Пермский филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Образовательная программа бакалавриата «Программная инженерия»

ОТЧЕТ
по преддипломной практике

Выполнил студент группы ПИ-20-2

Перминов Николай Александрович
(Фамилия, Имя, Отчество)

_____
(подпись)

Проверил
Руководитель практики
от НИУ ВШЭ – Пермь:

к.ф.м.н. доц.
(должность)

Плаксин Михаил Александрович
(Фамилия, Имя, Отчество)

_____ _____
(оценка) (подпись)

_____
(дата)

Пермь, 2024

АННОТАЦИЯ

Автор: студент НИУ ВШЭ Перминов Николай Александрович, Кафедра информационных технологий в бизнесе.

Тема работы: Разработка архитектуры и информационных подсистем системы краткосрочной аренды через вендинговые аппараты.

Вид выпускной квалификационной работы: проектная.

В связи с отсутствием программных продуктов позволяющих реализовать краткосрочную аренду пледов через вендинговые аппараты целью этой работы является разработка модулей системы краткосрочной аренды пледов через вендинговые, а именно модулей клиентского обслуживания, аутентификации, взаимодействия с вендинговыми автоматами. Обоснованием разработки является создание стартапа ООО “ГЭТ Э БЛАНКЕТ”, реализующего сервис посуточной и почасовой аренды пледов. Система реализуися с использованием клиент-серверной архитектуры, а именно микросервисно-микроядерной архитектуры.

В первой главе рассматривается технологические возможности реализации, возможности вендинговых автоматов, существующие подходы и программные продукты для автоматизации процесса краткосрочной аренды, изучение программных, аппаратных и архитектурных возможностей для разработки продукта, проведен функционально-стоимостной анализ с целью формирования требований к разрабатываемому программному продукту. Во второй главе описывается разработка архитектурного решения, выбор программных и аппаратных решений. В третьей главе указывается процесс разработки модулей, автоматизации развертывания и тестирования системы.

Работа состоит из 3 глав, включает 72 стр. основного текста, 33 рис., 5 табл., 39 источн. и 7 прил.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Экономика совместного потребления (шеринговая экономика) обеспечивается за счет технологий, с помощью которых люди могут обмениваться различными товарами и услугами. Ввиду бурного роста рынка шеринга и краткосрочной аренды [1], было проведено исследование рынка и перспектив продута. Таким образом, было определено, что наиболее перспективным будет направление развития краткосрочной аренды. Чтобы сделать данный бизнес рентабельным потребовалось сократить расходы на содержание точек раздачи, поэтому было принято решение использовать вендинговые аппараты.

Вендинговые аппараты получили широкое распространение ввиду нескольких факторов [2–4]:

— возможность установки в местах, где невыгодна или невозможна установка классических точек продажи;
— низкие эксплуатационные расходы;
— малая стоимость установки, по сравнению с организацией классической точки продаж.

Таким образом, использование вендинговых аппаратов для реализации краткосрочной аренды стало наиболее выгодным решением, но использование вендинговых аппаратов сопряжено с трудностями того, что аренда не является типичной бизнес-моделью для вендинга и реализуется очень редко, чаще с использованием разработанных специально для этого моделей [5]. Для контроля работы вендинговых аппаратов используются различные программно-аппаратные комплексы, относящиеся к типу систем управления мелкорозничной торговлей [6–10]. Однако существующие программные и аппаратные системы не позволяют реализовать систему аренды через вендинговые аппараты. Это обуславливает целесообразность разработки системы, которая бы позволила реализовать краткосрочную аренду пледов через вендинговые аппараты.

Следовательно, объект исследования – процесс краткосрочной аренды пледов через вендинговые аппараты, а именно организация автоматической выдача

и приема пледов. Предмет исследования – программная система обеспечения работы вендинговых аппаратов для сдачи пледов в краткосрочную аренду.

Программная система разрабатывается в интересах стартапа, сутью бизнес-модели которого является многоразовая сдача пледов. После каждого использования плед требуется изъять из автомата и отправить в химчистку. Поэтому в системе также требуется реализация модуля технического обслуживания с возможностью изъятия грязных пледов и выкладывания чистых. Система также должна позволять производить комплекс действий для реализации краткосрочной аренды. Похожими системами обладают сервисы шеринга самокатов [11, 12].

Исходя из этого был сделан вывод, что такая система слишком сложна для реализации одним человеком, таким образом, цель работы – разработка архитектуры системы и ее модулей реализующих взаимодействие с клиентами, аутентификация клиентов через номер телефона, а также модуля взаимодействия системы с множеством вендинговых аппаратов.

Для выполнения работы и достижения обозначенной выше цели требуется решить следующие задачи:

1) Формирование требований к разрабатываемому продукту:
    1.1) Формирование концепции продукта, а именно обоснование необходимости, анализ выгод от реализации, определение границ и ограничений реализации, определение пользователей системы, определение сценариев использования системы, определение требований пользователей к системе, анализ нормативных документов РФ.
    1.2) Определение функциональных требований.
    1.3) Определение нефункциональных требований.
    1.4) Формирование технического задания в соответствии с ГОСТ 19.201-78 [13].
2) Проектирование разрабатываемой системы:
    2.1) Определение архитектуры разрабатываемой системы.
    2.2) Определение программных решений, используемых в ходе разработки системы.
    2.3) Формирование концепции визуального облика системы.

2.4) Определение средств разработки и взаимодействия команды.

3) Разработка программного продукта:

3.1) Разработка пользовательского интерфейса, модуля аутентификации клиентов, модуля управления автоматами.

3.2) Формирование промежуточных выпусков программного продукта, а именно разработка алгоритмов автоматизированного развертывания, подготовка и настройка серверов для развертывания продукта, настройка вспомогательного программного обеспечения.

3.3) Проведение тестирования программного продукта.

3.4) Проведение документирования разработанных модулей, а именно написание руководства развертывания системы, описание функций API разработанных модулей в соответствии с нотацией OpenAPI.

4) Проведение опытной эксплуатации:

4.1) Проведение опытной эксплуатации системы в рамках лаборатории, подготовка и установка модуля системы на вендинговый аппарат.

4.2) Проведение опытной эксплуатации системы в рамках тестирования в парках с привлечением потенциальных клиентов.

4.3) Устранение обнаруженных проблем.

Технологии, используемые в разработке, должны быть свободно распространяемые и не должны быть подвержены влиянию санкций. Временные ограничения обусловлены требованием заказчика ООО “ГЭТ Э БЛАНКЕТ”, они не должны быть позже 31 марта 2024 года и должны соответствовать календарному плану проекта (см. приложение А).

На этапе разработки системы отсутствует доступ к вендинговому аппарату. Тестирование производится эмуляцией автомата, тестирование с использованием автомата производится на этапе проведения опытной эксплуатации.

Практическая значимость состоит в возможности применения системы для создания коммерческого проекта, так как реализация описанной выше бизнес-модели невозможна с использованием существующих программных решений, так как без применения автоматизации выдачи и сдачи пледов слишком высоки расходы на содержание точек выдачи и сдачи.

Глава 1. Формирование требований к системе на основе анализа предметной области

Формирование требований к системе – важная часть работы над любым программным продуктом, поэтому в данной главе будут рассмотрены:

— анализ структуры предприятия заказчика,
— анализ проблем, которые испытывает заказчик,
— финансовое обоснование реализации системы,
— анализ конкурирующих решений,
— методы решения поставленной проблемы,
— ожидания пользователей от системы,
— границы реализуемой системы.

Итогом анализа станет проведение функционально-стоимостного анализа, по результатам которого будут сформированы оптимальные функциональные и нефункциональные требования к системе.

1.1 Обоснование реализации системы

В ходе рассмотрения целесообразности реализации программной системы следует начать с рассмотрения заказчика. Система разрабатывается в интересах ООО “ГЭТ Э БЛАНКЕТ”. Данная компания основана студентами НИУ ВШЭ и фактически представляет собой “студенческий стартап”, что позволяет компании участвовать в программах государственных грантов. Цель компании – реализация системы шеринга (краткосрочной аренды) пледов как продукта для повседневного пользования. На февраль 2024 года над проектом работают 7 человек из Перми и Москвы. Финансирование компании осуществляется за счёт государственного гранта Фонда Содействия Инновациям “Студенческий стартап”.

Исходя из вышесказанного, компания занимается арендой пледов, для реализации этого требуются точки продаж, компания хотела бы размещать их в парках и на улицах городов. Но размещение классических точек продаж стоит слишком дорого, по мнению генерального директора Атаевой Сабины Валерьевны, так

как только средняя зарплата продавца в Москве от 70 до 100 тысяч рублей в месяц, без учета дополнительных сборов [14, 15]. Поэтому компания нацелена на снижение расходов на размещение точек, что привело к идее размещения пледов в вендинговых аппаратах.

Для доказательства востребованности продукта, компанией было проведено исследование рынка, которое показало, что потенциальный размер рынка достаточный для реализации проекта, а при высокой востребованности продукта, что позволит привлечь венчурные инвестиции. Предполагаемые общий объем целевого рынка (*TAM*) составляет 20 млрд рублей в год, доступный объем обслуживаемого рынка (*SAM*) составляет 2 млрд рублей в год, достижимый объем рынка (*SOM*) за последующие 7 лет с учетом дополнительных инвестиций 1 млрд рублей в год.

Исходя из построенной бизнес модели (см. приложение Г) рентабельность бизнеса составит 50% в сезон или 25% в год, что оправдывает риски создания нового бизнеса. Также использование уникальной бизнес-модели позволяет претендовать на грантовое финансирование, а также позволяет привлекать венчурный капитал. На данный момент компания существует на средства выделенные грантом Фонда Содействия Инновациям “Студенческий стартап”, на данный момент грант получен в полном объеме (1 млн рублей).

Из вышесказанного можно сделать вывод, что реализация бизнес-модели стартапа без использования автоматизированной цифровой программно-аппаратной системы учета аренды нецелесообразна, так как услуга предоставляемая по такой бизнес-модели стоила бы слишком дорого, что значительно уменьшило бы спрос, и сделало бы компанию убыточной.

Таким образом бизнес процесс, который хотел бы иметь заказчик: клиент пользуясь своим телефоном, берет плед в аренду, получая его через вендинговый аппарат, оплата должна производиться ежечасно или посуточно, возвращение пледа происходит аналогичным образом в любой из вендинговых аппаратов компании; система должна иметь возможность обсечения контроля за обслуживанием автомата: выкладки чистых и сбора грязных пледов. Более подробно бизнес процесс TO BE описан на диаграмме ландшафта (см. рисунок 1.1).

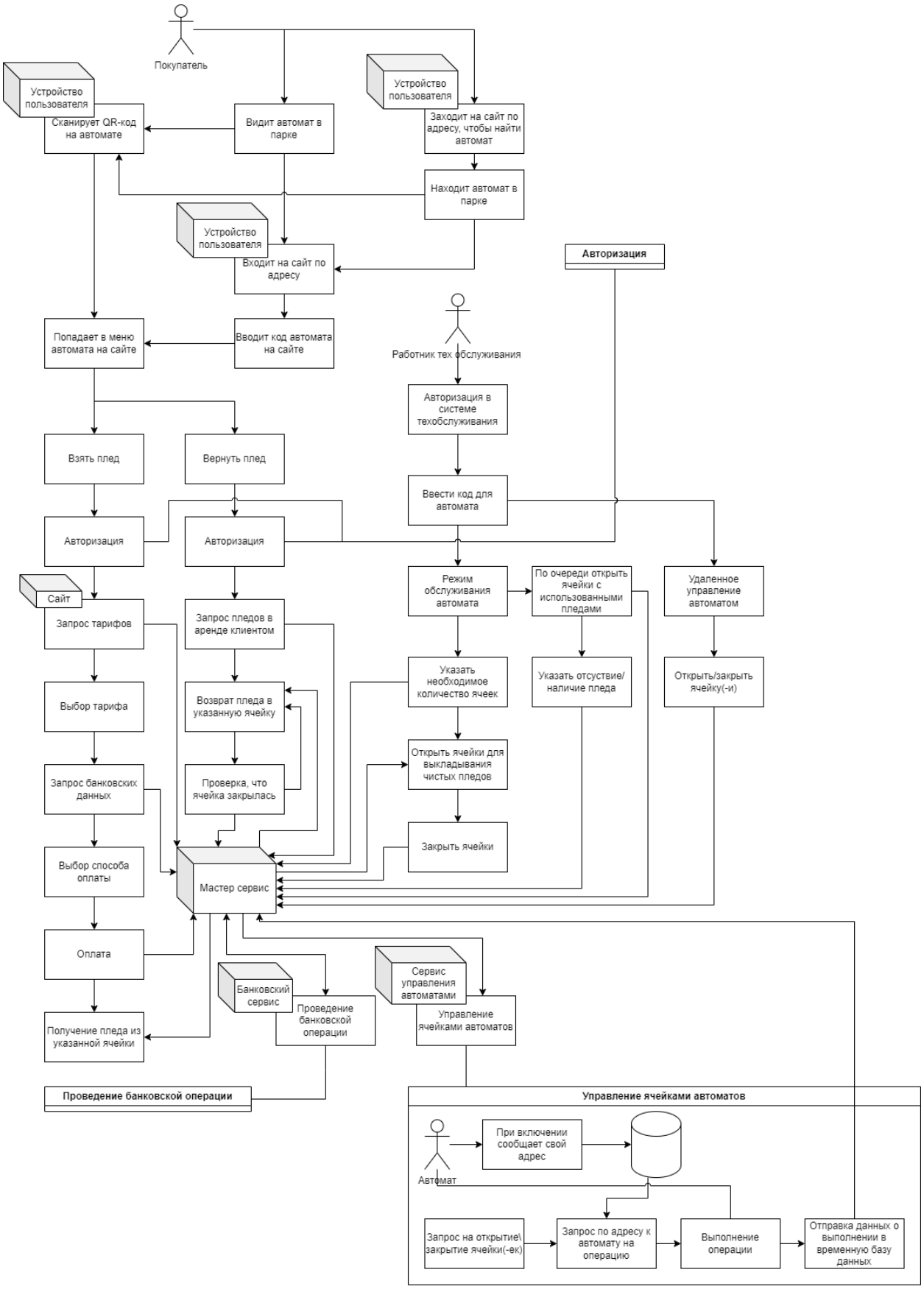

Рисунок 1.1 – Диаграмма ландшафта системы

Для реализации данного бизнес процесса требуется программно-аппаратная система, поэтому далее будут рассмотрены конкурирующие системы учёта вендинговых аппаратов.

1.2 Сравнительный анализ существующих систем

В ходе анализа рынка не было обнаружено систем, которые могли бы в полной мере реализовать бизнес процесс заказчика, поэтому главным критерием возможности применения рассматриваемых конкурирующих систем является возможность адоптации системы под рассматриваемый бизнес процесс.

Далее будут рассмотрены системы компаний:

— OTI Global – международная компания, специализирующаяся на разработке систем бесконтактных платежей в том числе с использованием вендинговых технологий [7];
— Cantaloupe – международная компания специализирующаяся на система управления IoT в сфере торговли [6];
— Новософт – российская компания, специализирующаяся на консалтинге в сфере интеграции программного обеспечения и IT-технологий в существующий бизнес партнеров [8];
— Hubex – российская компания, разработчик одноименной системы управления персоналом и обслуживания [9];
— ZetaSoft – российская компания, специализирующаяся на разработке систем управления и обслуживания в сферах автомобильного и вендингового бизнеса [10].

Далее будут рассмотрены особенности систем каждой из компаний.

OTI Global предлагает комплексную программно-аппаратную систему “OtiMetry System”, включающую в себя программное обеспечение учета продаж, контроллеры и терминалы NFC. Данная система позволяет гибко настроить систему оплаты, берет обеспечение автомата связью с интернетом на себя, имеет документированное API, позволяющее интегрировать ее в общую систему. Но

данная система не может обеспечить отслеживание возвратов пледов, а также не имеет возможности почасовой оплаты [7].

Компания Cantaloupe предлагает программно-аппаратную систему “Self-service Kiosks”, представляющую собой интеграцию кассы самообслуживания в вендинговые технологии. Данная система позволяет интегрировать внешние системы оплаты, обеспечивает отслеживание продажи и аренды, но не включает в себя системы обеспечения обслуживания, а также требует закупки дорогостоящего оборудования для работы (касс самообслуживания) [6].

Продуктом компании Новософт является “NERPA EAM BOX”, данная система предоставляет функции комплексной автоматизации обслуживания автоматов. Преимуществами продукта являются возможность учета выездного обслуживания, наличие внутренней системы управления персоналом, формирование отчетной документации по требованиям установленным на территории РФ. Недостатком является невозможность отслеживания аренды и интеграции платежных систем с почасовой оплатой [8].

Компания Hubex является разработчиком одноименного продукта, представляющего собой единую систему отслеживания процессов компаний ориентированных на предоставление услуг в различных сферах. Преимуществами данной системы являются встроенная гибкая система управления персоналом. Недостатками являются отсутствие системы отслеживания аренды, направленность системы на обслуживание автомата, в связи с чем отсутствие системы оплаты покупок [9].

Продукт Zeta Вендинг предоставляет возможности автоматизации работы компаний в сфере вендингового бизнеса. Система предоставляет возможности гибкого планирования обслуживания автоматов, автоматическое отслеживание продаж и составление на их основе планов обслуживания, создание отчетов по продажам, возможность интегрирования систем оплаты. Недостатками являются отсутствие систем отслеживания аренды, отсутствие систем взаимодействия с клиентом [6].

Для более наглядного сравнения конкурентов была построена сводная таблица 1.1.

Таблица 1.1 – Сравнение конкурентов по критериям

Критерий	**OTI Global**	**Cantaloupe**	**Новософт**	**Hubex**	**ZetaSoft**
Отслеживание покупок	+	+	+	+	+
Возможность интеграции внешней системы оплаты	-	+	-	-	+
Возможность отслеживания аренды	-	-	-	-	-
Система обслуживания автоматов	-	-	+	+	+
Возможность интеграции внешней системы обслуживания автоматов	+	-	-	-	-
Система управления персоналом	-	-	+	+	+
Возможность организации сбора грязных и выкладки чистых пледов	-	-	-	+	+
Возможность почасовой оплаты	-	-	-	-	-
Наличие системы обслуживания клиентов	+	+	-	-	-

Возможность интеграции внешней системы обслуживания клиентов	+	-	-	-	-

Исходя из сказанного выше можно сделать вывод, что существующие системы не могут выполнять бизнес процесс заказчика, так как ни одна из систем не дает возможности отслеживания аренды и возможностей почасовой оплаты.

1.3 Методы решения поставленной проблемы

Использование вендинговых технологий распространено на многие сферы, поэтому логично изучить научные работы на тему применения шеринговых и вендинговых технологий, а также применение технологий в сфере интернета вещей.

В ходе анализа были обнаружены научные работы исследовавшие темы смежные к теме данной работы. В основном работы на данные темы связаны с финансированием и инвестированием в вендинговые технологии [3, 4, 16–19] из чего можно сделать вывод, что программное обеспечение вендиговых аппаратов с научной точки зрения рассмотрено слабо. Работ по использованию вендинга для аренды не было найдено в открытом доступе, но существуют работы, описывающих использование шеринговых технологий в IT-бизнесе [20].

Также хорошо представлены работы описывающие использование облачных технологий и IoT [21–23]. В этих работах чаще рассматривается теоретическая архитектура решения, при этом физическая реализация этой архитектуры как правило не рассматривается. Практическое применение современных архитектурных паттернов представлено в работах [24, 25].

1.3.1 Современное использование вендинговых технологий

Торговые автоматы стали популярны во второй половине 20-го века. В то время в СССР использовались автоматы по продаже напитков и газет [26]. Технологии автоматической торговли широко используются во многих отраслях про-

мышленности. В настоящее время существует большое количество различных типов торговых автоматов, как специализированных для определенных целей, таких как автоматы компании “БериЗаряд”, для аренды внешних аккумуляторов (повербанков), так и универсальных. Классификацию по типу использования можно увидеть на рисунке 1.2.

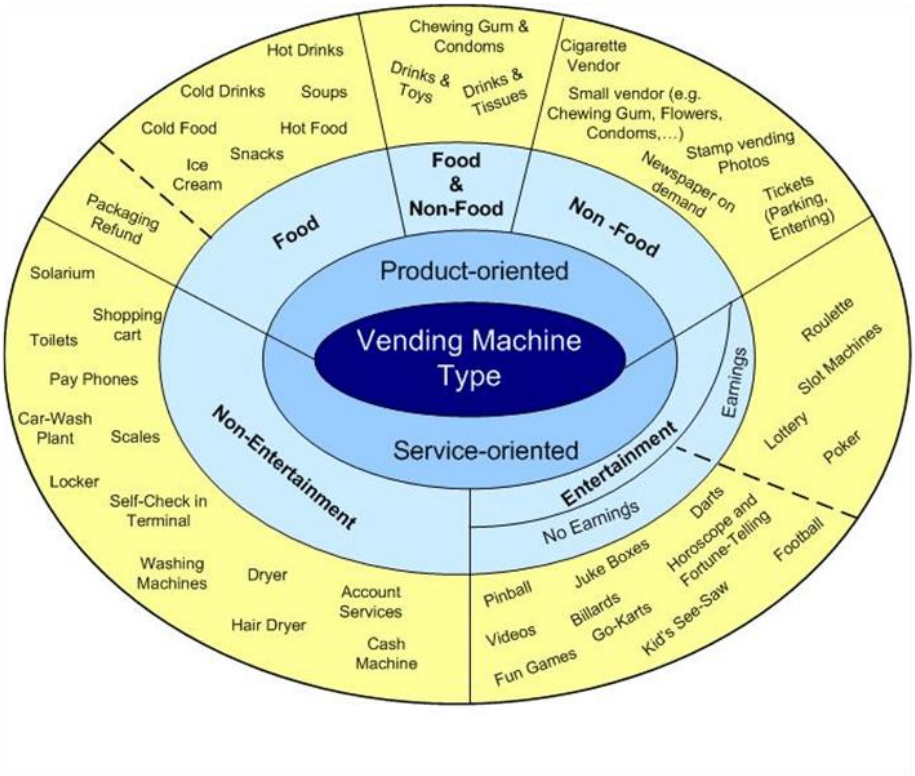

Рисунок 1.2 – Классификация вендинговых аппаратов по типу использования [27]

Основываясь на информации, представленной в статьях “The Commodity Vending Machine” [27] и “Design of a High-Tech Vending Machine” [28], можно сделать некоторые выводы о существовании различных типов торговых автоматов, используемых сейчас. Кроме того, можно сделать вывод, что в настоящее время использование торговых автоматов вышло за рамки торговли и распространилось на такие сферы, как производственный учет, развлечения, парковка и другие.

В соответствии с “Design of a High-Tech Vending Machine” [28], более подробно рассматриваются концепции, связанные с системами распределения торговых автоматов. Оценка этих систем осуществляется с помощью матрицы решений, при этом для этой цели используются такие критерии, как скорость доставки продукта, долговечность использования устройства, надежность и производительность.

Эти идеи полностью реализуются за счет использования модулей, облегчающих общение с мобильными устройствами через Интернет. Применение технологий Интернета вещей расширяет потенциал приложений для торговых автоматов, поскольку позволяет собирать данные для анализа и обработки с целью выявления различных зависимостей, включая предпочтения пользователей и спрос на торговые услуги в определенное время суток. Это также позволяет в режиме реального времени отслеживать состояние торгового автомата и получать уведомления о необходимости пополнения запасов, ремонта и технического обслуживания.

В статье “New Generation Artificial Intelligent Vending Machine System based on LoRaWan IOT Network” [29] описывается реализация этих систем и исследуется потенциал использования искусственного интеллекта (*Artificial Intelligence*) для определения предпочтений пользователей.

1.3.2 Современные подходы к проектированию программных систем

Чтобы определить оптимальный архитектурный дизайн для будущей системы, необходимо провести анализ научных достижений в области проектирования программных систем, использующих технологии Интернета вещей (*IoT*). Для классификации архитектуры систем, многие исследователи разделили проекты на различные абстрактные уровни. Как следствие, для каждого уровня абстракции были созданы различные шаблоны проектирования. Согласно статье “Landscape of Architecture and Design Patterns for IoT Systems” [30], были рассмотрены три уровня абстрактности: высокий уровень абстракции, средний уровень абстракции и низкий уровень абстракции.

Чтобы определить возможные архитектурные решения, которые могли бы обеспечить бесперебойную работу системы, исследователи часто критикуют использование монолитных архитектур в пользу микроядерных и микросервисных архитектур [31].

Рассматривая шаблоны проектирования наиболее часто используемых приложений, которые в настоящее время используют взаимодействие клиент-сервер, мы можем сделать вывод, что многие системы, которые в настоящее вре-

мя находятся в стадии разработки, используют архитектуру микросервисов. По мнению авторов статьи “The Comparison of Microservice and Monolithic Architecture” [32], многие системы, которые изначально разрабатывались как монолитные, в процессе развития часто используют шаблоны микросервисов в процессе длительной разработки. Авторы утверждают, что архитектуры таких систем следует называть микроядерными.

Согласно статье “Development Authentication and Authorization Systems of Multi Information Systems Based REst API and Auth Token” [33] на данный момент наиболее простым и масштабируемым архитектурным подходом является микросервисный.

1.3.3 Перспективное развитие систем совместного использования

Рассматривая тему технологий совместного использования, большинство статей посвящено экономическим показателям систем шеринга и применению такой бизнес-модели [34]. Обсуждаются потенциальные перспективы развития общества за счет использования общих предметов домашнего обихода, а также общие экономические выгоды от развития бизнеса в области технологий совместного использования [35].

Что касается реализации совместно используемых систем, авторы ссылаются на общие шаблоны проектирования систем Интернета вещей (*IoT*). Обращается внимание на необходимость дальнейшего развития технологий мобильной связи, таких как 5G и 6G, которые в настоящее время активно разрабатываются [36]. В статье “Vision, Requirements, and Technology Trend of 6G: How to Tackle the Challenges of System Coverage, Capacity, User Data-Rate and Movement Speed” [37] отмечается постоянное увеличение объема информации, передаваемой через мобильные сети, что приводит к перегрузке сети это существенно влияет на качество шеринговых систем.

1.4 Формирование требований к системе

В рамках формирования требований к системе требуется определить функциональные и нефункциональные требования, а также рамки разрабатываемой

системы. Для этого было проведено 2 глубинных интервью с заказчиком, а также функционально-стоимостной анализ (см. приложение Б).

1.4.1 Функциональные требования

После проведения интервью с заказчиком была поставлена цель реализуемой системы: сделать возможной автоматизированную краткосрочную аренду пледов (шеринга) через вендинговые аппараты, установленные на улицах и в парках города. Для этого необходимо:

— иметь возможность автономно сдать в аренду и вернуть плед,
— иметь возможность проводить обслуживание автоматов,
— иметь возможность собирать диагностические и иные данные с автоматов.

После проведение дополнительного интервью были поставлены задачи, которые должна выполнять система:

— обеспечить автономную и автоматическую аренду пледов через вендинговые автоматы:
  ◦ обеспечить возможность взымания платы с клиентов,
  ◦ обеспечить возможность взятия пледа из автомата,
  ◦ обеспечить возможность возврата пледа в автомат,
  ◦ обеспечить отслеживание пледов находящихся в аренде,
  ◦ обеспечить возможность нахождения ближайшего к клиенту автомата;
— обеспечить возможность технического обслуживания автоматов:
  ◦ обеспечить возможность открытия ячейки автомата,
  ◦ обеспечить возможность изъятия грязных пледов из автомата,
  ◦ обеспечить возможность выкладывания чистых пледов в автомат;
— обеспечить возможность мониторинга:
  ◦ обеспечить возможность просмотра графиков количества взятий пледов в аренду из автомата,
  ◦ обеспечить возможность просмотра графиков количества возвратов пледов из аренды в автомат;

Исходя из задач посаленных заказчиком и изучения структуры компании, описанной в пункте 1.1, были выявлены следующие типы пользователей:

Клиенты коммерческого проекта – люди, гуляющие вечерами, когда на улице начинает холодать, а форма одежды была выбрана на более теплое время суток.

Специалисты технического обслуживания автоматов – специалисты, задачами которых является поддержка рабочего состояния автоматов и выкладка/сбор пледов.

Системные администраторы – специалисты, занимающиеся поддержкой работы программно-аппаратного комплекса.

Аналитики – специалисты, занимающиеся анализом статистики взятий и возвратов пледов за определенный период.

Для того, чтобы формализовать требования к системе из желаний заказчика были построены UML представленные на рисунках 1.3, 1.4, 1.5.

Рисунок 1.3 – Диаграмма прецедентов, описывающая взаимодействие с системой клиентом коммерческого проекта

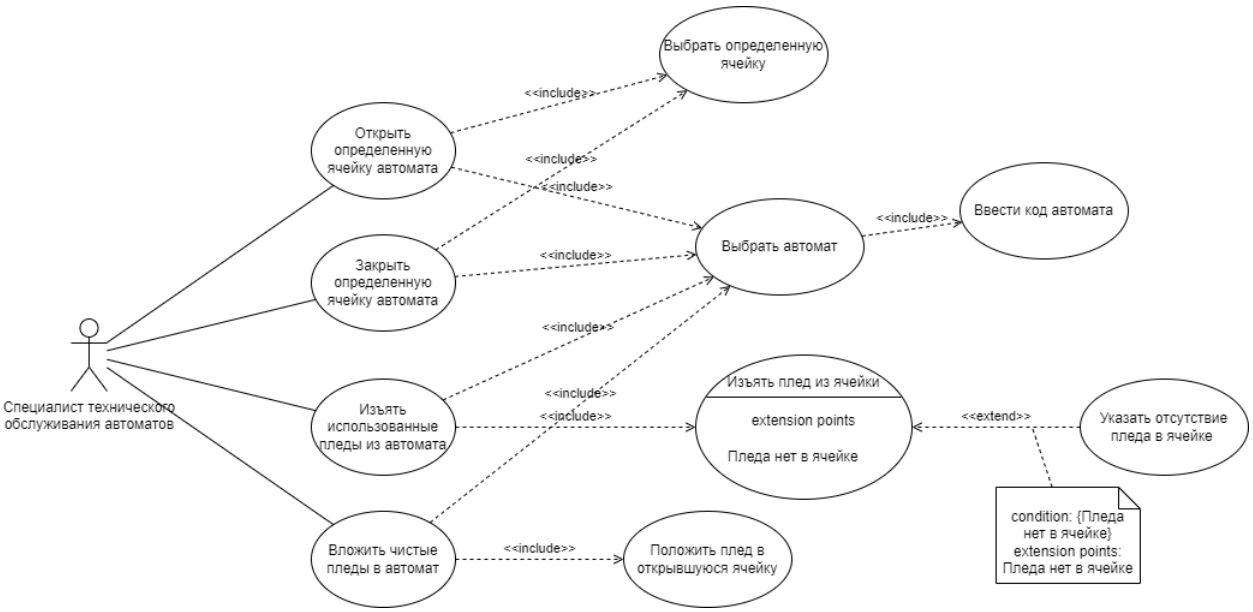

Рисунок 1.4 – Диаграмма прецедентов, описывающая взаимодействие с системой специалистом технического обслуживания

Рисунок 1.5 – Диаграмма прецедентов, описывающая взаимодействие с системой аналитиком и системным администратором

Далее представлены требования пользователей-клиентов коммерческого проекта:

— Система должна отображать номер ячейки автомата при взаимодействии клиента с ней.

— Система должна отображать карту с указанием местоположения автоматов и пользователя.

— Система должна отображать информацию о количестве доступных пледов в автоматах на карте.

— Система должна отображать информацию о пледах, находящихся в аренде пользователем на данный момент, а именно тариф, время пользования, общая цена аренды.

— Система должна отображать данные пользователя (номер телефона) на основной странице.

— Система должна позволять добавлять банковскую карту для оплаты с основной страницы.

— Система должна позволять указывать основную банковскую карту для оплаты.

— Пользователь должен иметь возможность выбрать автомат посредством ввода уникального кода автомата.

— Система должна предоставлять возможность выбора автомата посредством перехода по уникальной ссылке.

— Пользователь должен иметь возможность взять в аренду до 5 пледов.

— Система должна предоставлять возможность закрытия ячейки по указанию пользователя.

— Система должна предоставлять фотографии автомата.

— Система должна отображать список возможных тарифов для аренды пледа.

— Система должна позволять выбрать один из возможных тарифов для аренды пледа.

— Система должна позволять выбрать основную банковскую карту до оплаты тарифа.

Далее представлены требования пользователей-специалистов технического обслуживания автоматов:

— Система должна отображать номер ячейки автомата при взаимодействии специалиста с ней.

— Система должна позволять выбирать автомат один раз для выполнения нескольких операций.

— Система должна организовать изъятие использованных пледов из автомата.

— Система должна позволять отмечать отсутствие пледа в ячейке (данное действие требует подтверждения).

— Система должна организовать выкладывание чистых пледов в автомат.

— Система должна позволять открывать определенную ячейку автомата.

— Система должна позволять закрывать определенную ячейку автомата.

Далее представлены требования пользователей-аналитиков и пользователей- системных администраторов:

— Система должна отображать графики по количеству взятий пледов в аренду из определенного автомата за определенный период.

— Система должна отображать графики по количеству возвратов пледов в определенный автомат за определенный период.

— Система должна иметь возможность экспортировать данные в формате CSV.

1.4.2 Нефункциональные требования

Исходя из требований заказчика были сформулированы следующие требования:

— Система должна позволять блокировать доступ к определенной ячейке автомата, так как требуется иметь возможность отключать сломанные ячейки автомата.

— Система должна проводить авторизацию специалистов технического обслуживания, аналитиков, системных администраторов для обеспечения информационной безопасности.

— Система должна идентифицировать пользователя с помощью номера телефона.

— Система должна подтвердить, что номер телефона пользователя принадлежит ему.

— Открытие ячейки должно происходить не позднее, чем через 1 секунду после оплаты.

— Система должна позволять устанавливать таймеры для операций взятия и сдачи пледа, которые не будут считаться в время аренды.

— Авторизация персонала должна производиться посредством протокола *LDAP*, для обеспечения совместимости с системой учёта персонала.

Также в ходе анализа предметной области были выявлены нормативные акты регулирующие сферы деятельности затрагиваемые рассматриваемой системой. Таким образом, система должна соответствовать федеральным законам РФ: № 152-ФЗ “О персональных данных”, № 395-1 ”О банках и банковской деятельности”, № 115-ФЗ “О противодействии легализации (отмыванию) доходов, полученных преступным путем, и финансированию терроризма”.

1.4.3 Рамки реализации программной системы

Общие рамки реализации системы имеют следующие границы:

— Функциональные границы: автоматизация процесса аренды пледов.

— Географические границы: вендинговые аппараты, выдающие и принимающие пледы. Аппараты находятся в парках города Москва. Автоматы берутся в аренду с ПО от производителя, далее это ПО может быть модифицировано или заменено. Разрабатываемое ПО подключается к арендуемому автомату. Тестирование будет производиться в зависимости от того, удастся ли получить экземпляр вендингового аппарата. Если не удастся, то тестирование будет производиться посредством моделирования реального поведения вендингового автомата. В случае, если вендинговый аппарат получить удастся, то тестирование будет производиться непосредственно на нем. Второй вариант наиболее благоприятен, так как показывает результат test case в реальных условиях.

— Организационные границы: техник обслуживания(забирает/раскладывает пледы по аппаратам, техническое обслуживание), посетители парков.

1.5 Выводы по главе

В данной главе приведен анализ предметной области, приведены обоснования разработки и анализ конкурентов, формализованы требования заказчика:

1) Рассмотрена бизнес-модель заказчика, установлен бизнес процесс требующий автоматизации, сделан вывод о необходимости реализации программной системы.
2) Рассмотрены конкурирующие решения, сделаны выводы о слабых и сильных сторонах конкурентов, выявлены их слабости не позволяющие использование их для реализации автоматизации бизнес процесса.
3) Рассмотрены научные статьи относящиеся к предмету исследования.
4) Рассмотрены существующие методы решения проблемы автоматизации бизнес процесса.
5) Формализованы функциональные и нефункциональные требования, а также установлены рамки реализации программной системы.

Итогом первой главы является техническое задание, утвержденное заказчиком ООО “ГЭТ Э БЛАНКЕТ”, написанное в соответствии с ГОСТ 19.201-78 [13], представленное в приложении В.

Глава 2. Проектирование и формализация архитектуры программного решения

После формализации требований к системе, определения бизнес процессов и условий эксплуатации системы возможно приступить к проектированию системы. В рамках первой главы были рассмотрены существующие научные работы, был проведен функционально-стоимостной анализ, что окажет значительное влияние на формирование архитектурного облика системы.

В рамках этой главы следует:

— определить и обосновать выбор архитектурного дизайна системы, провести сравнительный анализ рассматриваемых архитектурных паттернов;
— распределить бизнес процессы на компоненты системы в соответствии с выбранным архитектурным дизайном и средствами разработки;
— определить и обосновать выбор средств разработки системы, провести общее проектирование компонентов системы, включая проектирование интерфейсов;
— распределить задачи реализации отдельных компонентов системы на команду разработчиков, определить методы управления командой разработки;
— выполнить полное проектирование компонентов системы рассматриваемых в рамках данной работы.

Итогом данной главы станут:

— формализованная архитектура всей системы описанная в соответствии с нотацией “C4 model”;
— макеты интерфейсов системы;
— диаграмма физической модели базы данных в сошествии с нотацией UML;
— полное описание архитектуры компонентов, разработка которых рассматривается в рамках данной работы;
— план коммуникации команды и распределения ответственности.

Для проектирования и формализации архитектуры были выбраны следующие средства:

— графический редактор Draw.io для построения диаграмм, был выбран по причинам: возможность проектирования с использованием шаблонов UML и C4, возможность совместного редактирования, все члены команды имеют опыт пользования данным программным продуктом, полностью бесплатен;

— графический редактор Figma для построения макетов интерфейсов, так как предоставляет возможности представления макетов визуального облика системы как демо приложение, что позволяет аналитикам проводить тестирование интерфейса на фокус-группе; данный редактор позволяет гибко настраивать взаимодействия компонентов макета между собой, что в значительной степени упрощает реализацию макетов виде кода.

2.1 Выбор архитектурного дизайна системы

Рассматривая научные работы в пункте 1.3.2 были выделены несколько научных работ, рассматривающих различные виды архитектурного дизайна с использованием технологии Интернета вещей. Во множестве работ выражался скепсис по отношению к использованию монолитной архитектуры [30–32]. В качестве аргументов приводились: малая возможность к масштабированию, сложность организации распределенных вычислений, низкая по отношению к распределенным архитектурам отказоустойчивость. Поэтому было принято решение использовать микросервисную архитектуру в качестве базового архитектурного дизайна.

Таким образом, далее следует определить архитектурный стиль, которого следует придерживаться в ходе разработки. Архитектурные стили — это высокоуровневые стратегии, которые представляют собой абстрактную структуру для систем [38]. Рассмотрим некоторые из них:

1) Многослойный. Типичный пример — многоуровневый стиль из 3 слоёв: слой представления, бизнес-логика и хранение данных. Плюсы: лёгкий в понимании, тестировании и поддержке. Минус: появление накладных расходов на производительность. Данный стиль требует создания монолитной клиент-серверной архитектуры, поэтому не является предпочтительным.

2) Компонентно-ориентированный. Разделяет функциональность на слои в рамках одной системы, организованной как повторно используемые и слабосвязанные компоненты. Плюсы: гибкость, поддерживаемость, повторное использование. Минус: сложность управления компонентами. Данный стиль требует создания множества универсальных библиотек, что делает его неподходящим для создания легковесных контейнеров, что может потребовать увеличения потребных вычислительных мощностей. Также данный стиль сложен для восприятия программистом, что может создать значительные сложности в рамках разработки начинающими специалистами. По этим причинам данный архитектурный стиль не является приемлемым.

3) Сервисно-ориентированный. Требует разработки ПО как набора взаимодействующих сервисов через сеть. Плюсы: слабая связанность, гибкость и возможность быстрого масштабирования. Минусы: сложность развёртывания, сетевая зависимость. Данный стиль требует разработки отдельных компонентов системы, как отдельных контейнеров, что требует разработки каждого компонента, фактически как отдельной системы, что приемлемо в рамках разработки командой. Сложность развёртывания нивелируется наличием в команде специалиста по поддержке и развёртыванию систем. Данный стиль можно считать приоритетным.

4) Распределённая система. Компоненты на сетевых компьютерах передают сообщения для взаимодействия и координирования действий, чтобы достичь цели. Плюсы: общий доступ, устойчивость к ошибкам и сбоям, возможность масштабирования. Минусы: сложность, сетевая зависимость. Данная архитектура потребовала бы размещения всех компонентов системы на каждом автомате, что усложняет масштабируемость системы, усложняет взаимодействие между автоматами, а также несет риски связанные с качеством сетевого соединения автоматов, поэтому данный архитектурный стиль не является приоритетным.

5) Предметно-ориентированный. Создаёт программные абстракции (модели предметных областей). Плюсы: ограниченные связи, целостность и взаимо-

связь. Минусы: требует глубокого понимания и опыта в рассматриваемой области. Данный архитектурный стиль требует глубокого понимания предметной области, но так как в команде отсутствует специалист по продажам и аренде с использованием вендинговых и шеринговых технологий, а также специалистов реализации систем с использованием паттернов технологий Интернета вещей, данный стиль нельзя считать приоритетным, но можно считать приемлемым.

Исходя из анализа можно сделать вывод, что наиболее подходящей станет сервисно-ориентированная микросервисная архитектура, так как, исходя из критериев требований описанных в техническом задании (см. приложение В), система должна обладать свойствами: способность к масштабированию, гибкость вносимых изменений, стоимость поддержки и развития, защита от внешнего вмешательства.

На основе выбранного дизайна была покроена упрощенная модель архитектуры, представленная на рисунке 2.1, построенная на основе бизнес-процессов описанных на Use Case диаграммах (см. рисунки 1.3, 1.4, 1.5).

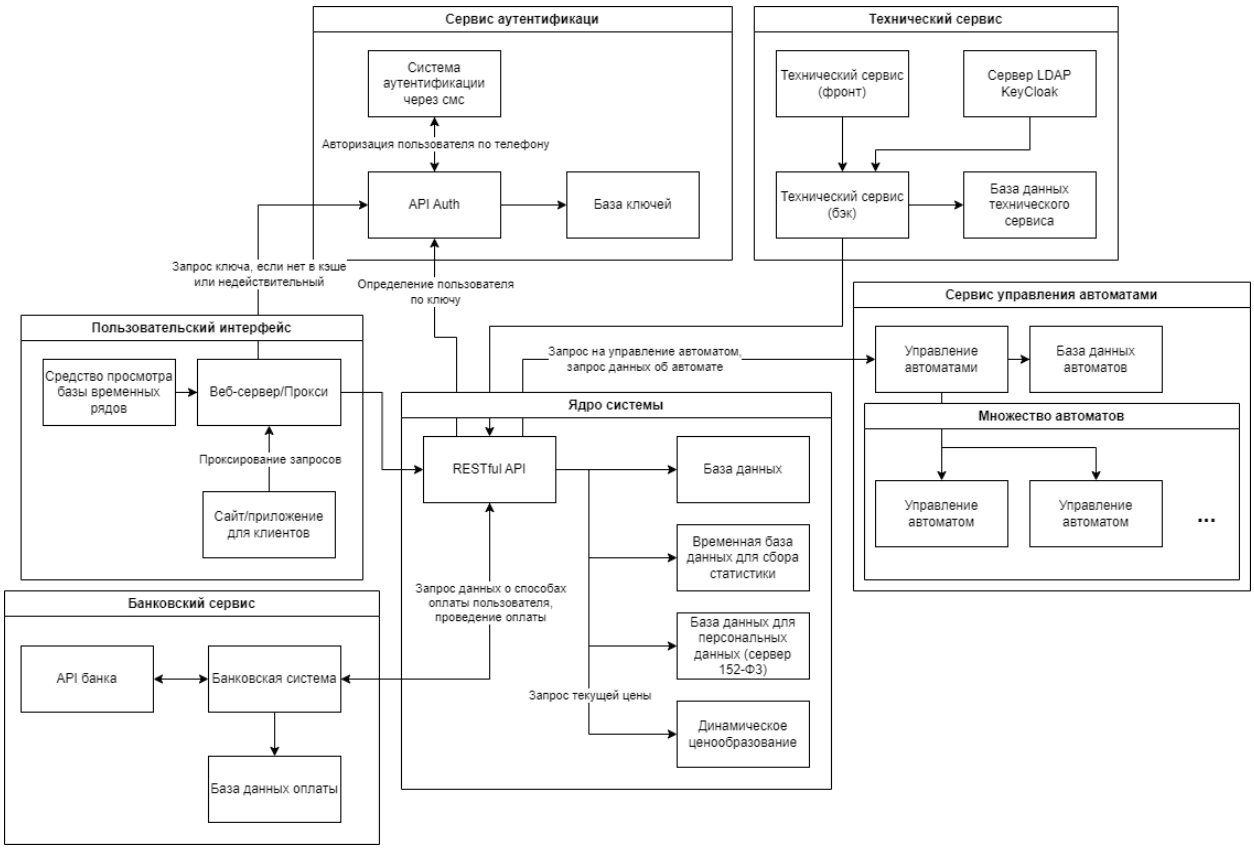

Рисунок 2.1 – Упрощенная модель архитектуры системы

2.2 Определение общей архитектуры системы и выбор средств разработки

Рассматривая средства разработки их следует делить на два типа: средства написания кода и взаимодействия команды, а также программные средства реализации системы (*SDK*). В данной части будут рассмотрены именно средства реализации.Средства написания кода, выбирались каждым участником команды индивидуально и будут рассмотрены в рамках 3 главы, средства взаимодействия команды будут рассмотрены отдельно в пункте 2.3.

Для построения диаграмм архитектуры системы была выбрана нотация C4. Для построения общей модели системы далее будет проведен анализ каждого компонента системы до второго уровня нотации (модели контейнеров).

На рисунке 2.2 представлена диаграмма контекста (диаграмма первого уровня по нотации C4).

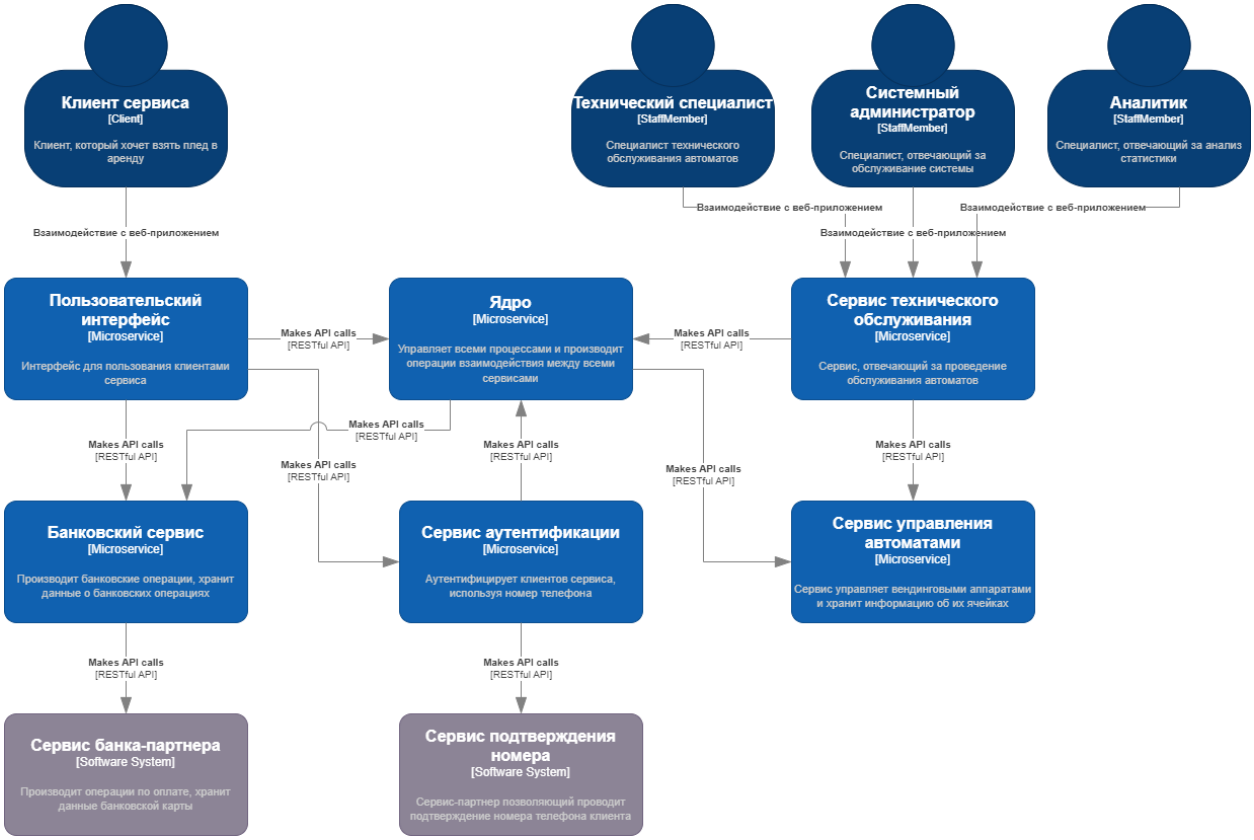

Рисунок 2.2 – Модель контекста

В ходе анализа бизнес процессов были выделены основные компоненты системы, представляющие собой микросервисы, которые требуют взаимодей-

ствия с друг-другом, но выполняют задачи независимо. Из диаграммы видно, что система имеет связи с внешними системами “Сервис банка-партнера” и “Сервис подтверждения номера”.

Для понимания взаимодействия между микросервисами и их внутренними контейнерами были построены диаграммы последовательности, представленные в приложении E.

Далее рассмотрим каждый микросервис в отдельности для более глубоко понимания взаимодействия контекстов.

Первыми следует рассмотреть контейнеры Ядра системы (см. рисунок 2.3), так как это связующее звено системы. Данный микросервис отвечает за отслеживание аренды, передачу задач на другие микросервисы, благодаря наличию данного микросервиса архитектуру можно назвать микроядерной. Данный микросервис состоит из API Server, СУБД общего назначения и Защищенной СУБД. Основная часть микросервиса (“API Server”) будет разработана с использованием библиотек ASP.NET Core, Entity Framework Core для языка C#, так как с этими технологиями знакомы все разработчики, при этом оно обеспечивает достаточную производительность. Использование данного стека позволяет автоматически генерировать документацию в соответствии с OpenAPI, что упростит документирование и взаимодействие разработчиков разных микросервисов системы. Для реализации базы данных будет использовано СУБД PostgreSQL, так как данная СУБД распространяется бесплатно по свободной лицензии, а также у разработчиком есть опыт работы с ней. Для реализации “СУБД временных рядов” была выбрана база данных InfluxDB, так как данная база данных имеет гибкую настройку автоматического удаления устаревших данных, а также к данной СУБД возможно подключение программного решения “Grafana”, позволяющего выполнять все задачи аналитика (см. рисунок 1.5). Реализация отдельной базы данных для персональных данных обусловлена законодательством РФ, которое обязывает хранить персональные данные на серверах, имеющих дополнительные степени защиты, из-за чего их использование обходится дороже, поэтому было принято решение вынести базу персональных данных на отдельный сервер. Реализация – СУБД PostgreSQL Pro входящая в перечень российского ПО.

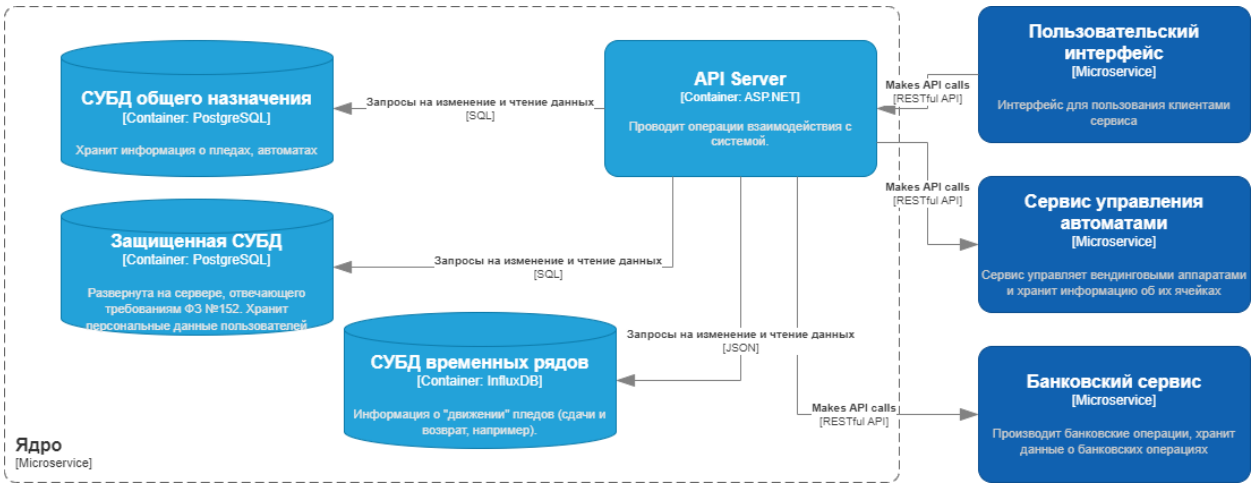

Рисунок 2.3 – Модель контейнеров Ядра системы

Микросервис “Пользовательский интерфейс” состоит из двух контейнеров “Интерфейс системы” и “Веб сервер” (см. рисунок 2.4). Для реализации “Интерфейс системы” выбрано программное решение на основе библиотеки Flutter на языке Dart. Это решение обосновано тем, что данный фреймворк позволяет компилировать приложение под платформы: Web, Android, iOS. Это значительно сокращает потенциальные трудозатраты на разработку отдельных приложений под каждую платформу. Для реализации “Веб сервер” выбрано программное решение nginx, так как данное решение имеет возможность полного проксирования запросов на другую машину и разработчики уже имеют опыт его использования.

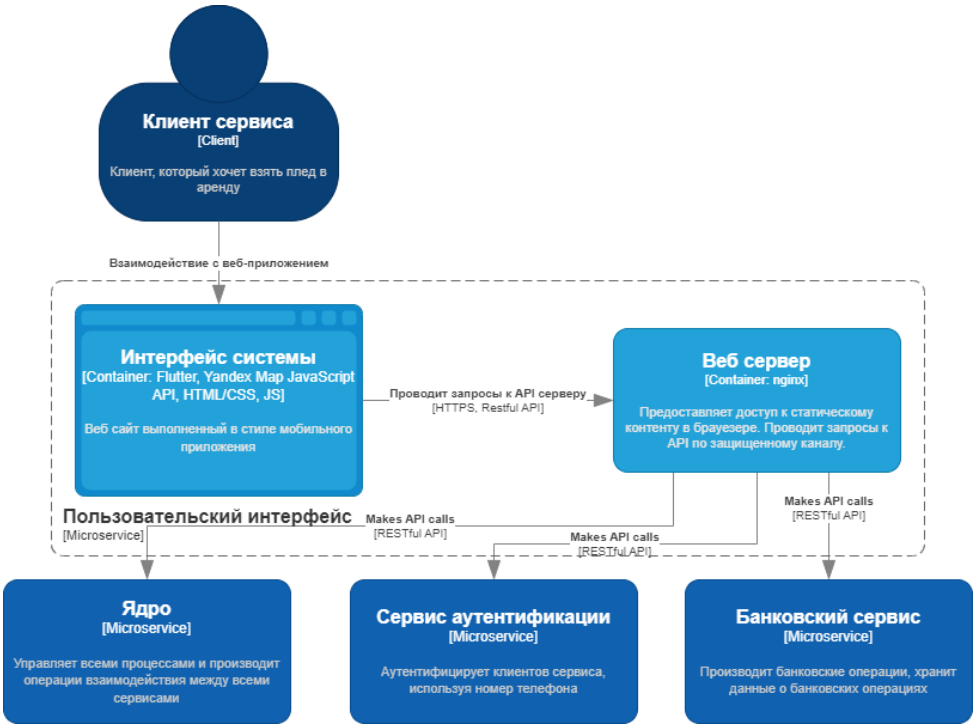

Рисунок 2.4 – Модель контейнеров Пользовательского интерфейса

Сервис аутентификации ответственен за выдачу JWT-токенов для интерфейса системы входящего в состав микросервиса “Пользовательский интерфейс”, также он осуществляет взаимодействие с внешней системой “Сервис подтверждения номера” (см. рисунок 2.5). Основная часть модуля (“API Server”) будет представлена в виде RESTful API и разработана с использованием библиотек ASP.NET Core, Entity Framework Core для языка C#, по причинам описанным ранее. Для реализации “СУБД сервиса аутентификации” будет использовано СУБД PostgreSQL.

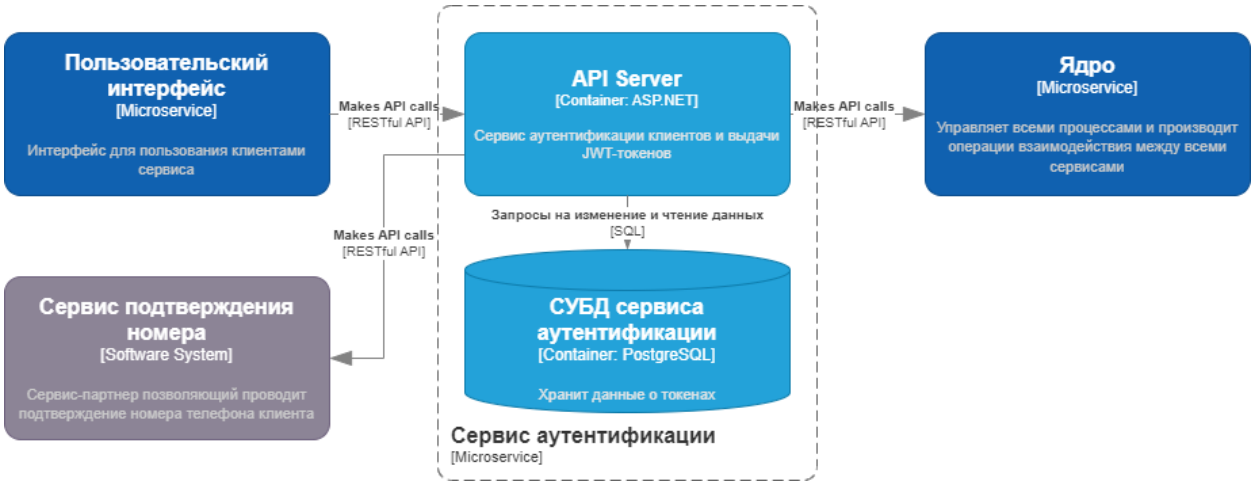

Рисунок 2.5 – Модель контейнеров Сервиса аутентификации

Банковский сервис ответственен за отслеживание и проведение банковских операций, сохранение и агрегирование информации о способах оплаты пользователей, также он осуществляет взаимодействие с внешней системой “Сервис банка-партнера” (см. рисунок 2.6). Основная часть модуля (“API Server”) будет представлена в виде RESTful API и разработана с использованием библиотек ASP.NET Core, Entity Framework Core для языка C#, по причинам описанным ранее. Для реализации “СУБД банковского сервиса” будет использовано СУБД PostgreSQL.

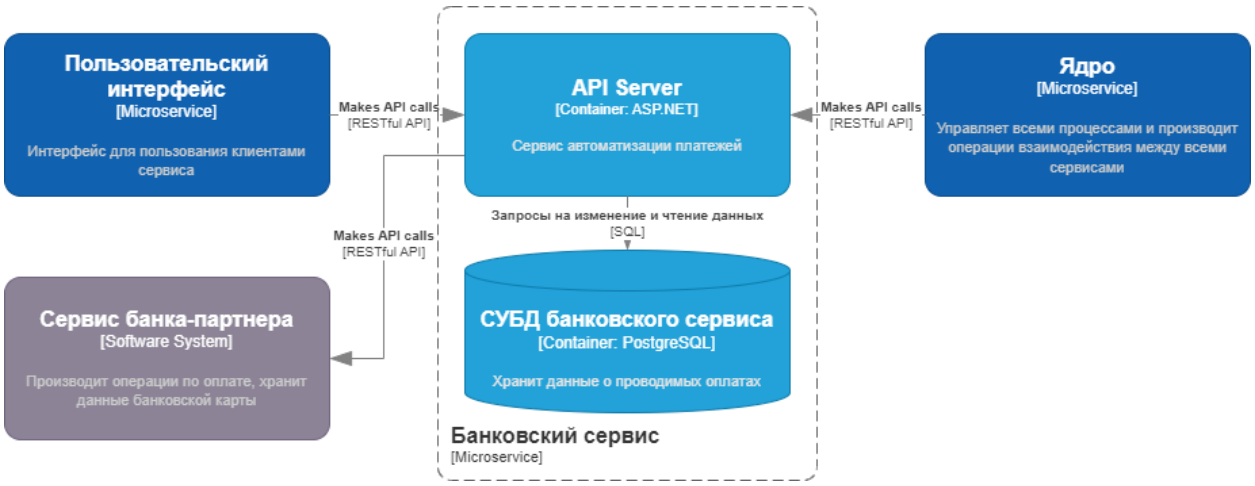

Рисунок 2.6 – Модель контейнеров Банковского сервиса

Технический сервис ответственен за техническое обеспечение работы бизнес-модели, включает в себя управление функциями выкладки и сбора пледов и другими задачами технического обслуживания (см. рисунки 1.4, 1.5). Модель контеинеров представлена на рисунке 2.7. Для реализации клиентской части (контейнер “Интерфейс системы”) будет использован язык TypeScript благодаря его статической типизации, что позволит избежать многих ошибок компиляции и позволит более четко определять структуры данных. В качестве целевого фреймворка будет использован React, так как разработчик имеет широкий опыт использования данного фреймворка, написание на фреймворке специально разработанном для написания веб-приложений ускоряет разработку и быстродействие итоговой системы (в сравнении с Flutter), а невозможность компиляции данного контейнера в мобильное приложение не является критичным в случае использования техническими специалистами. Для реализации “Веб сервер” выбрано программное решение nginx, так как данное решение имеет возможность полного проксирования запросов на другую машину и разработчики уже имеют опыт его использования. В качестве сервиса аутентификации принято решение использовать KeyCloak, так как это готовое инфраструктурное решение, позволяющее интегрировать в систему в будущем другие внешние компоненты использующие протоколы аутентификации OpenID и SAML2, при этом KeyCloak в полной мере отвечает поставленным требованиям к безопасности и поддерживает интеграцию с протоколом LDAP. “API Server” будет разработана с использованием библиотек ASP.NET Core, Entity Framework Core для языка C#, что позволит сохранить едино-

образие системы, так как все API сервера взаимодействующие с пользовательскими интерфейсами системы написаны на C#. Для реализации “СУБД технического сервиса” будет использовано СУБД PostgreSQL. В качестве аналитической системы используется программный продукт Grafana, так как это свободное программное обеспечение, позволяющее выполнять все задачи поставленные аналитикам (см. рисунок 1.5), а также имеет интеграцию авторизации через протокол SAML2, что позволит проводит авторизацию пользователя через KeyCloak.

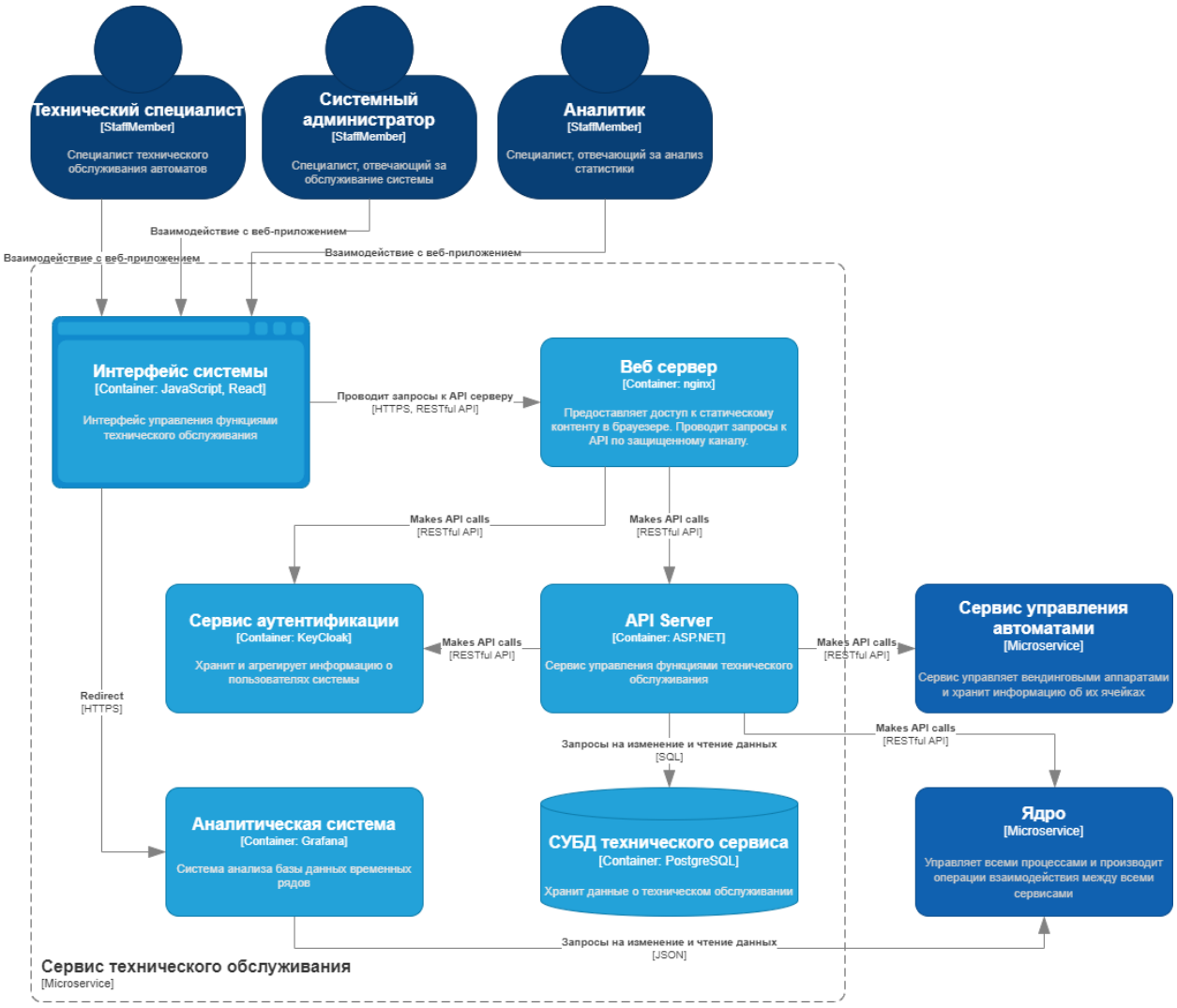

Рисунок 2.7 – Модель контейнеров Технического сервиса

Сервис управления автоматами состоит из 3 компонентов (см. рисунок 2.8): API Server (менеджер управления автоматами), Служба управления автоматами (устанавливается на каждый автомат) и СУБД правления автоматами. “API Server” представляет собой Restful API, взаимодействующий с автоматами посредством API запросов, по внутренней виртуальной сети OpenVPN. Разработка модуля бу-

дет с использованием библиотек ASP.NET Core, Entity Framework Core для языка C#. Для реализации “Базы данных автоматов” будет использовано СУБД PostgreSQL. “Служба управления автоматом” представляет собой программно-аппаратный комплекс, включающий в себя минимальный RESTful API взаимодействующий с “API Server”, модули взаимодействия с системой для управления автоматом. Реализован будет с использованием библиотеки Flask для языка Python, так как данный язык отличается наибольшей мультиплатформенностью и гибкостью из фреймворков известных разработчикам.

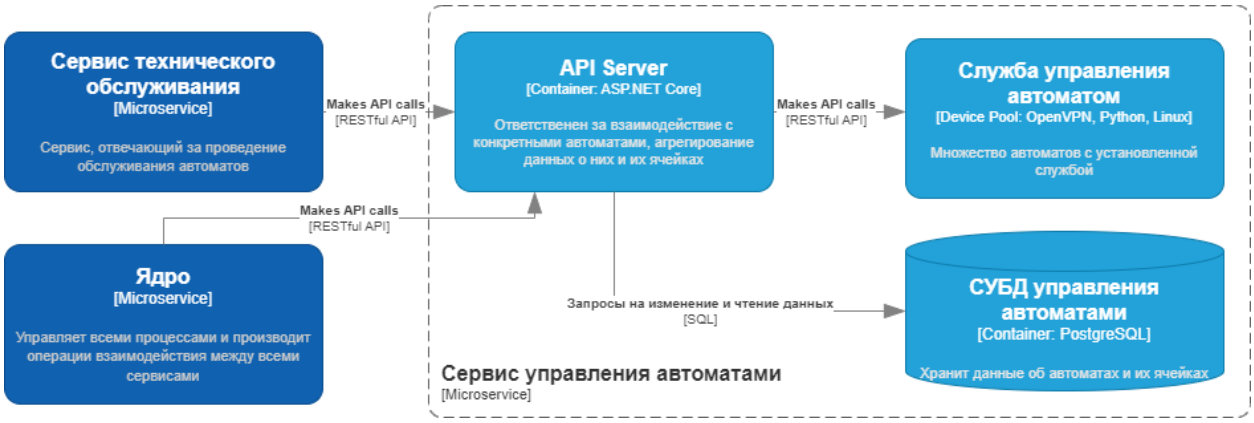

Рисунок 2.8 – Модель контейнеров Сервиса управления автоматами

Для обеспечения наиболее простого формирования контейнеров было принято решение об использовании Docker контейнеров, это в значительной степени упростит сборку компонентов системы. Для агрегирования отдельных контейнеров в микросервисы, а микросервисы в единую систему было принято решение использовать Kubernetes.

2.3 Методы управления командой разработки и распределение задач

Еще в ходе анализа предметной области было выявлено, что реализация всей системы одним разработчиком неприемлемо из-за объема работы. Таким образом, появляется потребность в организации работы команды разработчиков.

Следует выделить следующие цели, которые должны обеспечивать методы управления команды:

1) обеспечение равных нагрузок на разработчиков и обеспечение предсказуемости сроков выполнения задач;
2) обеспечение коммуникации разработчиков по спорным вопросам;
3) обеспечение контроля за равным качеством продукта в разных частях системы;
4) обеспечение связуемости микросервисов разрабатываемых разными разработчиками.

Таким образом, следует определить средства коммуникации команды, которые бы обеспечили выполнение данных целей.

Для обеспечения равных нагрузок в условиях плохой предсказуемости трудовой доступности членов команды и плохой предсказуемости уровня трудозатрат задач (в виду малого опыта разработчиков) было принято решение об использовании Agile подхода к разработке (см. рисунок 2.9) с временем спринта в одну неделю.

Рисунок 2.9 – Схема жизненного цикла по SCRUMBoK [39]

Планирование и ретроспектива проходят во время еженедельного совещания по проекту, на котором присутствуют все участники проекта, это позволяет также решить вторую цель.

Обеспечение контроля за равным качеством продукта осуществляется посредством ревью кода написанным разными участниками проекта, таким образом также осуществляется обеспечение связуемости микросервисов, так как проверкой кода занимаются разработчики ответственные за разработку систем взаимодействующих с проверяемой (см. пункт 2.2).

Обеспечение связумости также происходит и на уровне работы над итерацией, так как использование в ходе разработки Docker-контейнеров и автоматической документации OpenAPI позволяет в реальном времени отслеживать изменения всех компонентов системы, так как документация к системе остается всегда актуальной.

На основе вышесказанного составлен план коммуникации команды, представленный в приложении Д. Для определения сроков и нагрузки на участников проекта был составлен календарный план, представленный в приложении А.

В рамках данной работы рассматриваются задачи Full-stack разработчика и технического директора:

— составление и формализация требований заказчика;
— коммуникация с внешними подрядчиками (хостинг, внешние сервисы) и генеральным директором ООО “ГЭТ Э БЛАНКЕТ”;
— проектирование архитектуры;
— проектирование баз данных;
— разработка и проектирование клиентского интерфейса;
— проектирование макета интерфейса технического сервиса;
— разработка и проектирование системы аутентификации;
— разработка и проектирование системы управления автоматами;
— выполнение задач DevOps (CI/CD, развертывание, поддержка серверов);
— разработка автоматического тестирования разрабатываемых компонентов.

2.4 Проектирование компонентов системы

В рамках данной работы рассматривается разработка микросервисов Пользовательский интерфейс, Сервис аутентификации, Сервис управления автоматами (см. рисунок 2.10).

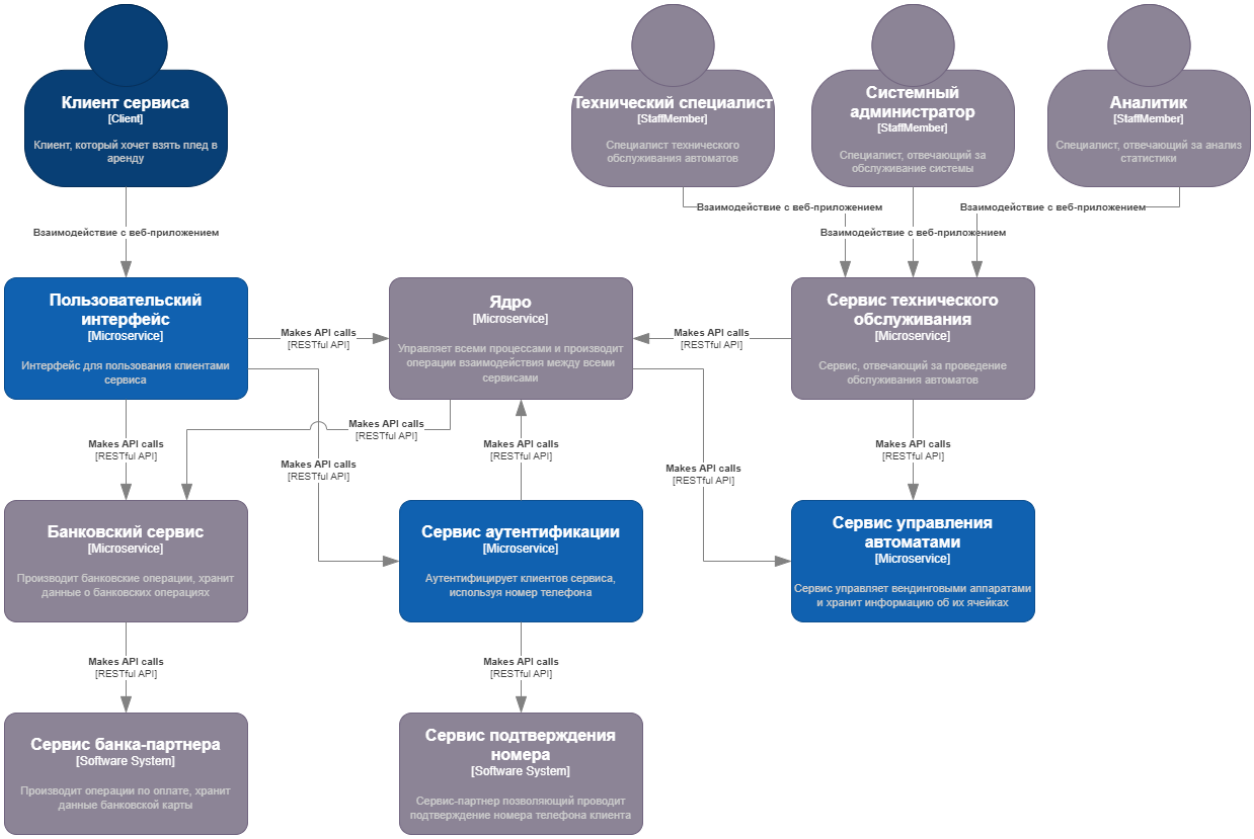

Рисунок 2.10 – Модель контекста, отмечены только компоненты, разработка которых рассматривается в рамках данной работы

Для проведения дальнейшего проектирования следует построить модели компонентов для рассматриваемых контейнеров.

2.4.1 Проектирование пользовательского интерфейса

Рассматривая контейнеры пользовательского интерфейса (см. рисунок 2.4), видно, что контейнер Веб сервер выполняет функцию предоставления доступа браузером клиента к интерфейсу системы, таким образом, контейнер Веб сервер не требует дальнейшей декомпозиции, так как он выполняет единственную функцию – предоставление доступа к статическим данным и API других микросервисов, а следовательно имеет единственный компонент – nginx сервер.

Интерфейс системы же возможно декомпозировать на компоненты (см. рисунок 2.11). Таким образом, контейнер делится на компоненты: Основной интерфейс приложения (представляющий собой набор виджетов формирующих приложение), Скрипты запросов к API, написанные на языке программирования Dart, и Виджет карты с автоматами, написанный на HTML/CSS и JavaScript с использованием API Yandex Map. Использование карт от Яндекс для HTML/CSS и JavaScript тем, что SDK карт для Flutter недоступен для компиляции под Web.

Рисунок 2.11 – Модель компонентов контейнера “Интерфейс системы”

Для проектирования интерфейса было использовано программное обеспечение Figma, пример макета для светлой и темной темы представлены на рисунке 2.12.

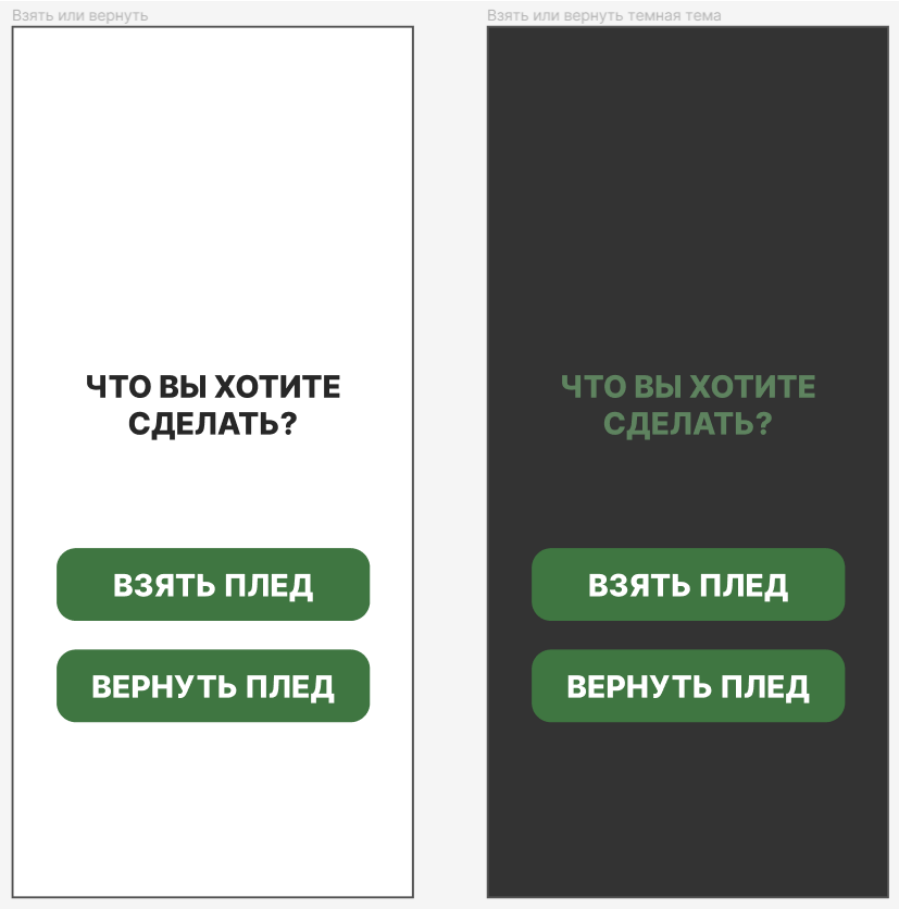

Рисунок 2.12 – Пример макета в Figma

2.4.2 Проектирование сервиса аутентификации

Декомпозиция контейнеров Сервиса аутентификации может производиться только в отношении контейнера API Server, так как декомпозиция СУБД невозможна, так как она выполняет только единственную функцию – хранение структурированных данных. За агрегирование и иное взаимодействие с данными полностью отвечает API Server, что видно на рисунке 2.13. API Server делится на компоненты: API controller (выполняет управление выдачи JWT-токенов), Библиотека подтверждения номера телефона (упрощает взаимодействие с внешней системой подтверждения номера телефона), ORM Postgres (упрощает взаимодействие с СУБД).

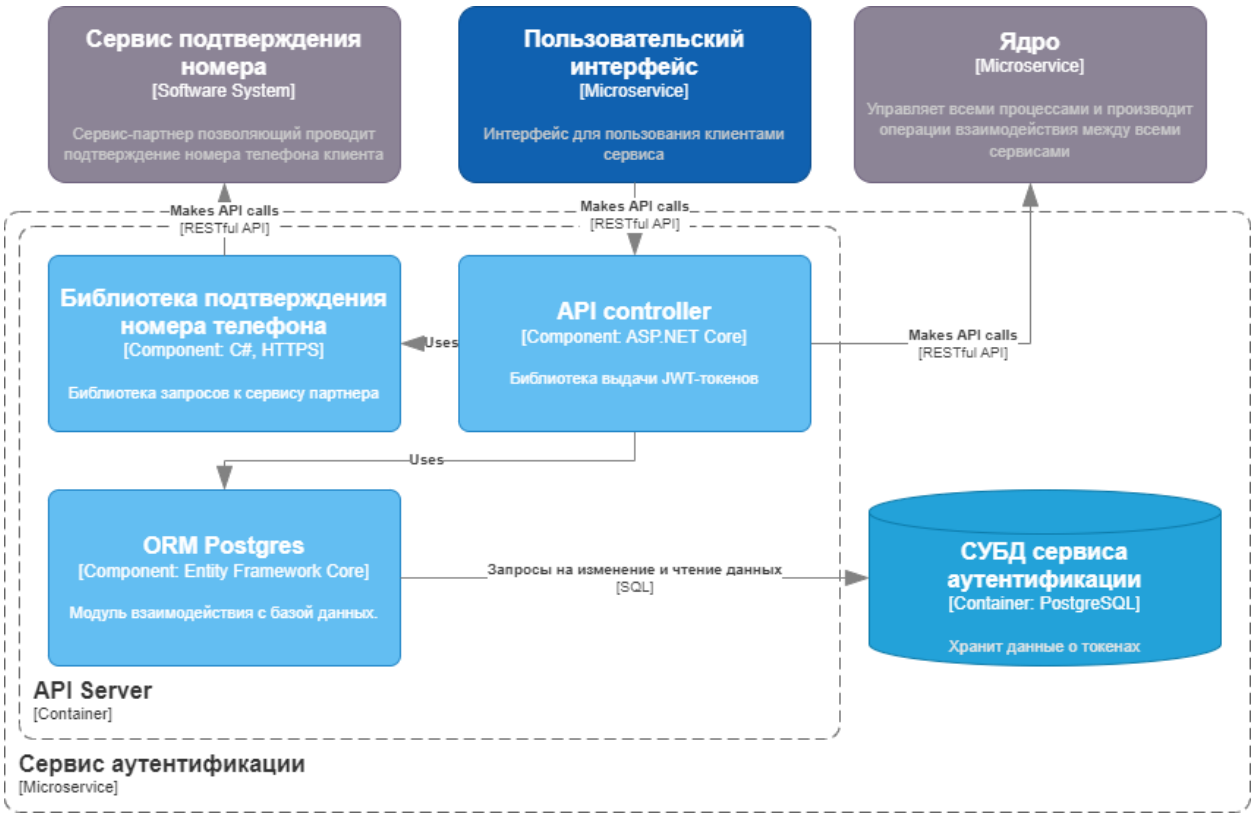

Рисунок 2.13 – Модель компонентов контейнера “API Server” Сервиса аутентификации

2.4.3 Проектирование сервиса управления автоматами

Внутри Сервиса управления автоматами возможна декомпозиция двух контейнеров: API Server и Служба управления автоматом, декомпозиция СУБД управления автоматами невозможна, так как она выполняет только единственную функцию – хранение структурированных данных.

Рассмотрим компоненты контейнера API Server (см. рисунок 2.14). Компонент API controller обеспечивает взаимодействие контейнера с внешними микросервисами, позволяя осуществлять управление функциями автоматов. Компонент Библиотека взаимодействия с автоматами предназначена для упрощения взаимодеиствия компонента API controller непосредственно с автоматами. Компонент ORM Postgres упрощает взаимодействие с СУБД.

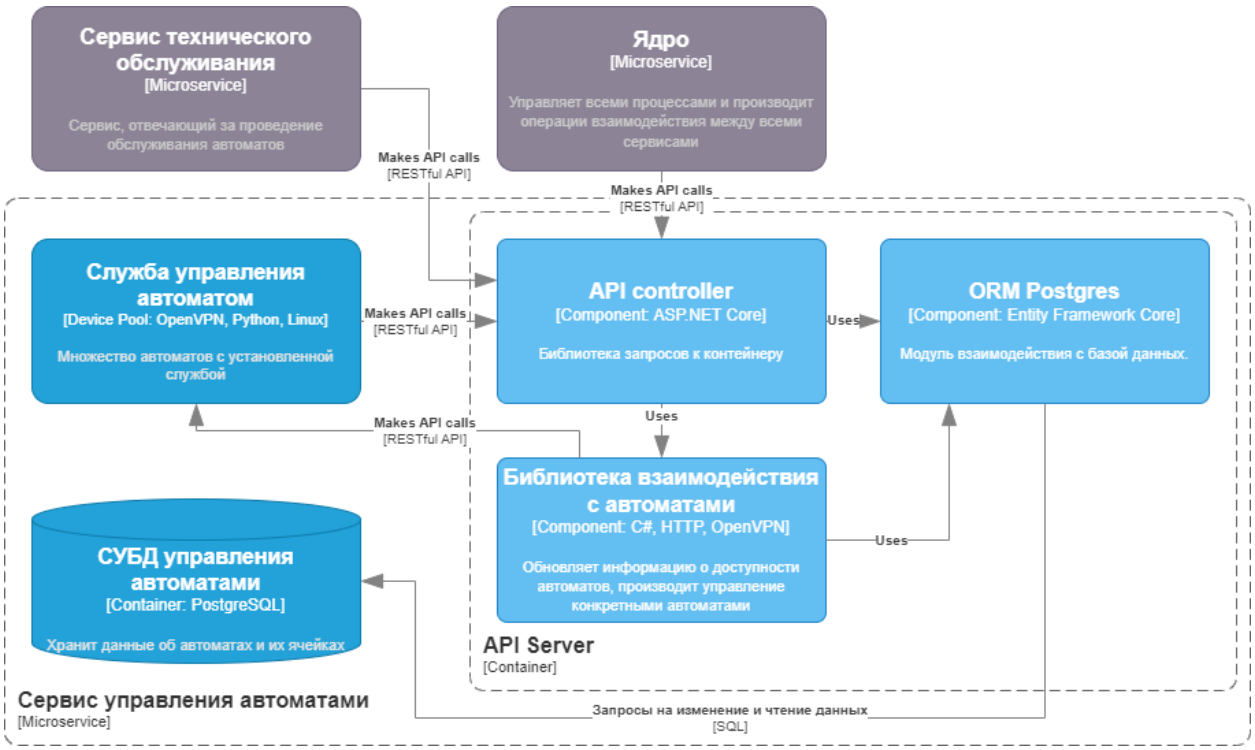

Рисунок 2.14 – Модель компонентов контейнера

Рассмотрим компоненты контейнера Служба управления автоматом (см. рисунок 2.15). Компонент API controller использует библиотеку Flask и реализует API для взаимодействия контейнера API Server с контейнером Служба управления автоматом. Компонент Библиотека управления автоматом упрощает взаимодействия с непосредственными скриптами управления автоматом, а также сообщает компоненту API Server о нахождении автомата в сети. Компонент Скрипты управления автоматом представляют собой низкоуровневые утилиты непосредственным управлением ячейками автомата.

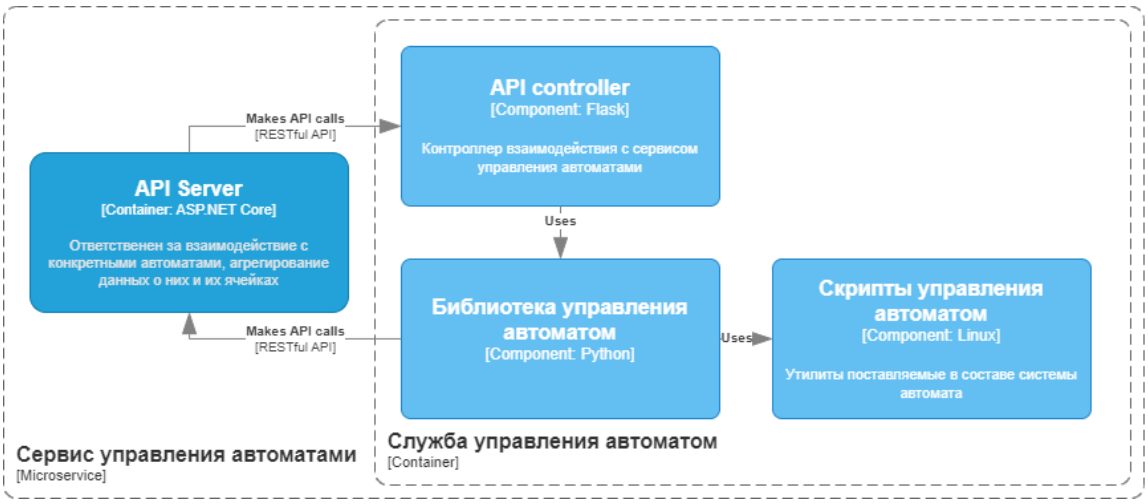

Рисунок 2.15 – Модель компонентов контейнера

2.4.4 Проектирование баз данных

Проектирование баз данных осложнено распределенностью контейнеров системы, что не позволит использовать единую схему базы данных. Таким образом схема базы данных должна быть построена так, чтобы минимизировать связи между данными, которые находятся в разных контейнерах. Поэтому необходимо построить набор схем баз данных, которые по отдельности будут отвечать 3 нормальной форме, а также не будут иметь внешних ключей относящихся к внешним базам данных. Схема базы данных представлена в соответствии с нотацией UML в приложении Ж.

Использование ORM Entity Framework Core в значительной степени упрощает описание и поддержку актуальности схемы базы данных. Было принято решение об использовании автоматических миграций, это позволит автоматически обновлять схему базы данных на тестовых и продакшн серверах. Что в значительной степени упростит развертывание и масштабируемость.

2.5 Выводы по главе

В данной главе рассмотрены процессы: выбора архитектурного дизайна системы, проектирования архитектуры системы до уровня контейнеров, выбора средств разработки, выбора методологии управления командной разработкой, полного проектирования архитектуры разрабатываемых компонентов системы, проектирования схемы базы данных.

Итогами данной главы являются:

— формализованная в соответствии с нотацией C4 архитектура системы;
— формализованные методы взаимодействия разработчиков внутри команды в виде плана коммуникаций (см. приложение Д);
— формализованы схемы базы данных в виде диаграммы в соответствии с нотацией UML (см. приложение Ж).

Глава 3. Реализация системы

В данной главе будут рассмотрены:

1) реализация Пользовательского интерфейса,
2) реализация Сервиса аутентификации,
3) реализация Сервиса управления автоматами,
4) тестирование разработанных компонентов,
5) автоматизация сборки и развертывания системы в целом,
6) внедрение в бизнес-процессы заказчика.

Также в рамках выполнения задач DevOps были подготовлены следующие средства коммуникации команды:

— создана организация на GitHub и подготовлены 11 репозиториев для работы над проектом;
— для репозиториев содержащих исходные коды контейнеров системы настроен CI/CD для автоматической сборки Docker контейнеров и агрегирования документации OpenAPI;
— создана организация в Notion для формирования бэклога (списка) задач к реализации по каждому компоненту системы;
— подготовлены средства коммуникации команды (составлен план коммуникаций (см. приложение Д), созданы каналы общения в Discord и Telegram).

Итогом данной главы станет акт о реальности проекта и акт о внедрении системы.

3.1 Реализация Пользовательского интерфейса

Разработка клиентского сайта производилась с разделением на виджеты, таким образом было выделено 12 виджетов (не считая информационные окна) (расположены в порядке пользования клиентом):

1) Виджет карты с автоматами.
2) Экран карты с автоматами.
3) Виджет кнопки клавиатуры.

4) Виджет клавиатуры.
5) Экран ввода кода автомата.
6) Экран автомата
7) Экран ввода номера телефона.
8) Экран подтверждения номера телефона.
9) Экран выбора тарифа и способа оплаты.
10) Виджет информации о ячейке.
11) Экран взятия пледа.
12) Экран возврата пледа.

Трудности в реализации клиентского сайта были связаны в первую очередь с реализацией виджета карты с автоматами. Использование API Яндекс карт, было проблемным из-за отсутствия интеграции MapKit SDK для Flutter Web, поэтому пришлось пользоваться JavaScript API, который можно считать в некоторой степени устаревшим. Это усложнило работу, так как интегрировать JavaScript в Flatter можно только посредством интеграции отдельной HTML страницы, CSS стилей и JavaScript скрипта. Для этого был создан отдельный виджет карты, поверх которого обрисовывался интерфейс Flatter Web.

Разработка с использованием JavaScript API не была беспроблемной, в первую очередь из-за неполной и слабой документации, а также неприспособленности Flutter Web для интеграции Web-приложений внутри себя. Наибольшие проблемы, помимо настройки карт, вызвало использование элементов интерфейса Flutter поверх HTML интеграции. Итоговый интерфейс имеет адаптивную вёрстку и приспособлен для использования устройствами размером от 4,7 до 7 дюймов. Изображение интерфейса представлено ниже на рисунке 3.1.

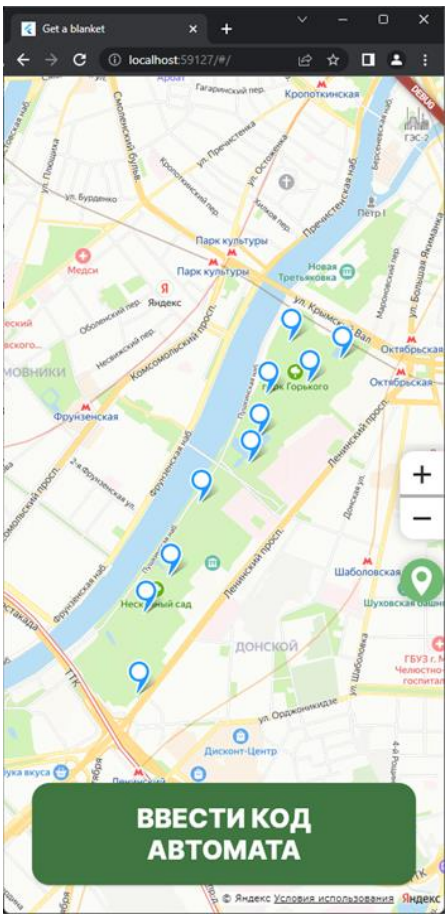

Рисунок 3.1 – Экран карты с автоматами

Остальные виджеты писались только с использованием языка Dart, возможностей Flatter. Разработка остальных интересов серьезных проблем не составила. Стоит только упомянуть, что все настройки шрифтов и цветов были вынесены в настройки темы приложения, что позволяет сделать приложение с настраиваемой темой (светлой и темной). Примеры экранов (в светлой версии) представлены ниже на рисунках 3.2, 3.3, 3.4, 3.5, 3.6.

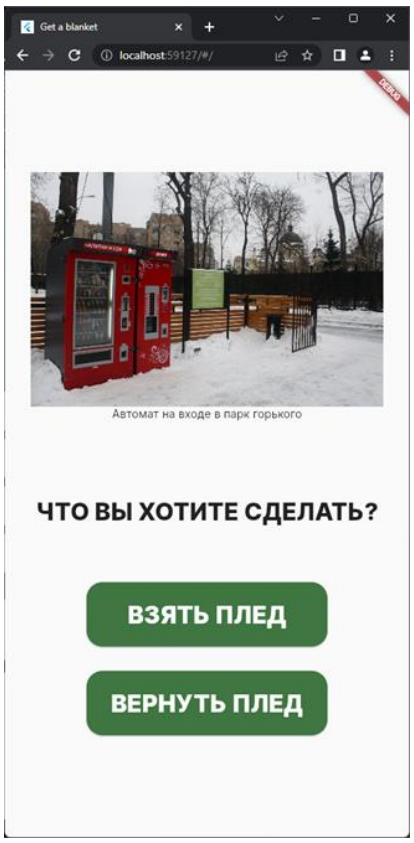

Рисунок 3.2 – Экран автомата

Рисунок 3.3 – Экран ввода номера телефона

Рисунок 3.4 – Экран подтверждения номера телефона

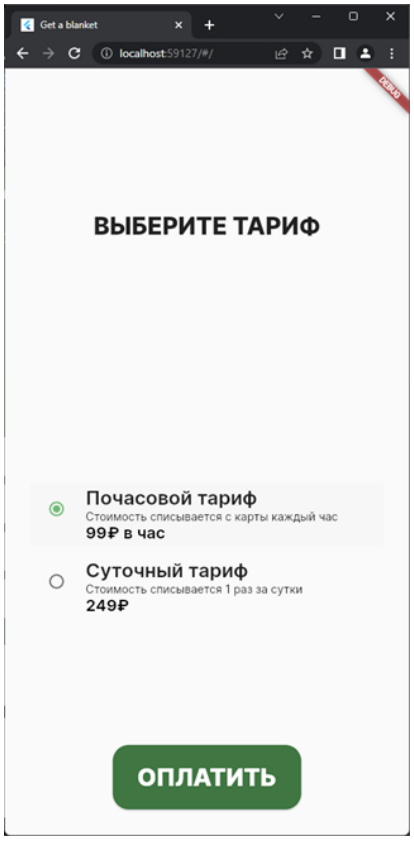

Рисунок 3.5 – Экран выбора тарифа и способа оплаты

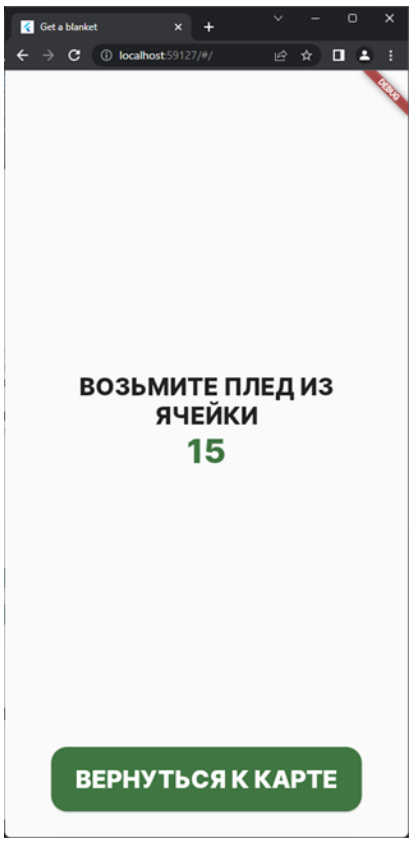

Рисунок 3.6 – Экран взятия пледа

Реализация контейнера веб сервера представляет собой файл шаблона конфигурации nginx, который формируется в ходе запуска Docker-контейнера.

Исходный код находится в открытом доступе на GitHub https://github.com/Get-a-blanket/ClientFrontend.

3.2 Реализация Сервиса аутентификации

Удобство использования ASP.NET для реализации RESTful API заключается в том, что фактически разработка RESTful API ничем не отличается от разработки классической библиотеки. Требуется лишь добавить атрибуты типа запроса к функциям. К классам следует добавить атрибуты, указывающие на адрес данного класса.

По итогу разработки получается RESTful API с готовой автоматически сгенерированной документацией OpenAPI (бывший Swagger).

Таким образом, реализация Сервиса аутентификации представляет собой написание 3 библиотек: библиотека запросов к внешнему сервису, библиотека

API, библиотека описывающая схему базы данных. В таблице 3.1 представлены примеры разработанных функций запросов к API.

Таблица 3.1 – Примеры функций запросов к API

Название функции	**Входные параметры**	**Выходные параметры**
AuthRequest	RegionCode, Number	AuthId
SendSMS	AuthId	–
CodeVerification	Code, AuthId	JWT-Token

Исходный код находится в открытом доступе на GitHub https://github.com/Get-a-blanket/Auth.

3.3 Реализация Сервиса управления автоматами

Разработка данного сервиса требует разработки двух значительно различающихся контейнеров: API Server и Служба управления автоматом. Поэтому будет целесообразно рассмотреть их отдельно.

3.3.1 Реализация API управления автоматами

API управления автоматами разрабатывается с использованием ASP.NET, что даёт те же преимущества для реализации RESTful API, что описаны в реализации Сервиса аутентификации.

В отличии от Сервиса аутентификации в состав контейнера API входит не только приложение, но и клиент OpenVPN для установления защищенного канала сетевой связи с множеством служб управления автоматами.

Таким образом, реализация API управления автоматами собой написание 3 библиотек: библиотека взаимодействия с автоматами посредством OpenVPN, библиотека API, библиотека описывающая схему базы данных. В таблице 3.2 представлены примеры разработанных функций запросов к API.

Таблица 3.2 – Примеры функций запросов к API управления автоматами

Название функции	**Входные параметры**	**Выходные параметры**
RequestInformationAboutCells	ListOfMachinesIds	ListOfMachinesCellsInformation
BlanketBackRequest	MachineId	CellNumber
BlanketReservationRequest	MachineId	ReservationId
BlanketGiveRequest	ReservationId	CellNumber
BlanketStopReservationRequest	ReservationId	–

Исходный код находится в открытом доступе на GitHub https://github.com/Get-a-blanket/VendingService.

3.3.2 Реализация Службы управления автоматом

Реализация службы управления значительно отличается от остальных компонентов системы. API имеет минимальный функционал (см. таблицу 3.3).

Таблица 3.3 – Функции запросов к API управления автоматом

Название функции	**Входные параметры**	**Выходные параметры**
RequestOpenCell	CellNumber	–
GetStatus	–	SystemInformation

Это позволяет минимизировать нагрузки на систему управления автоматом.

Исходный код находится в открытом доступе на GitHub https://github.com/Get-a-blanket/VendingMachine.

3.4 Тестирование

Для проведения тестирования программного продукта применены следующие виды и методы:

— Функциональное тестирование. Применяется для проверки основных функций системы и ее работоспособности. Функциональные испытания проводятся по методам «Черный ящик» и «Белый ящик».

— Нагрузочное тестирование. Необходимо для проверки системы в ситуациях, когда резко увеличивается количество пользователей.
— Тестирование производительности. Осуществляется оценка работы системы при нормальных условиях и при различных уровнях нагрузки.

Тестирование системы осуществляется с помощью ручных и автоматизированных тестов. Необходимо использовать средства и инструменты для обеспечения успешного проведения тестирования.

Для управления и организации тестирования:

— TestIT – система управления, позволяющая составлять тест-кейсы, хранить их в одном пространстве, создавать тест-планы;
— Telegram – средство общения с участниками команды для уточнения кейсов тестирования. Для тестирования API:
— Postman – сервис для комплексного составления и отправления запросов, а также их документирования;
— Swagger – средство просмотра документации и быстрой проверки запросов. Для автоматизированного тестирования:
— Unit-тестирование – тестирование, с помощью которого осуществляется проверка отдельных компонентов программы на корректность их работы;
— CI/CD – при выполнении pull request в основную ветку выполняется автоматическое тестирование.

3.4.1 Тестирование интерфейса

Поскольку рассчитывается, что сайтом будут пользоваться на телефонах, тестирование в первую очередь проводилось на видимость интерфейса на всех типах устройств. Было принято решение, что самый малый размер экрана устройства на данный момент (из устройств, имеющих достаточный объем рынка) это iPhone SE, а самый большой из вероятных устройств это iPad Air. Тестирование на видимость интерфейса проводилось как в автоматическом режиме с использованием функционала отладки Flutter, так и вручную, визуально. Для визуальной проверки использовался режим отображения «как на мобильном устройстве» встроенный в браузер Chrome. Для тестирования были выбраны «пресеты» 3

устройств: iPhone SE, iPhone 12 Pro, iPad Air. Способ тестирования показан на рисунке 3.7.

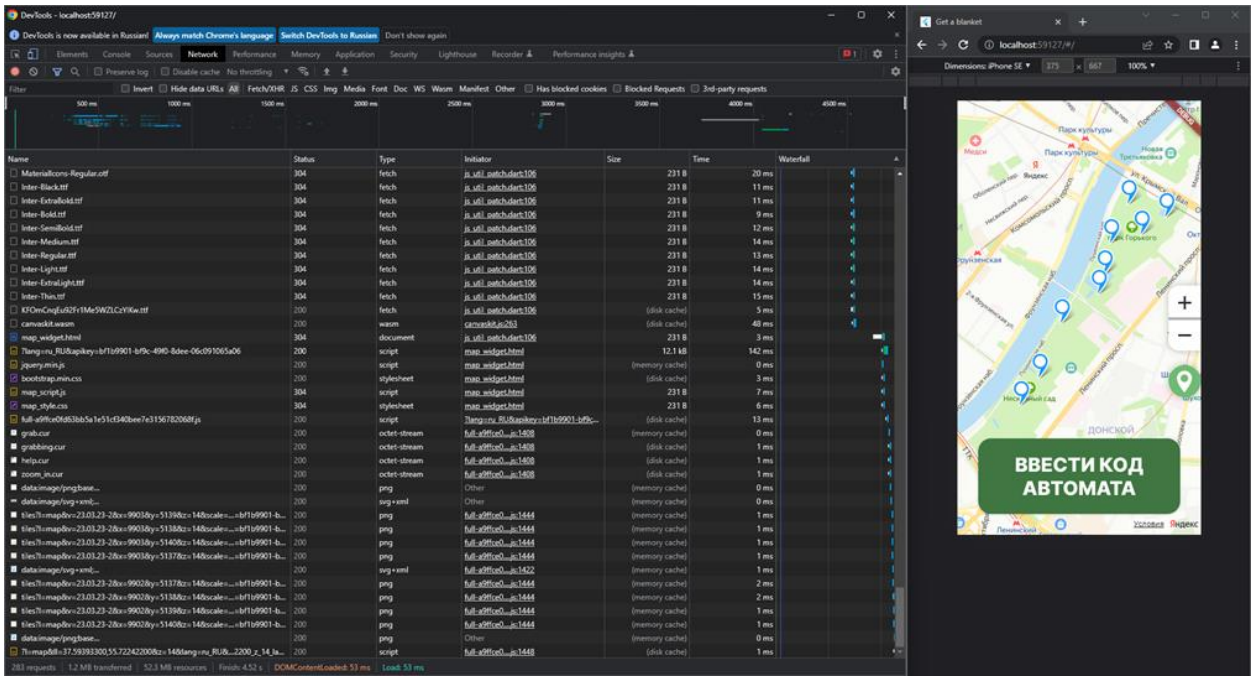

Рисунок 3.7 – Тестирование интерфейса

3.5 Автоматизация сборок и развертывания

Из-за того, что Front и Back проекты разделены на несколько репозиториев автоматизация сборок предполагает создание отдельных версий для каждого компонента системы. Было принято решение, что при коммите в ветку master Front проекта будет запускаться GitHub Action, состоящих из заданий:

1) Сборку приложения, проверив возможность этого.
2) Параллельно выполнит Unit-тесты.
3) После если прошлые этапы были выполнены без ошибок, будет выполнена сборка Docker-контейнера и его отправка в репозиторий Docker на GitHub.

GitHub Action Back-проектов будет выполняться так же при коммите в ветку master и состоять из задач:

1) Запуск Unit-тестов (проверка возможности сборки не требуется, так как она выполнится в любом случае при запуске тестов).

2) После если прошлые этапы были выполнены без ошибок, будет выполнена сборка Docker-контейнера и его отправка в репозиторий Docker на GitHub.

Обновление на контейнеров на тестовом и продакшн серверах выполняются автоматически в соответствии с правилами обновления: тест раз в день, продакшн раз в неделю с опозданием в неделю.

3.6 Выводы по главе

В ходе выполнения задач поставленных в начале главы было выполнено:

1) разработан Пользовательский интерфейс,
2) разработан Сервис аутентификации,
3) разработан Сервис управления автоматами,
4) проведено тестирование разработанных компонентов,
5) выполнена автоматизация сборки контеинеров.

ГЛОССАРИЙ

ГОСТ — Государственный стандарт

IoT — Internet of things. Интернет вещей. Концепция сети передачи данных между физическими объектами («вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой.

UML — Unified Modeling Language. Стандартный язык графического представления проектов.

ООО — Общество с ограниченной ответственностью. Форма коммерческой организации.

LDAP — Lightweight Directory Access Protocol. Протокол прикладного уровня для доступа к службе каталогов.

NFC — Near field communication. Технология беспроводной передачи данных малого радиуса действия.

SDK — Software development kit. Комплект для разработки программного обеспечения. Набор инструментов для разработки программного обеспечения, объединённый в одном пакете, обычно содержит комплект необходимых библиотек, компилятор, отладчик, иногда — интегрированную среду разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Макаренко Г.* Рынок шеринга в россии впервые превысил 1 трлн руб. [Электронный ресурс] / РБК Тренды. — 2021. — URL: https://trends.rbc.ru/trends/sharing/602e3a369a79477994233cb3 (дата обращения: 02.12.2023).

2. Бизнес план вендинговых аппаратов [Электронный ресурс]. — URL: https://www.beboss.ru/bplans-vending-machines (дата обращения: 02.12.2023).

3. *Макаренко Г.* Бизнес на вендинговых автоматах: выгодно ли это сейчас [Электронный ресурс] / СовкомБлог. — URL: https://journal.sovcombank.ru/biznesu/biznes-na-vendingovih-avtomatah-vigodno-li-eto-seichas (дата обращения: 02.12.2023).

4. Как заработать на вендинговых автоматах в 2023 году: обходим вместе все подводные камни [Электронный ресурс] / Финтолк. — URL: https://fintolk.pro/kak-zarabotat-na-vendingovyh-avtomatah-v-2023-godu/ (дата обращения: 02.12.2023).

5. Оборудование и продукты [Электронный ресурс] / Бери заряд. — URL: https://berizaryad.ru/business/products (дата обращения: 02.12.2023).

6. Cantaloupe, Inc. [Электронный ресурс]. — URL: https://www.cantaloupe.com/ (дата обращения: 02.12.2023).

7. OTI Global [Электронный ресурс]. — URL: https://otiglobal.com/ (дата обращения: 02.12.2023).

8. ИТ-компания Новософт [Электронный ресурс]. — URL: https://www.novosoft.ru/ (дата обращения: 02.12.2023).

9. Hubex [Электронный ресурс]. — URL: https://hubex.ru/ (дата обращения: 02.12.2023).

10. Компания ЗетаСофт [Электронный ресурс]. — URL: https://www.zetasoft.ru/ (дата обращения: 02.12.2023).

11. Юрент [Электронный ресурс]. — URL: https://urent.ru/ (дата обращения: 02.12.2023).

12. Whoosh [Электронный ресурс]. — URL: https://whoosh-bike.ru/ (дата обращения: 02.12.2023).

13. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению [Электронный ресурс]. — URL: https://www.swrit.ru/doc/espd/19.201-78.pdf (дата обращения: 02.12.2023).

14. Зарплаты продавцов в Москве [Электронный ресурс]. — URL: https://bdex.ru/moscow/?type=trading (дата обращения: 25.02.2024).

15. Зарплата Продавца в Москве [Электронный ресурс]. — URL: https://gorodrabot.ru/salary?p=продавец+в+тц&l=москва (дата обращения: 25.02.2024).

16. *Некрасов Максим Валерьевич*. Оценка эффективности инвестиций в вендинг-деятельность // π-Economy. — 2009.

17. *Хохлова Г.И., Томилова А.В.* Вендинговый бизнес в России и за рубежом: история и перспективы развития // Актуальные проблемы гуманитарных и естественных наук. — 2014.

18. *Вера Колерова*. Ларёк 2.0 // Бизнес-журнал. — 2014.

19. *Антипин Филипп Андреевич*. Вендинговая торговля в России: анализ и перспективы развития // Российское предпринимательство. — 2016.

20. *Лапидус Лариса Владимировна, Гостилович Александр Олегович*. УПРАВЛЕНИЕ КАЧЕСТВОМ ШЕРИНГОВЫХ УСЛУГ: ОЦЕНКА УРОВНЕЙ ЛОЯЛЬНОСТИ И ПОТРЕБИТЕЛЬСКОЙ УДОВЛЕТВОРЕННОСТИ РАЙДШЕРИНГОМ // Ars Administrandi. — 2023.

21. *Ю.В. Ядгарова*. Модель и алгоритм выбора программной архитектуры для систем Интернета вещей // Программные продукты и системы. — 2019.

22. *Гобарева Я.Л., Добриднюк С.Л., Касьянов М.Е.* НОВЫЕ ИНСТРУМЕНТЫ И СЕРВИСЫ В СИСТЕМЕ БЫСТРЫХ ПЛАТЕЖЕЙ, АРХИТЕКТУРА ПРОГРАММНЫХ РЕШЕНИЙ // Инновации и инвестиции. — 2023.

23. *Егорян В.В., Калугин А.В.* РОЛЬ АРХИТЕКТУРНОГО ПОДХОДА ПРИ РАЗРАБОТКЕ ПРОГРАММНЫХ ПРИЛОЖЕНИЙ // Столыпинский вестник. — 2022.

24. *Gos K., Zabierowski W.* The Comparison of Microservice and Monolithic Architecture // International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH). — 2020.

25. *Indra Gita Anugrah, Muhamad Aldi Rifai Imam Fakhruddin*. Development Authentication and Authorization Systems of Multi Information Systems Based REst API and Auth Token // INNOVATION RESEARCH JOURNAL. — 2020.

26. *Marshall I. Goldman*. Retailing in the Soviet Union // American Marketing Association. — 1960. — Vol. 24, no. 4.

27. *Susanne Gruber, Renate Buber, Bernhart Ruso, Johannes Gadner*. THE COMMODITY VENDING MACHINE // FORUM WARE international 2005/02. — 2005. — P. 32–42.

28. *Vennan Sibanda, Lorraine Munetsi, Khumbulani Mpofu, Eriyeti Murena, John Trimble*. Design of a high-tech vending machine // Procedia CIRP. — 2020. — Vol. 91. — P. 678–683.

29. *Chih-Chun Hsu, Yi-Chang Lin, Yeong-Long Shiue, Chi-Chia Sun*. New Generation Artificial Intelligent Vending Machine System based on LoRaWan IOT Network. — Yilan, Taiwan: IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW); IEEE, 2019.

30. *Hironori Washizaki, Shinpei Ogata, Atsuo Hazeyama, Takao Okubo, Eduardo B. Fernandez, Nobukazu Yoshioka*. Landscape of Architecture and Design Patterns for IoT Systems // IEEE Internet of Things Journal. — 2020. — Vol. 7, no. 10.

31. *Rebeca Campos Motta, Káthia Marçal de Oliveira, Guilherme Horta Travassos*. On challenges in engineering IoT software systems. — Sao Carlos, Brazil: SBES ’18: Proceedings of the XXXII Brazilian Symposium on Software Engineering; Association for Computing Machinery, 2018. — P. 42–51.

32. *Konrad Gos, Wojciech Zabierowski*. The Comparison of Microservice and Monolithic Architecture. — Lviv, Ukraine: International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH); IEEE, 2020. — P. 150–153.

33. *Indra Gita Anugrah, Muhamad Aldi Rifai Imam Fakhruddin*. Development Authentication and Authorization Systems of Multi Information Systems Based REst

API and Auth Token // INNOVATION RESEARCH JOURNAL. — 2020. — Vol. 1, no. 2. — P. 127–132.

34. *Virginija Grybaitė, Jelena Stankevičienė*. Factors affecting sharing economy growth // Oeconomia Copernicana. — 2018. — P. 635–654.

35. *Dinara Davlembayeva, Savvas Papagiannidis, Eleftherios Alamanos*. Mapping the economics, social and technological attributes of the sharing economy // Information Technology & People. — 2019. — P. 841–872.

36. *Ahmad, W. S. H. M. W.; Radzi, N. A. M.; Samidi, F. S.; Ismail, A.; Abdullah, F.; Jamaludin, M. Z.; Zakaria, M. N.* 5G Technology: Towards Dynamic Spectrum Sharing Using Cognitive Radio Networks // IEEE Access. — 2020. — P. 14460–14488.

37. *Shanzhi Chen, Ying-Chang Liang, Shaohui Sun, Shaoli Kang, Wenchi Cheng, Mugen Peng*. Vision, Requirements, and Technology Trend of 6G: How to Tackle the Challenges of System Coverage, Capacity, User Data-Rate and Movement Speed // IEEE Wireless Communications. — 2020. — P. 218–228.

38. Архитектура программного обеспечения [Электронный ресурс]. — URL: https://synergy.ru/akademiya/programming/arxitektura_programmnogo_obespecheniya (дата обращения: 15.04.2024).

39. *Кен Швабер. Джефф Сазерленд*. Исчерпывающее руководство по Скраму: Правила Игры.

ПРИЛОЖЕНИЕ А

Календарный план

Календарный план составлялся в системе управления ресурсов MS Project. Ниже представлен отчет сформированный 25 декабря 2023 года.

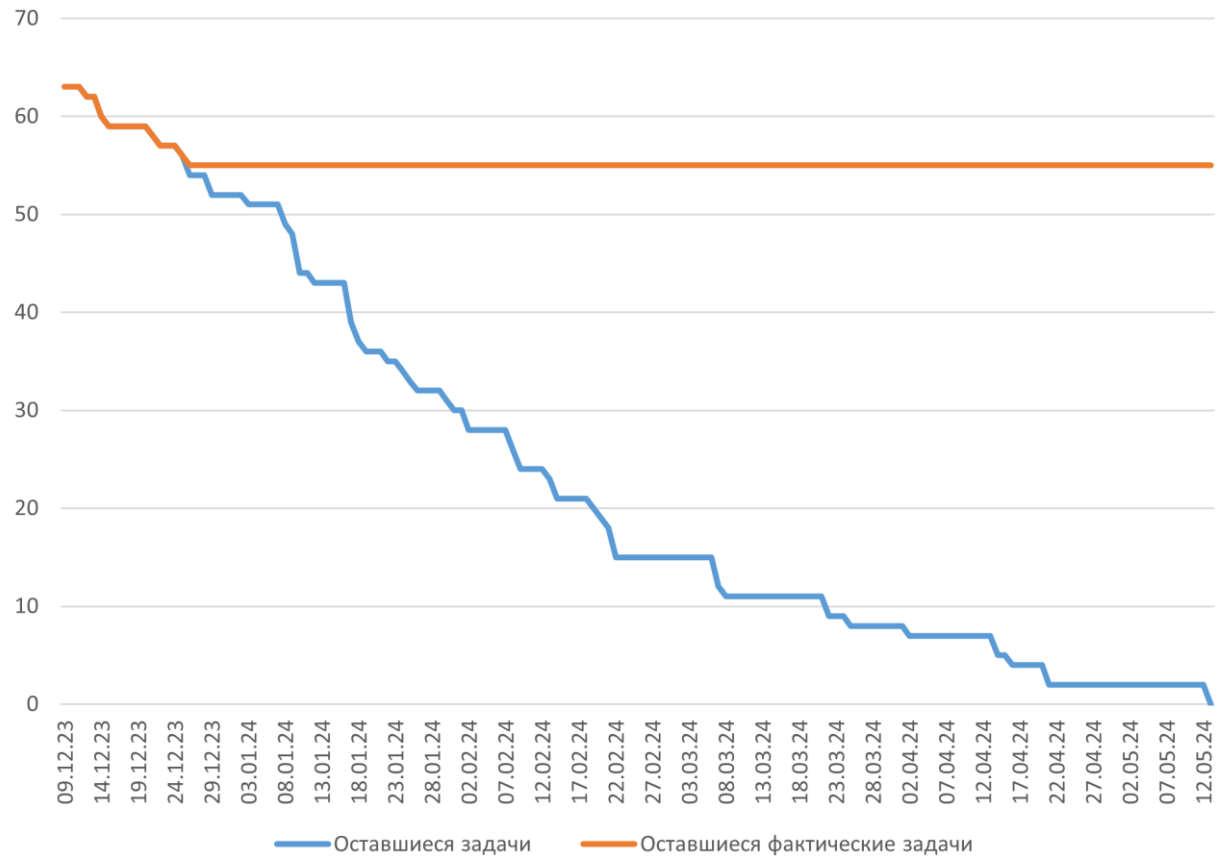

Рисунок А.1 – Диаграмма выполненных задач от времени

Таблица А.1 – Сроки выполнения основных работ

Название	**Начало**	**Окончание**	**% завершения**
Формирование требований проекта	Ср 11.10.23	Пн 20.11.23	100%
Проектирование разрабатываемой системы	Вт 21.11.23	Пт 15.12.23	100%
Описание внутреннего распорядка разработки	Пт 15.12.23	Пт 29.12.23	73%

Разработка программного продукта	Пн 25.12.23	Чт 07.03.24	0%
Проведение опытной эксплуатации	Пт 23.02.24	Вс 21.04.24	0%
Полноценная эксплуатация в рамках бизнес-модели	Пн 15.04.24	Пн 13.05.24	0%
Сдача ВКР	Пн 13.05.24	Пн 13.05.24	0%

Задачи представленные на рисунке А.1 и в таблице А.1 описаны более подробно в файле *УПП.mpp*, приложенном отдельно.

ПРИЛОЖЕНИЕ Б

Функционально-стоимостной анализ

Представлено в файле *ФСА.docx*.

ПРИЛОЖЕНИЕ В

Техническое задание

Представлено в файле *ТЗ.docx*.

ПРИЛОЖЕНИЕ Г

Финансовая модель на 5 лет

Представлено в файле *Финансовая модель.xlsx*.

ПРИЛОЖЕНИЕ Д

План коммуникации команды

Представлено в файле *План коммуникаций.docx*.

ПРИЛОЖЕНИЕ Е

Диаграммы последовательности

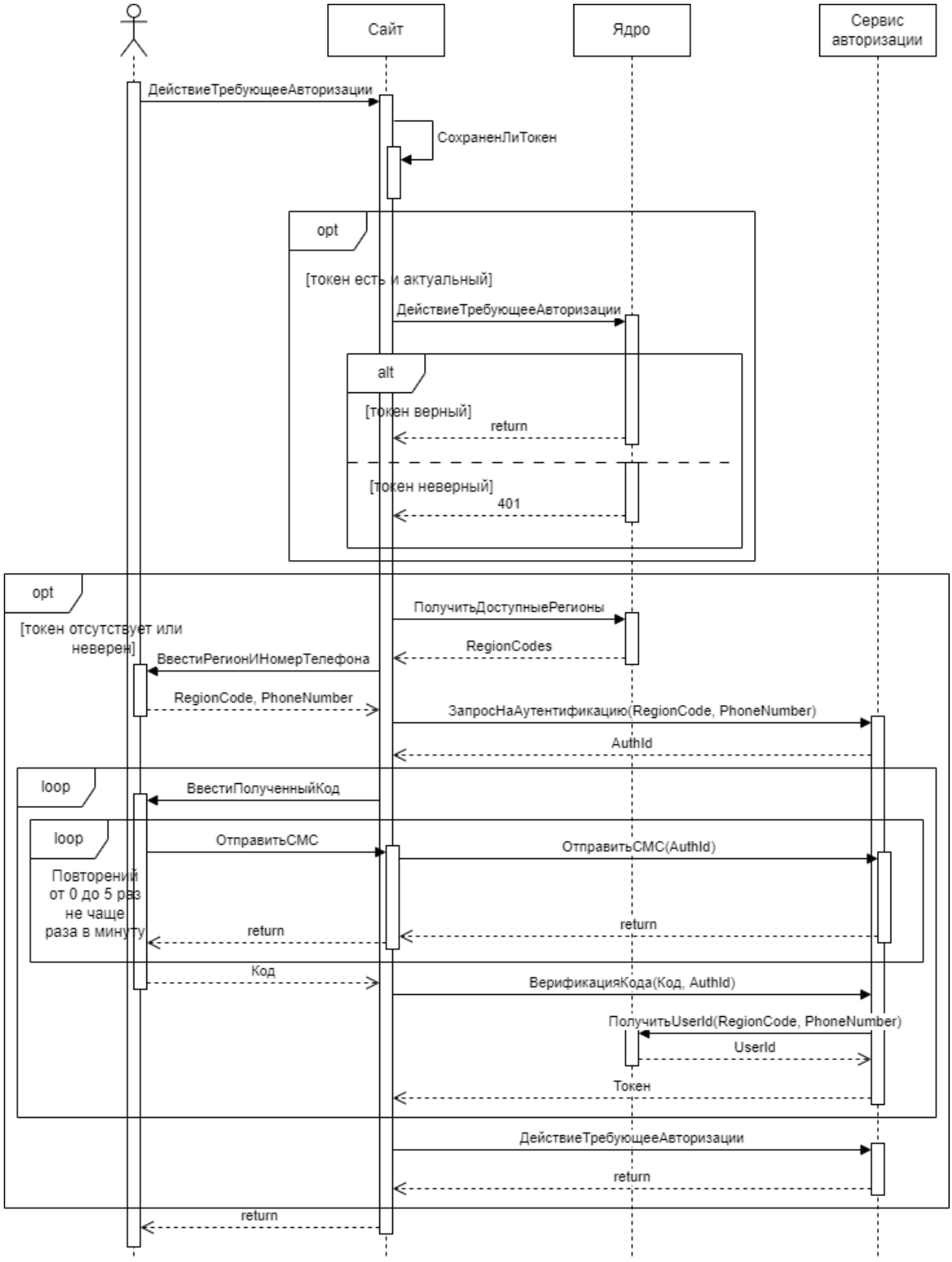

Рисунок Е.1 – Диаграмма последовательности “Авторизация по телефону”

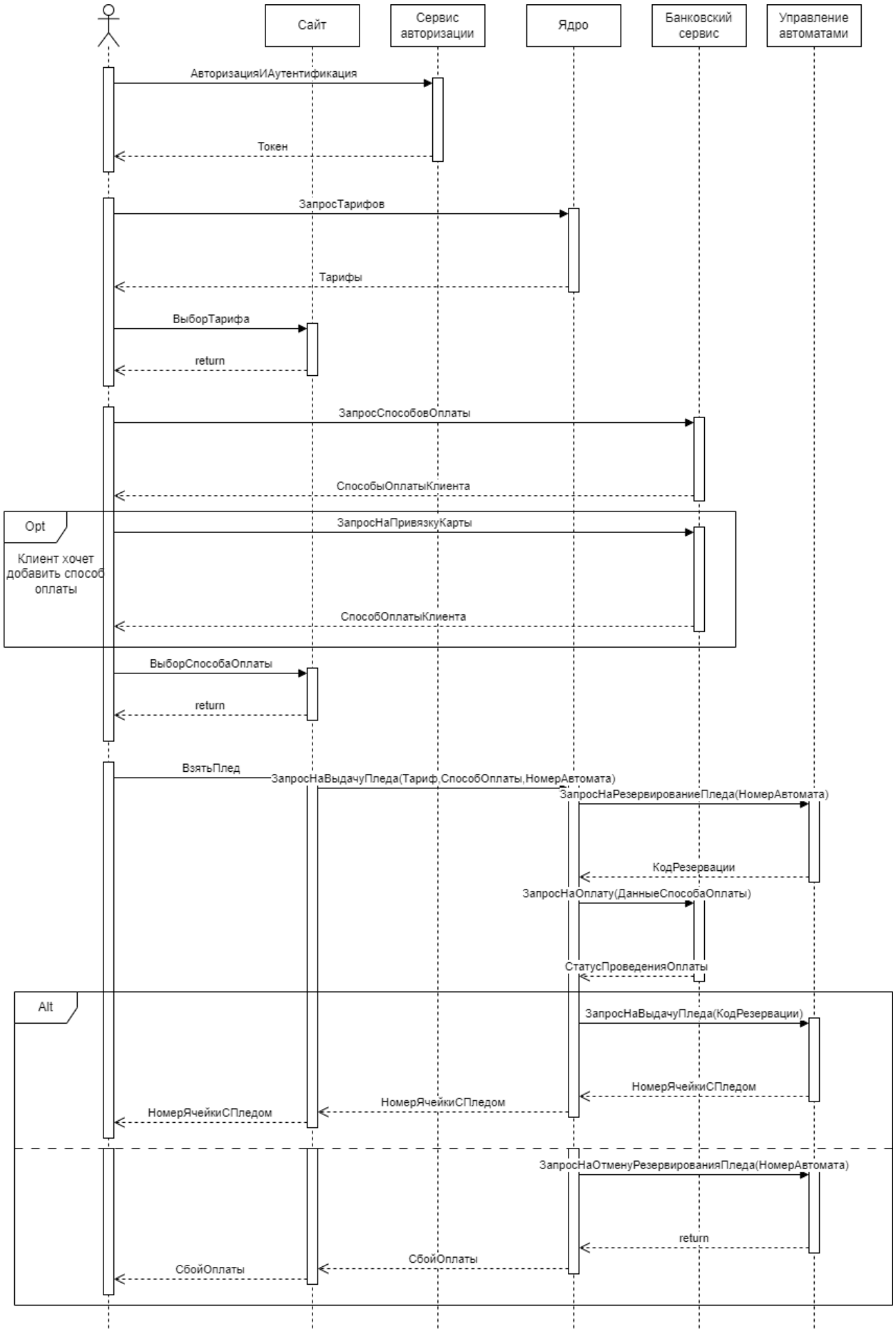

Рисунок Е.2 – Диаграмма последовательности “Взять плед”

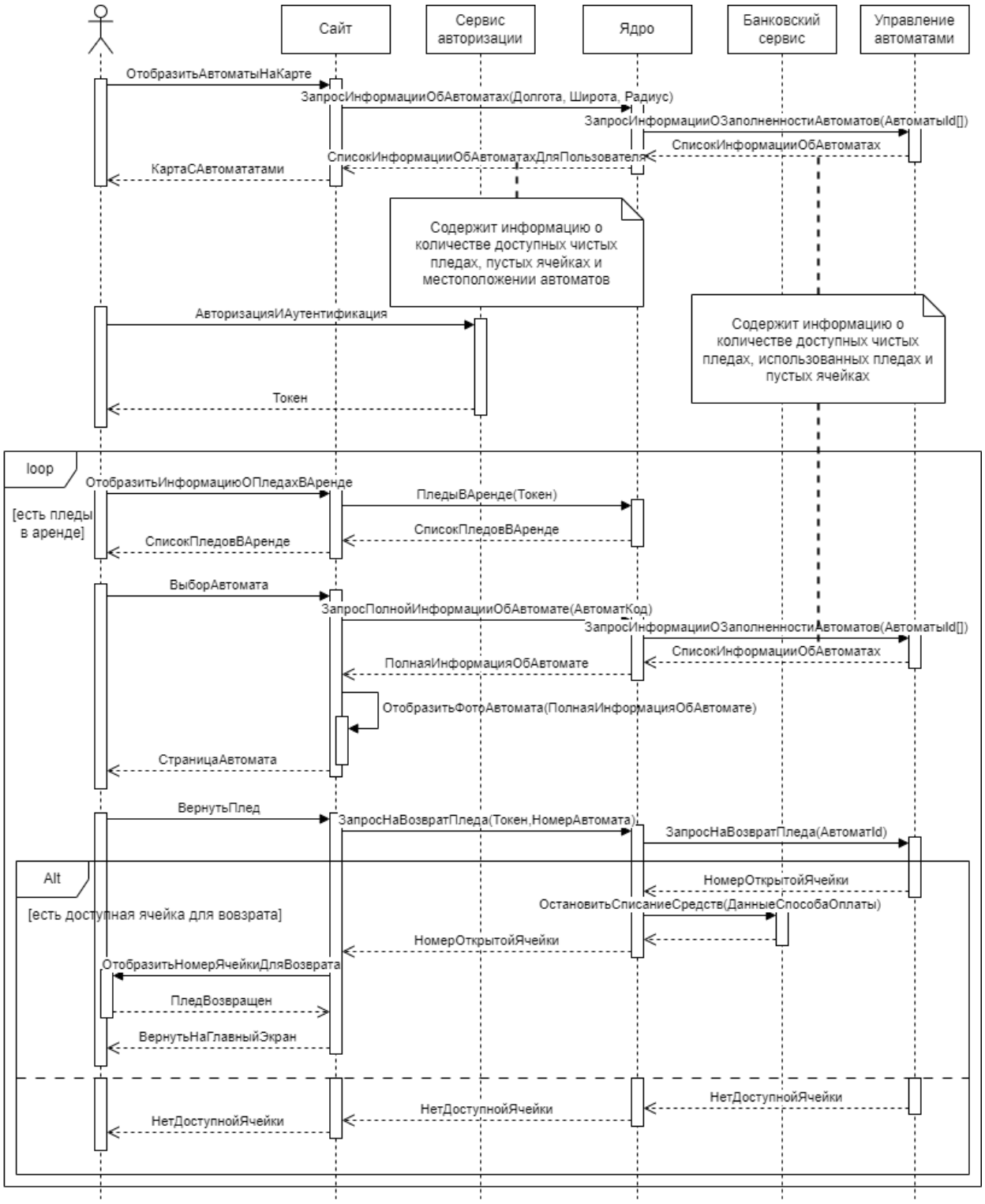

Рисунок Е.3 – Диаграмма последовательности “Вернуть плед”

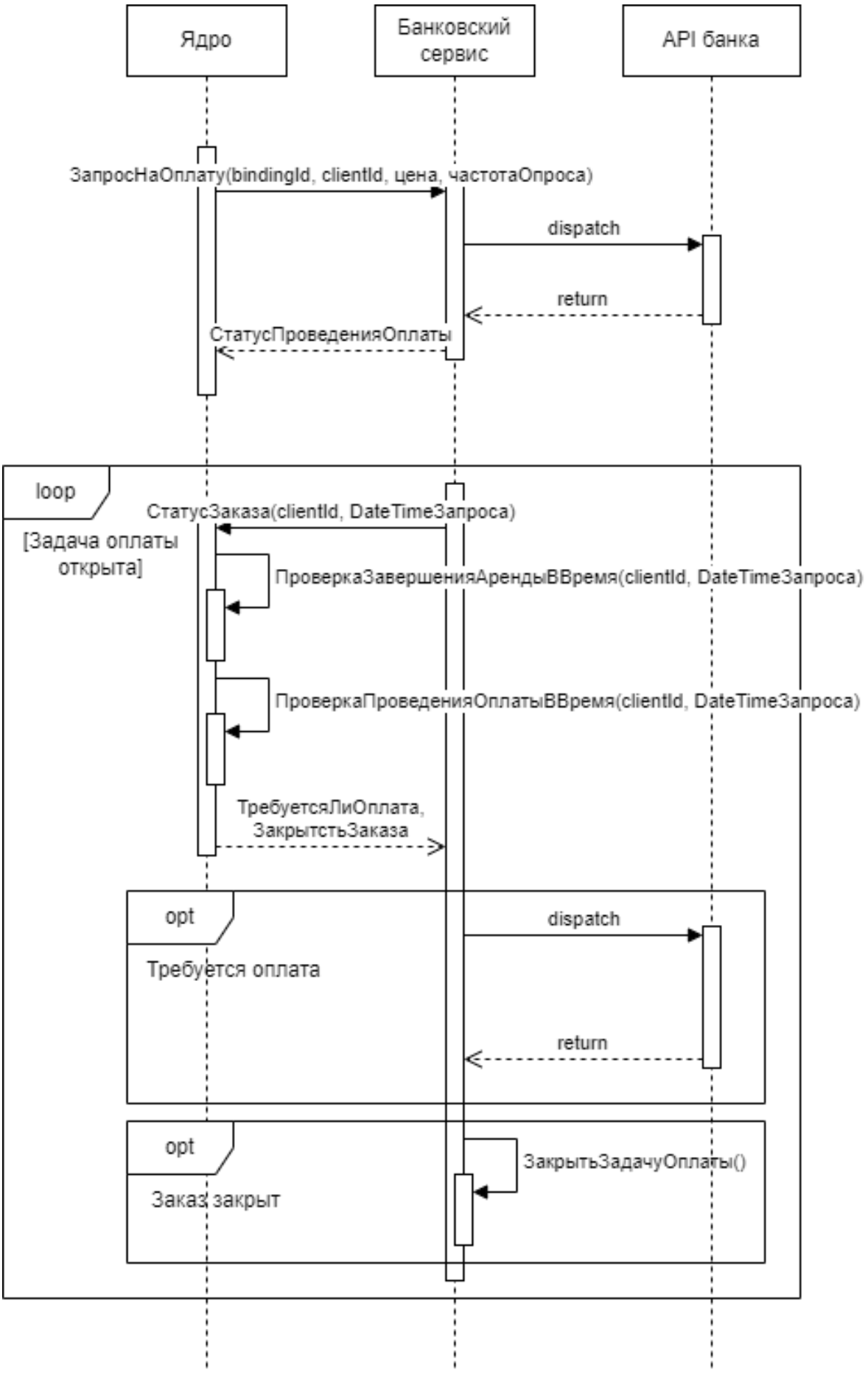

Рисунок Е.4 – Диаграмма последовательности “ЗапросНаОплату”

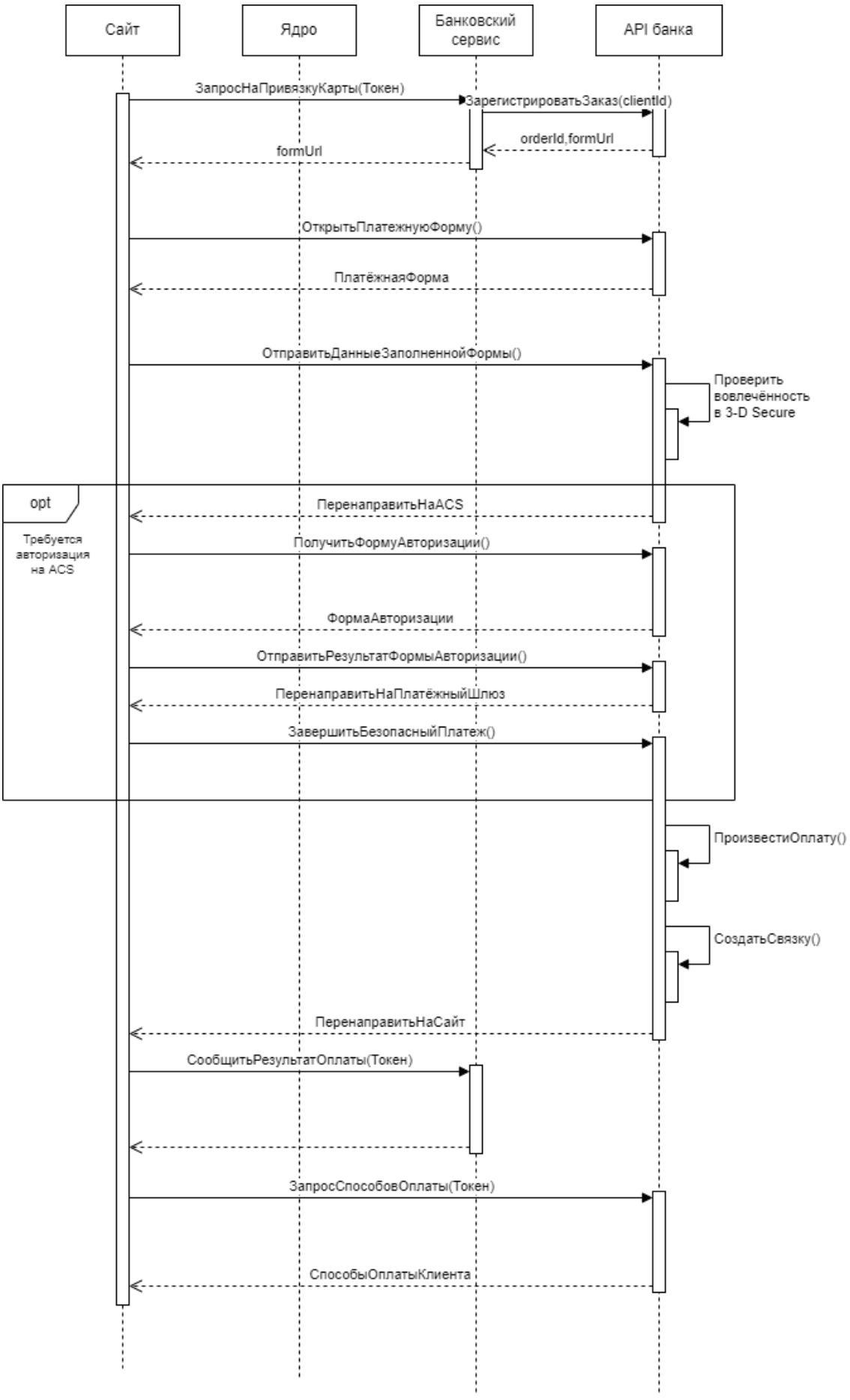

Рисунок Е.5 – Диаграмма последовательности “ЗапросНаПривязкуКарты”

ПРИЛОЖЕНИЕ Ж

Схема базы данных