger Fabric [27];

– Amazon – Американская крупнейшая компания в мире, деятельность которой связана с электронной коммерцией, облачными вычислениями и многими другими видами деятельности использует в собственных процессах управления цепочками поставок blockchain технологии на базе Hyperledger Fabric, а также предлагает управляемое решение для клиентов, которые хотят использовать частные сети [6];

– Everledger – компания-разработчик, предоставляющая технологические отраслевые решения для таких сфер, как искусство, минералы, бриллианты, мода, предметы роскоши и алкогольные напитки. Предоставляемые решения используются для повышения прозрачности цепочек поставок и идентификации объекта на базе решений IoT в сочетании с blockchain-технологиями [20, 45];

– IBM – компания-поставщик аппаратного и программного обеспечения, имеет отраслевое решение IBM Food Trust, сотрудничает с такими компаниями-производителями пищевой продукции как Carrefour и Nestle в рамках разработанной blockchain-платформы, позволяющей отслеживать продукты питания в цепочках поставок [7, 30].

В аналитических исследованиях использования blockchain технологий по странам лидирующими выступают Китай и США, а после страны Европы. Это во многом связано с уровнем технологической развитости производств [35]. Использование в России и большинства других развивающихся стран такого рода технологий на данный момент только обсуждается и рассматривается для пригодности применения в различных областях.

Несмотря на то, что blockchain, как инструмент для совершенствования процессов реального мира, имеет множество преимуществ и возможностей, существует ряд проблем и препятствий в его использовании.

Среди таких выделяют следующие:

– отсутствие правового регулирования и признания на государственном уровне, что не позволяет на законодательном уровне предоставить юридическую значимость смарт-контрактам;

– повышенная сложность разработки по сравнению с традиционными централизованными системами и как следствие недостаток предложения квалифицированных специалистов на рынке;

– сложность в выборе конкретной технологии, поскольку на данный момент существует множество конкретных технологических blockchain платформ, каждая из которых решает конкретные задачи, отличается предлагаемыми решениями, использованием языков программирования смарт-контрактов, скоростью и так далее.

Готовые решения, на данный момент применяются в ограниченном масштабе, такого рода решения лицензируются по модели подписки и не являются открытыми для любого пользователя или бизнеса. Клиентами

организаций, предоставляющих blockchain и формирующие готовые логистические процессы, являются организациями, имеющие существенные финансовые вложения в собственную технологическую инфраструктуру, которая и позволяет существенно автоматизировать логистические процессы.

Существуют также компании, специализирующиеся на разработке программного обеспечения, к которым обращаются организации, нуждающиеся в оптимизации логистических процессов. Примером такой компании может послужить ScienceSoft, которая разрабатывает клиентские прикладные приложения на базе смарт-контрактов и открытой частной blockchain платформы Hyperledger Fabric [11].

Крупные организации используют собственный blockchain для обеспечения собственной инфраструктуры в работе с контрагентами, смысл которого заключается прежде всего в хранении данных, создании цифрового двойника для сложных технических товаров [1].

2.4 Технология смарт-контрактов

Основой для внесения в blockchain информации и написания прикладных приложений являются смарт-контракты, которые обеспечивают выполнение условий соглашений и не требуют участия посредника. Смарт-контракты являются частью современных blockchain решений и представляют собой программу, которая работает в виртуальной машине blockchain, хранится в blockchain и имеет собственное состояние.

Выделяют следующие основные особенности смарт-контрактов:

– смарт-контракт имеет собственный адрес в сети, имеет баланс, может хранить в себе криптовалюту и состояние данных;

– автоматизация операций, связанных контекстом смысла и задачи, которую должен решать программный код. Смарт-контракты имеют ограничения автоматизации, которые выражаются в том, что вызов функций смарт-контракта должен осуществляться из вне, а значит он не имеет доступа к внешним данным для их самостоятельного анализа;

– в смарт-контракте все условия детерминированы, а значит и результат не является случайным и вполне прогнозируем;

– развернутый смарт-контракт не может быть изменен, что является как ограничением, так и гарантией, которая способствует доверию пользователем, а действия, которые пользователь осуществил со смарт-контрактом, остаются в blockchain;

– смарт-контракт и его код является общедоступным, что вносит прозрачность в отношении его смысла;

– формирование доверительных отношений между сложными взаимоотношениями между контрагентами, при таком случае стороны соглашения делегируют управление и контроль смарт-контракту, что более экономично и быстрее, чем работа с третьей стороной [8, 31].

Во многом работа смарт-контракта схожа с жизненным циклом транзакции, поскольку смарт-контракт работает поверх механизма транзакций в blockchain в виртуальной машине, которая исполняет программный код:

– изменения данных в смарт-контракте – запускает транзакцию, которая будет упакована в блок, который должен быть согласован при помощи консенсуса со всеми узлами в сети. В случае записи информации некоторая информация может быть записана в журнал событий, который можно прослушивать асинхронно;

– чтение информации в смарт-контракте – не изменяет состояние в blockchain [55].

На рисунке 14 представлена схема работы смарт-контракта и его связь с компонентами blockchain.

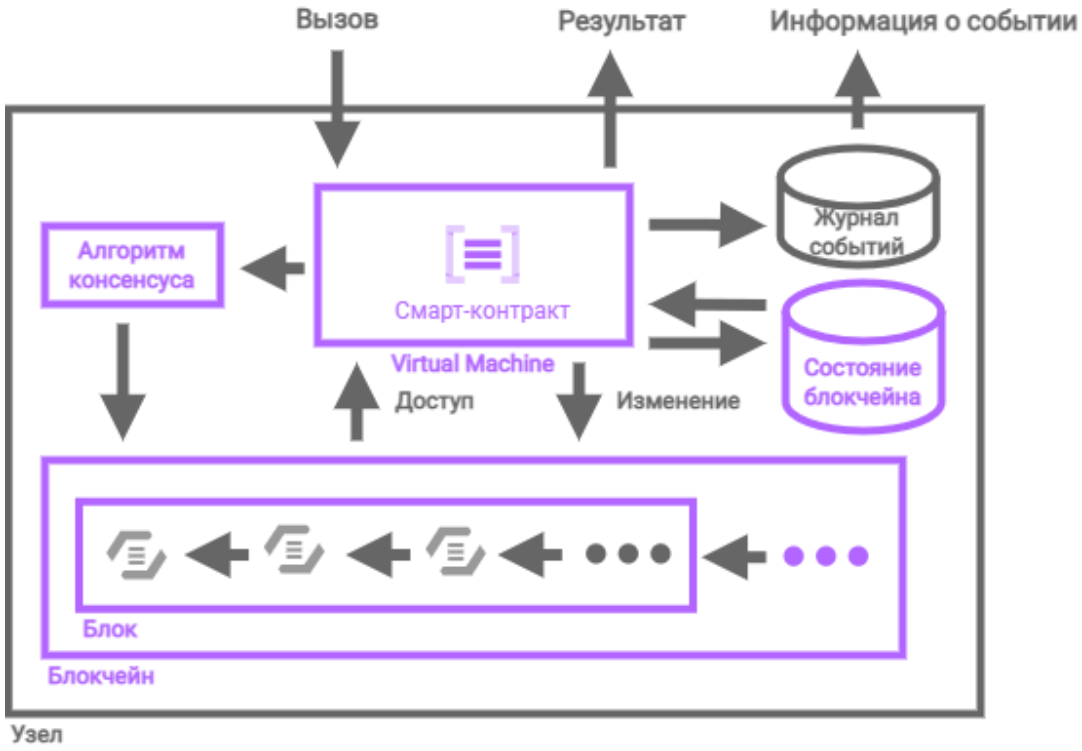

Рисунок 14 – Схема работы смарт-контракта

Смарт-контракты имеют следующие элементы:

– переменные – образующие состояние смарт-контракта. Переменные могут иметь 4 области видимости: частная, общедоступная, внешняя и внутренняя. Существуют следующие типы переменных: целые числа со знаком, целые числа без знака, логические значения, адреса, перечисления и байты;

– функции – исполняемые единицы кода, которые состоят из имени, тела, входных значений и выходного результата, опционального модификатора, а также область видимости, как в переменных. В функции определяется вся бизнес-логика.

– модификаторы – используются для устранения дублирования кода в случае проверки на соблюдение определенных разработчиком условий. Модификатор вызывается перед вызовом функции и в случае успеха дальше выполняется тело функции;

– события – генерируются при вызовах функций в смарт-контрактах и представляют из себя интерфейсы для вывода информации. События имеют следующие значения: идентификатор, хэш транзакций, тип события, хэш блока, в котором хранится информация о событии, индекс.

При разработке децентрализованных приложений на базе смарт-контрактов необходимо уделять особое внимание безопасности и тестировать смарт-контракты, поскольку чаще всего в смарт-контрактах хранится важная информация или средства, которые злоумышленник может похитить, так как код смарт-контрактов общедоступен для участников сети. Большинство существующих ошибок связаны с невнимательностью разработчика или непониманием того, как работает бизнес-процесс, автоматизируемый смарт-контрактом, поэтому в разработке децентрализованных приложения наибольшее внимание уделяется на этапе тестирования [55].

Поскольку из-за детерминированности смарт-контрактов и изолированности их о от внешних данных, которые находятся за пределами blockchain, при разработке децентрализованных программных решений на базе смарт-контрактов применяются оракулы – промежуточное программное обеспечение, которое представляет собой сервисы, соединяющие blockchain с внешними источниками данных, устройствами и сервисами [15].

Выделяют следующие виды blockchain-оракулов:

– программные – представляют из себя серверы, обрабатывающие и передающие данные в blockchain;

– аппаратные – представляющие собой физические устройства, IoT, RFID метки, датчики и передающие данные в blockchain. Отличаются тем, что заслуживают большего доверия, поскольку их сложнее скомпрометировать и полезны для процессов, в которых необходимо наблюдение за объектами реального мира;

– входящие – оракулы, передающие из внешних источников информации в blockchain;

– исходящие – такие оракулы передают данные из blockchain внешней стороне для наблюдения;

– основанные на консенсусе – оракулы, использующие специальные алгоритмы для получения фактических или приближенных к реальности данных и позволяют гарантировать, что данные, передаваемые в blockchain точны, и заслуживают доверия

– с поддержкой вычислений – вычисляют данные вместо вычислений их в виртуальных машинах blockchain, поскольку стоимость вычислений в них может оказаться слишком высокой;

– перекрестные – оракулы, обменивающиеся данными между разными blockchain-платформами;

– детерминированные – оракулы, построенные на основе надежных источников данных и применяющие алгоритмы для учета показателей репутации в процессе получения данных [15].

Оракулы необходимы для автоматизации процессов, обработки собираемых данных и их последующая фиксация в blockchain, но несмотря на свои достоинства, имеют такие проблемы, как:

– достоверность предоставляемых данных от оракула;
– порождают зависимость blockchain от функционирования оракула.

Следует отметить, что величина существующих проблем имеет значения не для каждой области применения blockchain и зависит от стоимости нарушения целостности системы.

ГЛАВА 3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Проектирование информационной системы является наиважнейшим этапом в разработке любой информационной системы, поскольку на этом этапе осуществляется рассмотрение возможностей и альтернатив в использовании программных продуктов, компонентов, функциональных особенностей, возможностей, взаимосвязей между частями системы, что в дальнейшем позволит эффективно удовлетворять потребности стейкхолдеров и бизнес-процессов.

3.1 Архитектура информационной системы

Разрабатываемая информационная система представляет из себя гибридное децентрализованное приложение для обеспечивающих процессов цепочки поставок. Архитектура разрабатываемой информационной системы представлена на рисунке 15.

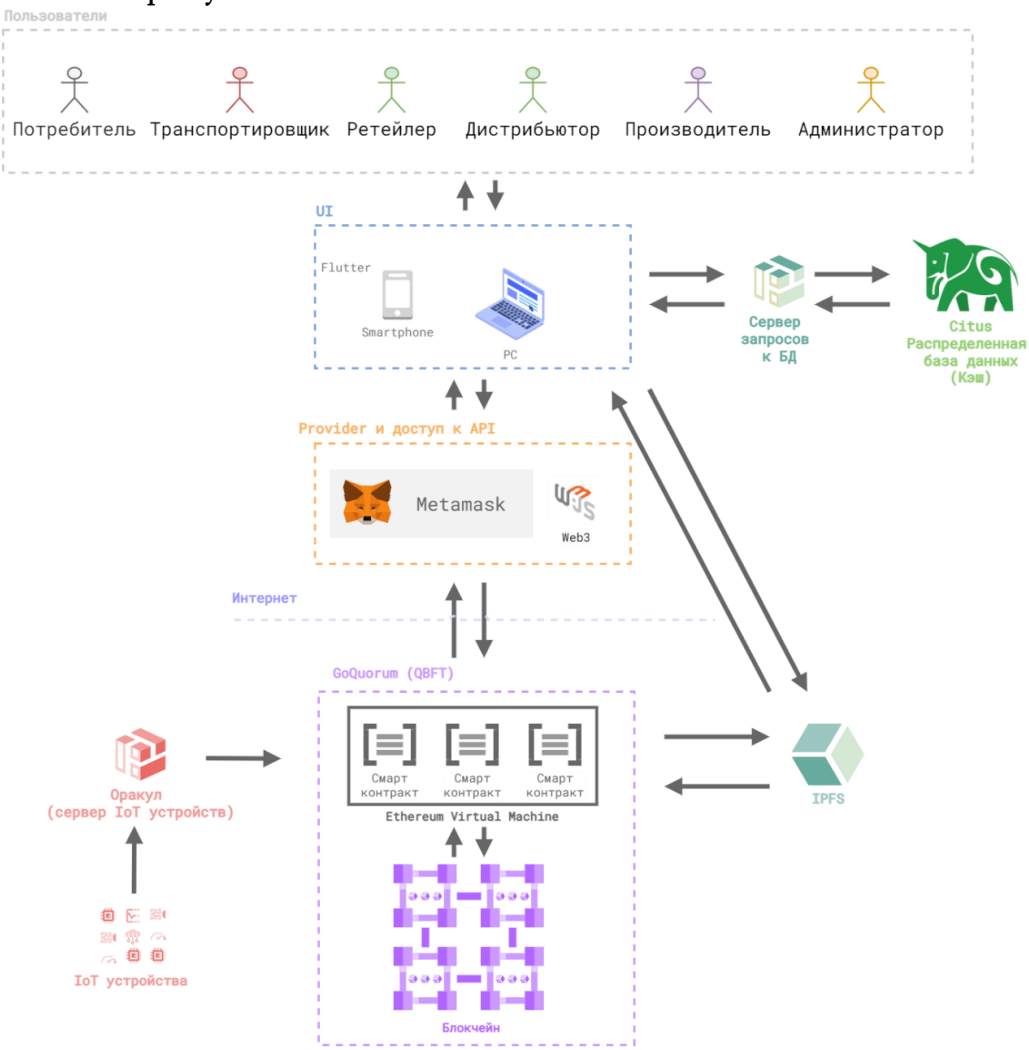

Рисунок 15 – Архитектура информационной системы

На представленном архитектуре системы присутствуют следующие компоненты:

– UI – кроссплатформенное приложение, позволяющее пользователю работать как на смартфонах, так и на компьютерах;

– Provider и доступ к API – обеспечение соединения клиентского приложения с blockchain при помощи JSON-RPC спецификации и Web3 библиотека для создания транзакций в blockchain;

– IPFS – хранилище файлов и других данных, которые слишком большие для хранения их в blockchain 32];

– Citus – расширение, позволяющее использовать СУБД PostgreSQL распределено [18];

– сервер запросов к БД – прослойка между СУБД Citus, необходимое только для обработки CRUD запросов к БД;

– GoQuorum – платформа для разработки промышленных информационных систем на базе blockchain [9];

– оракул (сервер IoT устройств) – доверенный сервер, который обрабатывает входящую информацию с IoT-устройств и фиксирует в blockchain местоположение перевозимого груза;

– IoT устройства – доверенные физические устройства, сенсоры и передатчики геолокации, устанавливаемые рядом с транспортируемым грузом и передающие информацию оракулу.

На рисунке 16 представлено взаимодействие основных компонентов системы между собой.

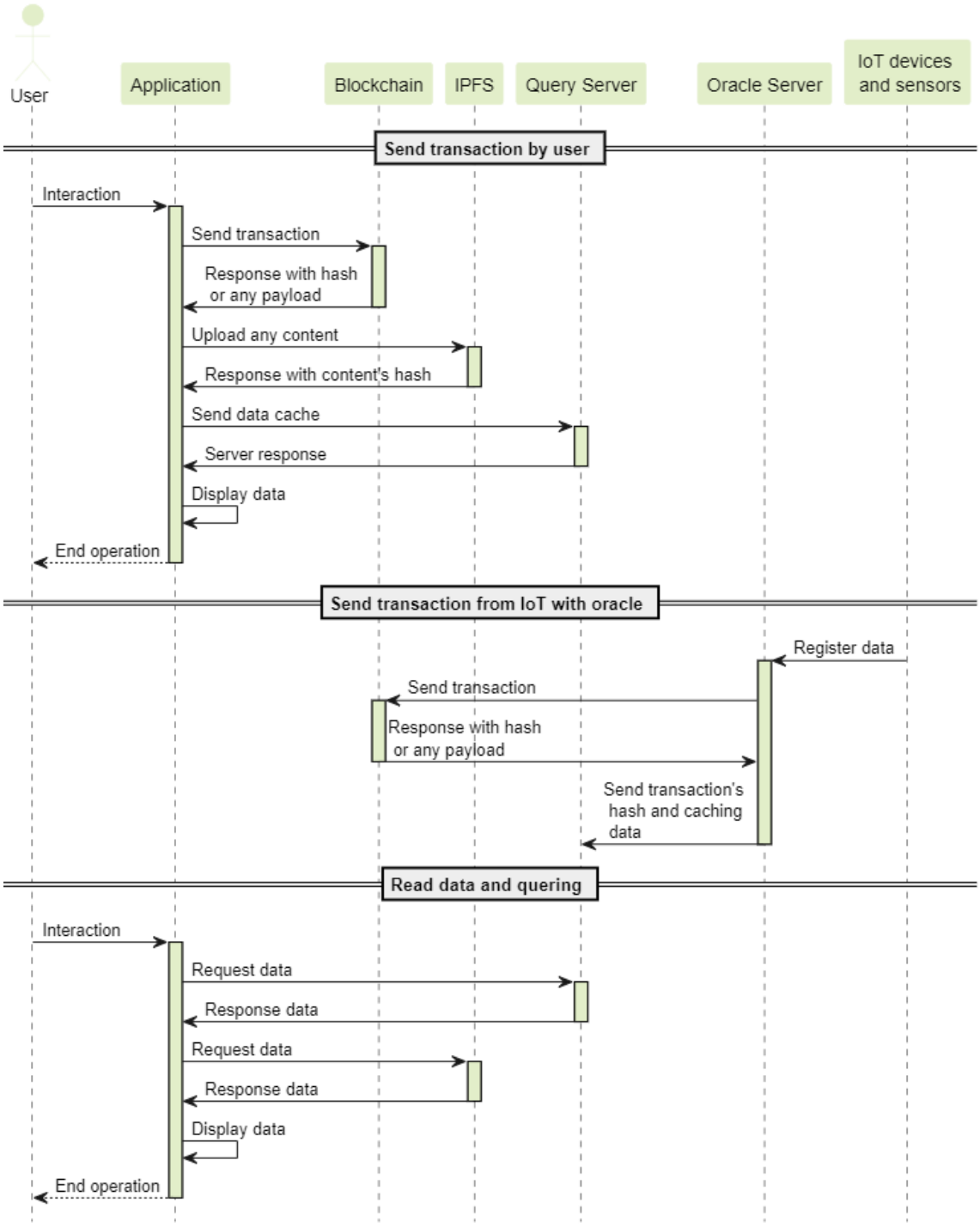

Рисунок 16 – Взаимодействие основных компонентов системы между собой

На представленной диаграмме последовательности отражены следующие основные возможные варианты:

– отправление транзакции пользователем – последовательности передачи данных между собой. Пользователь совершает действие в прикладном приложении после чего отправляется транзакция в blockchain, результатом которой становится возвращаемый хэш транзакции, после чего совершенные данные в системе кэшируются при помощи сервера запросов с распределённой базой данных. Результат совершенного действия отражаются в прикладном приложении;

– отправление транзакции оракул после поступление данных от IoT-устройства – от IoT-устройства, зарегистрированного в сервере IoT-устройств, поступают данные о перемещении заказа, после чего эти данные могут быть каким-то образом проверены или обработаны на достоверность и далее отправляется транзакция в blockchain с последующим результатом в виде хэша. Полученные данные кэшируются при помощи сервера запросов с распределённой базой данных;

– чтение данных и запросы – данная операция происходит при взаимодействии пользователя с прикладным приложением. В данном случае данные получаются пользователем от сервера запросов с распределённой базой данных.

3.2 Blockchain и смарт-контракты

3.2.1 Конфигурация blockchain

Для информационной системы на базе blockchain применяется GoQuorum – открытая платформа с технологией децентрализованного распределенного реестра, предназначенная для разработки децентрализованных приложений, функционирующих в корпоративной среде. Платформа отличается от существующих blockchain решений тем, что является улучшенным blockchain решением Ethereum, поскольку имеет модульную архитектуру, которая обеспечивает универсальность, оптимизацию и инновационность, что позволяет настроить платформу исходя из специфики промышленных бизнес-процессов самых различных областей [9].

GoQuorum, является платформой с опциональным контролем доступа, то есть возможно развернуть как публичную сеть, так и с разрешениями, поэтому с точки зрения целей существования распределенного реестра не могут полностью иметь взаимное доверие, но в случае применения определенной модели управления, когда все участники обеспечивают поддержку blockchain путем развертывания собственных разрешенных узлов возможно сформировать доверительные взаимоотношения между участниками и закрепить взаимоотношения юридическим соглашением.

GoQuorum имеет следующие основные компоненты:

– EVM, позволяющая разрабатывать и исполнять смарт-контракты на языке Solidity;

– хранилище для хранения данных, связанных с выполнением транзакций;

– одноранговая P2P сеть для связи с другими узлами blockchain;

– API для взаимодействия с blockchain;

– разрешения для участия;

– и другие компоненты для мониторинга, конфиденциальности и т. д [9].

Для реализации децентрализованного прикладного приложения предполагается настройка гибридного blockchain, доступ к которому является публичным, но для операций, связанных с смарт-контрактами и бизнес-процессами доступ будет не у всех участников: обычные покупатели товаров

могут использовать приложение для того, чтобы проверить цепочку поставок и историю владения товаром; пользователи, имеющие повышенный уровень доступа могут использовать функции системы, предусмотренные в рамках бизнес-процессов и их ролей.

Настройка blockchain сети предполагает следующие основные конфигурации:

– использование консенсусного протокола QBFT;

– использование модели расширенных разрешений для осуществления транзакций в сети;

– отключение газа для осуществления транзакций.

Для blockchain в данной работе применятся консенсусный протокол корпоративного уровня для частных сетей QBFT.

Схема работы QBFT такова, что валидаторы (утвержденные учетные записи), проверяют транзакции и блоки. Валидаторы по очереди создают следующий блок. Прежде чем вставить блок в цепочку, подавляющее большинство (более или равное 2/3) валидаторов должны сначала подписать блок. Существующие валидаторы предлагают и голосуют за добавление или удаление валидаторов. Для добавления или удаления валидатора требуется большинство голосов (более 50%) валидаторов.

Блок в QBFT создается каждый установленный период, независимо от наличия транзакций, что может привести к тому, что blockchain может быть объемным из-за огромного количества пустых блоков, особенно если установлен маленький период блоков, однако в GoQuorum можно уменьшить количество пустых блоков за счет дополнительной опции в файле конфигурации генезиса. Таким образом, если транзакции ожидающих обработки нет, то блоки создаются с заданным периодом времени [16].

Модель расширенных разрешений внутри сети blockchain основана на смарт-контрактах и разработана для нужд предприятий в части контроле доступа сотрудников к blockchain. При помощи смарт-контрактов происходит предоставление доступа сотрудников к blockchain, который на основе внутренних механизмов обеспечивает или запрещает доступ сотрудникам к осуществлению транзакций [17].

Также используемая конфигурация blockchain отключает газ для осуществления транзакций внутри сети, поскольку для частной сети этот механизм является бессмысленным и затратным [22].

3.2.2 Смарт-контракты

Для разработки смарт-контрактов используются объектно-ориентированные языки программирования. Для каждого blockchain предусмотрены различные варианты разработки смарт-контрактов.

В данной работе рассматриваются смарт-контракты в стандарте языка Solidity, который является объектно-ориентированным языком высокого уровня и предназначен для виртуальной машины Ethereum. Solidity имеет наибольшую поддержку сообщества в сравнении с другими языками для написания смарт-контрактов, а также является быстро-развивающимся.

В Solidty имеются следующие основные концепции:

– типизация – язык поддерживает строгую типизацию и следующие основные типы данных: bool – логическое значение true/false; int8-int256; uint8-uint256 – целочисленное значение без знака (с шагом размерности в 8 бит); address – стандартный адресный тип (20 байт) для адресации в blockchain контрактов/счетов; bytes1-bytes32 – массив байтов с указанным размером; bytes – динамический массив байтов; string – строка; enum – перечисление; struct – структура данных; массивы данных; mappings – хеш-таблица.

– разделение переменных и данных – в Solidity переменные и данные разделяются при помощи следующих ключевых слов: storage (хранилище) – переменные состояния, которые хранятся в blockchain; memory (память) – переменные, которые хранятся в рамках функции; calldata (вызываемые данные) – временное место для данных, где хранятся аргументы функций;

– множественное наследование – при множественном наследовании один контракт может быть унаследован от многих контрактов. У родительского контракта может быть несколько дочерних, а у дочернего контракта может быть более одного родителя. В таком формате удобно разделять программный код на модули, связанные между собой;

– события – сохраняет аргументы, переданные в журналы транзакций при создании. Как правило, события используются для информирования вызывающего приложения о текущем состоянии контракта с помощью средства регистрации EVM;

– области видимости – язык поддерживает следующие области видимости для функций: public, private, internal и external. Для переменных состояний доступны следующие области видимости: public, internal, private;

– модификаторы функций – С помощью модификаторов можно изменять поведение функций. Если задан модификатор, перед выполнением функции проверяется соответствующее условие;

– функции – В рамках контракта исполняемые единицы кода называются функциями. С помощью функций описываются отдельные действия, необходимые для выполнения общей задачи. Функции можно использовать повторно и вызывать из других исходных файлов, например библиотек. Поведение функций в Solidity аналогично другим языкам программирования [45].

Для описания структуры смарт-контракта и отношений между его компонентами в данной работе используется диаграмма классов в нотации UML, поскольку такое отображение может быть использовано в объектно-ориентированных языках, при помощи которых и разрабатываются смарт-контракты.

Рисунке 17 представлена диаграмма классов для смарт-контракта системы управления заказами в сокращении [40, 37]. В развернутом виде, диаграмма представлена в приложении А.

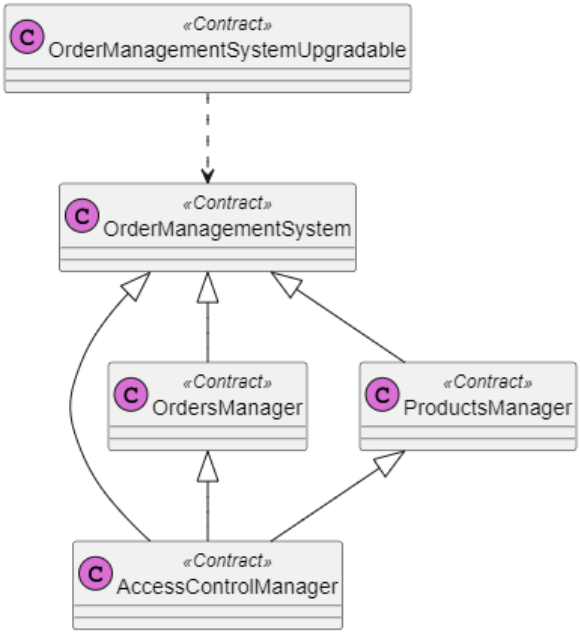

Рисунок 17 – Диаграмма классов для смарт-контракты

Таким образом смарт-контракт, на представленной диаграмме классов является целостным и при помощи возможностей языка Solidity разделен на следующие логические модули за счет механизма наследования: OrderManagementSystemUpgradable, OrderManagementSystem, AccessControlManager, OrdersManager, ProductsManager.

Разработанный смарт-контракт для системы управления заказами предусматривает следующую ролевую модель:

– владелец – аккаунт пользователя, развернувшего смарт-контракт в blockchain;

– администратор – аккаунт пользователя, контролирующего доступ к смарт-контракту и наделяющий других пользователей ролями;

– администратор организации – аккаунт пользователя, имеющего права на создание организации и наделяющий правами на распоряжение ресурсами организации, осуществление продаж;

– производитель – участник системы, регистрирующий в смарт-контракте новый продукт, а также оказывающий восстановление продукции, утратившей свои свойства, исходя из предоставляемого гарантийного срока;

– продавец – участник системы, организующий продажу продукции конечному клиенту, а также имеющий доступ к регистрации нового клиента в системе. Таким участником может являться также и оракул – система, которая обновляет состояния продуктов, заказов исходя из событий, происходящих в реальном мире. В качестве оракула выступает отдельный сервер, который обрабатывает входящие запросы от автоматизированных устройств и систем и совершает транзакций в blockchain;

– покупатель – конечный потребитель продукции, при приобретении товара, продавец регистрирует его в системе;

Важным стоит отметить то, что один и тот же аккаунт может иметь несколько ролей.

3.2.3 Тестирование смарт-контрактов

Поскольку развернутые смарт-контракты являются детерминированными, а это значит, что изменить программный код и внести новую бизнес-логику в них нельзя, что необходимо учитывать при разработке децентрализованных приложений. Именно поэтому необходимо уделить особое внимание тестированию смарт-контрактов на предмет наличия в них ошибок или уязвимостей, поскольку для системы, ориентированной на то чтобы стать единственным источником правды необходимо гарантировать, что не будет непредсказуемых ситуаций при использовании приложений, основанных на смарт-контрактах в реальных условий, ведь доступ к предполагается, что доступ к blockchain будет для большого количества пользователей.

Тестирование смарт-контрактов осуществляется при помощи развертывание программного кода смарт-контрактов внутри локального blockchain и может осуществляться при помощи следующих подходов:

– ручное тестирование – осуществляется путем последовательного вызова кода смарт-контрактов в специализированной IDE Remix для разработки смарт-контрактов на языке Solidity. Преимущество подхода заключается в том, что можно быстро опробовать бизнес-логику программы на предмет ее верного выполнения, но в долгосрочной перспективе подход занимает много времени для проверки всей бизнес-логики программы. Целесообразно использовать в простых программах или в сочетании с разработкой, когда необходимо быстро проверить работоспособность программы;

– автоматизированное тестирование – осуществляется при помощи написания программного кода, который в автоматизированном режиме тестирует смарт-контракты исходя из тест-кейсов, написанного тестировщиком. Преимущество подхода заключается в том, что такой способ тестирования экономит время при длительных этапах разработки, поскольку возможно написания тест-кейсов до того, как будет разработан смарт-контракт, а также нет необходимости в повторении одних и тех же действий. Целесообразно использовать для больших программных продуктов.

Для автоматизированного тестирования смарт-контрактов применяется специальное окружение Hardhat для профессиональной разработки смарт-контрактов.

На рисунке 18 представлена схема, отражающая процесс взаимодействия тест-кейсов с тестовым окружением для автоматизации тестирования.

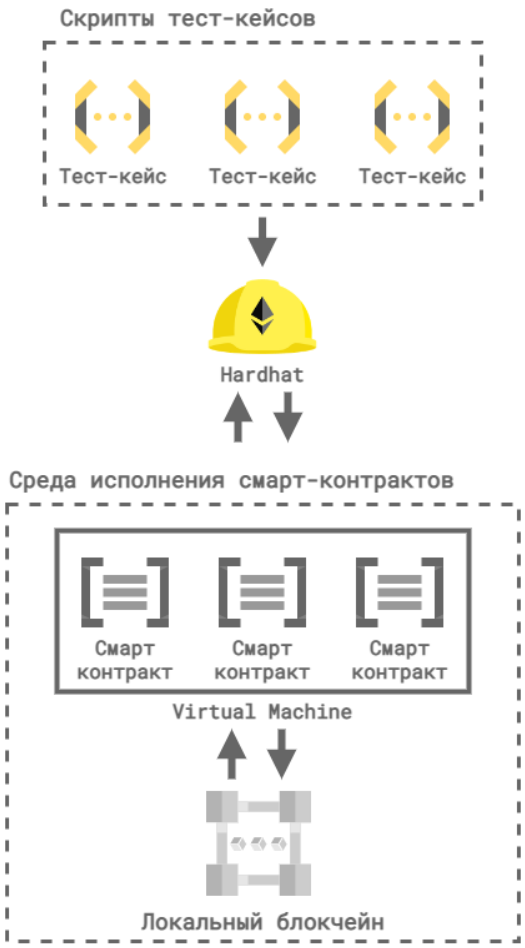

Рисунок 18 – Схема тестового окружения смарт-контрактов

3.3 IPFS

В децентрализованных приложениях важным аспектом является хранение объемных данных, например изображения, файлы, спецификации. В blockchain хранить такие данные не выгодно и избыточно.

Для решения проблемы хранения данных в децентрализованных приложениях используются IPFS – межпланетная файловая система, представляющая собой одноранговый гипермедиа-протокол, разработанный для того, чтобы сделать Интернет быстрее, безопаснее и более открытым.

Особенности IPFS состоят в следующем:

– IPFS децентрализована, потому что загружает контент с тысяч пиров, а не с одного централизованного сервера. Каждая часть данных криптографически хешируется, в результате чего создается безопасный уникальный идентификатор контента;

– целостность содержимого IPFS можно проверить криптографически;

– содержимое IPFS дедуплицируется и при загрузке двух одинаковых файлов на одном и том же узле IPFS, они будут сохранены только один раз, что устранит дублирование, поскольку их хэш будет создавать идентичный CID [10, 32].

В данной работе IPFS применятся для загрузки файлов со спецификациями о продукции, различных изображений, используемых в системе. После загрузки

файла в хранилище, хэш файла передается в смарт-контракт для обеспечения подлинности файла.

3.4 Распределенная СУБД Citus для кэширования данных

На данный момент смарт-контракты имеют существенный недостаток в виде отсутствия индексации хранимых данных и отсутствия специального языка, который мог бы обеспечивать сложные запросы к хранимым данным в смарт-контракте. Именно по этим причинам, для разработки полноценных, удобных в использовании приложений все еще необходимы классический базы данных.

В данной работе, использование вспомогательной базы данных означает не перенос всей основной функциональности приложения, связанного с хранением данных на классические базы данных, а использование свойств этих баз данных, связанных с сложными запросами, поскольку существующие методы доступа к данным в смарт-контрактах не достаточно гибкие, что сильно усложняет структуру смарт-контрактов. Применение реляционных баз данных в децентрализованных приложениях позволяет кэшировать данные транзакций для удобного и быстрого взаимодействия с ними.

В данной информационной системе, используется Citus, который представляет из себя расширение, позволяющее не только использовать СУБД PostgreSQL распределено, но и предоставляет удобные механизмы для администрирования распределенных баз данных, что позволяет каждый узел blockchain определить вместе с базой данных, которая индексирует и кеширует данные в смарт-контракте, тем самым скорость доступа к системе остается на высоком уровне, возможно осуществлять анализ данных при помощи сложных запросов, а данные в blockchain служат неизменным доказательством, совершенных действий в автоматизируемых процессах [18].

На рисунке 19 представлена логическая структура распределенной базы данных Citus, каждый узел которой сопровождается вместе с узлом blockchain.

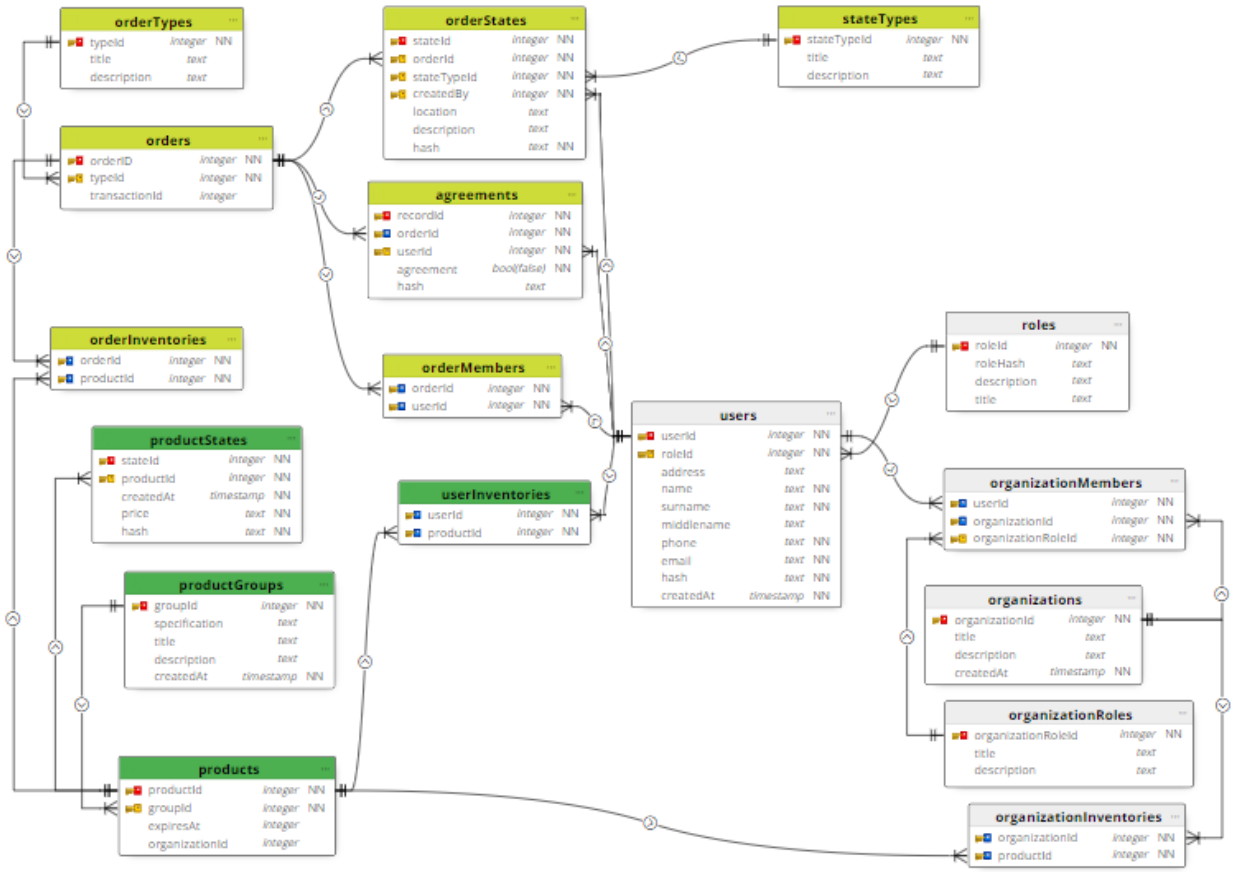

Рисунок 19 – Логическая схема базы данных для кеширования данных в смарт-контрактах

Логическая схема данной базы данных имеет третью нормальную форму, и достаточную, для атомарности сущностей в автоматизируемых процессах.

В базе данных присутствуют таблицы (сущности), представленные в таблице 2.

Таблица 2 – Таблицы в логической схеме базы данных

Название таблицы в логической схеме	Описание
roles	Роли пользователей
users	Пользователи
organizations	Организации
organizationMembers	Участники организации
organizationRoles	Роли в организации
organizationInventories	Инвентарь организации, хранящий список всех имеющихся продуктов, которые организация имеет в своем распоряжении

Продолжение таблицы 2

Название таблицы в логической схеме	Описание
inventories	Инвентарь пользователя, список всех продуктов, которыми владеет пользователь
products	Продукты, внесенные в систему производителями
productGroups	Различные группы продуктов с описанием
productStates	Состояния продуктов
orders	Заказы, обеспечивающие товарообмен между контрагентами внутри одной сети
orderTypes	Типы заказа
stateTypes	Различные типы состояний заказа
orderInventories	Продукты в заказе
orderMembers	Участники заказа
agreements	Соглашения на передачу товара
orderStates	Состояния заказа

Таким образом, данная логическая структура базы данных, позволяет обеспечить полное кеширование данных, хранимых в blockchain и повторяет структуру смарт-контракта, но хранимые данные в базе данных при этом являются атомарными, за счет чего возможны гибкие запросы к данным.

3.5 Оракул IoT-устройств

Оракул необходим для автоматизации некоторых процессов, связанных с происходящими в мире событий, такими как внесение товара в систему, географическое местоположение товара, определение его на складе, необходим сервер, который будет прослушивать события, поступающие из вне, от датчиков или операторов, которые занимаются транспортировкой груза. При осуществлении событий будут получены данные и запущены процессы связанные с добавлением информации в blockchain или считывании их для проверки. Оракул возможно разработать на любой существующей серверной платформе, но в данной архитектуре предполагается использование платформы Node.js – Node.js является серверной платформой, которая использует JavaScript как основной язык программирования и используется для разработки web-серверов, серверов для IoT-устройств, а также разрабатывать клиентские веб-приложения. Платформа также поддерживает Typescript – разработанный язык программирования, который является надмножеством JavaScript и обеспечивает более безопасную разработку [46].

3.6 Provider и доступ к API

Provider – необходим для взаимодействия с blockchain и реализующий спецификацию JSON-RPC. Для записи в blockchain необходимо подписать транзакцию при помощи закрытого ключа, что позволяет сделать Web3, который хранит закрытые ключи пользователя в устройстве.

Metamask в данном случае используется как инструмент для хранения закрытых ключей пользователя, что позволяет упрощенно получить доступ пользователем к системе.

Библиотека Web3 хранит в себе инструменты, которые позволяет взаимодействовать с узлом blockchain, т.е. осуществлять транзакции, получать данные с blockchain и слушать события.

3.7 Клиентское кроссплатформенное приложение

Для взаимодействия с blockchain и смарт-контрактами в целях автоматизации процессов электронного документооборота в децентрализованных приложениях необходима разработка клиентской части приложения.

В данной работе акцент смещен в сторону кроссплатформенности и отзывчивости клиентского приложения для конечного пользователя. Такого результата можно достичь, используя современные инструменты разработки, поэтому в данном случае принято решение применять для разработки приложений Flutter, который ориентирован на разработку кроссплатформенных приложений с реализованной концепцией пользовательского интерфейса Material Design.

Flutter – фреймворк созданный Google в 2017 году и поддерживаемый как самой Google, так и сообществом. Технология используется для разработки кроссплатформенных приложений, использующий язык программирования Dart. Использование этого фреймворка позволяет быстро разработать и внедрить клиентскую часть системы за счет упрощения и гибкости разработки, а также большого количества библиотек, предлагающих готовые решения. За счет своих достоинств Flutter делает разработку эффективной, экономичной и быстрой для бизнеса [21].

Для клиентского приложения применяется архитектурный паттерн MVC+S, суть которого, как и в MVC заключается разделении логики приложения на компоненты.

MVC+S содержит следующие основные компоненты:

– модель (Model) – классы данных, которые представляют собой сущности, имеющие собственную структуру данных исходя из бизнес-процессов;

– представление (View) – слой, реализующий графический интерфейс приложения и главным образом отвечает за представление данных пользователю;

– контроллер (Controller) – слой, реализующий бизнес-логику, манипулирующий моделями и передающий их в слой представления, а также отвечающий за управление состоянием приложения;

– службы (Services) – слой, использующийся для организации запросов к внешним API или источникам данных мира по отношению к приложению. Службы вызываются из бизнес-логики, которая содержится в контроллере [21].

На рисунке 20 представлена архитектура клиентского приложения MVC+S, связанные приложения и основные библиотеки, реализующие взаимодействие между компонентами системы.

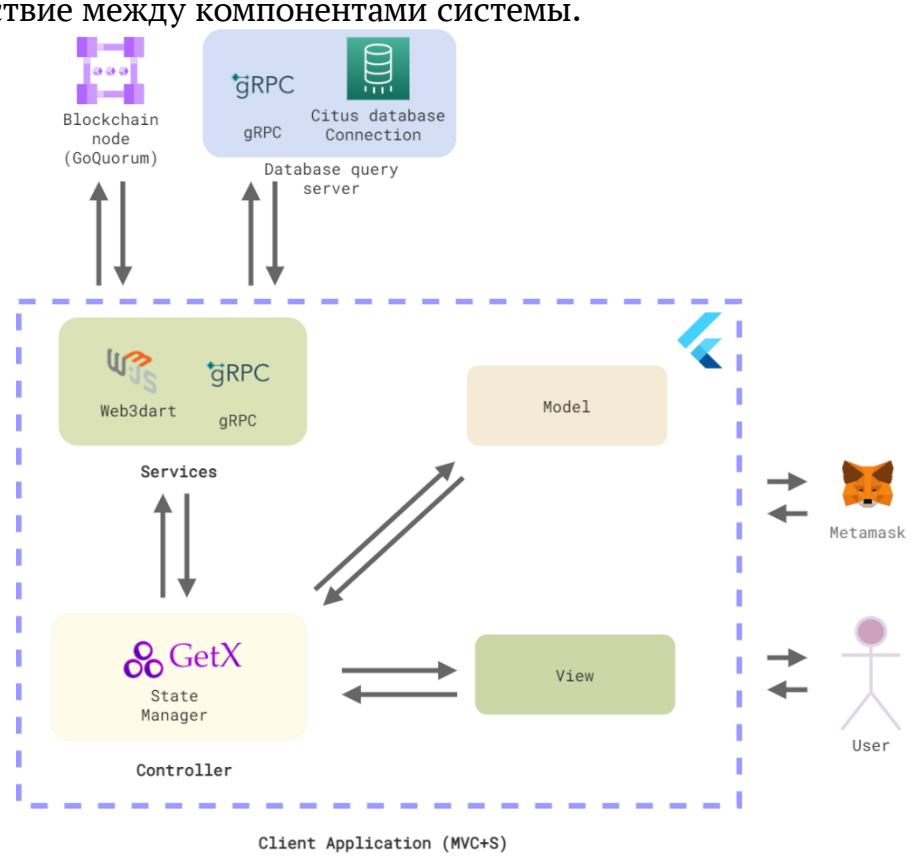

Рисунок 20 – Архитектура MVC+S клиентского приложения

За реализацию представленной архитектуры отвечают следующие основные фреймворки и библиотеки:

– web3dart – библиотека для подключения и взаимодействия с blockchain на базе EVM, необходимая для отправки транзакций, прослушивания событий, взаимодействия со смарт-контрактами [54];

– GetX – фреймворк для управления состояниями в приложении Flutter с дополнительными функциональными возможностями взаимодействия с UI. Данный инструмент позволяет реализовать архитектуру MVC+S [24];

– gRPC – современная высокопроизводительная платформа удаленного вызова процедур, которая может работать на любой платформе и независима от языка программирования. В данной информационной системе используется с целью кроссплатформенности при взаимодействии между компонентами системы [25].

Таким образом, такой архитектурный паттерн позволит достичь следующие цели: структурировать код, облегчить тестирование, организовать независимую работу различных участников процесса разработки и упростить поддержку и масштабирование за счет возможности модификации отдельных компонентов независимо друг от друга.

На рисунке 21 представлена карта основных экранов приложения, которые должны быть разработаны в рамках клиентского графического интерфейса.

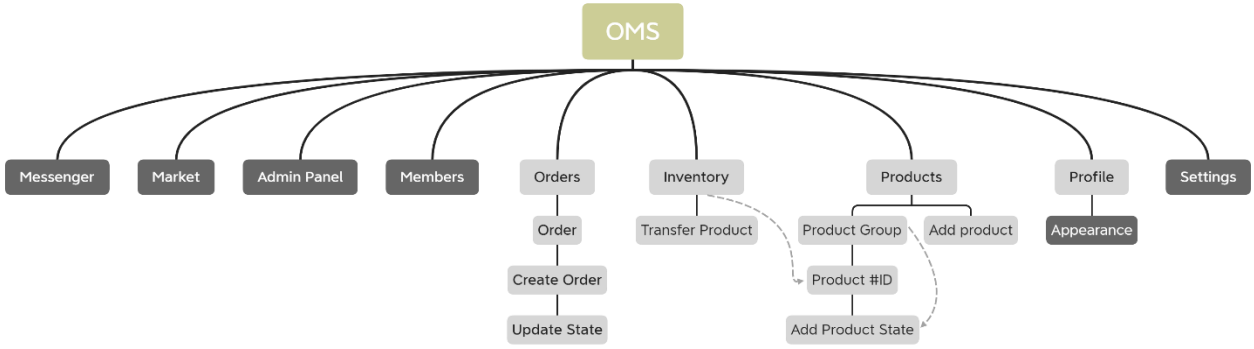

Рисунок 21 – Карта основных экранов приложения

Спроектированный графический интерфейс экранов системы представлен в приложении Б.

ГЛАВА 4 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

В данной работе акцент разработки смещен в область blockchain и смарт-контрактов, то есть реализации логики работы основной части системы управления заказами.

Данная глава посвящена разработке распределенного приложения для управления заказами на базе корпоративного blockchain и смарт-контрактов, а также развертыванию смарт-контракта в blockchain.

Программный код всей системы и дополнительные файлы проекта представлены в открытом репозитории автора диссертационной работы [38].

4.1 Конфигурация blockchain

На рисунке 22 представлено содержимое директории blockchain, которая содержит конфигурацию тестовой сети blockchain. В данной директории особый интерес для разработчика имеют следующие файлы и директории:

– bash-скрипты (run, stop, remove, restart, resume) – необходимы для развертывания, запуска, остановки и перезапуска blockchain в контейнера Docker;

– *docker-compose.yml* – файла для Docker Compose, который описывает подробности запуска и развертывания узлов blockchain в виртуальном контейнере [19];

– *config/goquorum/data* – директория, содержащая генезис-файлы для blockchain, а также файлы с информацией о статичных, разрешенных и неразрешенных узлах;

– *config/nodes* – директория, содержащая сгенерированный набор из первоначальных аккаунтов (адреса, закрытый и открытый ключ), который используется узлами blockchain (валидаторами), а также участниками.

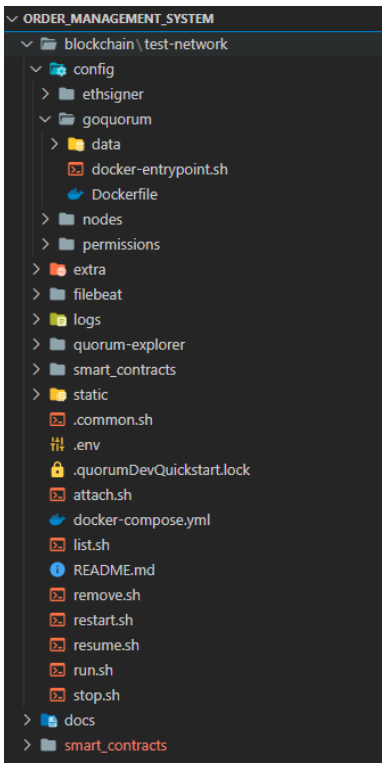

Рисунок 22 – Структура проекта информационной системы управления заказами

Для разработки информационной системы необходимо подготовить blockchain. В данном случае будет использован локальный blockchain с 4 узлами и консенсусом QBFT. Такой blockchain в свой работе ничем не отличается от корпоративного, который распределен на разных серверах, однако для разработки и тестирования распределенного приложения этого будет достаточно и позволит ускорить разработку.

Для генерации тестовой сети blockchain использована утилита «Quorum Dev Quickstart» [41]. Утилита вызывается в определенной разработчиком директории через терминал командой *npx quorum-dev-quickstart*. После серии вопросов и ответов тестовая сеть будет сгенерирована.

Предполагается использование консенсуса QBFT, поэтому необходимо сконфигурировать файл для генезис-блока.

Генезисный блок – это начало blockchain, а *genesis.json* – это файл, который его определяет. Это похоже на «настройки» для blockchain. Например, конфигурация цепочки, уровень сложности добычи блоков и т. д.

Генезис файл содержит следующие основные параметры:

– *chainId* – Идентификатор для сети;

– *qbft.Epochlength* – количество блоков, после которого обнуляются все голоса;

– *qbft.blockperiodseconds* – минимальное время блокировки в секундах, т.е. интервал между созданием нового блока;

– *qbft.emptyBlockPeriodSeconds* – задает период между созданием пустых блоков и тем самым уменьшает количество пустых блоков, когда нет никаких транзакций;

– *qbft.requesttimeoutseconds* – тайм-аут для каждого раунда консенсуса перед изменением раунда в секундах;

– *qbft.policy* – политика выбора предлагающего, которая равна 0 (круговой алгоритм) или 1 (фиксация);

– *qbft.ceil2Nby3Block* – у станавливает номер блока, из которого следует использовать обновленную формулу для расчета количества неисправных узлов;

– *txnSizeLimit* – максимальный размер транзакции;

– *isQuorum* – разрешение клиенту geth работу от имени GoQuorum и выполнение дополнительных проверок, например для проверки того, что плата за газ равна нулю;

– *timestamp* – дата и время создания блока;

– *gasLimit* – блокировка лимита газа. Общий лимит газа для всех транзакций в блоке.

– *gasUsed* – использованное количество газа;

– *alloc* – определяет счета с балансами или контрактами.

Конфигурация генезис блока представлена на рисунке 23.

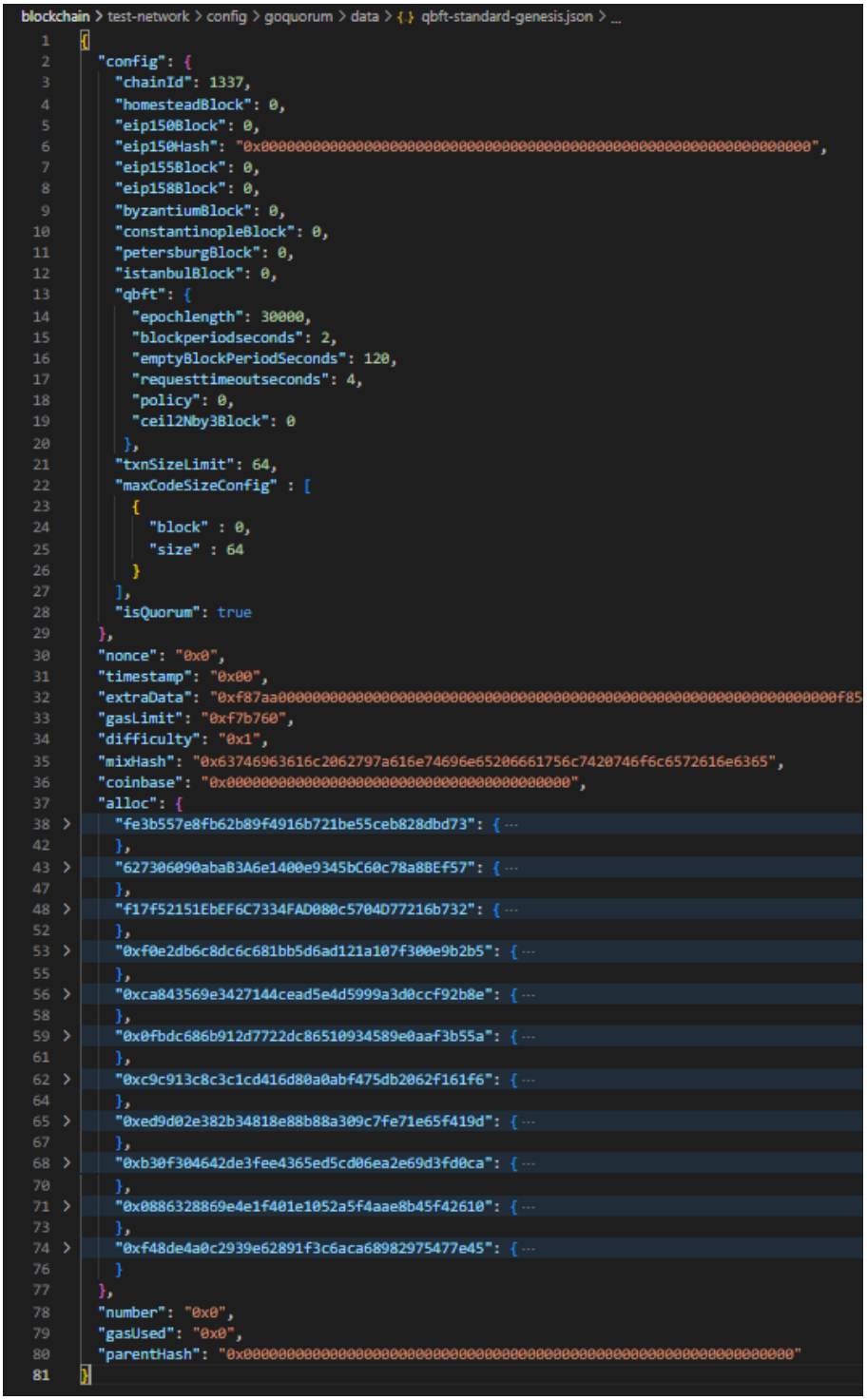

Рисунок 23 – Содержимое генезис файла

Стоит отметить, что генезис файл у всех узлов blockchain должен быть одинаковым.

После конфигурации генезис-файла при помощи скрипта *run.sh* осуществляется запуск локального blockchain с 4 узлами в контейнере Docker (рисунок 24).

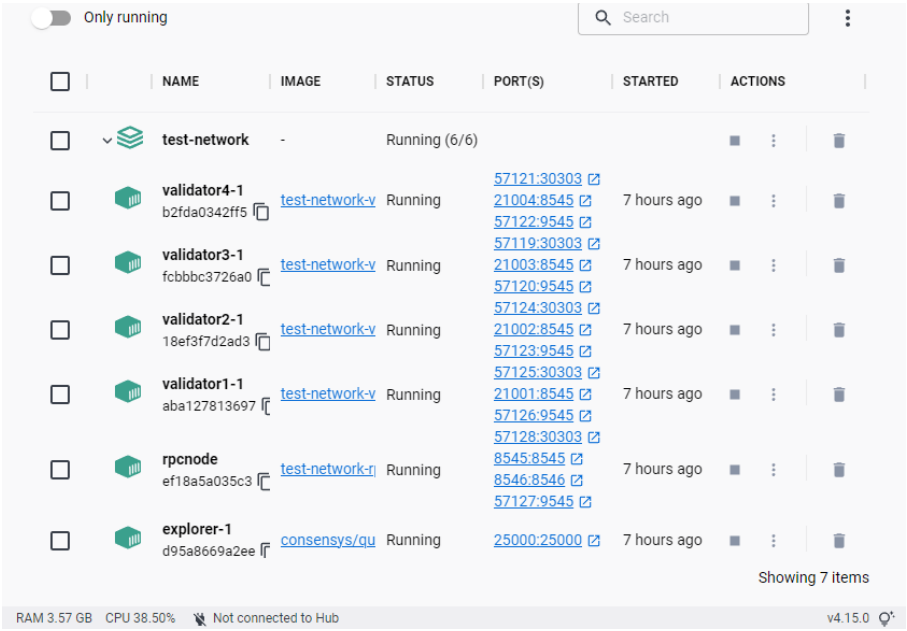

Рисунок 24 – Запущенная тестовая сеть в Docker-контейнере

В обозревателе блоков, доступному по ссылке *http://localhost:25000/explorer* можно просматривать состояние blockchain, генерацию блоков, транзакции.

На рисунке 25 представлен блок генезиса, созданный при создании первого узла blockchain.

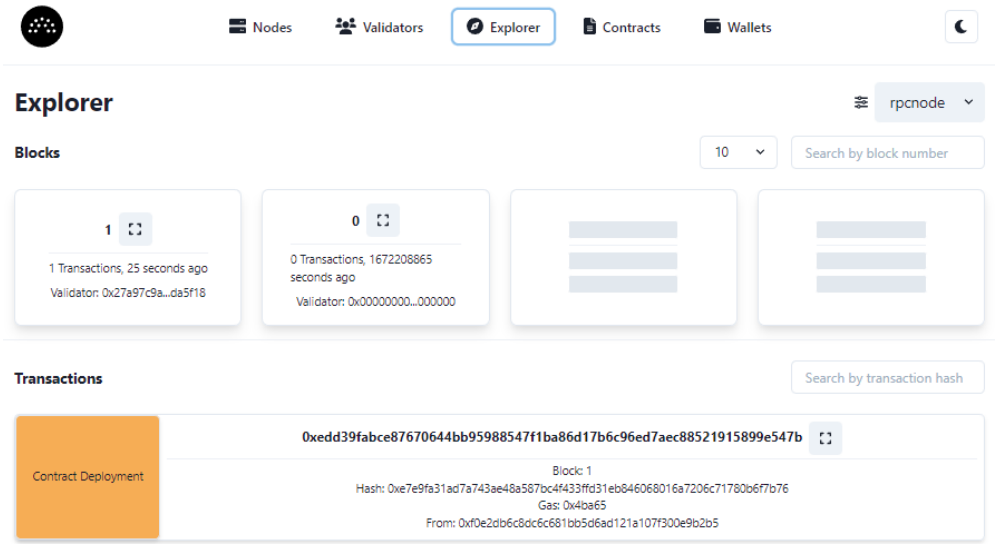

Рисунок 25 – Отображение развернутого локального blockchain в обозревателе блоков

В случае использования Metamask как инструмента для хранения закрытого ключа необходимо осуществить подключение к blockchain при помощи добавления новой сети со следующими настройками:

– Network name – название сети, понятное для человека;

– New RPC URL – адрес узла blockchain;

– Chain ID – идентификатор цепочки;

– Currency symbol – символ криптовалюты, в данном случае не используемой.

На рисунке 26 представлена конфигурация сети для подключения.

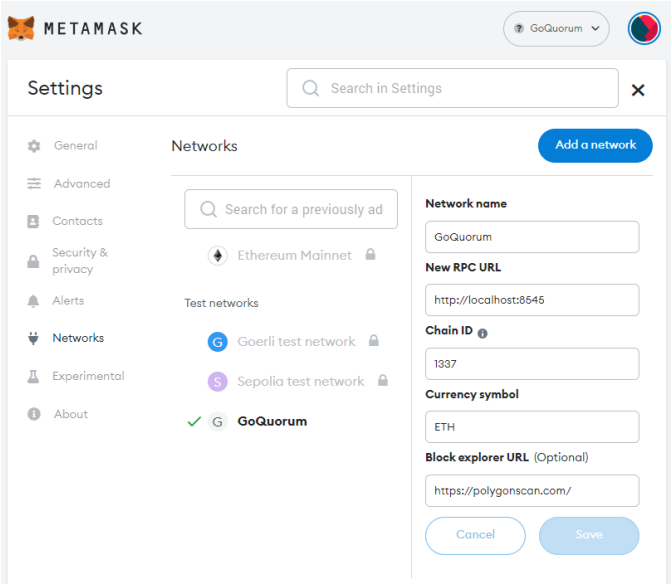

Рисунок 26 – Подключение к локальному blockchain при помощи провайдера Metamask

Таким образом, тестовая сеть blockchain была развернута и подготовлена для дальнейшей разработки.

4.2 Смарт-контракты

На рисунке 27 представлен сгенерированная структура для проекта фреймворка Hardhat, необходимая для разработки и тестирования смарт-контрактов Solidity.

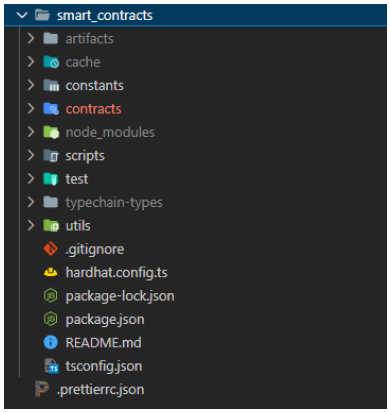

Рисунок 27 – Структура проекта, сгенерированная фреймворком Hardhat

Структура проекта содержит следующие основные файлы и директории, представляющие интерес для разработчика:

– *constants* – добавленная директория, содержащая константы (ссылки на приватные ключи);

– *contracts* – директория, содержащая смарт-контракты на языке программирования solidity;

– *scripts* – директория, содержащая программный код для развертывания смарт-контрактов в blockchain;

– *test* – директория, содержащая программный код для тестов смарт-контрактов. Тесты могут быть написаны на JavaScript или TypeScript;

– *utils* – директория, содержащая вспомогательные функции и иной программный код, который может быть вызван из других мест в структуре проекта;

– *hardhat.config.ts* – файл, содержащий настройки для фреймворка Hardhat.

На рисунке 28 представлено в сокращении содержимое модуля OrderManagementSystemUpgradable, который представляет из себя обновляемый смарт-контракт, доступ к которому получает пользователь в прикладном приложении. Особенность данного смарт-контракта в том, что, используя Proxy механизм, он предоставляет доступ пользователя к смарт-контракту OrderManagementSystem. Данный контракт нужен для возможности обновления смарт-контракта в будущем на случай, если необходимо будет обновление функций основного смарт-контракта.

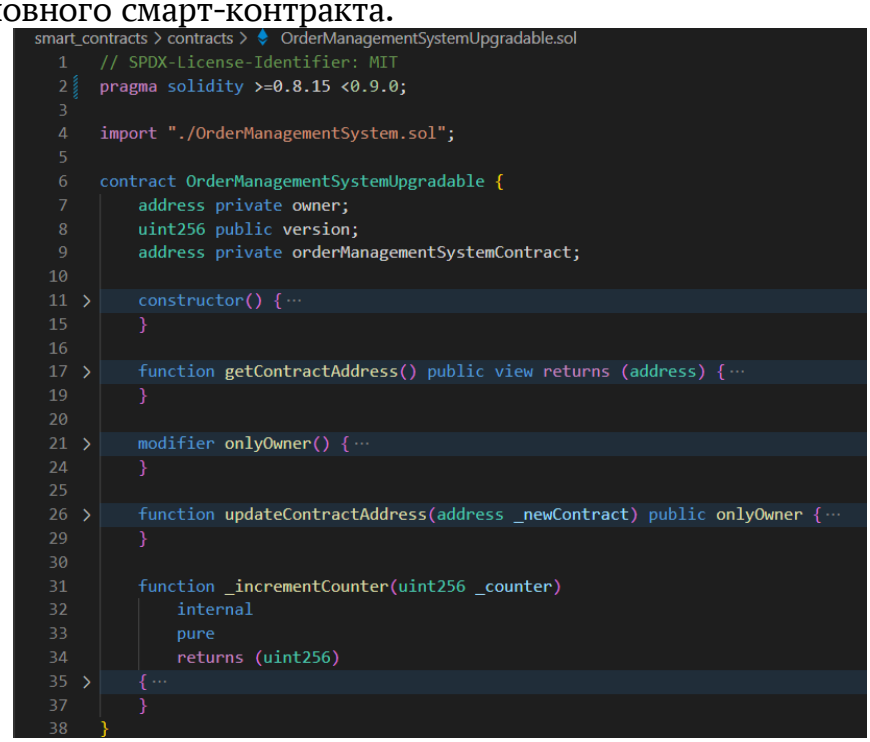

Рисунок 28 – Часть программного кода для модуля «OrderManagementSystemUpgradable»

На рисунке 29 представлено содержимое модуля OrderManagementSystem, который является основным смарт-контрактом, который при помощи механизма наследования Solidity наследует весь функционал других смарт-контрактов, являющихся, по своей сути модулями.

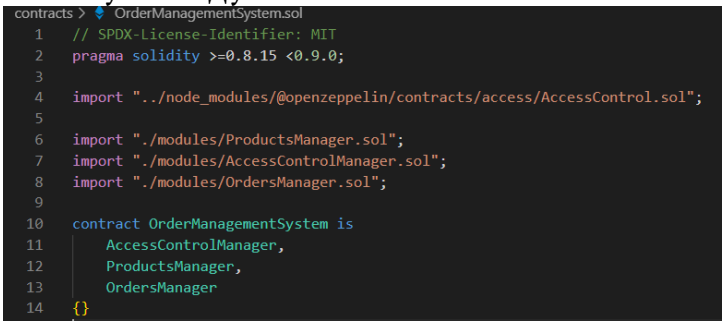

Рисунок 29 – Часть программного кода для модуля «OrderManagementSystem»

На рисунках 30 и 31 представлено в сокращении содержимое модуля AccessControlManager. Данный модуль обеспечивает:

– осуществляет регистрацию пользователей в системе, а также позволяет обновлять информацию;

– предоставляет администратору функции для наделения пользователей ролями;

– контролирует доступ к методам смарт-контрактов только для пользователей, участвующих в процессах доступных для конкретных ролей;

– позволяет пользователям отзывать доступ к приложению на случай, если их аккаунт (закрытый ключ) был скомпрометирован.

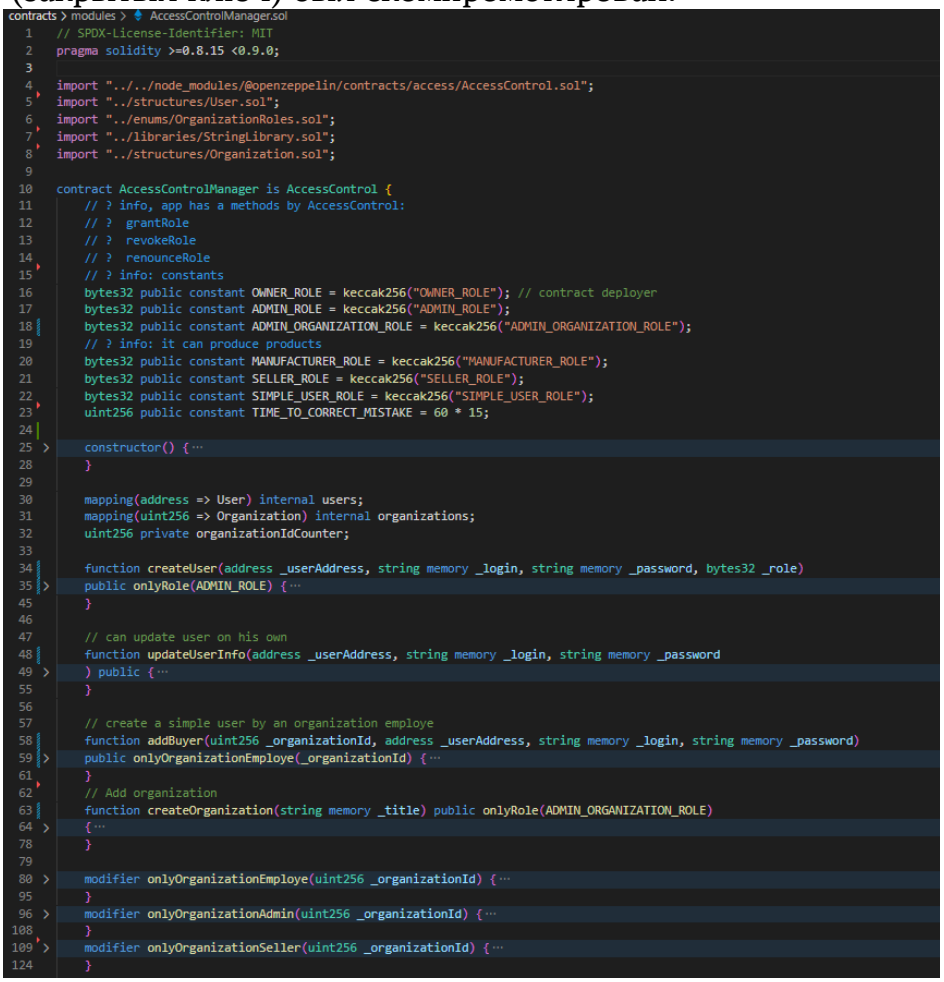

Рисунок 30 – Часть программного кода для модуля «AccessControlManager» (Начало)

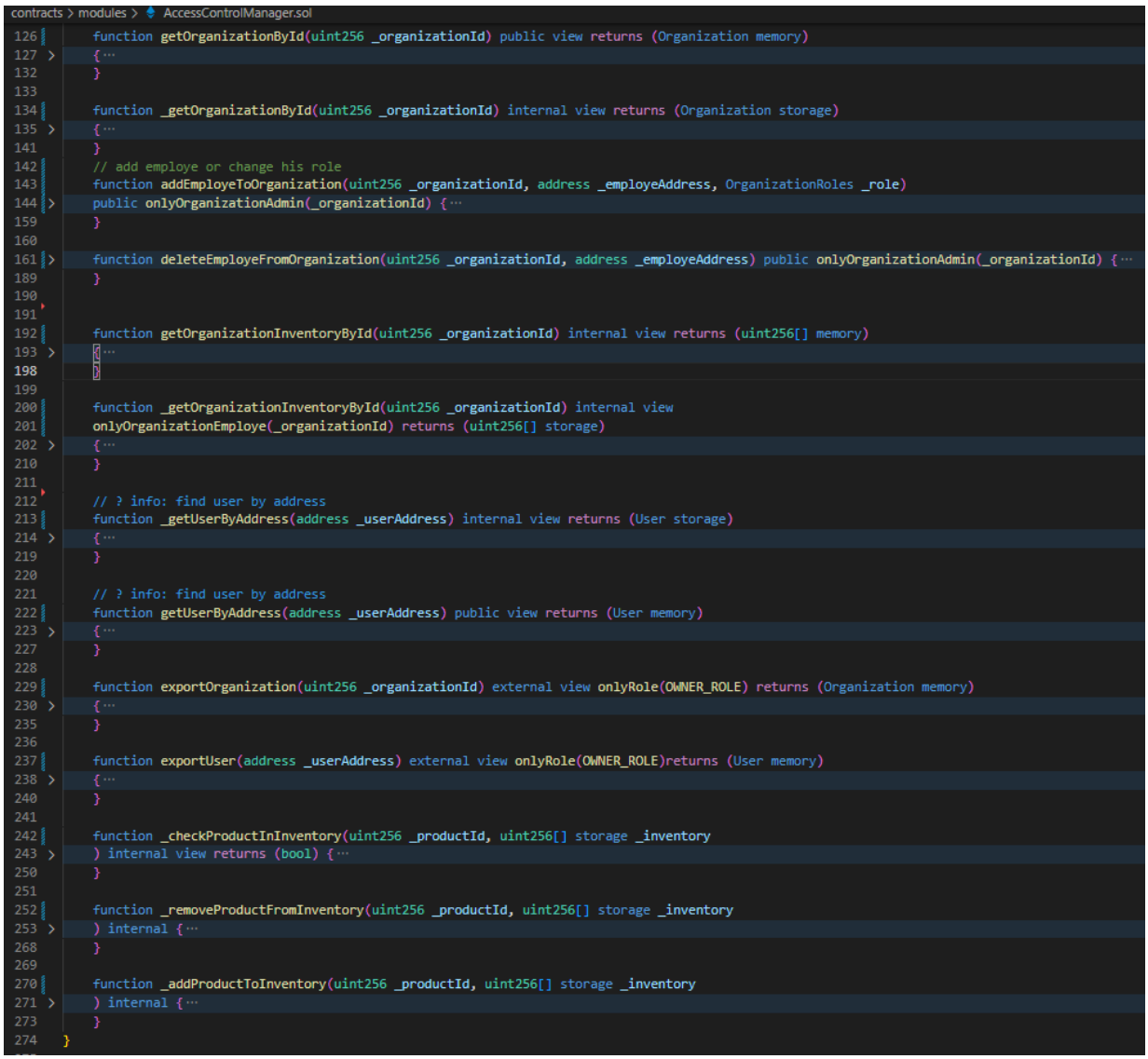

Рисунок 31 – Часть программного кода для модуля «AccessControlManager» (Окончание)

На рисунке 32 представлено в сокращении содержимое модуля OrdersManager, который имеет следующие основные функции:

– создание заказов и добавление в историю статуса заказа, а также обновление статуса продукции;

– управление блокировкой продуктов в составе заказа;

– управление перемещением продуктов между счетами заказчика и продавца.

Рисунок 32 – Часть программного кода для модуля «OrdersManager»

На рисунке 33 и 34 представлено в сокращении содержимое модуля ProductsManager, который является менеджером управления продукцией, обеспечивающий процессы внесения продукции в систему, а также организующий управление передачей от одного пользователя к другому, за счет чего обеспечиваются учетный функции.

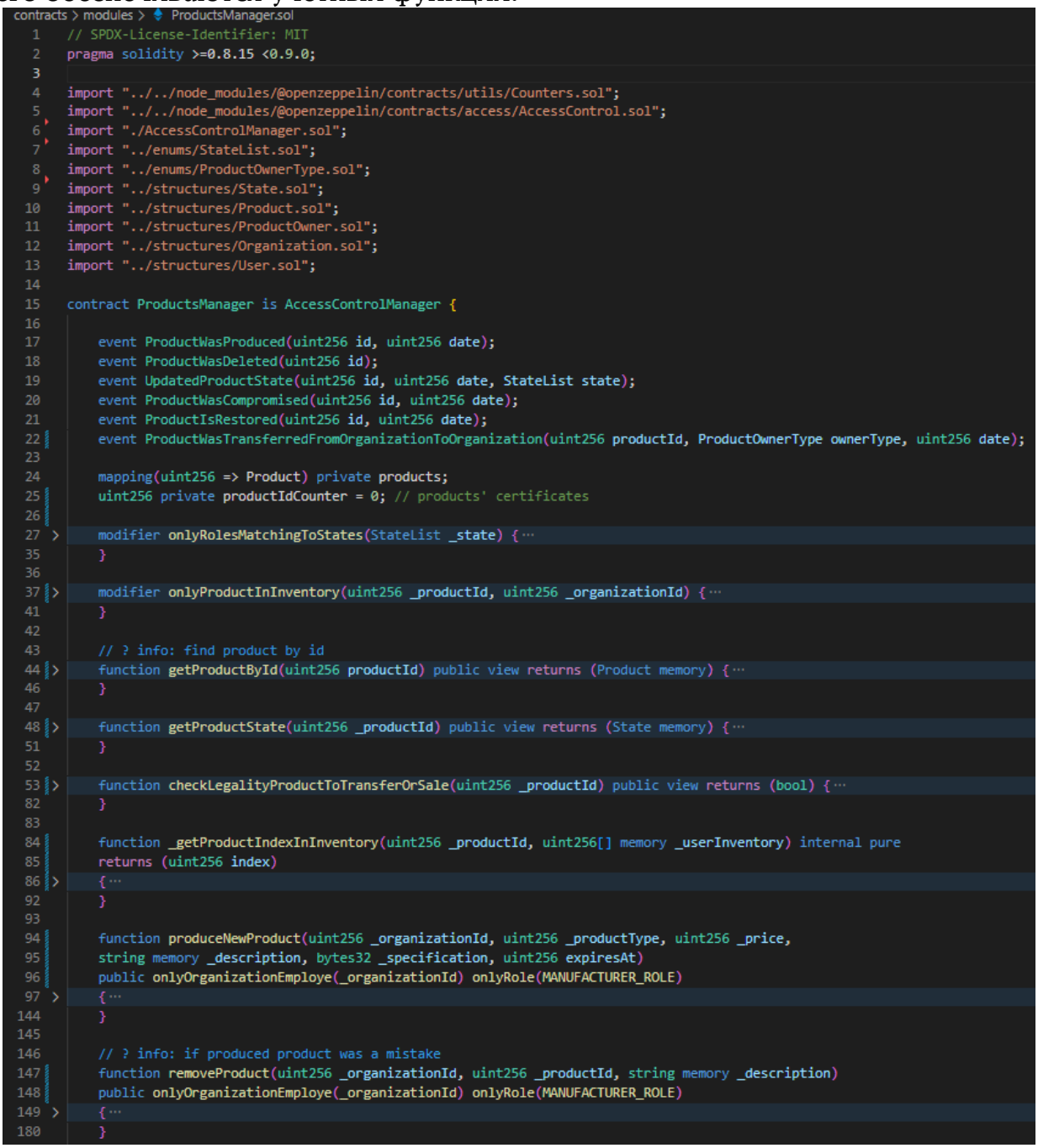

Рисунок 33 – Часть программного кода для модуля «ProductsManager» (Начало)

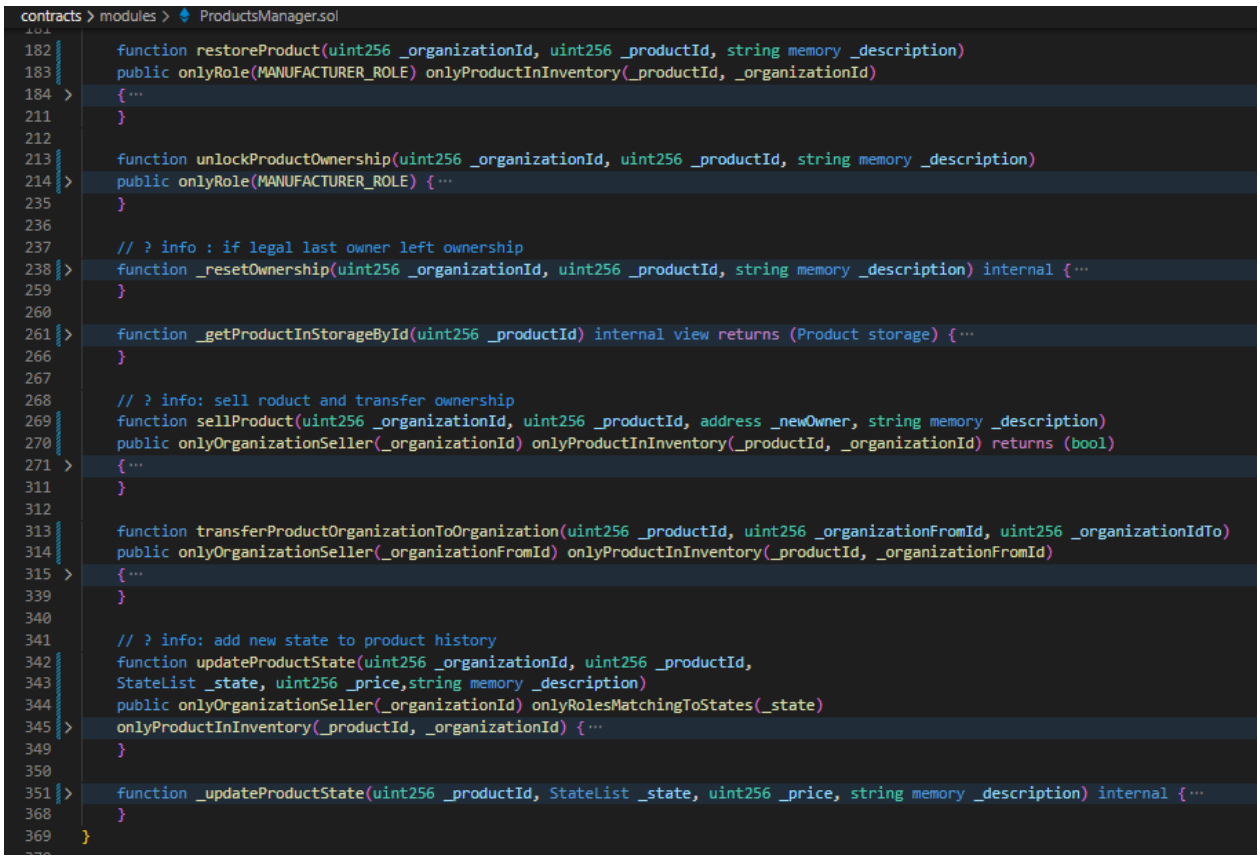

Рисунок 34 – Часть программного кода для модуля «ProductsManager» (Окончание)

Стоит отметить, что на рисунках выше представлен не весь программный код смарт-контрактов. Весь код представлен в открытом репозитории Github [38].

Таким образом смарт-контракты являются основным компонентом децентрализованных приложений, и вся логика взаимодействия пользователей с системой реализуется при помощи обращении через API библиотеки web3 к смарт-контракту, развернутому в blockchain.

4.3 Развертывание смарт-контрактов

На рисунке 35 представлен программный код файла «*hardhat.config.ts*» для настройки фреймворка Hardhat.

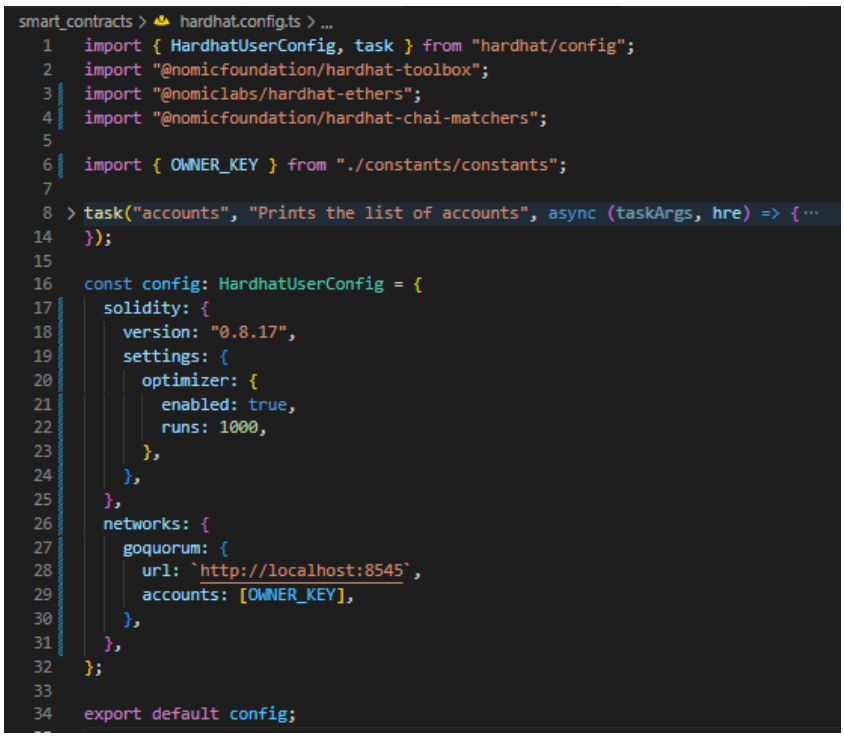

Рисунок 35 – Содержимое файла опций «*hardhat.config.ts*» для фреймворка Hardhat

В коде, представленном на рисунке 35 наибольшее значение представляет раздел config > networks. В данном участке кода описывается сети blockchain, с их адресом и указанием аккаунта пользователя. При помощи данной опции осуществляется развертывание смарт-контрактов в необходимой сети blockchain, а также исполняются скрипты, которые автоматизирует какие-либо действия, связанные с смарт-контрактом.

На рисунке 36 представлено содержимое файла «*deploy.ts*», который необходим для осуществления развертывания смарт-контракта в сети blockchain.

```
smart_contracts > scripts > deploy.ts > ...
import { ethers } from "hardhat";

async function main() {
  console.log(`----------------- DEPLOY SCRIPT START -----------------`);

  const [deployer] = await ethers.getSigners();
  console.log(`Deploying contracts with the account: ${deployer.address}.`);

  // deploy the library
  const StringLibrary = await ethers.getContractFactory("StringLibrary");
  const stringLibrary = await StringLibrary.deploy();
  await stringLibrary.deployed();

  const OrderManagementSystemUpgradable = await ethers.getContractFactory(
    "OrderManagementSystemUpgradable",
    {
      signer: deployer,
      libraries: {
        StringLibrary: stringLibrary.address,
      },
    }
  );
  // deploy contract
  const orderManagementSystemUpgradable =
    await OrderManagementSystemUpgradable.deploy();

  console.log(
    `OrderManagementSystemUpgradable contract was deployed to ${orderManagementSystemUpgradable.address} address.`
  );
  console.log(`----------------- DEPLOY SCRIPT END -----------------`);
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Рисунок 36 – Скрипт «deploy.ts» для развертывания смарт-контрактов в локальном тестовом blockchain

Для исполнения скрипта «*deploy.ts*» необходимо вызвать команду: *npx hardhat run scripts/deploy.ts --network goquorum*.

На рисунке 37 представлен результат выполнения скрипта «*deploy.ts*».

```
----------------- DEPLOY SCRIPT START -----------------
Deploying contracts with the account: 0xf0E2Db6C8dC6c681bB5D6aD121A107f300e9B2b5.
OrderManagementSystemUpgradable contract was deployed to 0x695Baaf717370fcBb42aB45CD83C531C27D79eF1 address.
----------------- DEPLOY SCRIPT END -----------------
```

Рисунок 37 – Результат выполнения скрипта «*deploy.ts*»

На рисунке 38 представлена в обозревателе блоков выполненная транзакция с развертыванием контракта.

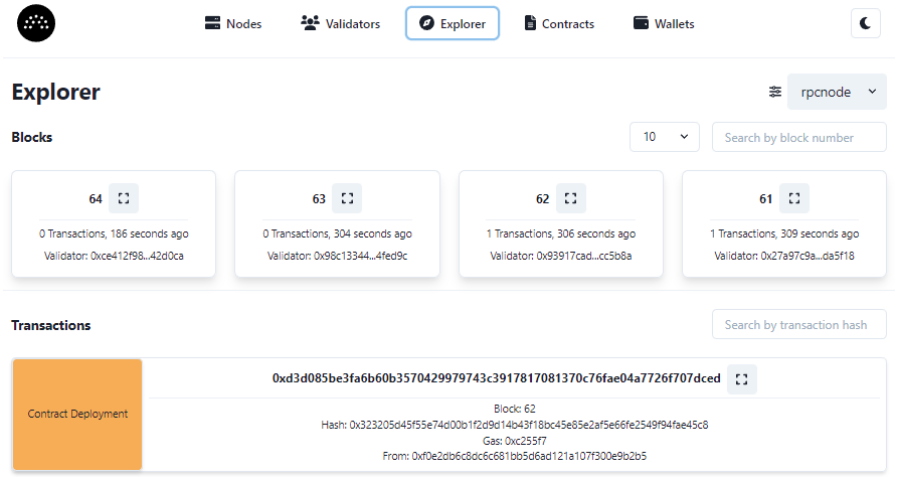

Рисунок 38 – Отображение в обозревателе блоков транзакции с развертыванием смарт-контракта

Таким образом, разработанный смарт-контракт был развернут в сети blockchain при помощи скрипта. За счет такого подхода к разработке и развертыванию смарт-контрактов возможно осуществлять быструю разработку распределенных приложений, а также при помощи скриптов автоматизировать какие-либо действия с смарт-контрактом после их развертывания, например осуществлении первоначальной настройки, наделение доступом к смарт-контракту различным участникам сети, осуществление миграции между смарт-контрактами или даже blockchain-сетями, а также тестирование функционала смарт-контрактов на предмет наличия уязвимости и ошибок.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было исследовано применения blockchain технологии в области управления заказами, спроектирована архитектура распределенной системы управления заказами, а также разработано децентрализованное приложение на базе смарт-контрактов, которое является главным звеном всей системы и ценность которого заключается в обеспечении прозрачности, связанных с процессами заказа, потоков информации, а также учета всех активов в системе и историю действий, которые были с ними совершены.

В процессе работы были решены все поставленные задачи:

1) проанализированы процессы информационного обеспечения процессов управления заказами в цепочке поставки, в результате чего были обобщены такие несовершенства этих процессов как: проблемы масштабирования, сложность контроля и учета при территориальном разбросе; проблемы прозрачности и доверия, возможности мошенничества, ненамеренные нарушения в процессах транспортировки, контрафактная продукция. Определена модель REA, рассматривающая процессы реального мира с точки зрения информационного обмена и событий, порождающих ценную информацию для участников процессов и сформированы требования к информационной системе;

2) проанализированы технологические решения, разрешающие проблемы в рассмотренных ранее процессах управления цепочками поставок и заказов. Были сделаны выводы, что современные технологии в своем совместном использовании позволят получить значительное преимущество от использования;

3) рассмотрена технология blockchain и возможности, которые он может предоставить для совершенствования процессов управления заказами. Применение blockchain в прикладных приложения реализуется за счет смарт-контрактов, которые являются неотъемлемой частью blockchain и представляют собой программный код, написанный на доступном для конкретного blockchain языке написания смарт-контрактов. Кроме того, смарт-контракты имеют ограничения в хранении информации и разработке приложений;

4) спроектирована архитектура и функциональные составляющие проблеморазрешающей системы, которая представляет собой гибридное децентрализованное приложение;

5) разработаны смарт-контракты, представляющие собой основу распределенного приложения, базовая конфигурация тестового варианта blockchain, и скрипты развертывания смарт-контрактов.

Поскольку все задачи были успешно решены, можно сделать вывод, что цель данной работы достигнута.

Стоит отметить, что по теме выпускной квалификационной работы была написана статья «Проектирование распределенной системы управления

заказами и методика ее тестирования» и опубликована в международной научно-практической конференций «Концепция «общества знаний» в современной науке» [4].

В своей концепции распределенное приложение состоит из следующих компонентов: графический интерфейса, с которым взаимодействует пользователь; blockchain GoQuorum, использующий язык программирования Solidity для смарт-контрактов; сервер запросов к распределенной базе данных; распределенная база данных Citus, которая кеширует все данные, связанные с процессами управления заказами; сервер IoT-устройств, принимающий и обрабатывающий сигналы и фиксирующий их в blockchain; IPFS для организации децентрализованной сети обмена файлами. Такая архитектура системы является оптимальной, поскольку каждый разрешенный участник может развернуть собственный узел из всех компонентов системы, а вся важная информация из распределённой базы данных может быть проверена при обращении к blockchain.

Разработанные смарт-контракты занимают главное место во всей распределенной системе управления заказами, поскольку от их реализации зависит устройство клиентского приложения.

В результате проектирования и разработки они имеют следующие функциональные особенности:

– обеспечивают управление доступа к системе управления заказами и разграничивают доступ к функциям распределенного приложения;

– обеспечивают полный учет всех имеющихся активов организаций, подключенных к системе;

– обеспечивает процессы передачи активов между организациями;

– обеспечивают процессы создания заказов, сохраняют историю передвижений;

– обеспечивают привилегированных пользователей в системе возможностями производства продукции, ее восстановление и выход из эксплуатации;

– смарт-контракт спроектирован таким образом, чтобы обеспечить его масштабирование при развертывании в производственной среде.

Важным стоит отметить, что все действия в процессах, описанных программным кодом в смарт-контрактах, защищены от посторонних пользователей, что было подтверждено тестированием смарт-контракта на уязвимость, поэтому в своей текущей реализации он является безопасным.

Таким образом, цель данной диссертационной работы в совершенствовании процессов управления заказами при помощи децентрализованной информационной системы была достигнута. Спроектированное и разработанное распределенное приложение может быть улучшено путем добавления функций продажи активов конечным потребителями с наделением прав на приобретенный продукт, а также внесением функций для оказания сервис-поддержки приобретенных продуктов.

В заключении также можно сделать вывод о том, что несмотря на особенность и смысл существования технологии blockchain в децентрализованности, на текущий момент существует множество ограничений при разработке прикладных приложений для разработчика, например такие, как: отсутствие возможности простых запросов к данным, хранимых в blockchain; отсутствие конфигурации виртуальной машины, в которой исполняется код смарт-контрактов; ограниченность некоторых реализаций языков программирования смарт-контрактов и другие. Все эти ограничения так или иначе вносят зависимость распределенных приложений от других компонентов, например от сервера базы данных, выполняющего только роль кеширования данных для быстрого и простого доступа к ним. Но, несмотря на это, технология все еще развивается и существуют различные открытые проекты, каждый из которых вносит свой вклад в общую тенденцию развития технологии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Blockchain и интернет вещей в логистике: что это такое?. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://logists.by/blockchain/blokcheyn-i-internet-veschey-v-logistike-chto-eto-takoe (дата обращения: 14.01.2022)

2. Крылатков, П.П. Управление цепью поставок (SCM): учебное пособие / П.П. Крылатков, М.А. Прилуцкая. – Екатеринбург : Изд-во Урал. ун-та, 2018. – 140 с. – Текст : электронный: [сайт]. – URL: https://elar.urfu.ru/bitstream/10995/59184/1/978-5-7996-2269-5_2018.pdf (дата обращения: 14.01.2022)

3. Методология функционального моделирования IDEF0 – Текст : электронный : сайт. – URL: https://advanced-quality-tools.ru/assets/idef0-rus.pdf (дата обращения: 14.01.2022)

4. Шубин, Д. В. Проектирование распределенной системы управления заказами и методика ее тестирования / Д. В. Шубин, А. А. Момот. – Текст : электронный // Концепция «общества знаний» в современной науке. – 2022. – URL: https://www.elibrary.ru/item.asp?id=50007875&pff=1 (дата обращения: 14.01.2022)

5. Artificial Intelligence (AI). – Текст : электронный // Официальный сайт и блог компании «IBM» : сайт. – URL: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (дата обращения: 14.01.2022)

6. Amazon Managed Blockchain. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://aws.amazon.com/ru/managed-blockchain/ (дата обращения: 14.01.2022)

7. An Overview of Blockchain in Supply Chain: What's the Link?. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://www.foley.com/en/insights/publications/2021/08/overview-blockchain-in-supply-chain-whats-link (дата обращения: 14.01.2022)

8. Andrianova, N. V. Intelligent contracts in engineering enterprise supply management / N. V. Andrianova, P. A. Nechaeva. – DOI 10.21603/2500-3372-2021-6-4-496-505 // Bulletin of Kemerovo State University. – Текст : электронный. – URL: https://vestnik-pses.kemsu.ru/en/nauka/article/48235/view#gost (дата обращения: 14.01.2022)

9. Architecture GoQuorum. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://consensys.net/docs/goquorum//en/latest/concepts/architecture/ (дата обращения: 14.01.2022)

10. A Technical Guide to IPFS – the Decentralized Storage of Web3. – Текст : электронный // Сайт сообщества программистов с руководствами : сайт. – URL: https://www.freecodecamp.org/news/technical-guide-to-ipfs-decentralized-storage-of-web3/ (дата обращения: 14.01.2022)

11. Blockchain for Supply Chain: The Essence. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://www.scnsoft.com/blockchain/supply-chain#service-options (дата обращения: 14.01.2022)

12. Business Process Model and Notation (BPMN). Version 2.0 – Текст : электронный // Сайт разработчика стандарта : сайт. – URL: https://www.omg.org/spec/BPMN/2.0/PDF (дата обращения: 14.01.2022)

13. Consensus Algorithms in Blockchain. – Текст : электронный // A Computer Science portal for geeks : сайт. – URL: https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/ (дата обращения: 14.01.2022)

14. Caldarelli, G. Blockchain 1.0 to Blockchain 4.0 – The Evolutionary Transformation of Blockchain Technology / G. Caldarelli, J. Ellul. – DOI 10.1007/978-3-030-69395-4_3 // In book: Blockchain Technology: Applications and Challenges. – Текст : электронный. – URL: https://www.researchgate.net/publication/351263701_Blockchain_10_to_Blockchain_40-The_Evolutionary_Transformation_of_Blockchain_Technology (дата обращения: 14.01.2022)

15. Caldarelli, G. The Blockchain Oracle Problem in Decentralized Finance – A Multivocal Approach / G. Caldarelli, J. Ellul. – DOI 10.3390/app11167572 // New Trends in Blockchain Technology. – Текст : электронный. – URL: https://www.mdpi.com/2076-3417/11/16/7572 (дата обращения: 14.01.2022)

16. Configure QBFT consensus. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://consensys.net/docs/goquorum/en/latest/configure-and-manage/configure/consensus-protocols/qbft/ (дата обращения: 14.01.2022)

17. Configure enhanced permissions. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://consensys.net/docs/goquorum/en/latest/configure-and-manage/configure/permissioning/enhanced-permissions/ (дата обращения: 14.01.2022)

18. Citus : официальный сайт : сайт – . – URL: https://www.citusdata.com/ (дата обращения: 14.01.2022)

19. Docker Compose Overview : официальный сайт с документацией : сайт – . – URL – Режим доступа: https://docs.docker.com/compose/ (дата обращения: 14.01.2022)

20. Everledger: официальный сайт : сайт – . – URL: https://everledger.io/technology/ (дата обращения: 14.01.2022)

21. Flutter: State Management using an MVC+S Architecture : сайт – . – URL: https://blog.gskinner.com/archives/2020/09/flutter-state-management-with-mvcs.html (дата обращения: 14.01.2022)

22. Free gas networks. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL:

https://consensys.net/docs/goquorum/en/latest/concepts/free-gas-network/ (дата обращения: 14.01.2022)

23. Flutter : официальный сайт : сайт – . – URL: https://flutter.dev/ (дата обращения: 14.01.2022)

24. GetX : репозиторий проекта : сайт – . – URL: https://github.com/jonataslaw/getx (дата обращения: 14.01.2022)

25. gRPC : официальный сайт : сайт – . – URL https://grpc.io/ (дата обращения: 14.01.2022)

26. How transaction flow on the ethereum blockchain. – Текст : электронный // Платформа со статьями для специалистов из различных областей : сайт. – URL: https://learn.block6.tech/how-transaction-flow-on-the-ethereum-blockchain-8130f3f42a6a (дата обращения: 14.01.2022)

27. How Walmart Canada Uses Blockchain to Solve Supply-Chain Challenges. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://hbr.org/2022/01/how-walmart-canada-uses-blockchain-to-solve-supply-chain-challenges (дата обращения: 14.01.2022)

28. Hall, A. J. Accounting Information Systems: 8th edition / A. J. Hall – Mason: South-Western Cengage Learning, 2012. – 840 p. – direct text.

29. Hardhat : официальный сайт с документацией: сайт – . – URL: https://hardhat.org/ (дата обращения: 14.01.2022)

30. IBM Supply Chain Intelligence Suite: Food Trust. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://www.ibm.com/blockchain/solutions/food-trust (дата обращения: 14.01.2022)

31. Introduction to smart contracts. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://ethereum.org/en/developers/docs/smart-contracts/ (дата обращения: 14.01.2022)

32. IPFS : официальный сайт с документацией : сайт – . – URL: https://docs.ipfs.tech/ (дата обращения: 14.01.2022)

33. Modern supply chain overview: Definition, models, challenges. – Текст : электронный // Блог об электронной коммерции : сайт. – URL: https://www.the-future-of-commerce.com/2022/03/09/supply-chain-definition-models-challenges/ (дата обращения: 14.01.2022)

34. Manoshi Das Turjo, Smart Supply Chain Management Using the Blockchain and Smart Contract / Manoshi Das Turjo, Mohammad Monirujjaman Khan, Manjit Kaur, Atef Zaguia. – DOI 10.1155/2021/6092792 // Scientific Programming. – Текст : электронный. – URL: https://www.hindawi.com/journals/sp/2021/6092792/ (дата обращения: 14.01.2022)

35. Moosavi, J. Blockchain in supply chain management: a review, bibliometric,and network analysis / J. Moosavi, L. M. Naeni, A.M. Fathollahi-Fard, U. Fiore. – DOI 10.1007/s11356-021-13094-3 // Environmental Science and Pollution Research

36. Metamask : официальный сайт с документацией : сайт – . – URL: https://metamask.io/ (дата обращения: 14.01.2022)

37. OMG Unified Modeling Language. Version 2.5.1. – Текст : электронный // Сайт разработчика стандарта : сайт. – URL: https://www.omg.org/spec/UML/2.5.1/PDF (дата обращения: 14.01.2022)

38. Order management system : репозиторий проекта : сайт – . – URL: https://github.com/YoKawaiiK/order_management_system (дата обращения: 14.01.2022)

39. OpenZeppelin : официальный сайт с документацией : сайт – . – URL: https://docs.openzeppelin.com/ (дата обращения: 14.01.2022)

40. PlantUML : официальный сайт : сайт – . – URL: https://plantomg.com/ru/ (дата обращения: 14.01.2022)

41. Quorum Dev Quickstart : репозиторий проекта : сайт – . – URL: https://github.com/ConsenSys/quorum-dev-quickstart (дата обращения: 14.01.2022)

42. Robotic Process Automation (RPA). – Текст : электронный // Официальный сайт и блог компании «UiPath» : сайт. – URL: https://www.uipath.com/rpa/robotic-process-automation (дата обращения: 14.01.2022)

43. Study of Blockchain Application in the Logistics Industry. – Текст : электронный // One of the largest Open Access journal publishers : сайт. – URL: https://www.scirp.org/journal/paperinformation.aspx?paperid=115674 (дата обращения: 14.01.2022)

44. Smits M., Smart Supply Chain Management Using the Blockchain and Smart Contract / M. Smits, J. Hulstijn. – DOI 10.3389/fbloc.2020.00005 // Tilburg School of Economics and Management. – Текст : электронный. – URL: https://www.frontiersin.org/articles/10.3389/fbloc.2020.00005/full (дата обращения: 14.01.2022)

45. Solidity : официальный сайт с документацией: сайт – . – URL: https://docs.soliditylang.org/en/v0.8.17 (дата обращения: 14.01.2022)

46. Should You Use Node.js for Your IoT Development Project?. – Текст : электронный // Сайт компании разработчика ПО : сайт. – URL: https://relevant.software/blog/node-js-for-iot-development/ (дата обращения: 14.01.2022)

47. The Supply Chain: From Raw Materials to Order Fulfillment. – Текст : электронный // Веб-сайт финансовых СМИ : сайт. – URL: https://www.investopedia.com/terms/s/supplychain.asp (дата обращения: 14.01.2022)

48. The Four Technologies Shaping Next-Gen Supply Chains. – Текст : электронный // Информационный ресурс по управлению цепями поставок : сайт. – URL: https://www.supplychainbrain.com/blogs/1-think-tank/post/33709-the-four-technologies-shaping-next-gen-supply-chains (дата обращения: 14.01.2022)

49. Transactions. – Текст : электронный // Официальный сайт с документацией blockchain Ethereum : сайт. – URL: https://ethereum.org/en/developers/docs/transactions/ (дата обращения: 14.01.2022)

50. The growing list of applications and use cases of blockchain technology in business and life. – Текст : электронный // Сайт компании, занимающаяся исследованиями рынка : сайт. – URL: https://www.insiderintelligence.com/insights/blockchain-technology-applications-use-cases/ (дата обращения: 14.01.2022)

51. Types of Blockchains. – Текст : электронный // Сайт разработчика корпоративных blockchain-решений и инструментов разработки : сайт. – URL: https://data-flair.training/blogs/types-of-blockchain/ (дата обращения: 14.01.2022)

52. What is IoT?. – Текст : электронный // Официальный сайт и блог компании «Oracle» : сайт. – URL: https://www.oracle.com/cis/internet-of-things/what-is-iot/ (дата обращения: 14.01.2022)

53. What is blockchain technology?. – Текст : электронный // Официальный сайт и блог компании «IBM» : сайт. – URL: https://www.ibm.com/topics/what-is-blockchain (дата обращения: 14.01.2022)

54. Web3dart : репозиторий проекта : сайт – . – URL: https://github.com/xclud/web3dart (дата обращения: 14.01.2022)

55. Zhou, H., The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support / H. Zhou, A. F. Milani, A. Makanju. – DOI 10.3390/jcp2020019 // Journal of Cybersecurity and Privacy. – Текст : электронный. – URL: https://www.mdpi.com/2624-800X/2/2/19 (дата обращения: 14.01.2022)

ПРИЛОЖЕНИЕ А

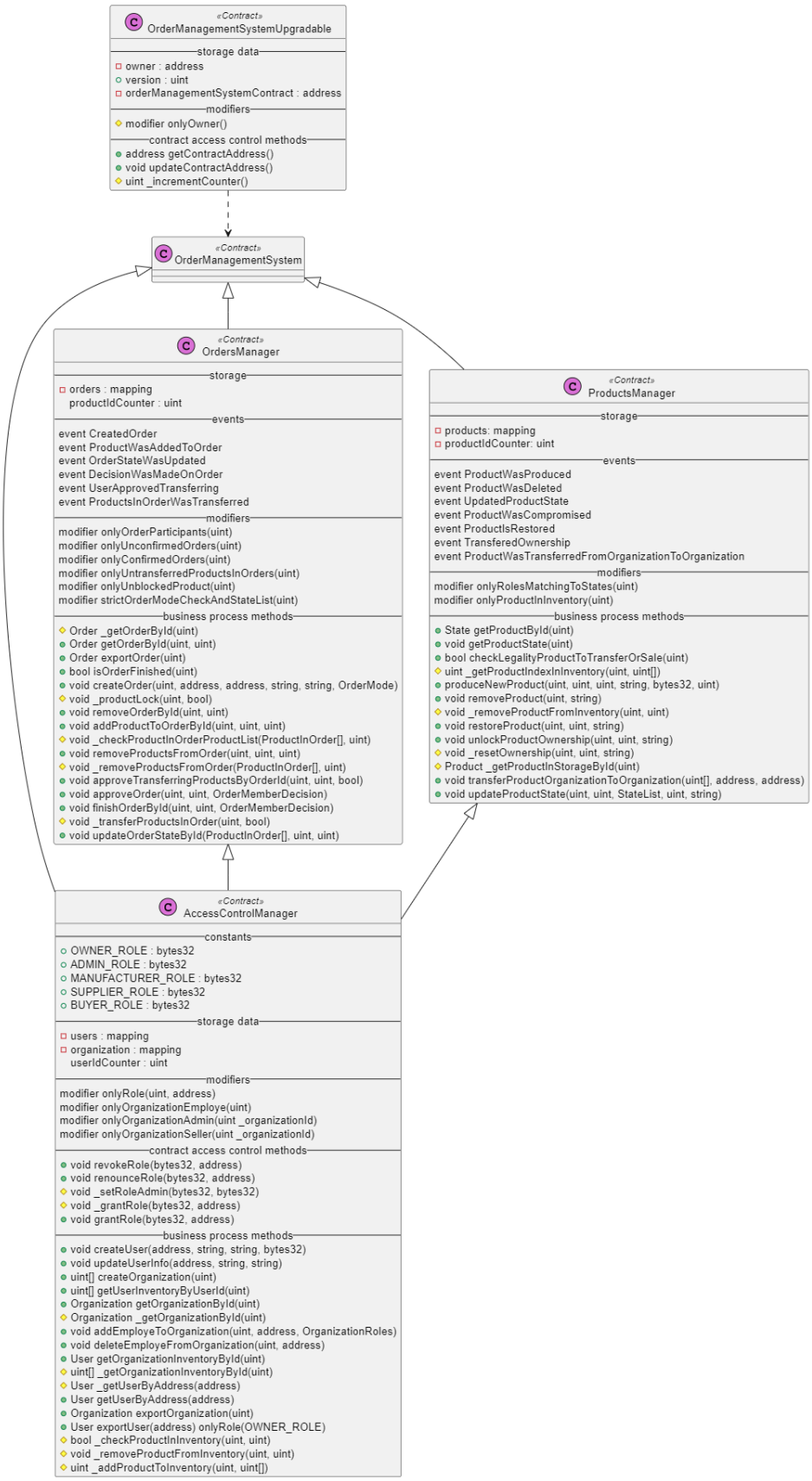

Рисунок А.1 – Развернутая диаграмма классов для смарт-контрактов

ПРИЛОЖЕНИЕ Б

На рисунке Б.1 представлен пользовательский интерфейс страницы перечня заказов. Пользователь на данной странице может просматривать заказы, фильтровать заказы, переходить между вкладками всех заказов, завершенных, активных, отмененных, а также перейти к созданию нового заказа.

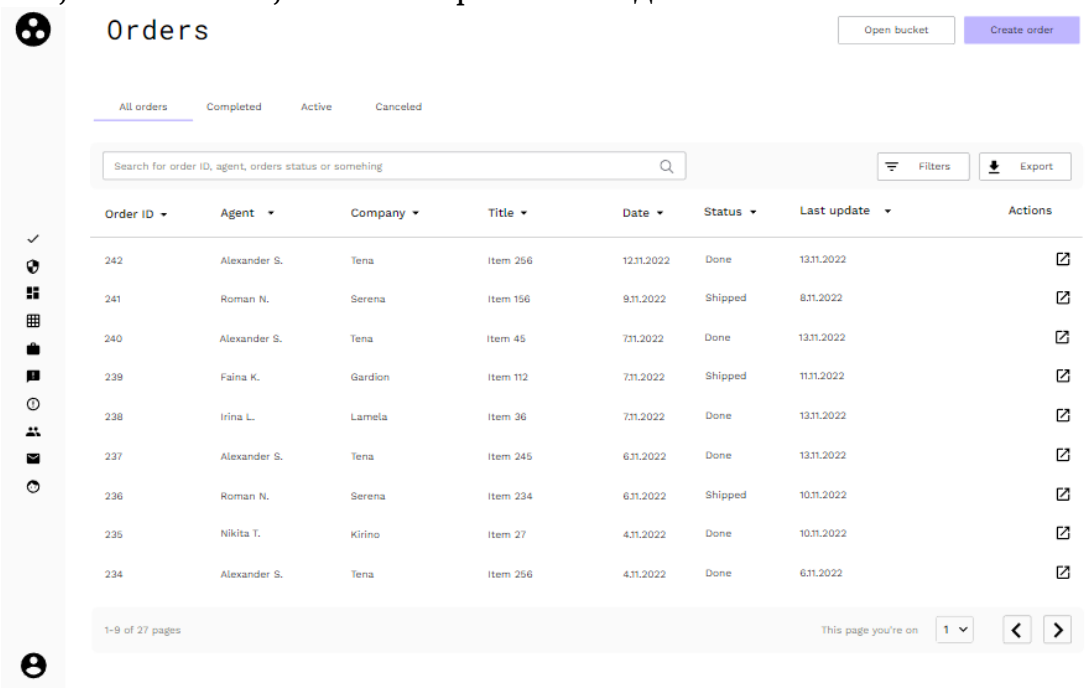

Рисунок Б.1 – Пользовательский интерфейс страницы перечня заказов

На рисунке Б.2 представлен пользовательский интерфейс страницы истории созданного заказа. Пользователь может просмотреть событие заказа, отфильтровать события.

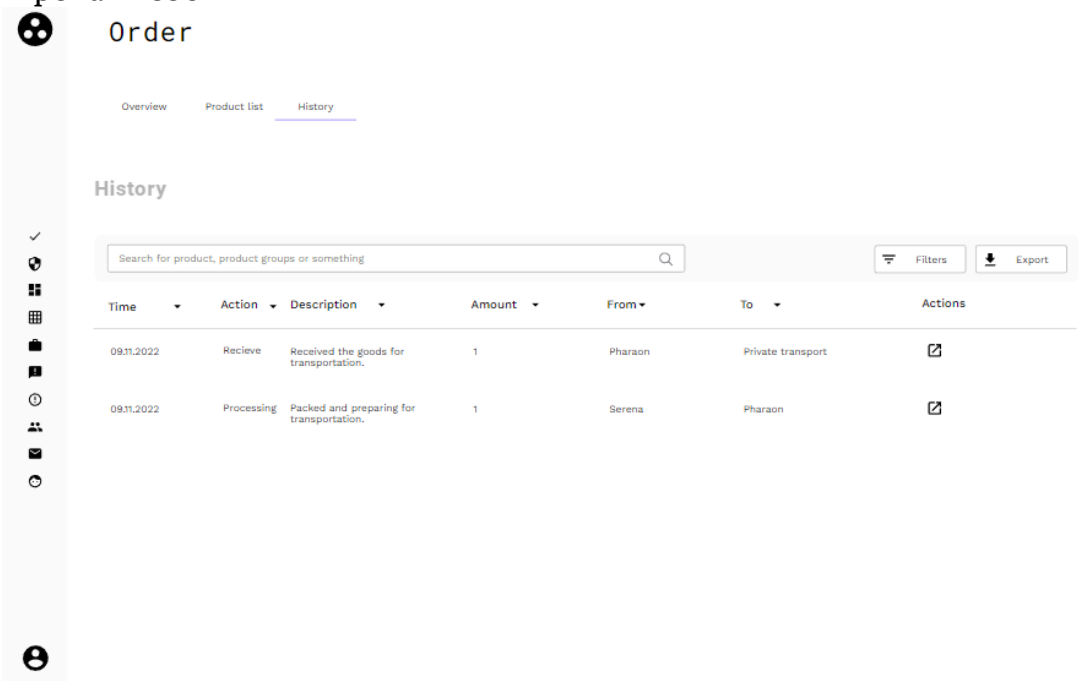

Рисунок Б.2 – Пользовательский интерфейс страницы истории созданного заказа

На рисунке Б.3 представлен пользовательский интерфейс страницы созданного заказа, вкладка перечня продуктов в заказе. Пользователь может осуществлять фильтрацию списка продуктов, а также перейти к просмотру деталей продукта.

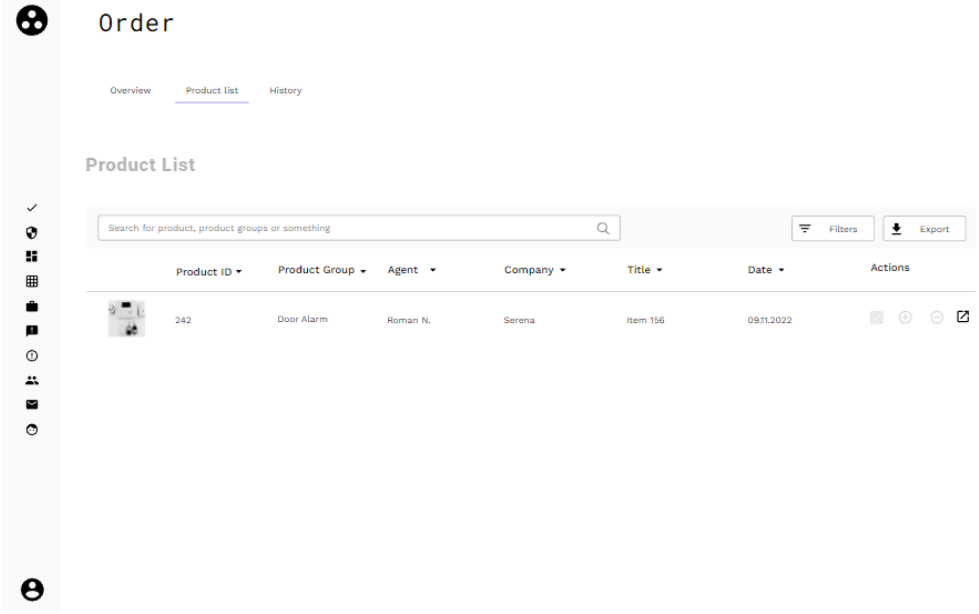

Рисунок Б.3 – Пользовательский интерфейс страницы созданного заказа, вкладка перечня продуктов в заказе

На рисунке Б.4 представлен пользовательский интерфейс страницы создания заказа от роли поставщика. На странице пользователь может осуществлять заполнение информацией заказа, загружать файлы, которые необходимо прикрепить к заказу.

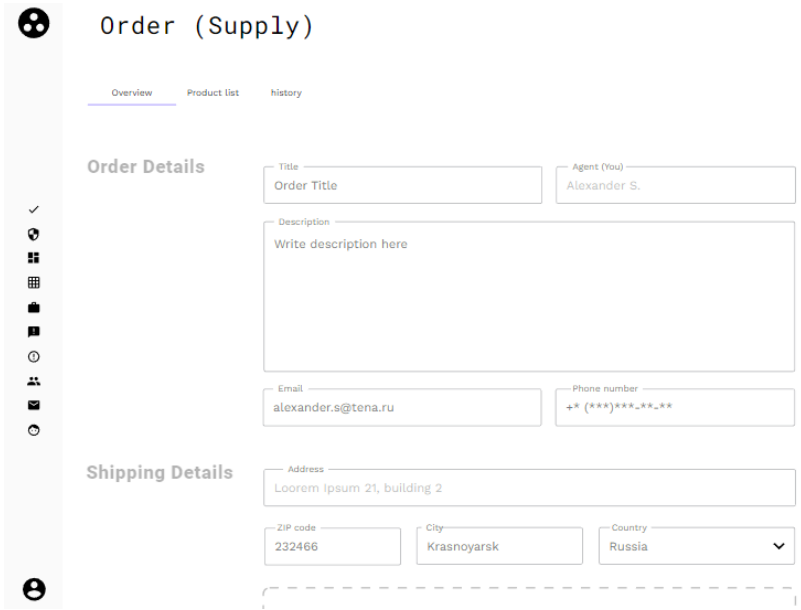

Рисунок Б.4 – Пользовательский интерфейс страницы создания заказа от роли поставщика

На рисунке Б.5 представлен пользовательский интерфейс страницы истории передвижения продукта. На странице пользователь может выгрузить историю перемещения продукта.

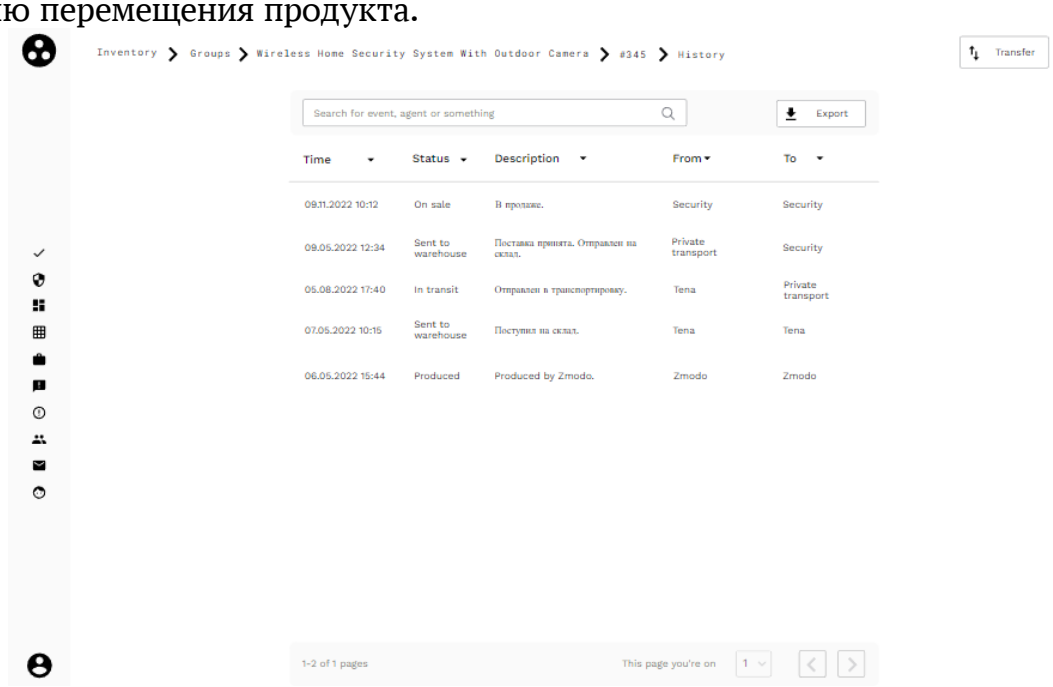

Рисунок Б.5 – Пользовательский интерфейс страницы истории передвижения продукта

На рисунке Б.6 представлен пользовательский интерфейс страницы обзора информации о конкретном продукте. На данной странице пользователь может осуществить просмотр характеристик продукта, перейти к истории перемещения продукта, а также его передачи другому агенту.

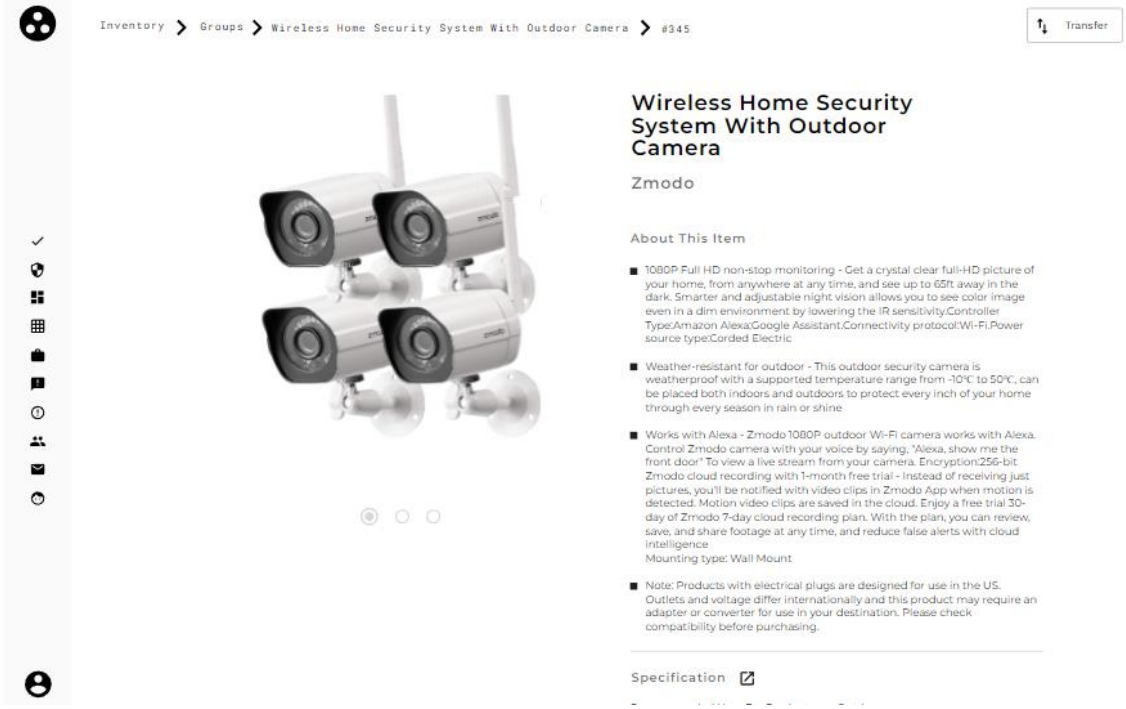

Рисунок Б.6 – Пользовательский интерфейс страницы обзора информации о конкретном продукте

На рисунке Б.7 представлен пользовательский интерфейс страницы ручного обновления состояния продукта. Пользователь может осуществить заполнение полей информацией, а затем подтвердить обновление состояния продукта.

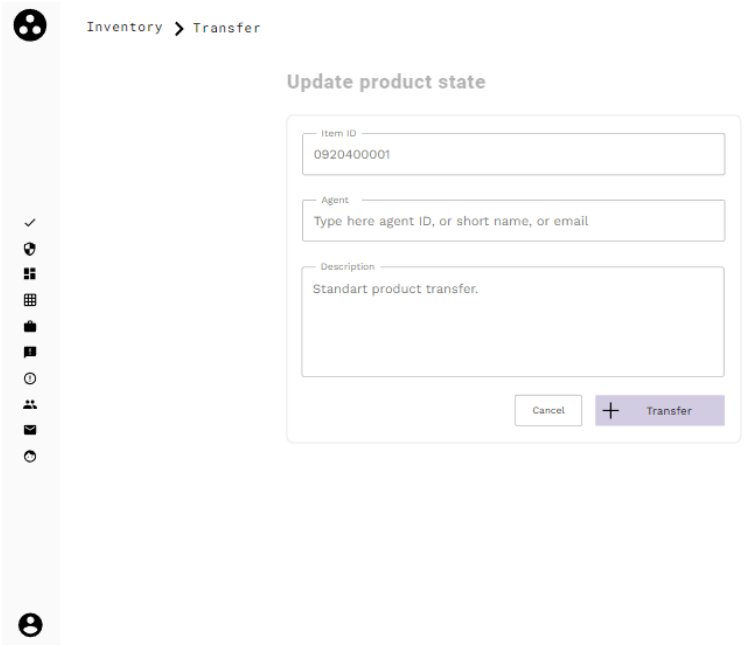

Рисунок Б.7 – Пользовательский интерфейс страницы ручного обновления состояния продукта

На рисунке Б.8 представлен пользовательский интерфейс страницы групп продукции. На данной странице пользователь может осуществить переход к конкретной группе продукта, выгрузить перечень групп продукции, а также его фильтровать и осуществлять поиск.

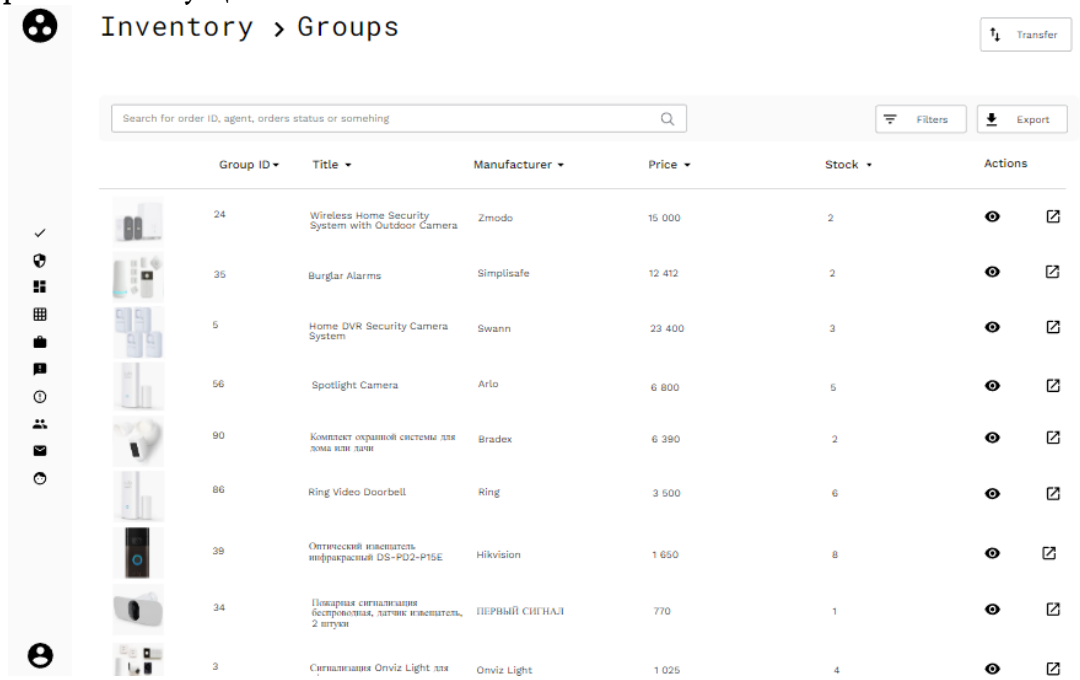

Рисунок Б.8 – Пользовательский интерфейс страницы групп продукции

На рисунке Б.9 представлен пользовательский интерфейс страницы группы продукта. На данной странице пользователь может обновить изображения группы товара, установить общую стоимость, а также просмотреть количество продукции, которая доступна контрагенту.

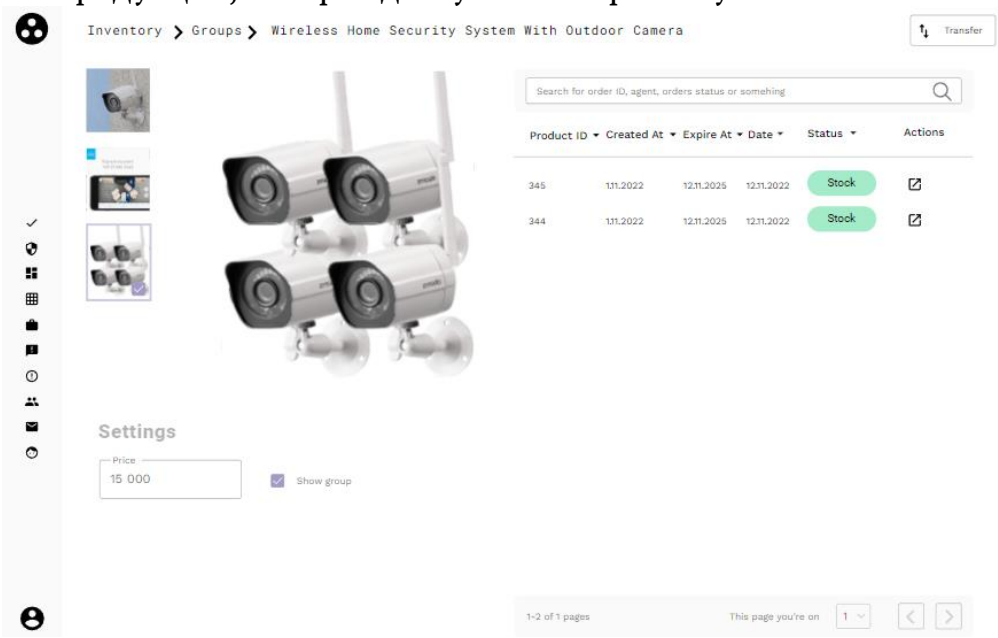

Рисунок Б.9 – Пользовательский интерфейс страницы группы продукта

На рисунке Б.10 представлен пользовательский интерфейс страницы изменения группы продукции, ее описания, категории, спецификации, а также обновлять изображения.

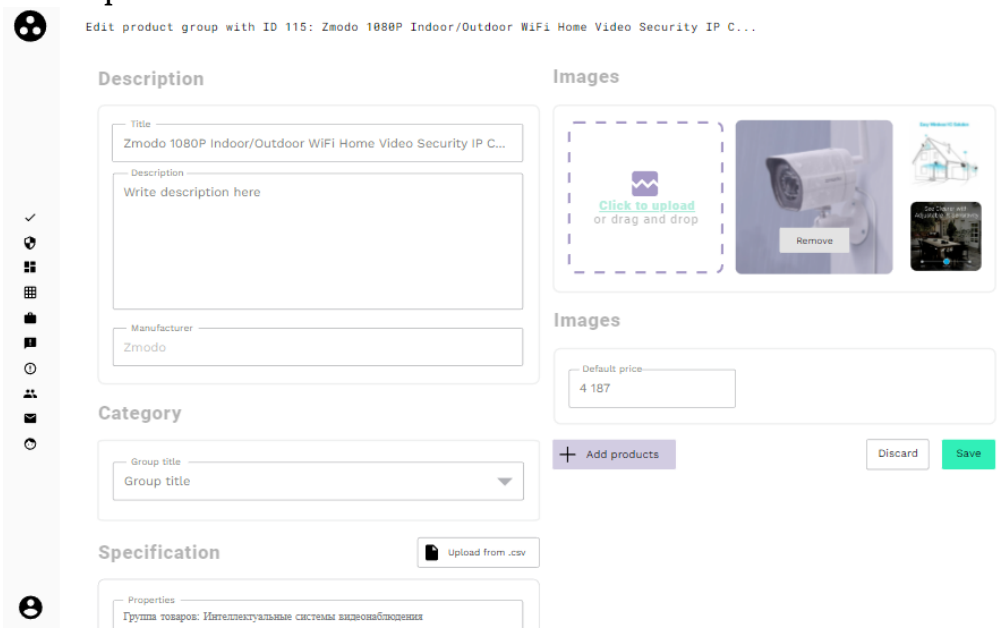

Рисунок Б.10 – Пользовательский интерфейс страницы изменения группы продукции

На рисунке Б.11 представлен пользовательский интерфейс страницы ручного добавления нового продукта в группу продуктов. Данное действие может осуществить только пользователь с ролью производитель. Пользователь может осуществить добавление продукта при помощи внешнего файла.

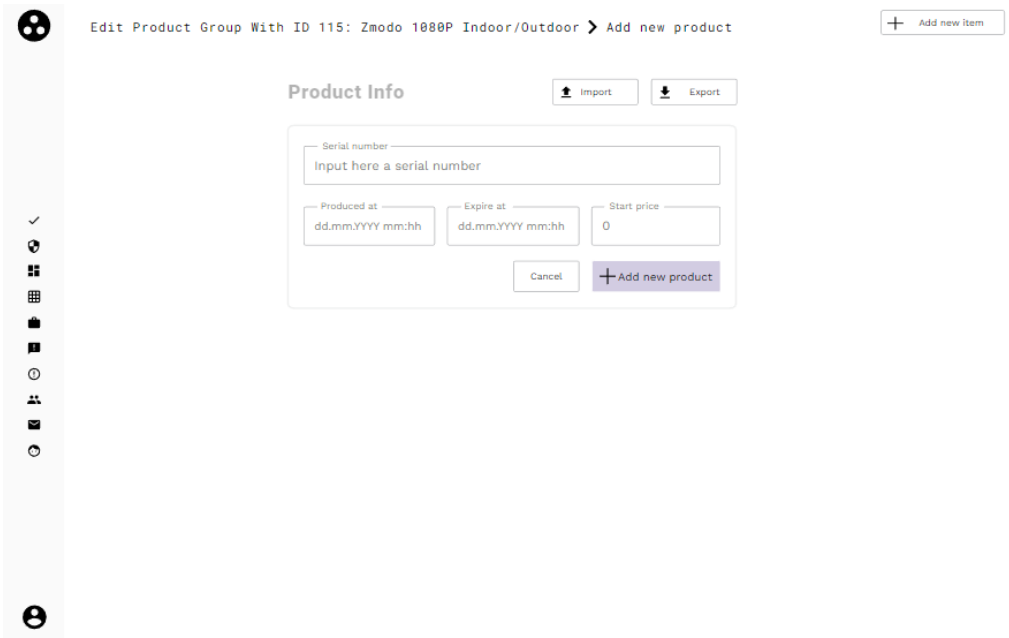

Рисунок Б.11 – Пользовательский интерфейс страницы ручного добавления нового продукта в группу продуктов

На рисунке Б.12 представлен пользовательский интерфейс страницы ручного изменения некоторой информации о продукте: стоимость, состояние, локацию, описание.

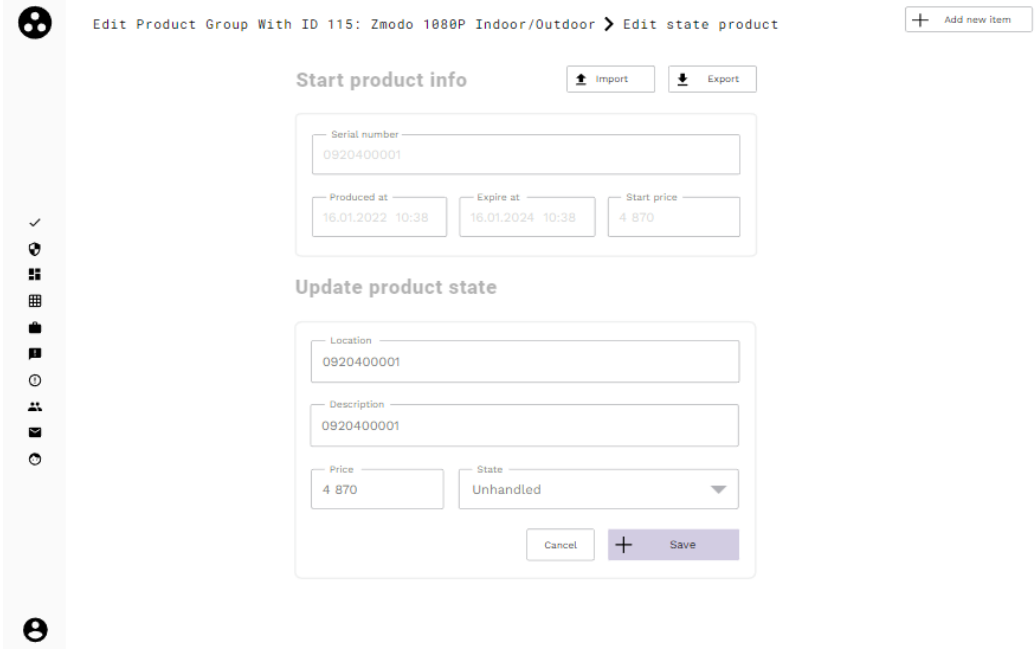

Рисунок Б.12 – Пользовательский интерфейс страницы ручного изменения некоторой информации о продукте

На рисунке Б.13 представлен пользовательский интерфейс страницы просмотра перечня продукции в группе продуктов. Пользователь с ролью производителя может осуществлять импортирование списка продукции в файл, а также создавать продукцию путем внесения файла.

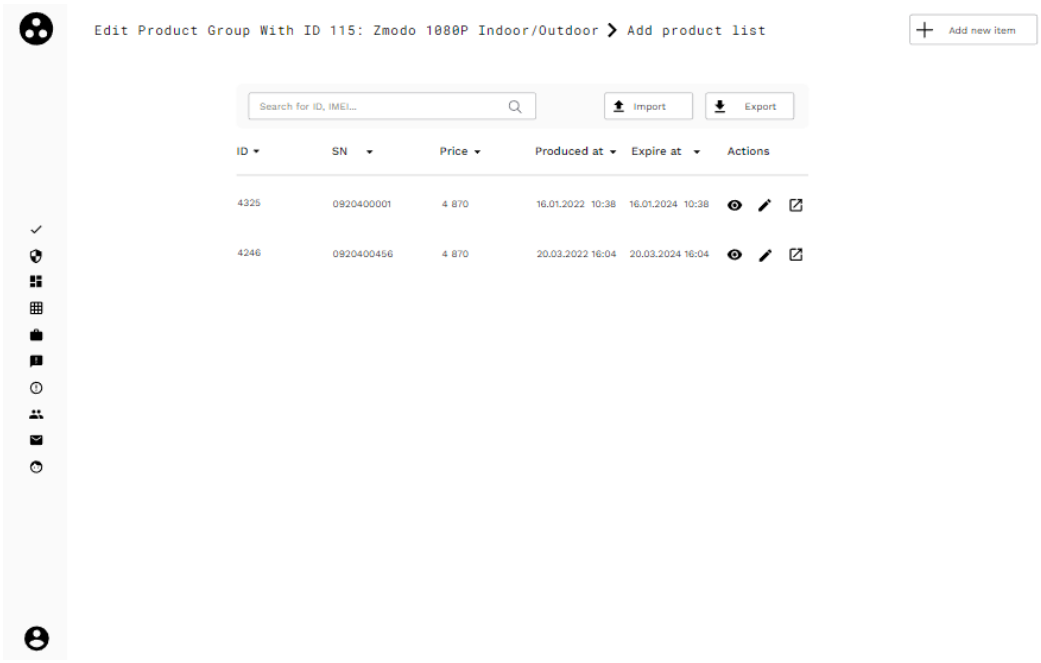

Рисунок Б.13 – Пользовательский интерфейс страницы просмотра перечня продукции в группе продуктов

На рисунке Б.14 представлен пользовательский интерфейс страницы просмотра групп продукции. Пользователь может осуществлять фильтрацию и экспорт списка продукции.

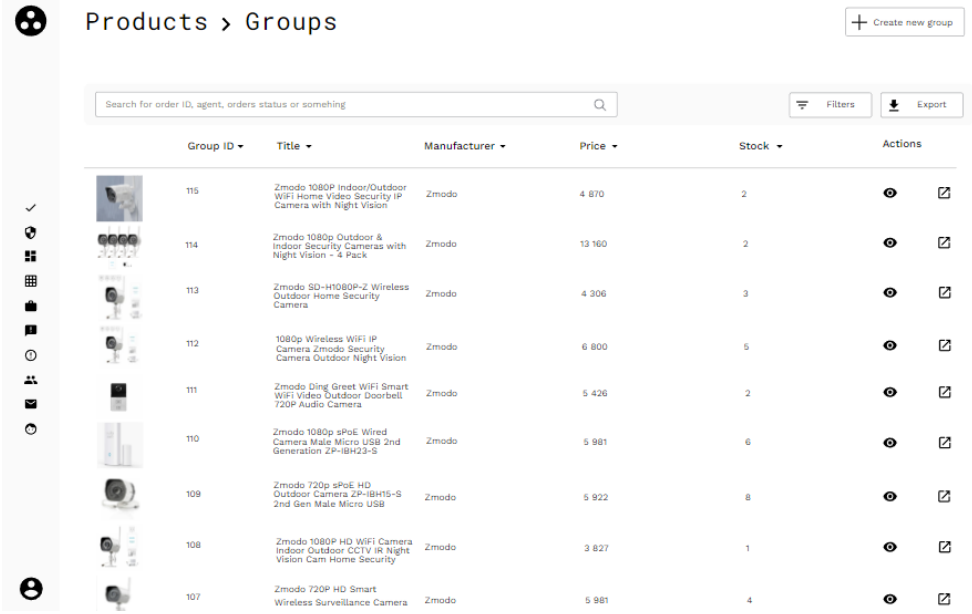

Рисунок Б.14 – Пользовательский интерфейс страницы просмотра групп продукции

На рисунке Б.15 представлен пользовательский интерфейс страницы обзора информации о контрагенте. Пользователь может осуществлять просмотр информации блога контрагента, а также перейти к списку продукции, доступную для заказа.

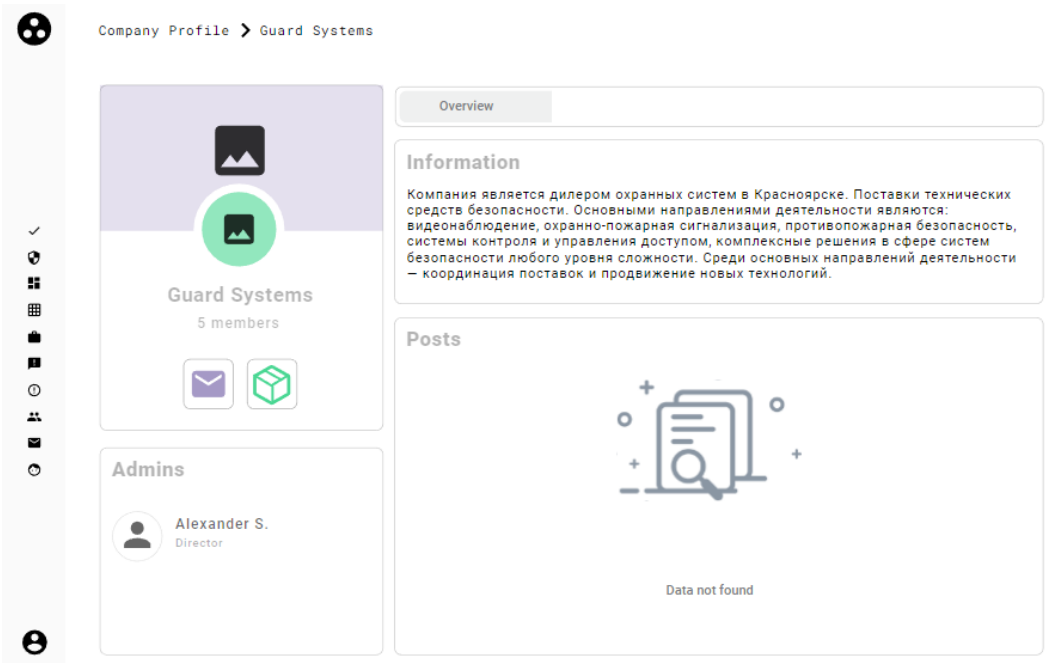

Рисунок Б.15 – Пользовательский интерфейс страницы обзора информации о контрагенте

На рисунке Б.16 представлен пользовательский интерфейс страницы групп продукции выбранного контрагента. Пользователь может осуществлять добавление продукции в свою корзину для последующего заказа, осуществлять импорт списка продукции.

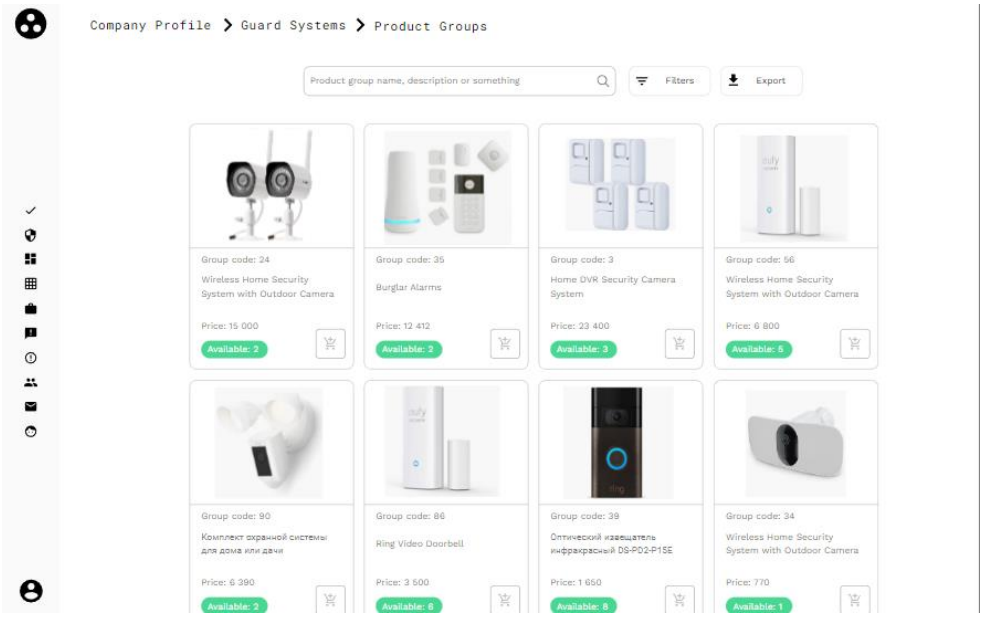

Рисунок Б.16 – Пользовательский интерфейс страницы групп продукции выбранного контрагента