**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc1)

[1 СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ, СУЩЕСТВУЮЩИЕ СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ, ОСНОВАННЫЕ НА АЛГОРИТМАХ ZKP 5](#_Toc2)

[1.1 Система перевода средств между электронными кошельками и её общие характеристики 5](#_Toc3)

[1.2 Обзор существующих аналогов 8](#_Toc4)

[1.3 Обоснование выбора инструментальных средств 11](#_Toc5)

[Выводы к разделу 1 13](#_Toc6)

[2 CИСТЕМОТЕХНИЧЕСКИЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ НА ОСНОВЕ АЛГОРИТМОВ ZKP 15](#_Toc7)

[2.1 Построение диаграмм потоков данных (DFD) в системе перевода средств между электронными кошельками на основе алгоритмов ZKP 15](#_Toc8)

[2.2 Разработка функциональной модели IDEF0 системы перевода средств между электронными кошельками на основе алгоритмов ZKP 18](#_Toc9)

[2.3 Разработка обобщенной архитектуры и алгоритма функционирования системы перевода средств между электронными кошельками на основе алгоритмов ZKP 20](#_Toc10)

[Выводы к разделу 2 22](#_Toc11)

[3 РАЗРАБОТКА СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ НА ОСНОВЕ АЛГОРИТМОВ ZKP 23](#_Toc12)

[3.1 Выбор и обоснование технологических средств разработки системы перевода средств между электронными кошельками на основе алгоритмов ZKP 23](#_Toc13)

[3.1.1 Платформа с технологией распределенного реестра Hyperledger Fabric 23](#_Toc14)

[3.1.2 Язык программирования Java 25](#_Toc15)

[3.1.3 Фреймворк Spring и расширение Spring Boot 25](#_Toc16)

[3.1.4 Язык программирования Go 26](#_Toc17)

[3.1.4 Платформа контейнеризации приложений Docker 26](#_Toc18)

[3.1.5 Система контроля версий Git 27](#_Toc19)

[3.1.6 Документирование REST API при помощи Swagger 27](#_Toc20)

[3.2 Инструментальные средства 28](#_Toc21)

[3.2.1 NetBeans 28](#_Toc22)

[3.2.2 Visual Studio Code 28](#_Toc23)

[3.2.3 Docker Desktop 28](#_Toc24)

[3.3 Разработка API 28](#_Toc25)

[ЗАКЛЮЧЕНИЕ 30](#_Toc26)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc27)

# ВВЕДЕНИЕ

В современном мире наблюдается рост тенденций на внедрение блокчейн технологий в различных отраслях, в том числе и в экономической сфере. Это связано с тем, что данные технологии позволяют не только упростить и увеличить эффективность финансовых операций, но и обеспечить безопасность и прозрачность транзакций, что является неотъемлемым аспектом при проведении операций, связанных с денежными средствами.

При использовании приложений для электронных платежей остро стоит проблема о конфиденциальности данных, об их секретности. Большинство таких систем включают в себя сервисы для аутентификации, невозможности отрицания авторства. Такие возможности могу быть реализованы при помощи простых криптографических протоколов, например, TLS (Transport Layer Security). Протокол TLS – это протокол шифрования и аутентификации, он работает на транспортном уровне сетевой модели OSI, где отвечает за создание безопасных сессий обмена данными между браузером и сервером.

Однако существуют такие операции, которые невозможно защитить при помощи простых протоколов шифрования, вроде TSL. К таким операциям можно отнести электронные платежи, при проведении которых сохранение конфиденциальности является неотъемлемой частью, ведь при несоответствии приложения стандартам безопасности возникают риски утечки личных и финансовых данных пользователей. Также участники таких систем могут стать жертвой мошенничества, поскольку простые протоколы шифрования могут иметь уязвимости, из-за которых становится возможным получить доступ к персональным данным пользователей.

Более тонкие операции, такие как проведение электронных платежей, требуют внедрения более сложных криптографических алгоритмов, чтобы избежать возможности утечки пользовательских данных.

В связи с этим, актуальным является задача разработки системы перевода средств между электронными кошельками на основе алгоритмов доказательства с нулевым разглашением (Zero Knowledge Proof), что и является целью данной курсовой работы. Доказательство с нулевым разглашением (ZKP) представляет собой криптографический протокол, который позволяет одной стороне (доказывающему, prover, P) убедить другую сторону (проверяющего, verifier, V) в истинности какого-либо утверждения, не раскрывая при этом никакой информации, подтверждающей это утверждение. ZKP может быть интерактивным, когда доказывающий повторяет процесс доказательства для каждого проверяющего, а также неинтерактивным, когда доказывающий создаёт доказательство, которым может воспользоваться каждый человек, использующий то же доказательство.

Существует множество вариантов реализации алгоритмов доказательства с нулевым разглашением, например, протокол парольной аутентификации SRP (Secure Remote Password Protocol), или неинтерактивный криптографический протокол zk-SNARKs, но в данной курсовой работе будет использоваться набор криптографических протоколов Idemix.

# 1 СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ, СУЩЕСТВУЮЩИЕ СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ, ОСНОВАННЫЕ НА АЛГОРИТМАХ ZKP

## **1.1 Система перевода средств между электронными кошельками и её общие характеристики**

Предметная область настоящей курсовой работы – Система перевода средств между электронными кошельками на основе алгоритмов ZKP.

Основной целью данной области является обеспечение приватности и безопасности проводимых финансовых транзакций, путём внедрения в систему сложных криптографических протоколов, а именно алгоритмов доказательства с нулевым разглашением.

Одной из задач криптографии является двусторонняя интерактивная игра, в которой доказывающая сторона доказывает истинность некоторого утверждения другой стороне, проверяющей, не раскрывая сущности доказательства. Такая игра называется протоколом интерактивного доказательства или IP-протоколом (interactive proof – IP). Доказательство, получаемое при помощи такого IP-протокола, является секретным, потому что, во-первых, проверяющая сторона, убедившись в истинности доказанного утверждения, не способна самостоятельно повторить доказательство, и, во-вторых, после завершения протокола никто извне не способен понять сообщения, которыми обменивались стороны в процессе доказательства.

Такая конфиденциальность необходима во многих приложениях, в которых проводятся операции с использованием персональных данных, раскрытие которых может привести к фальсификации, подделыванию или краже личных данных. Например, при проведении голосований необходимо сохранять анонимность голосующих, чтобы предотвратить возможное принуждение или подтасовывание результатов, или при проведении электронных платежей, в процессе которых используются финансовые данные пользователей.

Доказательство с нулевым разглашением должно обладать следующими свойствами:

* Полнота: при корректном утверждении и при соблюдении обеими сторонами принципов протокола, проверяющий может однозначно убедиться в корректности и истинности утверждения;
* Устойчивость: при ложном утверждении проверяющая сторона должна иметь возможность убедиться в ложности утверждения;
* Нулевое разглашение: при истинности утверждения проверяющая сторона не сможет узнать ничего, кроме того факта, что утверждение верно. Другими словами, простого знания утверждения (а не секрета) достаточно, чтобы убедиться в том, что доказывающий знает этот секрет.

ZKP относится к «вероятностным доказательствам». Это класс криптографических протоколов, в которых проверяющая сторона может сделать вывод о справедливости утверждения лишь с некоторой вероятностью.

Протоколы ZKP можно разделить на интерактивные и неинтерактивные. В интерактивных протоколах стороны должны быть постоянно «на связи» и обмениваться сообщениями (это называется раунды взаимодействия). А в неинтерактивных протоколах взаимодействие между участвующими сторонами сводится к одному раунду, пересылке ZK-доказательства от доказывающего к проверяющему.

Использование интерактивных протоколов в распределенных реестрах ограничено необходимостью постоянного обмена сообщениями между доказывающим и проверяющим для проведения множества раундов взаимодействия. На практике такие протоколы не изучаются и не применяются в распределенных реестрах.

В отличие от них, неинтерактивные протоколы лучше подходят для распределенных реестров, так как требуют лишь одного раунда взаимодействия: отправки доказательства и ответа при его проверке.

Распределенные реестры характеризуются открытым хранением данных, при этом информация о транзакциях не скрывается. Однако нельзя скрыть отдельные транзакции, данные о них или некоторые поля транзакции.

На рисунках 1.1 и 1.2 представлены алгоритмы работы интерактивного и неинтерактивного протоколов доказательства с нулевым разглашением.

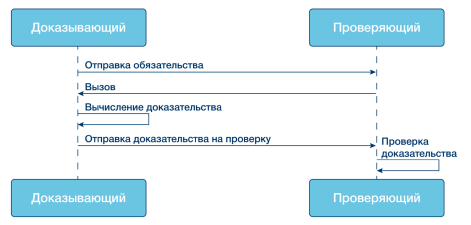


Рисунок 1.1 – Алгоритм работы интерактивного протокола доказательства с нулевым разглашением



Рисунок 1.2 – Алгоритм работы неинтерактивного протокола доказательства с нулевым разглашением

## **1.2 Обзор существующих аналогов**

Алгоритмы доказательства с нулевым разглашением находят практическое применение в различных областях, таких как блокчейн, криптовалюты и децентрализованные финансы (DeFi). ZKP позволяет проверять информацию без её раскрытия, что повышает конфиденциальность и безопасность. Многие проекты DeFi уже используют технологию ZKP для обеспечения большей конфиденциальности в сферах кредитования, займов или торговли. Некоторые блокчейны также внедряют роллапы на основе ZKP или zkEVM (Zero Knowledge Ethereum Virtual Machine).

Для анализа были выбраны только некоторые из существующих решений, а именно: криптовалюта Monero, криптовалюта ZCash.

**1.2.1 Криптовалюта Monero**

Monero (официальный тикер – XMR) – это криптографическая валюта, которая была создана на базе блокчейна ByteCoin, в 2014 году. Данная криптовалюта основана на таком протоколе как CryptoNote, при помощи которого обеспечивается анонимность при проведении транзакции.

Большинство криптовалют, например Bitcoin или Ethereum, построены на прозрачных блокчейн-сетях, то есть все транзакции в них прозрачны и могут быть отслежены любым пользователем сети, а адреса отправителя и получателя могут быть связаны с реальной личностью человека. Monero же использует криптографические протоколы для защиты адресов пользователей, с целью сокрытия конфиденциальных данных, а также для самих транзакций.

Все транзакции, проводимые в Monero по умолчанию «запутывают» адрес отправителя и получателя, а также суммы транзакции. Такая конфиденциальность означает, что деятельность каждого пользователя Monero повышает конфиденциальность всех других пользователей сети, в отличие от выборочно прозрачных криптовалют, как ZCash, в которых можно выбрать тип проводимой транзакции.

CryptoNote в сочетании с обфускацией («запутыванием») транзакций приводят к «пассивному смешиванию», когда все транзакции в сети анонимны и в случае необходимости каждый участник сможет применить правдоподобное отрицание. На рисунке 1.3 представлен принцип шифрования транзакции по протоколу CryptoNote.

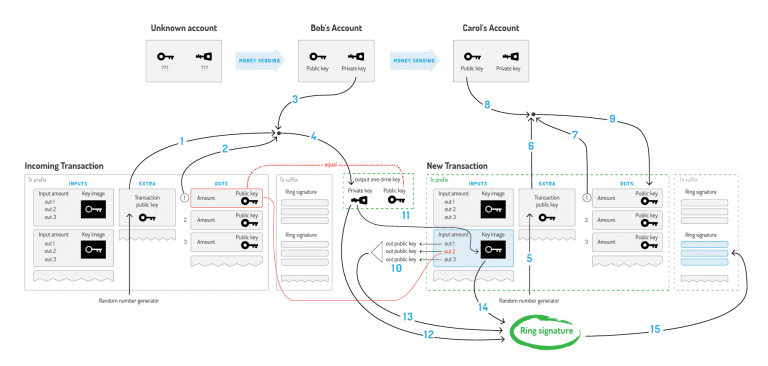


Рисунок 1.3 – Принцип работы протокола CryptoNote

Можно выделить следующие недостатки данной криптовалюты:

* Monero много раз подвергалась критике за свою анонимность, поскольку власти считают, что это способствует распространению нелегальной информации и товаров. Однако анонимность в сети не является абсолютной, и существуют уязвимости, которые могут быть использованы для нарушения приватности транзакций;
* Транзакции в сети являются довольно тяжеловесными, их объём в 8 раз больше чем транзакции Bitcoin, что отражается на скорости выполнения транзакций;
* Также, в 2018 году была обнаружена уязвимость, благодаря которой было возможным «сжигать» депозит, выраженный в криптовалюте, но эта проблема была устранена после выпуска патча с названием Devs Patch.

**1.2.2 Криптовалюта ZCash**

ZCash (официальный тикер – ZEC) – криптовалюта, ориентированная на конфиденциальность которая является форком (fork) Bitcoin.

Форк (fork) – копирование исходного кода с внесением в него изменений.

Основной особенностью ZCash является то, что можно настроить прозрачность проводимых транзакций. В данной сети пользователи сами могут выбирать как провести транзакцию, с сокрытием данных (частная транзакция) или же без сокрытия (публичная транзакция).

Конфиденциальность в ZCash достигается при помощи алгоритмов доказательств с нулевым разглашением. А именно при помощи zk-SNARK – набора инструментов с нулевым кратким неинтерактивным аргументом знаний, при помощи которого обеспечивается шифрование транзакций в блокчейне, при этом проверяя их легитимность посредством консенсуса установленного в сети.

ZCash использует предоставляет доступ к двум типом адресов кошельков: z-адреса (частные) и t-адреса (прозрачные). В зависимости от адресов, используемых отправителем и получателем, существуют 4 типа транзакций:

* Частные транзакции: обе стороны используют z-адреса кошельков, сохраняя детали и суммы транзакций анонимными;
* Публичные транзакции: обе стороны используют t-адреса кошельков, благодаря чему все детали транзакции прозрачны;
* Деэкранирующие транзакции: отправитель использует z-адрес, а получатель – t-адрес, в результате чего раскрываются данные о получателе и сумме транзакции;
* Экранирующие транзакции: отправитель использует t-адрес, а получатель – z-адрес, в этом случае общедоступными становятся данные отправителя, а также сумма транзакции.

На рисунке 1.4 представлен процесс проведения транзакции в сети ZCash

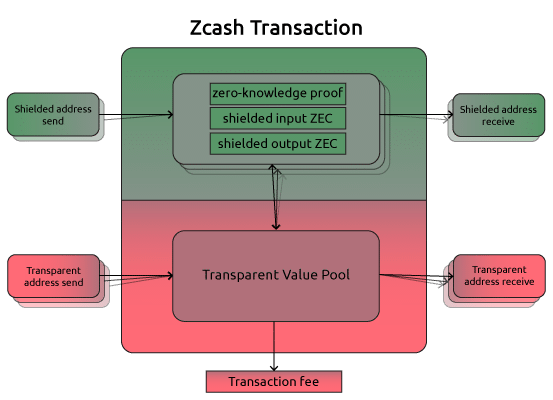


Рисунок 1.4 – Принцип проведения транзакции в ZCash

К недостаткам данной криптовалюты можно отнести низкую скорость проведения транзакций, из-за множества сложных математических вычислений, а также то, что официально поддерживается только на операционной системе Linux.

## **1.3 Обоснование выбора инструментальных средств**

Реализацию системы перевода средств между электронными кошельками на основе алгоритмов ZKP было решено проводить с использованием Hyperledger Fabric и Identity Mixer.

Hyperledger Fabric – это блокчейн-платформа, написанная на языке программирования Go с открытым исходным кодом от Linux Foundation. С её помощью можно создавать блокчейн-приложения и управлять доступом и разрешениями для данных в цепочке блоков. Также, как и другие блокчейн-технологии он использует реестр, смартконтракты и систему, с помощью которых участники управляют своими транзакциями.

Основное отличие HLF (Hyperledger Fabric) от других блокчейн-систем заключается в том, что он является «закрытым и контролируемым». В отличие от открытых систем, позволяющих неидентифицированным пользователям участвовать в работе сети (например, механизм консенсуса PoW в Bitcoin), участники блокчейнов Hyperledger Fabric обязаны проходить регистрацию с получением сертификатов. В дальнейшем, уровень доступа к отдельным структурам блокчейна может быть настроен с помощью анализа идентификаторов, например, отдельные группы участников могут создавать каналы, которые будут являться отдельным регистром транзакций, доступным только им.

Реестр Hyperledger Fabric имеет состоящую из двух компонент подсистему: world state (состояние мира) и transaction log (журнал транзакций). Каждый участник имеет копию реестра каждой сети Hyperledger Fabric, в которой он состоит.

Компонент world state описывает состояние реестра в определенный момент времени. Это база данных реестра. Компонент transaction log записывает все транзакции, которые привели к текущему world state; это обновленная история для world state. В реестре, тогда, это комбинация базы данных world state и истории transaction log.

Реестр имеет сменное хранилище данных для world state. По умолчанию, это база LevelDB, работающая со структурами данных типа «ключ — значение». В сменном transaction log нет необходимости, он просто записывает значения «до» и «после» базы данных реестра при использовании блокчейн-сети.

Все смартконтракты Hyperledger Fabric написаны через чейнкоды и вызываются через внешнее приложение, когда ему требуется провзаимодействовать с реестром. В большинстве случаев, чейнкод взаимодействует только с базой данных реестра – компонентом world state (например, совершая поисковые запросы).

Чейнкод может быть написан на нескольких языках программирования. В настоящее время поддерживаются Go и Node.js.

Identity Mixer (Idemix) – набор криптографических протоколов, обеспечивающих надежную аутентификацию вкупе с сохраняющими конфиденциальность свойствами, такими как анонимность, возможность совершать транзакции без выдачи сторон транзакции, и несвязность (unlinkability), возможность совершить одной identity несколько транзакций, не выдав, что все эти транзакции были совершены одной и той же identity.

В процессе Idemix участвуют три актора: пользователь (user), издатель (issuer) и верификатор (verifier). На рисунке 1.5 представлен процесс взаимодействия акторов.

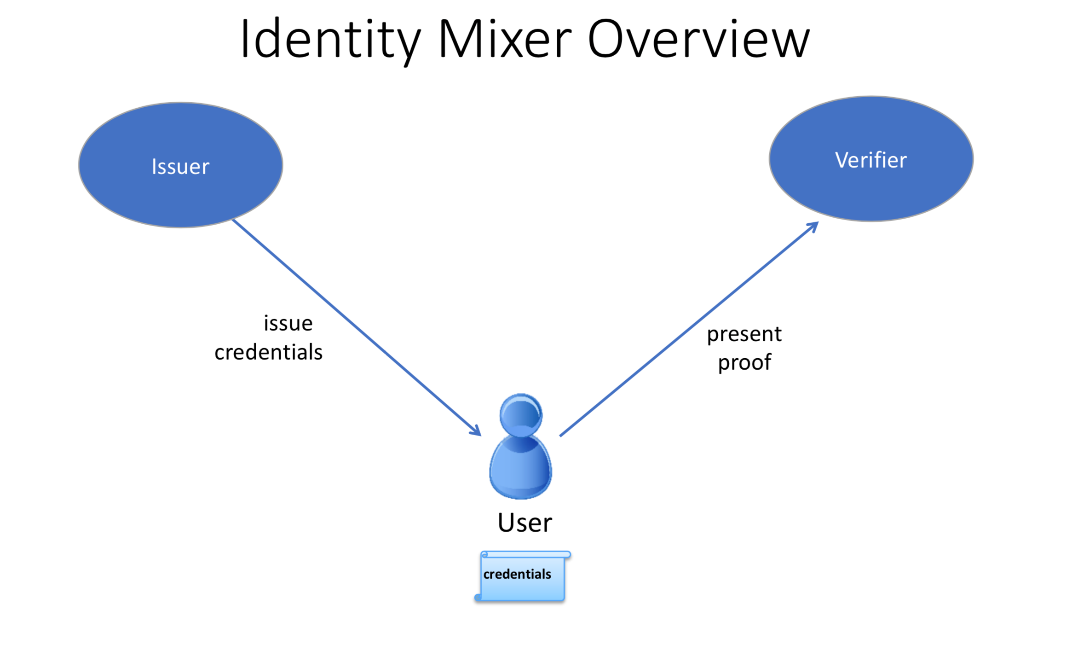


Рисунок 1.5 – Процесс взаимодействия акторов в Idemix

Издатель удостоверяет набор атрибутов пользователя - выпускает цифровой сертификат, далее именуемый «удостоверение» (credential).

Потом пользователь генерирует доказательство с нулевым разглашением, что он владеет удостоверением и также выборочно раскрывает атрибуты по собственному желанию. Так как доказательство ничего не разглашает, оно не раскрывает никакой другой информации про верификатора, издателя или кого-либо еще.

## **Выводы к разделу 1**

В данном разделе был проведён анализ системы перевода средств между электронными кошельками на основе алгоритмов ZKP. Были рассмотрены принципы работы протоколов доказательства с нулевым разглашением.

Было выяснено, что стандартные протоколы шифрования не могут обеспечить полную безопасность конфиденциальных данных в системах перевода средств между электронными кошельками, и для обеспечения лучшей защищенности необходимо использовать более тяжелые криптографические протоколы, например, доказательства с нулевым разглашением.

Также были рассмотрены некоторые аналоги данной курсовой работы. Были рассмотрены две криптографические валюты Monero и ZCash, изучены принципы их функционирования, а также алгоритмы протоколов, которые используются ими для шифрования данных. Также были выделены основные недостатки данных аналогов.

# 2 CИСТЕМОТЕХНИЧЕСКИЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ НА ОСНОВЕ АЛГОРИТМОВ ZKP

## **2.1 Построение диаграмм потоков данных (DFD) в системе перевода средств между электронными кошельками на основе алгоритмов ZKP**

В системе перевода средств между электронными кошельками на основе алгоритмов ZKP основными процессами являются: регистрация пользователей оператором системы, а также создание кошельков, которые могут иметь различные категории; перевод пользователями средств на счета операторов переводов только определённой категории; проверка отнесённости кошелька к определённой категории, без раскрытия персональных данных пользователя; установка лимитов на переводы получателем переводов.

Оператор системы должен иметь возможность выполнять следующие операции:

1. регистрация пользователей в системе. При регистрации генерируется уникальный адрес, а также задаётся ФИО и другие данные пользователя;

2. просмотр пользовательских профилей. При просмотре профилей, оператор может получить всю необходимую информацию о пользователе, но без возможности её изменения;

3. создание категорий кошельков. Оператором могут создаваться специфические категории кошельков, которые могут потребовать дополнительной информации от пользователя, подтверждающей право владения кошельком определённой категории;

4. создание кошельков. Оператор может создавать кошельки различных категорий;

5. просмотр списка категорий кошельков. Оператор может просмотреть все созданные им категории кошельков;

6. просмотр списка кошельков. Оператор может просмотреть какими кошельками владеет определённый пользователь.

Пользователям системы доступны следующие возможности:

1. создание кошелька определённой категории. Пользователь может выбрать какую категорию кошелька ему необходимо создать, а также может предоставить необходимую дополнительную информацию, если того требует выбранная категория кошелька;

2. перевод средств на счет оператора переводов. У пользователей имеется возможность выполнить перевод средств со своего кошелька, на другой, но только одинаковой категории.

Получатели переводов имеют следующие возможности:

1. установка лимитов на переводы. Получатели переводов имеют возможность установить лимит на размер получаемых переводов, например, они могут установить лимит на месячный или ежедневный перевод.

Были выделены внешние сущности данной предметной области, а также определены потоки данных, которыми обмениваются процесс и внешние сущности.

На основе этого была построена DFD-диаграмма основного процесса. Данная диаграмма позволяет продемонстрировать процессы с точки зрения данных, отображает потоки данных между системами.

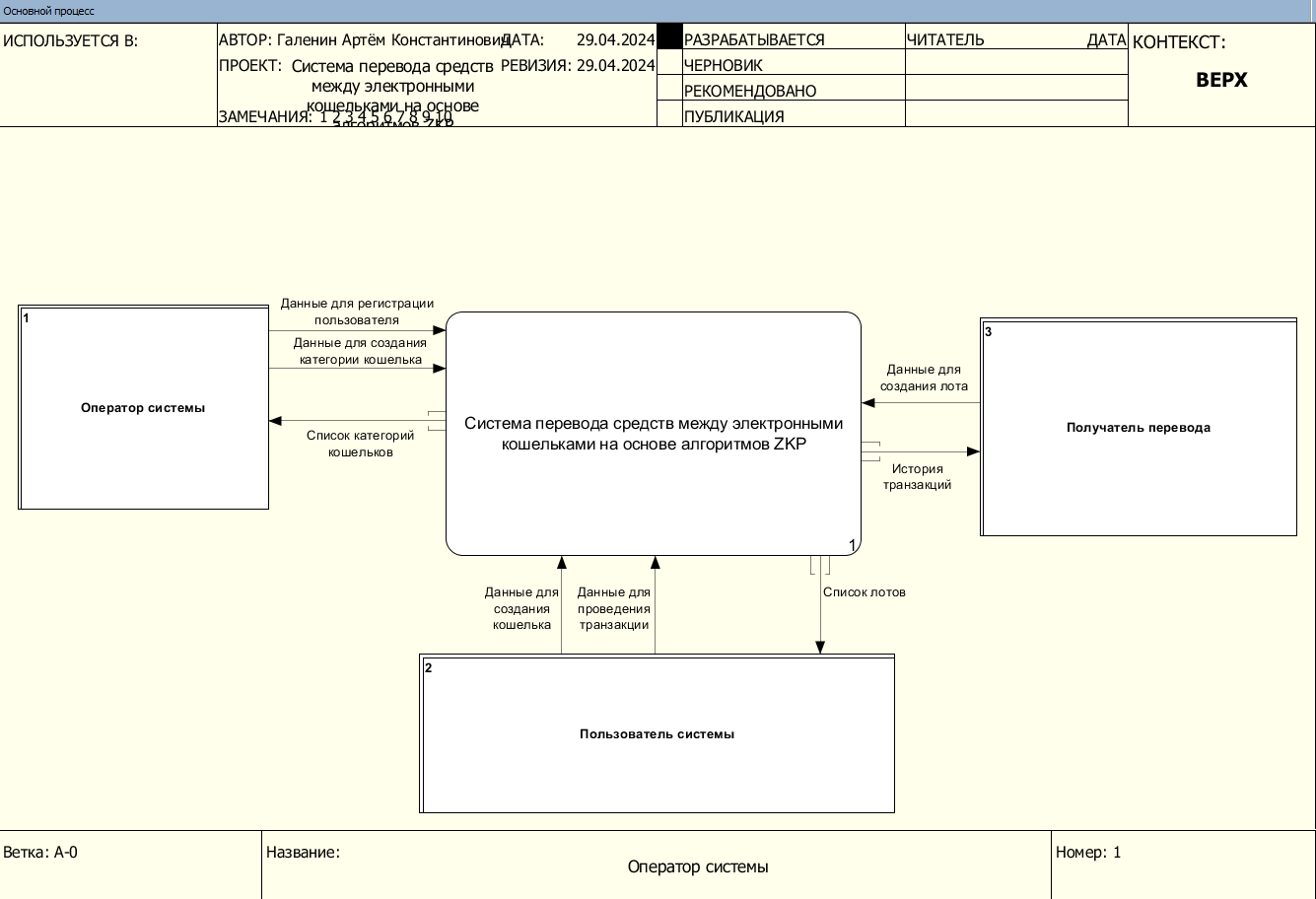


Рисунок 2.1 – DFD-диаграмма основного процесса

Далее была разработана DFD-диаграмма возможных действий внешних сущностей системы, а именно оператора системы, пользователя и получателя переводов.

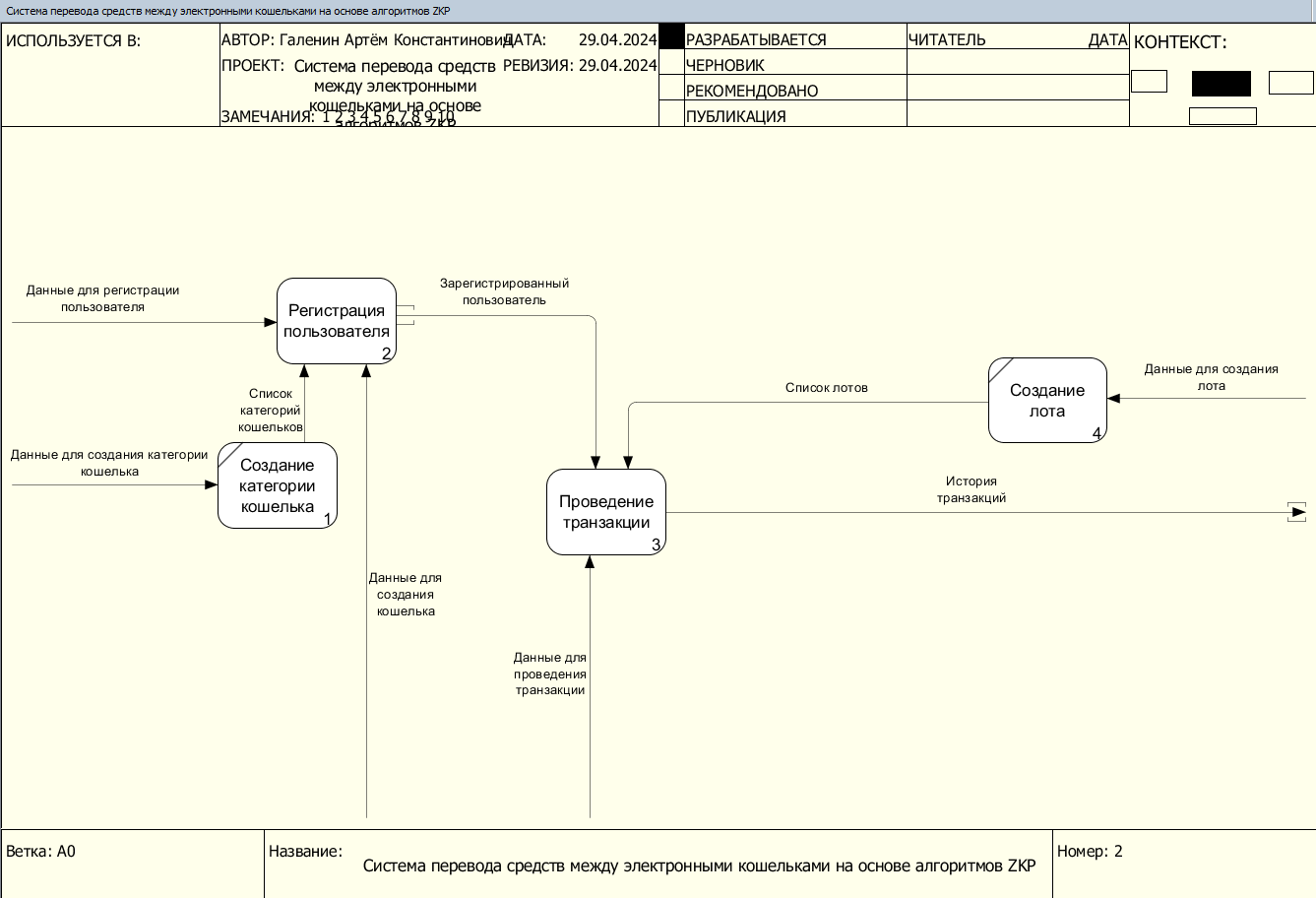


Рисунок 2.2 – DFD-диаграмма возможных действий внешних сущностей системы

После чего была разработана DFD-диаграмма процесса регистрации пользователя в системе.

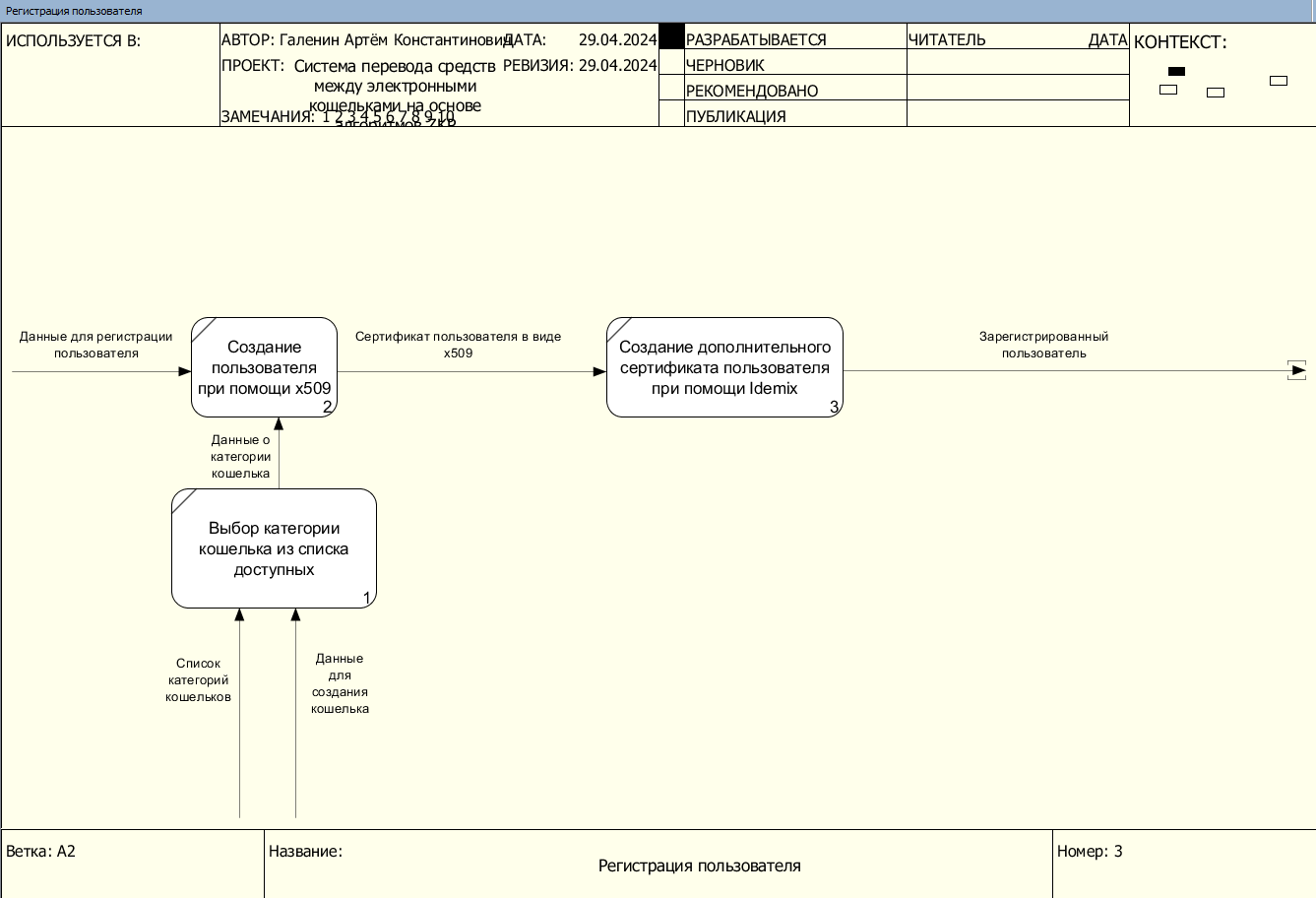


Рисунок 2.3 – DFD-диаграмма действия регистрация пользователя

Затем была разработана DFD-диаграмма процесса проведения транзакции.

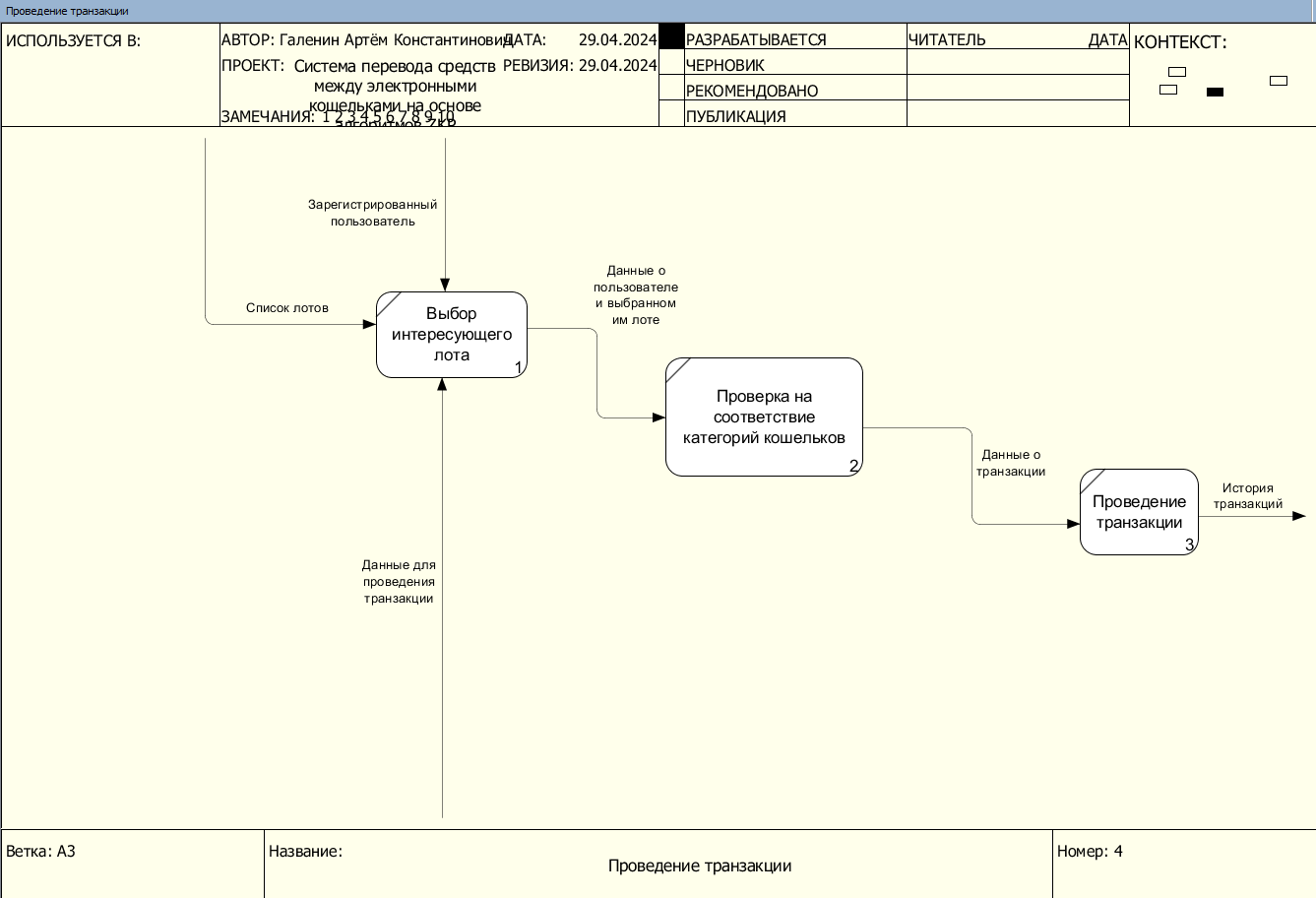


Рисунок 2.4 – DFD-диаграмма действия проведение транзакции

## **2.2 Разработка функциональной модели IDEF0 системы перевода средств между электронными кошельками на основе алгоритмов ZKP**

Была построена IDEF0-диаграмма основного процесса, затем была произведена её декомпозиция и получена IDEF0-диаграмма первого уровня.

IDEF0 это нотация для моделирования бизнес-процессов, она помогает проанализировать процессы, позволяет уточнить специфические требования к разрабатываемому проекту, а также поддерживает разработку на уровне систем и на уровне задач по интеграции.

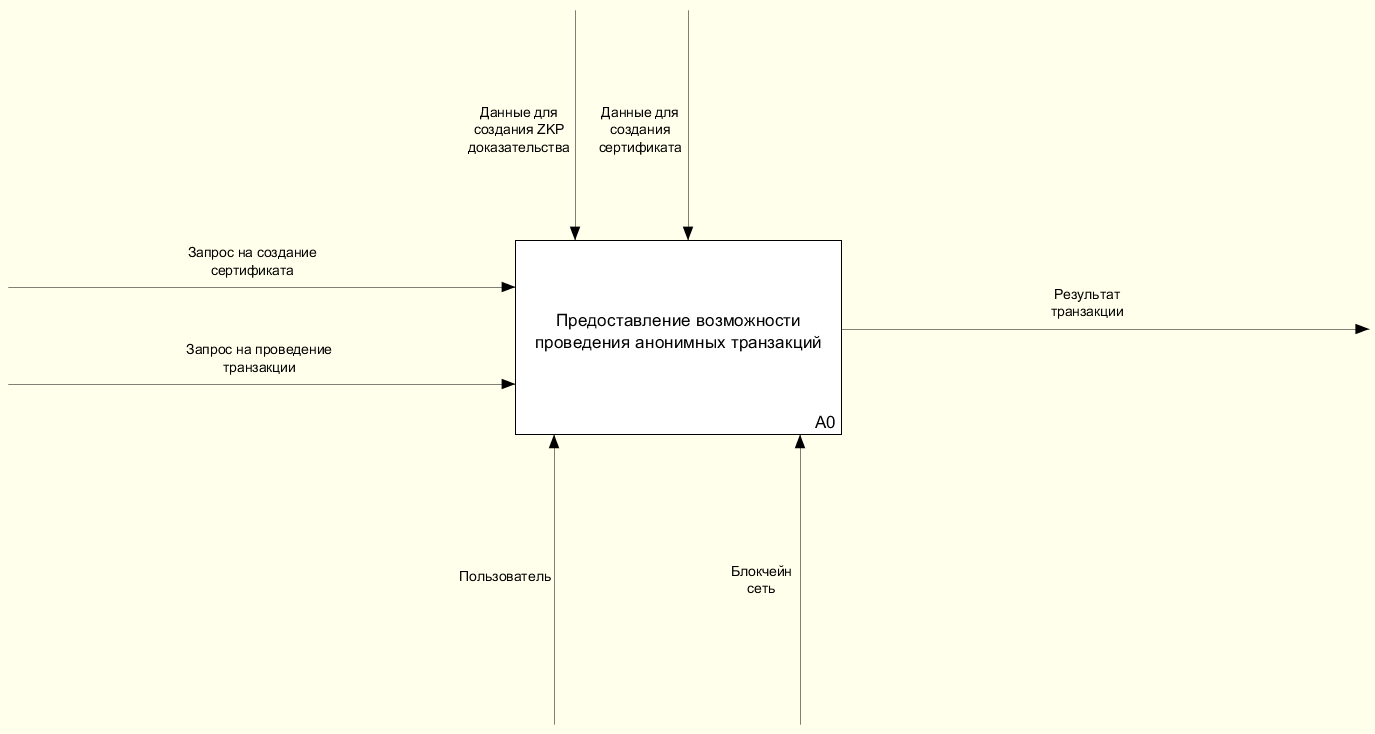


Рисунок 2.5 – IDEF0-диаграмма основного процесса

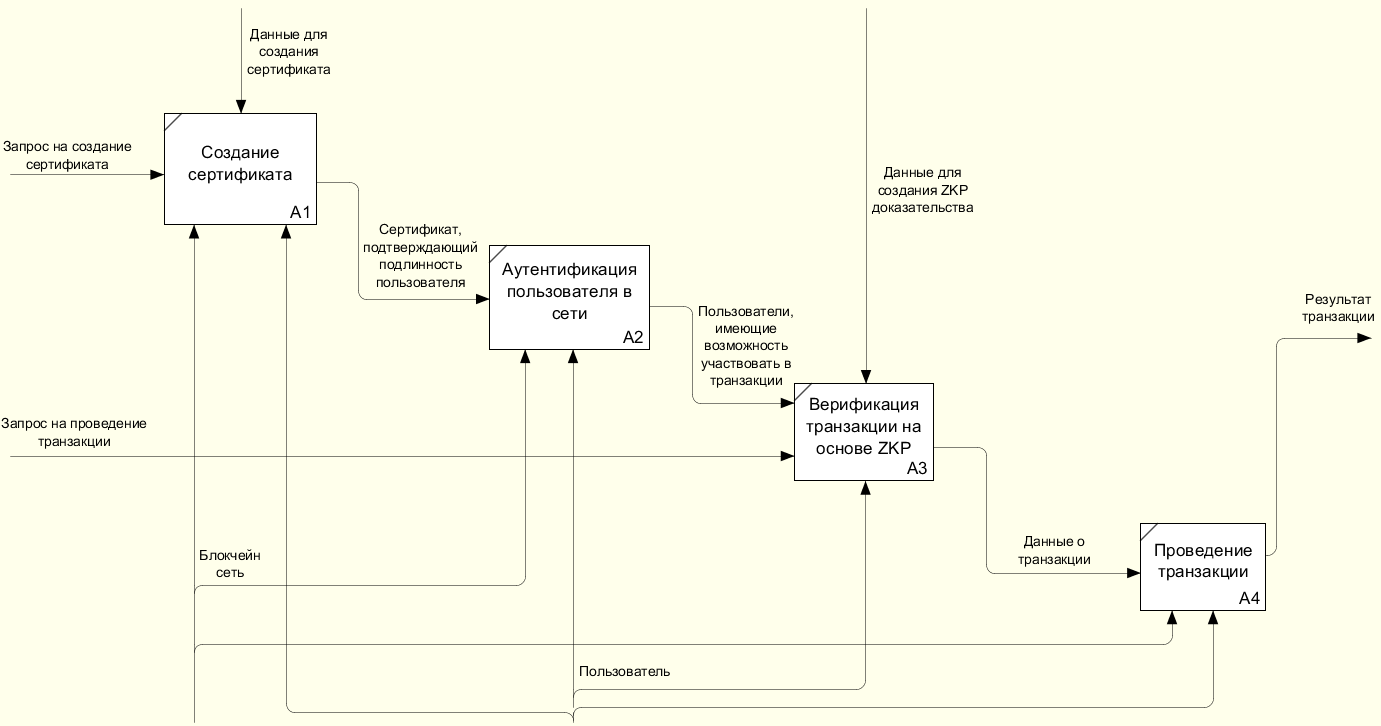


Рисунок 2.6 – IDEF0-диаграмма декомпозиции первого уровня

После чего была произведена декомпозиция процесса «Создание сертификата», результат продемонстрирован на рисунке 3.

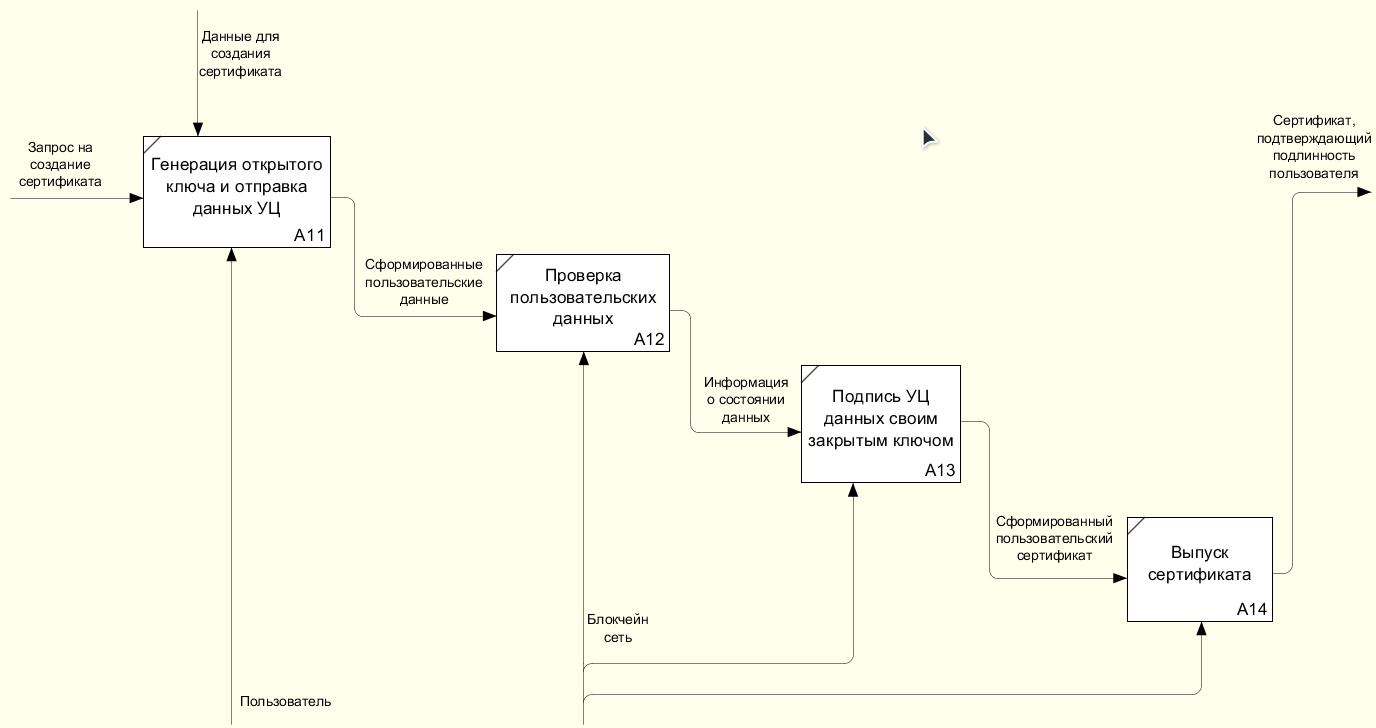


Рисунок 2.7 – IDEF0-диаграмма декомпозиции второго уровня

## **2.3 Разработка обобщенной архитектуры и алгоритма функционирования системы перевода средств между электронными кошельками на основе алгоритмов ZKP**

Проектируемая информационная система должна обладать набором компонентов и функций, позволяющих эффективно проводить транзакции между участниками сети на основе алгоритмов ZKP. Можно разделить функционал между сущностями системы: оператором, пользователем и получателем переводов.

Оператор системы это сущность, которая создается в единственном экземпляре, во время создания для данной сущности создаются необходимые сертификаты, при помощи которых в дальнейшем можно будет инициировать выполнение смарт-контракта в блокчейн сети, а также проводить регистрацию пользователей и получателей переводов. Также оператор системы должен иметь возможности создания категорий кошельков. Категория кошелька это сущность, которая обладает следующими атрибутами: название, id, дневной лимит, месячный лимит. Работа с категориями должна быть организована в виде смарт-контракта для того, чтобы вся информация о созданных категориях хранилась в Hyperledger Fabric леджере. Также оператор должен иметь возможность просматривать всех зарегистрированных пользователей, все созданные им категории кошельков, а также просматривать информацию о пользовательских кошельках.

Пользователь системы должен иметь возможность авторизации в системе для дальнейшей возможности проведения транзакций. А также пользователь должен иметь возможность создания кошельков определенной категории, и, при необходимости должен предоставлять дополнительную информацию для подтверждения права относиться к выбранной категории.

Получатели переводов должны иметь возможность создания кошельков определенной категории, а также возможность изменять в них дневной или же месячный лимит на переводы.

Были определены объекты данных (документы, программы и базы данных, инструменты и материалы).

Таблица 2.1 – Список задач, действующих лиц, объектов данных и показателей эффективности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № задачи | Название задачи | Список действий, составляющих решение задачи | Участник, составляющий решение задачи | Объекты данных |
| 1 | Регистрация пользователей | Генерация пользовательских атрибутов и ключей и дальнейшая проверка их УЦ | Оператор системы, | Распределенный реестр пользователей |
| 2 | Создание категории кошелька | Ввод данных для создания категории кошелька | Оператор системы | Распределенный реестр категорий кошельков |
| 3 | Создание кошелька | Ввод данных для создания кошелька | Пользователь, блокчейн-сеть | Распределенный реестр пользовательских кошельков |
| 4 | Выпуск сертификатов | Получение пользовательских данных, их шифрация УЦ | Блокчейн-сеть | Распределенный реестр сертификатов |
| 5 | Проведение транзакции | Подтверждение пользовательских сертификатов, а также выполнения ZKP алгоритма | Пользователь, блокчейн-сеть | Распределенный реестр транзакций |

Далее были построены упрощенная модель бизнес-процесса, а также усложненная модель бизнес-процесса (BPMN-диаграмма).

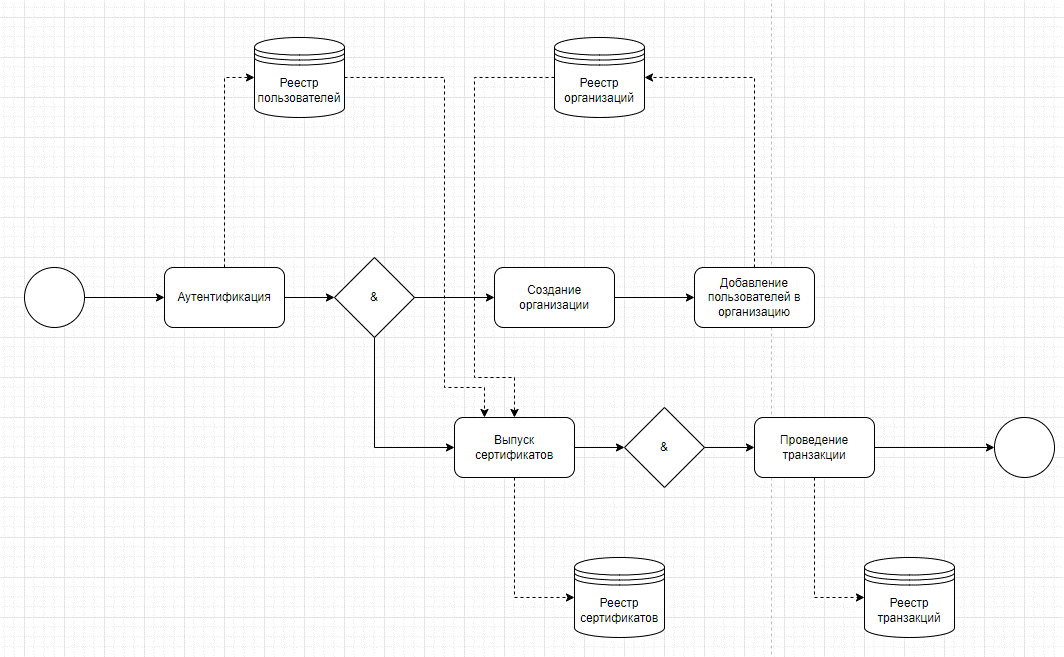


Рисунок 2.8 – Упрощенная модель бизнес-процесса

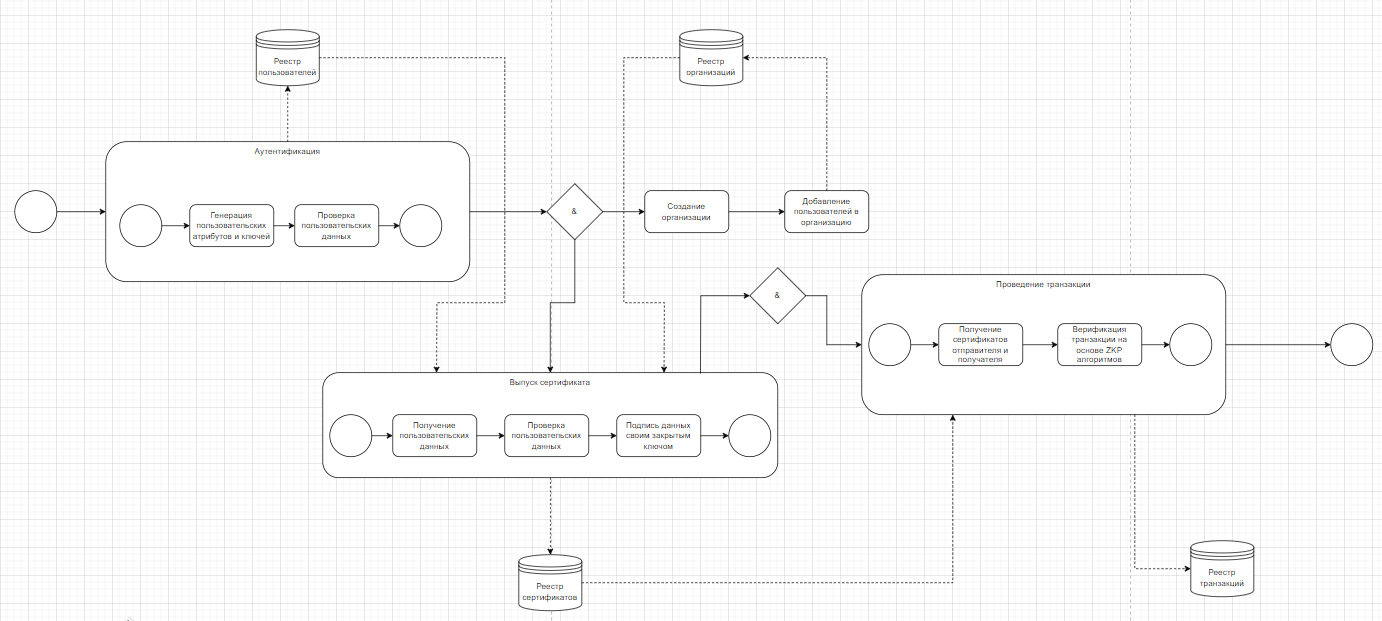


Рисунок 2.9 – Усложненная модель бизнес-процесса (BPMN-диаграмма)

## **Выводы к разделу 2**

В данном разделе была разработана идея архитектуры и внутренней структуры предполагаемого программного обеспечения. Был проведен анализ внешних компонентов приложения, включая взаимодействие с пользователем, обмен данными между различными системными блоками и последовательность событий при различных действиях пользователя. Кроме того, были выделены основные сущности приложения, их характеристики и взаимосвязи.

Для визуального представления результатов проектирования использовались различные нотации, включая DFD для потоков данных, IDEF0 для процессов взаимодействия с пользователем и BPMN для бизнес-процессов

Этот этап подготовки позволил четко определить основные принципы и механизмы функционирования будущего приложения. Теперь, имея четкое представление о его структуре и логике работы, можно приступать к разработке программного продукта.

# 3 РАЗРАБОТКА СИСТЕМЫ ПЕРЕВОДА СРЕДСТВ МЕЖДУ ЭЛЕКТРОННЫМИ КОШЕЛЬКАМИ НА ОСНОВЕ АЛГОРИТМОВ ZKP

В данном разделе пояснительной записки рассматривается полный процесс разработки системы перевода средств между электронными кошельками на основе алгоритмов ZKP. Данный раздел содержит в себе: Выбор и обоснование инструментальных средств разработки проектируемой системы; разработка запросов интерфейса клиент-серверного взаимодействия; описание API разрабатываемой системы и тестирование программных модулей.

## **3.1 Выбор и обоснование технологических средств разработки системы перевода средств между электронными кошельками на основе алгоритмов ZKP**

### **3.1.1 Платформа с технологией распределенного реестра Hyperledger Fabric**

Платформа Hyperledger Fabric (HLF)– это распределенная блокчейн-сеть, которая состоит из различных модулей или компонентов, которые устанавливаются на узлы сети. Это платформа обладает открытым исходным кодом, а также основная её область применения – корпоративные сети. Конфигурацию компонентов можно настроить в зависимости от потребностей разрабатываемой информационной системы, что обеспечивает универсальность для применения данной технологии в различных областях, таких как здравоохранение, финансовые операции, системы голосования.

Также в Hyperledger Fabric имеется возможность написания смарт-контрактов на языках программирования общего назначения, таких как Java, NodeJS, Go, что позволяет разработчикам использовать уже известный для них стек технологий, без затрат времени на изучение какого-либо специфического языка программирования.

Благодаря модульности блокчейн-сети Hyperledger смарт-контракты устанавливаются на узлы сети, на которых выполняется вся бизнес-логика приложения, а также хранится состояние распределенного реестра (Ledger Data). Данные смарт-контракты работают как доверенные и распределенные приложения, доверие которых обеспечивается блокчейном а также консенсусом, установленным между одноранговыми узлами сети.

На следующем рисунке представлена общая архитектура сети Hyperledger Fabric.

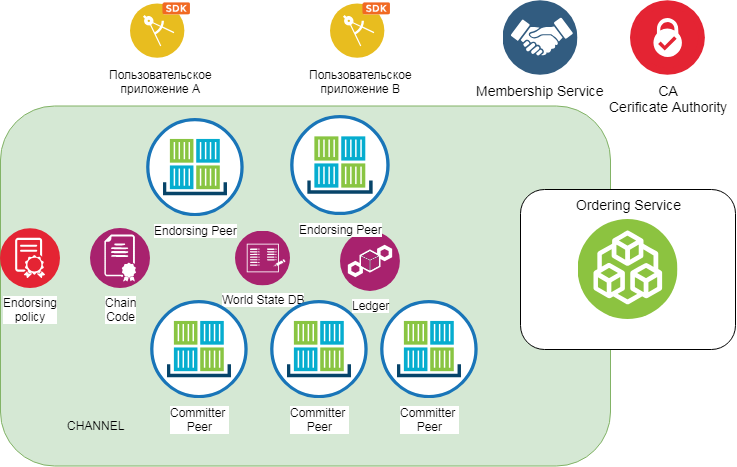


Рисунок 3.1.1 – Общая архитектура сети Hyperledger Fabric

Любой пользователь в Hyperledger Fabric сети сперва должен быть зарегистрирован, а также у него должны быть сгенерированы сертификаты, то есть он должен иметь возможность подтвердить свою Identity. Identity – цифровой идентификатор пользователя, который содержит в себе пользовательские атрибуты. Также все идентификаторы должны быть инкапсулированы в цифровом сертификате X509. Чтобы Identity в дальнейшем можно было проверить, она должна исходить от доверенного органа. В Hyperledger Fabric таким доверенным органом является Membership Service Provider (далее MSP). По умолчанию, в MSP используются сертификаты X509, также основанные на архитектуре ключевой иерархической модели Public Key Infrastructure (PKI).

### **3.1.2 Язык программирования Java**

Java – один из языков программирования общего назначения или универсальный. Это означает, что данный язык можно применять в совершенно разных прикладных областях. Также данный язык программирования кроссплатформенный, что позволяет писать приложения под разные устройства и архитектуры, позволяет разработать мобильные приложения, десктопные, а также веб-приложения. Кроссплатформенность достигается благодаря механизму виртуализации Java Virtual Machine (далее JVM), при помощи которого написанный код сначала компилируется в байт код, затем создаётся файл класса с типом .class, а после чего данный класс интерпретируется виртуальной машиной Java. И благодаря такому механизму, написанный один раз класс может запускаться на любой операционной системе или платформе.

Также преимуществом языка программирования Java является объектно-ориентированность. И всё взаимодействие происходит через объекты, которые в дальнейшем можно будет легко расширять и масштабировать.

### **3.1.3 Фреймворк Spring и расширение Spring Boot**

Spring – это фреймворк для языка программирования Java. Данный фреймворк предназначен для того, чтобы облегчить разработку приложений на Java, поскольку обладает уже готовым набором инструментов, написание которых с нуля заняло бы большое количество времени. Spring является модульным, то есть может состоять из отдельных компонентов (модулей), и поэтому при конфигурации проекта необходимо тщательно продумывать какие компоненты необходимо использовать.

Spring Boot – это модуль, который можно подключить к проекту Spring. Основной целью Spring Boot является упрощение этапа конфигурации Spring. При помощи данного компонента можно ускорить процесс управления зависимостями, поскольку он неявно упаковывает их в Starter-пакеты, и данные наборы дескрипторов зависимостей в дальнейшем можно без особых проблем добавлять в проект.

### **3.1.4 Язык программирования Go**

Go – компилируемый язык программирования, который имеет открытый исходный код, а также обладает таким важным достоинством, как многопоточность. Поскольку данный язык программирования компилируемый, то исходный код преобразуется в машинный при помощи компилятора, который поддерживается UNIX-подобными системами.

В языке Go имеется строгая статическая типизация, а также механизмы безопасности, например управление памятью. Синтаксис этого языка очень простой и легко читается, что может позволить разным разработчикам работать над одним проектом, не заполняя множество документации, пояснений к тому или иному модулю.

Также данный язык очень популярен при написании смарт-контрактов в Hyperledger Fabric, есть официальная SDK для языка Go, при помощи которой можно запросто интегрировать разработанный смарт-контракт с блокчейн-сетью.

### **3.1.4 Платформа контейнеризации приложений Docker**

Docker – это платформа, которая позволяет контейнеризовать (упаковать в контейнер) приложение и всё его окружение, включая зависимости, а затем запустить в целевой системе. Полученный контейнер приложения является изолированным как от операционной системы, так и от других приложений.

Для полноценной работы, в Docker существует несколько сущностей, таких как: Docker Image (неизменяемый образ, шаблон приложения, который содержит исходный код определенной зависимости или библиотеки); Dockerfile (конфигурационный файл, который содержит всю необходимую информацию по настройке контейнера приложения); Docker Container (это уже запущенное программной обеспечение, которое основывается на образах и следует конфигурации, прописанной в конфигурационном файле).

В блокчейн-сети Hyperledger Fabric, все необходимые компоненты сети представлены в виде образов Docker, соответственно использование Docker является неотъемлемой частью при разработке системы перевода средств между электронными кошельками на основе алгоритмов ZKP. Каждый узел сети представляется Docker-контейнером, в котором содержатся все необходимые компоненты для функционирования приложений.

### **3.1.5 Система контроля версий Git**

Разработка комплексной и сложной информационной системы подразумевает долгий и комплексный процесс, во время которого выполняется постоянное внесение правок в код, добавление новой функциональности, или добавление новых внешних зависимостей. И контролировать процесс разработки может быть довольно трудно, в связи с этим была использована система контроля версий Github.

Постоянная выгрузка проекта в репозиторий позволит вернуть проект в состояние до изменения кода или добавления новых компонентов или функциональности, которые породили какие-либо конфликты. Системы контроля версий очень сильно помогают при разработке комплексных программных продуктов, поскольку можно легко просматривать изменения от версии в версию и отслеживать конфликтные места, которые привели к каким-то ошибкам.

### **3.1.6 Документирование REST API при помощи Swagger**

Создание документации вручную для API разработанного приложения иногда может быть довольно затратным по времени процессом, особенно когда программа предполагает наличие большого числа компонентов, и, соответственно запросов.

В связи с этим было решено использовать инструмент автоматического документирования Spring Boot API – Swagger. Данный фреймворк предоставляет множество инструментов, существенно упрощающих процесс создания интерактивной документации API.

Swagger автоматически генерирует документацию для всех доступных в приложении endpoint’ов, форматов запросов/ответов и любых других дополнительных деталей, которые указывает разработчик. Это позволяет существенно ускорить процесс разработки и минимизировать риск возможных ошибок.

## **3.2 Инструментальные средства**

### **3.2.1 NetBeans**

...

### **3.2.2 Visual Studio Code**

...

### **3.2.3 Docker Desktop**

...

## **3.3 Разработка API**

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Mao, W. (2003). Modern Cryptography: Theory and Practice. Prentice Hall.
2. Goldwasser, S., Micali, S., and Rackoff, C. (1989). The Knowledge Complexity of Interactive Proof Systems. SIAM Journal on Computing, 18(1), pp. 186–208.
3. Blum, M., Feldman, P., and Micali, S. (1988). Noninteractive Zero-Knowledge and Its Applications. Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, pp. 103–112.