МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему Исследование методов и моделей интеграции мобильных приложений в корпоративные информационные системы предприятий

Обучающийся А.И. Таиров
(И.О. Фамилия) (личная подпись)

Научный руководитель доцент, Т.А. Раченко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Содержание

Введение

В современном мире большинство систем работают, взаимодействуя друг с другом и обмениваясь данными. Такой способ работы вызывает вопрос интеграции этих систем и сервисов. «От выбранного способа, метода и модели интеграции зависит скорость работы систем и общее качество продукта» [1].

Сложность интеграции систем заключается в необходимости учета платформ, которые эти системы используют. Так, мобильное приложение может хранить кеш, но мобильные устройства могут использовать только Wi-Fi или мобильный Интернет, поэтому обмен большим количеством данных или частые запросы могут негативно сказаться на скорости работы мобильного приложения. В то же время веб-приложениями могут пользоваться как с мобильного устройства, так и с компьютеров. Тип используемой платформы необходимо учитывать при интеграции систем.

Интеграция мобильного приложения с ERP-системой является способом повышения комфортабельности работы с ERP-системой. Для руководителей и владельцев бизнеса использование мобильного приложения позволяет иметь удаленный доступ к отчетам в режиме реального времени, а также получать и утверждать заявки, находясь в любой точке. Для тех сотрудников, которые много времени проводят в командировках и должны отчитываться о своих расходах, использование мобильного приложения позволит сэкономить время и упростить процесс оставления отчетности. С помощью мобильных инструментов ERP-систем можно настроить автоматическую отправку всех расходов с помощью своего смартфона или планшета.

Большие возможности открывает использование мобильных приложений в логистике и на производстве. Если груз задержался в пути или случилась какая-либо поломка, менеджеры могут получать информацию об этих проблемах незамедлительно, а не только тогда, когда находятся рядом со своим компьютером.

Для реализации функций ERP-системы в мобильном приложении нет

необходимости в имплементации всей логики фронтенда и бэкенда ERP-системы в мобильном устройстве, так как это повлечет слишком большую, невозможную с технической точки зрения, нагрузку для пользователей мобильных устройств. В качестве решения предлагается использовать модель, при которой мобильное приложение, обладающее собственным интерфейсом для работы с пользователем, отправляет запросы с небольшим количеством данных в ERP-систему, которая может обработать запрос на своем сервере и при необходимости получить дополнительные данные из связанных систем, не затрагивая при этом мобильное приложение. Необходимо также учитывать, что мобильное приложение и ERP-система могут изменяться, менять сетевые адреса, порты и расширяться при росте компании. Или компания может отказаться от использования одной ERP-системы, предпочитая ее какой-либо другой. В таком случае потребуется большое количество времени и усилий для адаптации всех мобильных приложений под новую систему.

Вопрос интеграции систем, в том числе мобильного приложения с внешними системами, является актуальным в связи с высокой и растущей популярностью мобильных приложений, а также в связи с повсеместным использованием ERP-систем.

Объектом исследования является взаимодействие мобильного приложения и корпоративной информационной системы.

Предметом исследования являются модели и методы интеграции, применяемые технологии и инструменты для создания взаимодействия систем.

Целью работы является исследование и анализ существующих методов и моделей интеграции мобильных приложений с корпоративными информационными системами и поиск решения, которое сможет обеспечить высокую эффективность их совместной работы и оптимальную модель интеграции.

Научная новизна работы заключается в улучшенной модели интеграции, позволяющей связать между собой различные системы.

1. Обзор систем и методов их интеграций

1.1 Способы интеграции систем

На многих предприятиях и в организациях, в том числе и российских, созданы ИТ-инфраструктуры, большое множество прикладных систем (персональный учет, бухгалтерский учет, ERP и CRM-системы, биллинг). «Эти ИТ-системы собирают исходную информацию для поддержки принятия оперативных решений. Однако трудность заключается в том, что эти «необработанные» данные должны быстро собираться и обрабатываться (проверяться, отслеживаться, агрегироваться, приводить к единообразным форматам и анализироваться), и для этого необходимо использовать интеграционные и аналитические технологии» [2].

Существует и другая проблема. «По мере увеличения числа систем на предприятиях и в организациях проблема их интеграции выходит на первый план: внедрение новых, замена или модернизация существующих систем обычно требует либо создания интеграционных связей с нуля, либо изменения связей между системами. Более того, постоянное развитие бизнес-процессов приводит к постоянным изменениям в интеграционных связях между приложениями, даже при том условии, что новые приложения не появляются» [3].

На сегодняшний день наиболее успешным подходом к решению задач интеграции систем является разработка корпоративной информационной системы (КИС) предприятия в соответствии с концепцией сервис-ориентированной архитектуры (Service-Oriented Architecture, SOA), появившейся в конце 1980-х в идеях CORBA, DCOM, DCE.

Можно выделить несколько сценариев применений сервис-ориентированной архитектуры:

– управление услугами и бизнес-процессами,

– управление крупными территориально-распределенными

компаниями,

– управление процессами, требующими интенсивной обработки документов.

Существует еще один подход к построению КИС - платформа SAP, которая была разработана для комплексной автоматизации крупных предприятий. Структура платформы представлена на рисунке 1.

Рисунок 1 - Структура платформы SAP

«SAP ERP реализована по архитектуре клиент/сервер, что позволяет масштабировать систему в зависимости от желания организации. Эта система позволяет объединять несколько предприятий или филиалов в единую информационную среду независимо от их местонахождения» [4].

«Одним из наиболее значимых направлений развития современных корпоративных информационных систем является их ориентация на интеграцию друг с другом за счет формирования единого целостного информационного пространства (ЕИП) конкретного предприятия» [5]. Схема организации информационной системы предприятия с помощью ЕИП представлена на рисунке 2.

Рисунок 2 - Обобщенная схема ЕИП

«С точки зрения практической значимости ЕИП направлено на сохранение целостности данных и возможность их дальнейшего использования разными пользователями в различных информационных системах в соответствии с профилем их деятельности. Используя ЕИП, представляется возможным выстроить процессный подход к управлению, который будет преследовать цель, заключающуюся в устранении фрагментации в работе, путем сокращения организационных и информационных недостатков и недочетов. Организации единого информационного пространства активно изучаются учеными последние несколько лет, но основная проблема состоит не только в выборе и развитии технологий интеграции ИС, но также и в создании неразрывного, единого представления данных. Следовательно, новизна существующих исследований, связанных с ЕИП, опирается на исследование методов структурирования информации, получаемой из различных информационных систем. Для разработки этой концептуальной основы ЕИП использует принципы теории множеств, графов, управления проектами и т.д.» [5].

Единое информационное пространство можно организовать путем интеграции всех информационных систем предприятия с целью поддержания

целостности данных в любой момент времени, так как сейчас на любом предприятии существует множество информационных и программных систем, и их можно объединить в единое пространство, сохраняя консистентность данных.

Как правило, информационная система состоит из следующего набора компонентов:

– платформа, на которой работают базовые компоненты системы, включая аппаратное и системное программное обеспечение;

– приложения, которые реализуют бизнес-логику для работы с системой данных. В свою очередь, она включает такие компоненты как: бизнес-логика пользовательского интерфейса, компоненты поддержки (интерфейсы) и серверы приложений, которые обеспечивают хранение и доступ к компонентам приложения;

– данные, с которыми работает система, состоящая из различных СУБД (система управления базами данных);

– бизнес-процессы, представляющие собой сценарии работы пользователя с данной системой.

Архитектура информационной системы в общем виде представлена на рисунке 3. Протоколы взаимодействия приложений определяют, как передаются данные между приложениями.

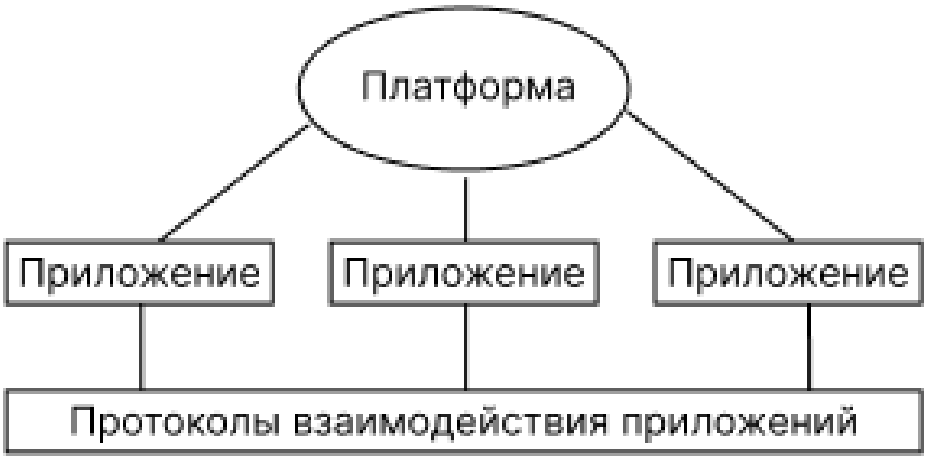

Рисунок 3 - Архитектура информационной системы

«Существуют различные подходы для интеграции приложений, и в каждом из этих подходов могут быть использованы различные технологии:

– технология виртуализации,

– удаленный вызов процедур (REST, RPC, Web-сервисы и пр),

– ПО промежуточного слоя (Microsoft.Net, Java Runtime)» [7].

Исходя из определения, информационная система работает с разнообразными хранилищами данных. Интеграция на уровне данных использует обмен данными из различных систем. Эти данные могут быть связаны более простым образом на этом уровне, нежели на уровне приложений, так как современные СУБД могут эффективно поддерживать различные протоколы для разных платформ, что значительно упрощает доступ к данным.

Есть два основных подхода к интеграции данных:

– хранилище данных,

– универсальный доступ к данным.

«Концепция хранилища данных представляет собой объединение данных из нескольких разнородных источников, которые хранятся с использованием различных технологий и обеспечивают единое представление данных» [7]. Интеграция данных становится все более важной и необходимой

в случае слияния систем двух компаний или объединения приложений в одной компании для обеспечения единого представления данных компании.

«Наиболее известной реализацией интеграции данных является создание warehouse (хранилища данных предприятия). Это позволяет компании самостоятельно проводить необходимые анализы на основе данных в хранилище данных. Это было бы невозможно сделать с данными, доступными только в исходной системе. Причина невозможности реализации состоит в том, что исходные системы могут не содержать соответствующих данных из других систем, и даже если данные имеют одинаковые имена, они могут ссылаться на абсолютно разные объекты» [8].

«Интеграция на уровне приложений (или интеграция корпоративных приложений) – это разделение процессов и данных между различными приложениями. Как для небольших, так и для крупных организаций стало критически важно и приоритетно подключение разрозненных приложений и использование совместной работы приложений в масштабах целого предприятия для повышения общей эффективности бизнеса, повышения масштабируемости и снижения затрат на ИТ» [9].

Существуют следующие подходы к интеграции приложений:

– сервис-ориентированная архитектура,

– обмен сообщениями (корпоративная сервисная шина),

– интерфейсы,

– интеграция юзер-интерфейсов (на уровне представления).

«Программный интерфейс (API - application programming interface) - это набор инструментов, определений и протоколов для построения и интеграции прикладного программного обеспечения. Он позволяет приложению или сервису взаимодействовать с другими продуктами и услугами, не зная, как они реализованы» [10].

«Сервис-ориентированная архитектура (SOA) - модульный подход к разработке программного обеспечения, основанный на использовании распределенных, слабо связанных компонентов, оснащенных

стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам» [11].

Она включает в себя набор принципов проектирования, которые структурируют разработку системы и предоставляют средства для интеграции компонентов в целостную и децентрализованную систему.

Архитектура объединяет все взаимодействующие сервисы, которые могут быть интегрированы в программные системы, принадлежащие отдельным бизнес-доменам.

Сервис-ориентированная архитектура содержит две важные роли. Сервис-провайдер представляет из себя сервис, которые предоставляет сервисы и функционал для использования другими.

Сервис-консьюмер, который содержит в себе мета-данные и все необходимые для работы клиентские компоненты.

При таком походе приложение состоит из нескольких уровней. Эту структуру можно видеть на рисунке 4.

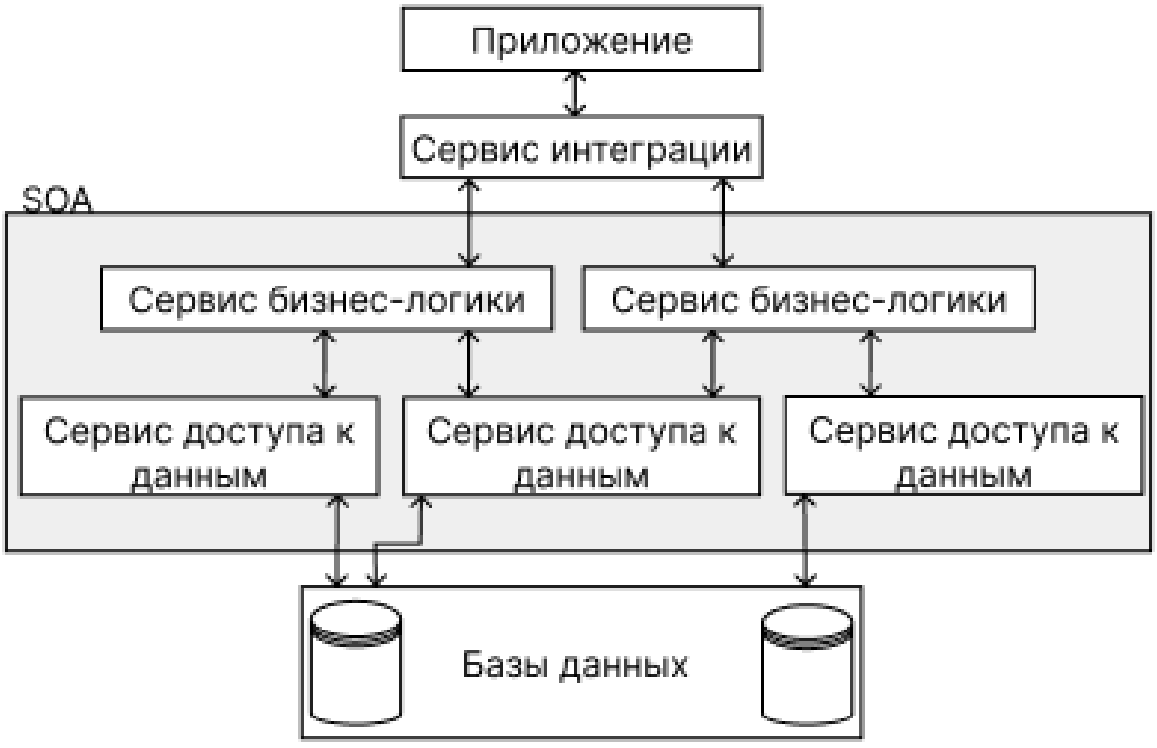

Рисунок 4 - Уровни сервис-ориентированного приложения

Самый верхний уровень содержит службы интеграции, каждая из которых контролирует потоки запросов. Каждый сервис интеграции вызывает один или несколько бизнес-сервисов.

Второй уровень состоит из бизнес-сервисов, каждый из которых выполняет бизнес-задачу более низкого уровня.

«Третий уровень приложения состоит из служб доступа к данным, каждая из которых выполняет задачу чтения и записи в области хранения данных, такие как очереди сообщений и базы данных. Служба доступа к данным обычно вызывается из бизнес-уровня» [11].

«Наиболее целостный подход к системной интеграции - это интеграция на уровне бизнес-процессов. На этом уровне интеграции происходит интеграция данных, интеграция приложений и людей. Предприятия могут использовать интеграцию приложений для определения взаимодействия отдельных приложений с целью автоматизации важных бизнес-процессов, что приводит к снижению вероятности человеческих ошибок, более быстрой доставке товаров и услуг для клиентов и снижению эксплуатационных расходов» [12].

Основными преимуществами интеграции на уровне бизнес-процесса является возможность гибкой настройки взаимодействия систем и независимость от реализаций этих систем.

К недостаткам можно отнести высокую вероятность ошибок при создании интеграции и необходимость дополнительно реализовывать модули, если существующих не хватает для создания взаимодействия.

Целью моделирования является устранение недостатков интеграции на уровне бизнес-процессов путем ввода платформы, способной связать между собой различные системы без разработки дополнительных модулей.

Задачи, которые необходимо для этого решить, следующие:

– выбрать способ взаимодействия систем с платформой,

– изучить и выбрать наилучший способ передачи данных между приложениями,

– решить проблему зависимости платформы от конкретных систем.

Существуют следующие модели создания системной интеграции:

– Point-to-point («точка-точка») - прямой канал связи между двумя системами без посредников или промежуточных узлов. Подход является самым простым, поскольку системы взаимодействуют и обмениваются данными напрямую. Существенным минусом такого подхода является сложность масштабирования, поскольку все системы являются взаимосвязанными. Пример связи систем с применением данной модели указан на рисунке 5;

– Hub-and-spoke («звезда») - отличается от модели «точка-точка» тем, что в такой модели используется дополнительный компонент - «концентратор», который является центральным узлом для всех других систем, объединенных в сеть. Концентратор отвечает за прием и передачу данных между различными системами, которые называются «спицами». Спицы взаимодействуют с концентратором, но не друг с другом. Наибольший риск вызывает выход концентратора из строя, поскольку в таком случае вся сеть окажется выведенной из строя. Для предотвращения подобной проблемы могут использоваться несколько концентраторов. Пример использования такой модели представлен на рисунке 6;

– Enterprise Service Bus - метод системной интеграции, который служит централизованным посредником между различными системами. Данная модель добавляет к использованию концентратора дополнительные функции, такие как преобразование данных, маршрутизацию и безопасность.

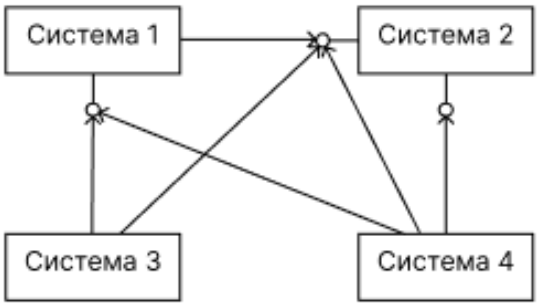

Рисунок 5 - Интеграция «точка-точка»

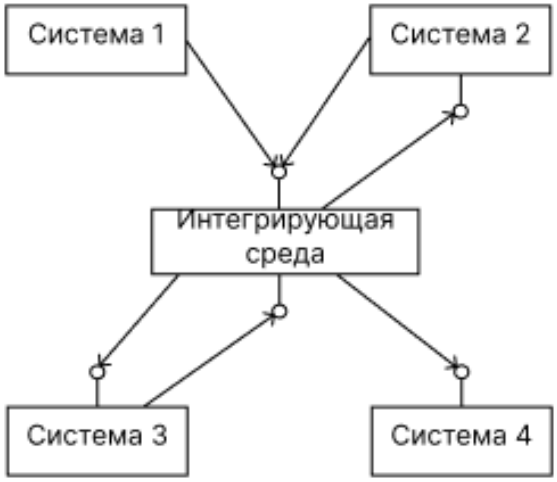

Рисунок 6 - Интеграция «звезда»

Для интеграции мобильного приложения с внешними системами лучше всего подходит интеграционная модель «звезда», поскольку концентратор обеспечивает одинаковые условия работы для всех пользователей и в случае появления каких-либо ошибок их будет проще увидеть с помощью логов концентратора, чем в мобильном приложении.

1.2 Обзор и классификация ERP-систем

«Enterprise resource planning (ERP) - это класс систем, разработанных в 60-х годах прошлого века и предназначавшихся для ведения

производственного учета и организации процессов производства и являющихся расширением такого понятия как MRP (MRP-II) для разных областей бизнеса» [10].

С самого начала ERP-системы позиционировались как системы, которые решают вопросы организации данных и оптимизации процессов, связанных с функционированием бэк-офиса и решают задачи планирования ресурсов. На рисунке 7 представлены составы ERP-систем от самого их начала и до введения термина ERP.

Рисунок 7 - Состав ERP-систем

Типичная ERP-система представляет собой набор модулей, каждый из которых управляет определенным процессом: продажами, закупками, производством, кадровыми процессами, бухгалтерским и налоговым учетом, поддержкой клиентов, складской логистикой CRM и т.д. При этом система охватывает основные процессы всех направлений деятельности предприятия.

Можно сделать вывод, что ERP-система представляет собой комплексную информационную систему управлений информацией внутри организации, решающую комплекс задач регламентированного, управленческого и других видов учета. На рисунке 8 представлены функциональные возможности одной из самых популярных ERP-систем в

России - 1С:ERP.

На сегодняшний день международный рынок ERP-систем огромен и постоянно продолжает расти. Преимущественно на рынке представлены отраслевые решения, разработанные под нужды определенных отраслей промышленности: логистика, розничная торговля, конкретные направления производства. Однако в лидерах международного рынка продолжают оставаться компании, предлагающие максимально комплексные решения.

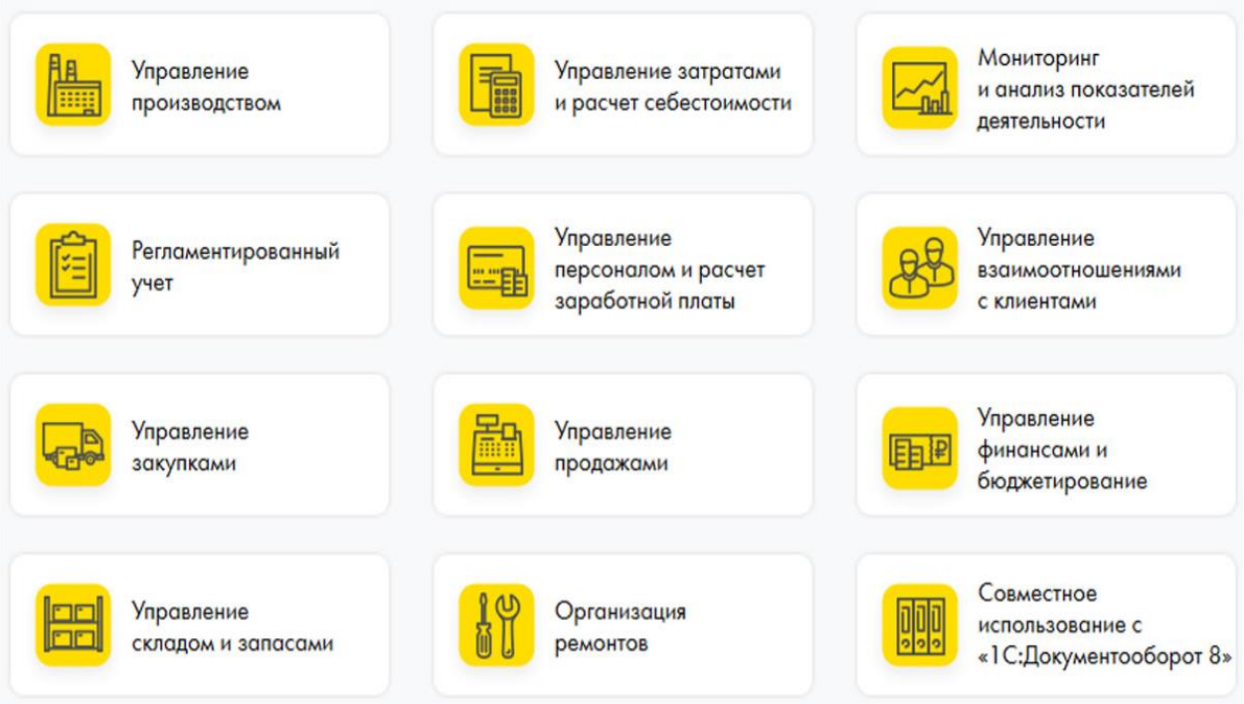

Рисунок 8 - Функциональные возможности ERP-системы

По области применения ERP-системы можно разделить на следующие группы:

– комплексные информационные системы. В данной группе находятся универсальные ERP-системы. Их можно адаптировать к процессам самых разных по величине компаний. Такие системы имеют широкий диапазон настроек и развитые механизмы интеграции, чтобы быть универсальными и соответствовать требованиям различных отраслей бизнеса. Примеры таких систем: Oracle, SAP, Netsuite, 1С;

– отраслевые информационные системы. Системы в этой группе ориентированы на конкретную отрасль или даже на узкое направление внутри отрасли. Такие продукты обычно выпускают стартапы или компании, которые не могут конкурировать с крупными компаниями из прошлого пункта и пытаются найти свою нишу и занять в ней лидирующее положение. Примеры: Microsoft Dynamics AX, Brightpearl, Epicor Retail;

– ERP для малого бизнеса. Системы из этой группы не совсем являются ERP-системами, но небольшим компаниям хватает функциональности для удовлетворения всех своих нужд при более низкой стоимости системы. Такие ERP-системы являются модульными и обладают урезанным функционалом. Пример: PeopleSoft.

Основными преимуществами, которые дает внедренная на предприятии ERP-система, являются:

– встроенные инструменты прогнозирования, используемые для принятия обоснованных решений о дальнейших шагах по развитию бизнеса;

– более глубокое и детальное понимание процессов, происходящих в компании, и уменьшение времени реакции на изменения. Благодаря консолидации всей ключевой информации в одной единой системе, появляется возможность оперативного получение управленческой отчетности по всем аспектам деятельности предприятия в режиме реального времени;

– консолидация данных и операций в одной системе является обеспечением сопоставимости данных, устранением дублирования и формированием единого видения происходящих процессов у всех участников;

– единообразный IT-ландшафт, позволяющий ERP-системе повысить безопасность при хранении данных, упростить задачи ограничения доступа и тем самым повысить уровень информационной безопасности;

– минимальные издержки за счет внедрения единых сквозных бизнес-процессов, автоматизации трудоемких задач, ликвидации избыточных процессов, а также упрощения процедур обучения и адаптации новых пользователей.

Однако ERP-системы обладают также большим количеством недостатков. К ним относятся:

– недостаточная универсальность. Несмотря на то, что ведущие производители пытаются сделать решения максимально комплексными и гибкими, подходящими под любые требования бизнеса, это не всегда удается. На рынке может отсутствовать полностью подходящее решение, поэтому часто требуется адаптация продукта под конкретные нужды конкретной организации, и это значительно увеличивает стоимость проекта;

– высокая стоимость внедрения и владения. Классическая схема внедрения ERP-систем подразумевает крупные первоначальные затраты при внедрении, причем до того, как система заработает и бизнес получит свои преимущества;

– высокие риски внедрения. Существует большое количество различных сложностей при внедрении, например: особенности работы программного обеспечения, которое необходимо учесть при внедрении нового решения, трудность принятия изменений для персонала и отсутствие квалифицированных кадров на предприятии, которые способны организовать и провести процесс перехода и дальнейшей поддержки.

Таким образом, ERP-системы вносят существенный вклад в инфраструктуру предприятия, но необходимо также учитывать сложность их внедрения и большие затраты времени и ресурсов.

1.3 Особенности мобильных систем и их интеграций

Для интеграции мобильного приложения с ERP-системой предприятия необходимо выявить отличия мобильного приложения от веб-приложения.

Ключевыми отличиями являются:

– простота доступа - для использования веб-приложения необходимо использовать браузер и иметь постоянное подключение к Интернету. Для дальнейшего использования, как правило, веб-приложение требует

регистрации пользователя. Однако мобильное приложение также требует предоставления персональной информации. Отличием является то, что браузер обычно устанавливается на смартфон ещё до продажи и веб-приложение одинаково работает на любом устройстве, в то время как мобильное приложение необходимо установить из соответствующего плеймаркета;

– скорость обновления - для обновления мобильного приложения разработчики сначала должны загрузить обновление во все плеймаркеты и дождаться, когда пользователи обновят приложение на своем устройстве. Веб-приложения обновляются централизованно. То есть, если в приложение был добавлен новый функционал или были совершены ещё какие-то изменения, то при следующем обращении к сайту пользователь будет работать с уже обновленной версией;

– стоимость продвижения - веб-приложения ищутся через поисковик браузера, автоматически индексируясь, и пользователи могут взаимодействовать с сайтом сразу после перехода на него. Мобильное приложение необходимо продвигать через плеймаркеты и учитывать рейтинг, который ставят пользователи. В то же время нет никаких гарантий, что пользователь решит установить мобильное приложение;

– быстродействие - скорость работы веб-приложения зависит не только от его оптимизации, но и всегда зависит от скорости Интернета;

– оффлайн-доступ - мобильные приложения зачастую обладают функциями, которые доступны без подключения к Интернету;

– интеграция с операционной системой мобильного устройства - веб-приложения могут запросить доступ к камере или отправке пуш-уведомлений, но после закрытия вкладки или браузера эти функции будут недоступны. Мобильные приложения используют большее число функций мобильных устройств: открываются без дополнительной авторизации, принимают данные с камеры и микрофона, отслеживает геолокацию, отправляют пуш-уведомления.

Для интеграции мобильного приложения с одной и более системой часто используется промежуточный сервер, представляющий собой дополнительный бэкенд, который объединяет всю доступную информацию из нескольких сервисов в формат, доступный для использования в мобильном приложении.

Такой подход имеет следующие преимущества:

– упрощенное подключение сложной инфраструктуры,

– легкость масштабирования,

– низкая нагрузка на мобильного клиента.

Общий принцип связи систем с помощью промежуточного сервера представлен на рисунке 9.

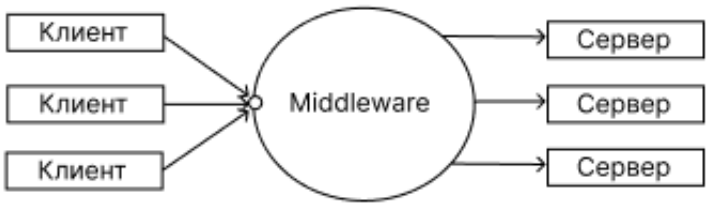

Рисунок 9 - Схема взаимодействия сервисов через middleware

Данный способ интеграции довольно удобен при использовании в сложной архитектуре, но обладает следующими недостатками:

– промежуточный бэкенд не может решить проблему производительности и не может работать быстрее исходного API,

– не предоставляет никакой защиты от багов бизнес-логики,

– сложность обнаружения источника ошибок.

В рамках первой главы были рассмотрены существующие методы и модели интеграций систем, выявлены отличия мобильных приложений от других систем, изучены особенности корпоративных информационных систем.

2. Построение модели интеграции систем

2.1 Анализ методологий для разработки моделей и алгоритмов

Любое программное обеспечение, ИТ-решение и пр. разрабатывается по определенному алгоритму. В том случае, если разработчик с самого начала не придерживается никакой последовательности и дисциплины, которые позволяют выделить на каждом этапе конкретные задачи и поставить определенные подцели, то проследить взаимосвязь между задачами очень сложно. Такой подход является пошаговым и начинается обычно со спецификации, которая получается в результате анализа общей задачи. На каждом этапе разработки алгоритма принимается небольшое число решений, детализирующих и уточняющих структуру алгоритма. Таким образом, на выходе получается информативный и детально проработанный алгоритм, отражающий последовательность работы итоговой версии программы.

Метод пошаговой разработки позволяет разбить алгоритм на модули, каждый из которых отвечает за какую-то небольшую подзадачу (по аналогии с микросервисной архитектурой). Такой подход дает возможность сосредоточиться на решении подзадачи, которая может быть реализована в виде отдельной функции или процедуры. Связь между модулями и их управление осуществляются с помощью вызовов, а передача параметров происходит через глобальные переменные и параметры.

Наряду с использованием метода пошаговой разработки необходимо также иметь в виду следующие факторы, которые могут существенно повлиять на разрабатываемый алгоритм:

– возможности, предоставляемые языком программирования, который будет использоваться для разработки алгоритма. Например, в языке Java будет использоваться исключительно объектно-ориентированное программирование, могут использоваться шаблонные типы и асинхронные потоки, в то время как C++ позволяет использовать императивное

программирование и создавать вручную всю структуру данных. От выбранного языка программирования будут существенно различаться возможности разрабатываемого алгоритма;

– структуры данных, на которые ориентирован алгоритм. Этот фактор оказывает огромное влияние на эффективность разрабатываемого алгоритма;

– модель - объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств последнего; упрощенное представление системы для её анализа и предсказания, а также получения качественных и количественных результатов, необходимых для принятия правильного управленческого решения.

Объектом исследования в данной работе является модель интеграции, поэтому этот фактор стоит рассмотреть более внимательно.

При решении конкретной задачи необходимо постоянно выявлять определенные свойства изучаемого объекта. В таком случае модель оказывается зачастую единственным инструментом исследования. Исследуемый объект в одно и то же время может иметь множество моделей, а одна модель может описывать разные объекты.

«Под термином «моделирование» обычно понимают процесс создания точного описания системы; метод познания, состоящий в создании и исследовании моделей. Моделирование облегчает изучение объекта с целью его создания, дальнейшего преобразования и развития. Оно используется для исследования существующей системы, когда реальный эксперимент проводить нецелесообразно из-за значительных финансовых и трудовых затрат, а также при необходимости проведения анализа проектируемой системы, т.е. которая еще физически не существует в данной организации» [12]. Обычно различают реальное (материальное, предметное) и мысленное (идеализированное, концептуально-методологическое) моделирование.

Для формирования модели используются:

– структурная схема объекта,

– структурно-функциональная схема объекта,

– алгоритмы функционирования системы,

– схема расположения технических средств на объекте,

– схема связи и др.

Для проектирования ИС используют информационные модели, представляющие объекты и процессы в форме рисунков, схем, чертежей, таблиц, формул, текстов.

Информационная модель – это модель объекта, процесса или явления, в которой представлены информационные аспекты моделируемого объекта, процесса или явления.

При изучении нового объекта сначала обычно строится его описательная модель, затем она формализуется, то есть выражается с использованием математических формул, геометрических объектов и т.д.

Концептуально-методологическое моделирование представляет собой процесс установления соответствия реальному объекту некоторой абстрактной конструкции, позволяющий получить характеристики объекта. Данная модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности.

2.2 Особенности взаимодействия систем

В мире современных технологий одну из ключевых ролей играют способы, которые используют приложения для связи между внутренними компонентами и с другими системами. От неправильного выбора способа взаимодействия системы могут некорректно передавать друг другу данные, терять их и нарушать весь процесс. Кроме того, неправильный способ взаимодействия может очень сильно тормозить системы и показывать множество ошибок, что является неприемлемым способом устройства современной системы.

Рассмотрим самые популярные способы связи систем и их преимущества и недостатки.

REST – это стиль проектирования API, передающий состояние представления. Для передачи и получения информации используется предопределенный набор операций, таких как GET, POST, PUT и т.д.

Главными преимуществами REST являются его масштабируемость и гибкость, простота в использовании.

Основными недостатками REST являются избыточность данных и недостаточная их выборка. К примеру, если пользователь хочет получить номер телефона клиента, то получит всю его личную информацию. В то же время пользователь вынужден делать несколько запросов на различные API для получения всей необходимой информации.

GraphQL – стиль проектирования API, в котором все рассматривается как граф, в котором данные взаимосвязаны. Такая связь позволяет получить из одной конечной точки всю требуемую информацию.

Преимуществами GraphQL являются быстрая разработка запросов на клиенте и то, что пользователь получает только ту информацию, которую он запрашивает. Учитывая то, что структура данных обычно определяется в самом начале разработки проекта, то разработка клиентской и серверной частей может развиваться отдельно, что очень удобно для развития проектов.

Недостатками GraphQL являются сложность настройки (для небольших приложений эта сложность является избыточной) и отсутствие кеширования, поскольку GraphQL использует всего одну точку входа и это не позволяет следовать спецификации HTTP-кеширования.

Web-сокеты – это протокол веб-коммуникации, который представляет из себя полнодуплексный канал коммуникации поверх TCP-соединения.

Основным преимуществом такого подхода является минимизация затрат для обмена данными. Благодаря этому преимуществу, технология обычно используется в чатах, социальных и новостных лентах и приложениях, работающих на основе местоположения.

Однако данный подход обладает рядом недостатков:

– лишняя сетевая нагрузка - клиент вынужден постоянно отправлять

запросы для поддержания соединения, даже если новой информации нет;

– высокие накладные расходы при установке нового соединения;

– низкая частота обновления данных, так как сервер обновляет данные только тогда, когда получает запрос от клиента. Если сделать частоту слишком высокой, сильно вырастает сетевая нагрузка.

Очередь сообщений - это форма асинхронной коммуникации между сервисами. Преимущества использования очереди сообщений следующие:

– логическое отделение независимых компонентов друг от друга,

– улучшение масштабируемости,

– балансировка нагрузки,

– повышение надежности,

– улучшение безопасности.

Однако очередь сообщений также обладает недостатками:

– очередь сообщений является отдельной системой, которую нужно покупать и устанавливать;

– при выходе из строя брокера сообщений может остановиться работа многих систем;

– требуется добавление системы трассировки для отладки системы;

– требуется выбор стратегии передачи сообщений.

Асинхронные операции могут выполнять несколько задач параллельно, в одном или нескольких потоках, в то время как синхронные программы имеют очередь задач, в которой каждая вторая задача остается бездействующей, пока первая задача не будет завершена.

Разница синхронного и асинхронного взаимодействий представлена на рисунке 10.

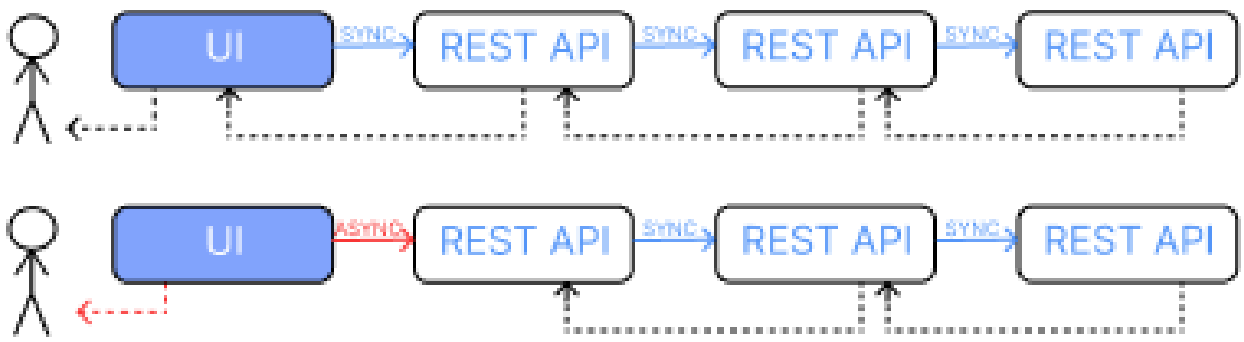

Рисунок 10 - Разница синхронного и асинхронного взаимодействий

Асинхронный способ взаимодействия значительно ускоряет взаимодействие клиента с сервером, но есть большой риск того, что в случае сбоя в одном из API информация, которую отправил клиент, будет потеряна. Также недостатком является то, что для корректной работы очереди сообщений необходимо использовать брокер сообщений.

Все эти подходы имеют свои преимущества и недостатки. Для устранения большинства недостатков требуется создать такое решение, которое позволит гибко настраивать точки входа и будет взаимодействовать с внешними системами.

2.3 Построение модели интеграции

Для разработки новой модели интеграции различных систем предлагается использовать метод пошаговой разработки. Для этого необходимо разделить алгоритм на небольшие подзадачи.

Первым шагом для построения модели интеграции необходимо провести поиск и анализ существующих решений.

Самым простым способом интеграции является интеграция «точка-точка». В процессе анализа данного способа мы неизбежно придем к решению, отраженному в интеграции с использованием ESB - использование концентратора для повышения возможностей настройки интеграций и

добавление к концентратору дополнительных модулей, таких как модуль авторизации или модуль шифрования.

Следующим этапом развития интеграции систем является разбиение ESB на микросервисы, где каждый микросервис будет отвечать за отдельную бизнес-логику и самостоятельно ее реализовывать. Примером реализации такой системы является «7TECH Integra».

«Данное решение представляет собой платформу, позволяющую постепенно интегрировать системы, не нарушая целостность выстроенной ИТ-инфраструктуры, предоставляет low-code интерфейс для конфигурации платформы под конкретного заказчика, является распределенной, что обеспечивает высокую надежность и скорость работы» [10].

Возможности, которые предоставляет интеграционная платформа, следующие:

а) административная панель:

1) управление контейнерами,

2) управление брокером сообщений,

3) управление BPMN процессами,

4) управление потоками,

5) просмотр файлов логирования,

6) меню;

б) управление правами доступа:

1) управление пользователями,

2) права доступа;

в) инфраструктурные отчеты:

1) доступность узла,

2) утилизация CPU,

3) утилизация разделов HDD,

4) показатели нагрузки ввода вывода,

5) утилизация RAM;

г) прикладные отчеты:

1) статистика сбоев в работе адаптеров,

2) тестовый сервис,

3) сводный отчет по работе адаптеров,

4) отчет по превышению времени исполнения адаптеров,

5) отчет по прохождению процесса в адаптерах;

д) реестр сервисов:

1) описание сервиса,

2) отчет о работе сервиса.

Следующим шагом для построения модели интеграции является анализ решения, найденного на прошлом шаге, и выявление способов его улучшения. Существует несколько моментов, которые могут быть улучшены:

– добавление асинхронного взаимодействия,

– упрощение построения модели бизнес-процесса,

– добавление валидации запросов в адаптерах,

– улучшение логики работы с мобильными приложениями.

Следующим действием по улучшению модели интеграции является применение изменений, реализующих найденные способы улучшения. Модель интеграционной платформы представлена на рисунке 11.

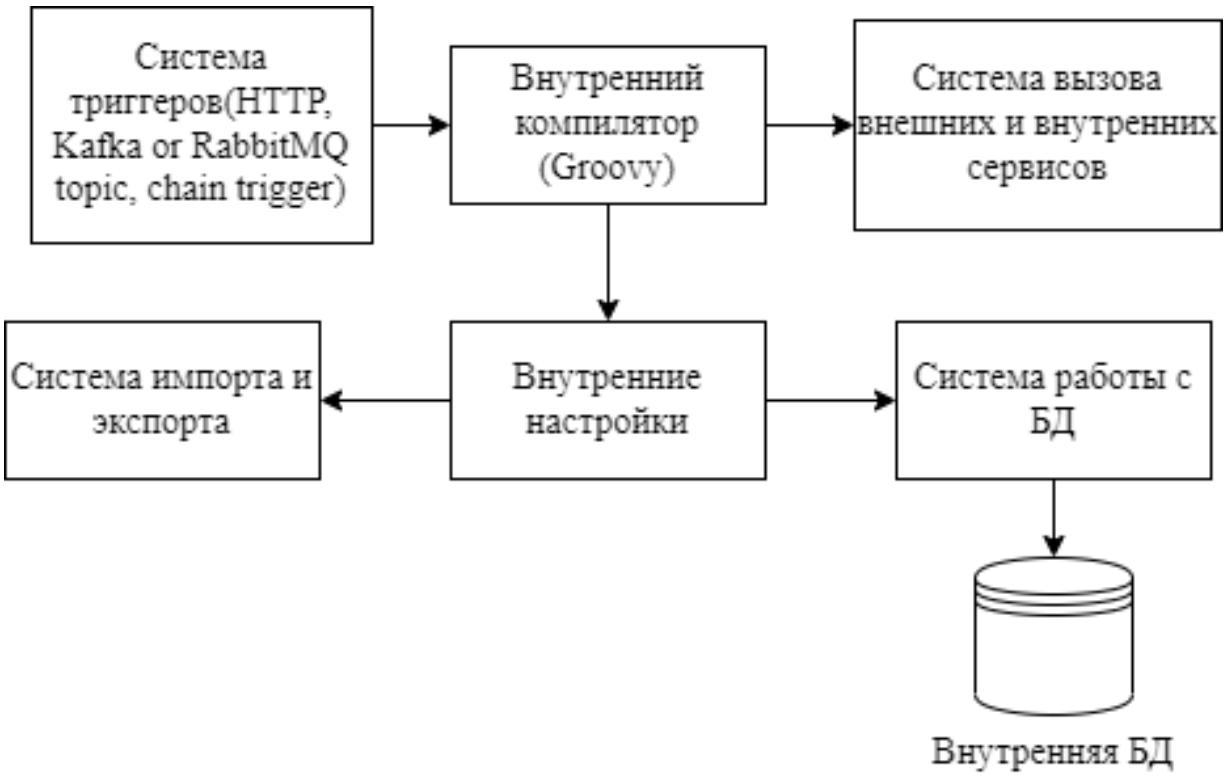

Рисунок 11 - Улучшенная модель интеграционной платформы

Таким образом, на основе интеграционной модели была построена её улучшенная версия. Следующим шагом является применение измененной модели на практике и проверка ее эффективности.

В рамках второй главы были изучены подходы к построению моделей и алгоритмов, выбран метод пошаговой разработки в силу его наглядности, выявлены особенности, преимущества и недостатки методов интеграции систем, рассмотрено и проанализировано существующее решение от компании «7TECH», и на основе интеграционной модели этого решения была получена улучшенная модель взаимодействия мобильного приложения и корпоративной информационной системы.

3. Применение разработанной модели интеграции

3.1 Проектирование и разработка мобильного приложения

Для применения и оценки модели интеграции необходимо использовать мобильное приложение и ERP-систему.

В первую очередь, необходимо реализовать нативное мобильное приложение. Для этого будет использоваться среда разработки Intellij IDEA и язык программирования Java. Выбор сделан с учетом того, что подавляющее большинство людей используют мобильные устройства с операционной системой Android. Об этом говорит статистика за 2022 год, представленная на рисунке 12.

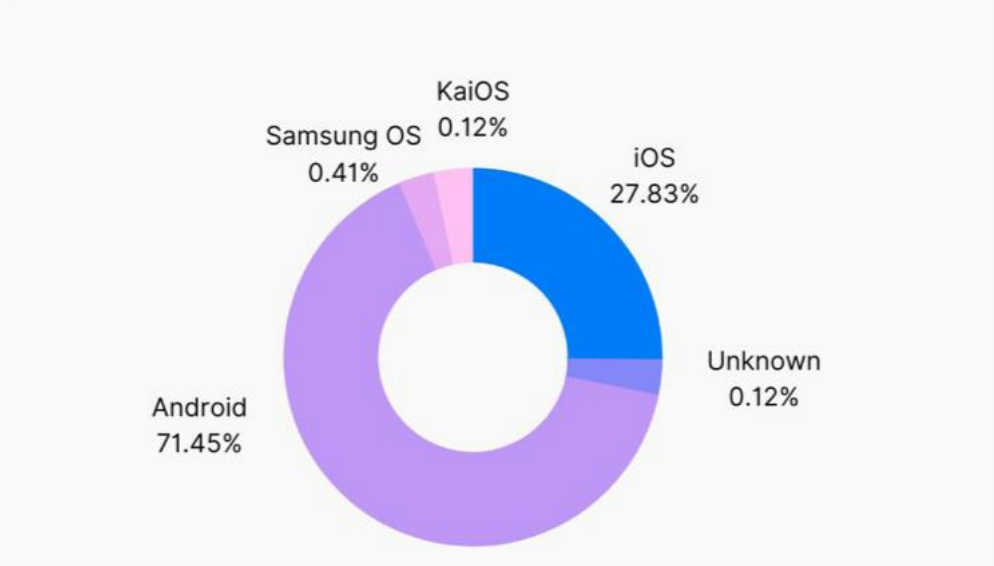

Рисунок 12 - Отношение пользователей Android к другим ОС

В дальнейшем необходима также реализация приложения для операционной системы iOS с использованием языка программирования Swift.

Для реализации промежуточного слоя используется следующий стек технологий:

– в качестве языка для разработки интеграционной платформы был выбран язык программирования Java. «Java - язык программирования,

разработанный компанией Sun Microsystems» [18]. Выбор был сделан из-за поддержки объектно-ориентированного подхода и наличия механизма работы с потоками. Для данного языка существует множество фреймворков, которые ускоряют разработку крупных систем, в том числе с микросервисной архитектурой;

– для реализации работы в асинхронном режиме будут использоваться Kafka и RabbitMQ. Наличие обеих систем необходимо, так как они реализуют принципиально разные модели доставки сообщений. «Kafka - pull (получатели сами достают из топика сообщения), а RabbitMQ - push (отправляет сообщения получателям)» [19];

– в качестве СУБД будет использоваться PostgreSQL, поскольку данная СУБД обладает надежными механизмами транзакций и репликаций и поддерживает БД неограниченного размера. Такие особенности помогут более эффективно реализовать систему кеширования данных на платформе;

– для конфигурации пользователей и их прав доступа будет использоваться Keycloak;

– для логирования запросов, данных и системной информации будет использоваться Grafana совместно с Graylog. «Grafana - это платформа с открытым исходным кодом для визуализации, мониторинга и анализа данных» [20];

– для загрузки и работы платформы на сервере будет использоваться Docker.

Данный стек позволяет полностью реализовать все функциональные возможности, включая логирование запросов, синхронный и асинхронный режим работы, конфигурацию пользователей и их ролей, время выполнения операций, контейнеризацию для дальнейшего развертывания.

Мобильное приложение будет содержать в себе интерфейс, отображающий функциональные возможности ERP-системы, кеш для хранения информации и запросов, необходимые настройки.

Для интеграции была выбрана система 1С:ERP, поскольку данный

продукт является одним из самых популярных и поддерживает функции, доступные в том числе в прочих ERP-системах.

Функции, которые должно поддерживать мобильное приложение, следующие:

а) финансы:

1) бухгалтерский учет,

2) налоговый учет,

3) формирование регламентированной отчетности;

б) управление персоналом и расчет заработной платы:

1) просмотр штатного расписания,

2) составление графика работы и отпусков,

3) учет рабочего времени,

4) отражение изменений условий труда,

5) расчет заработной платы,

6) формирование кадровой отчетности;

в) производство:

1) составление согласованного комплекта планов (производства, снабжения, продаж),

2) балансировка планов,

3) управление графиком производства,

4) управление ресурсами,

5) управление логистикой,

6) анализ состояния,

7) диагностика этапов,

8) планирование и выполнение операций;

г) цепочка поставок:

1) инвентаризация,

2) управление запасами.

В первую очередь, для использования мобильного приложения и доступа к ERP-системе, необходимо пройти авторизацию для получения

доступа к персональным данным и дальнейшему использованию функций ERP. Для авторизации достаточно указать логин и пароль пользователя. Окно авторизации представлено на рисунке 13.

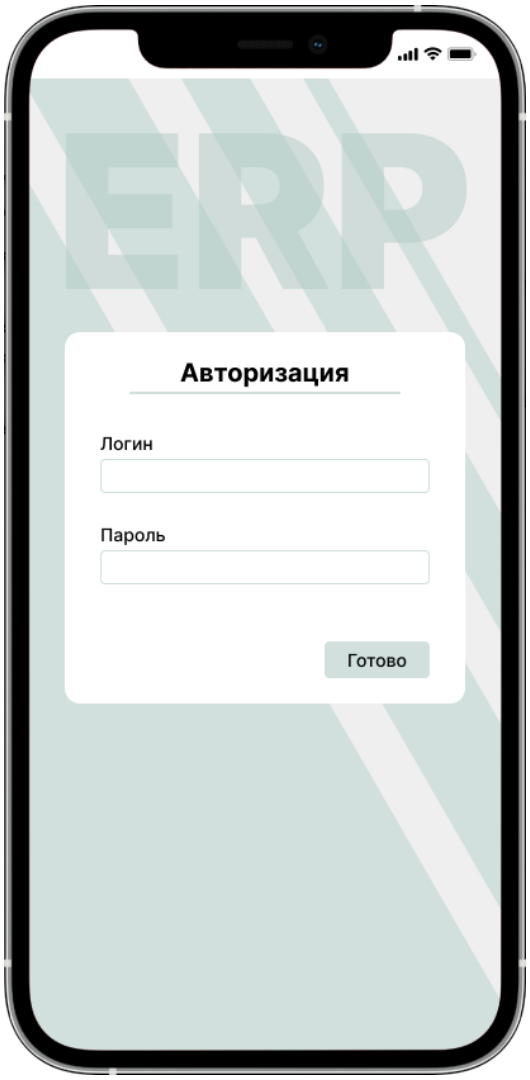

Рисунок 13 - Окно авторизации

После авторизации пользователь видит основное меню, откуда может перейти к функциям, которыми он пользуется чаще всего. Также в основном меню отображаются уведомления для пользователя. Вид основного меню представлен на рисунке 14.

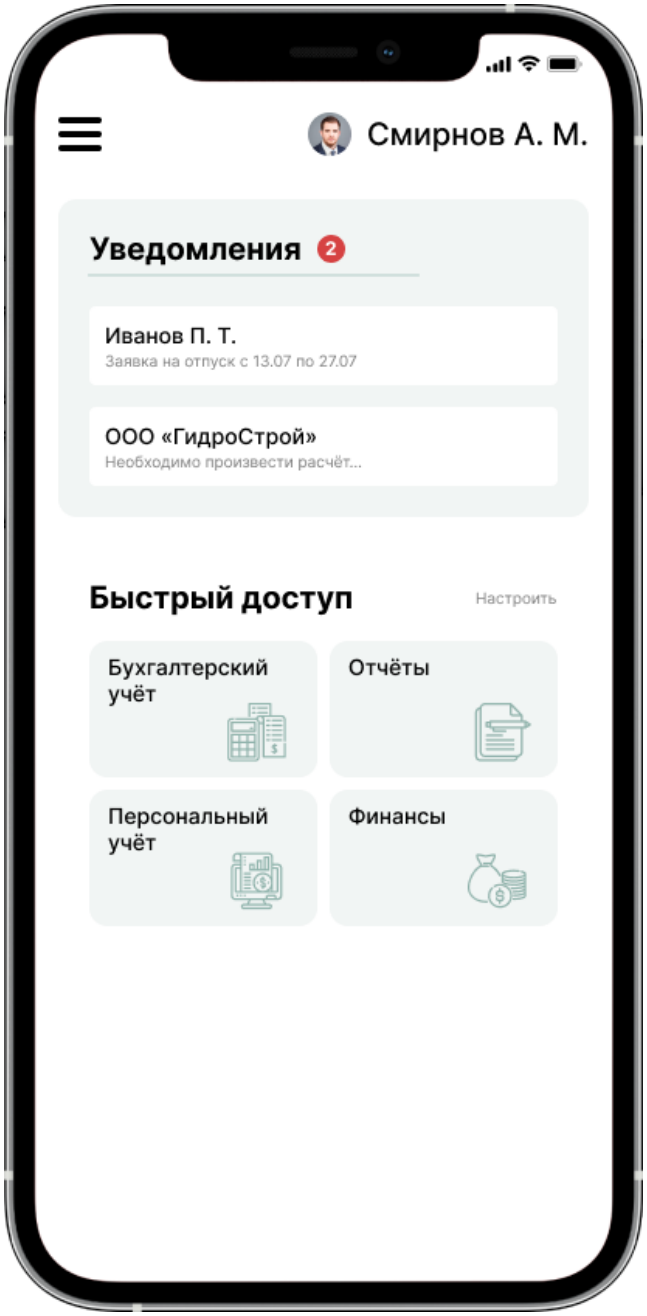

Рисунок 14 - Стартовый экран приложения

Для доступа ко всем функциям ERP-системы необходимо открыть меню, представленное на рисунке 15.

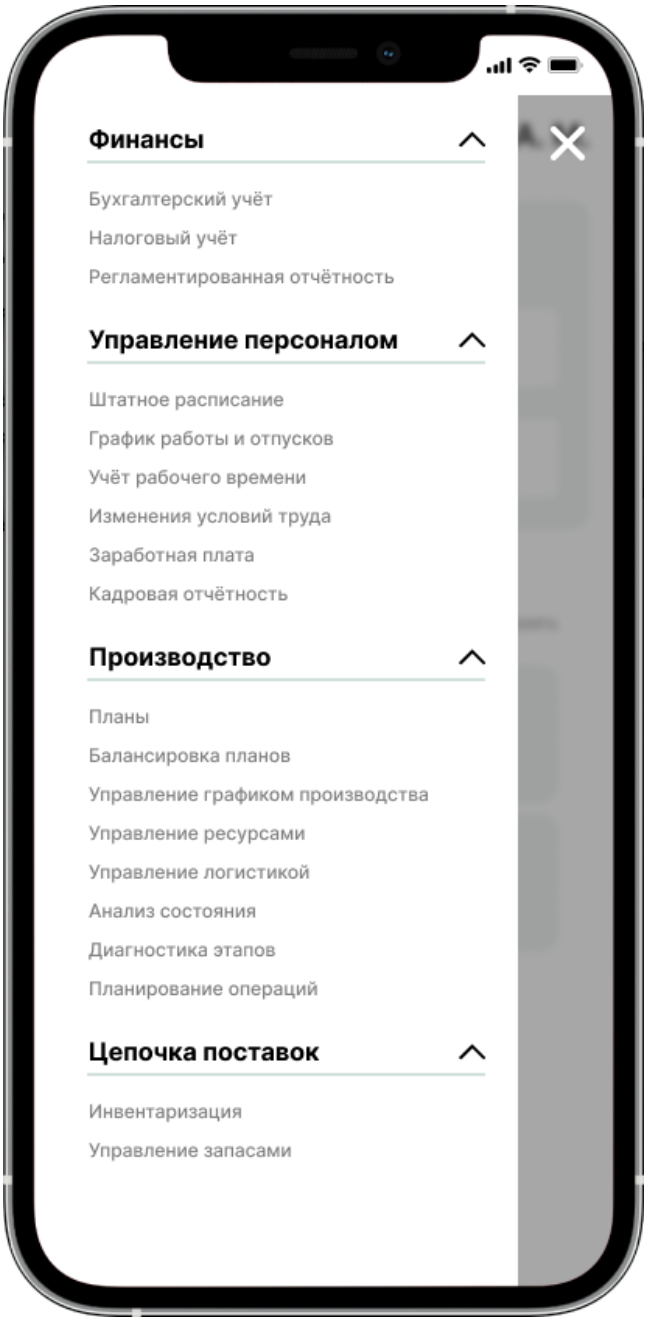

Рисунок 15 - Меню приложения

На рисунке 16 изображен пример создания заказа. После обработки со стороны интеграционной платформы и успешного создания заказ попадает в список заказов, доступный также из бокового меню, и дальше с ним можно работать с помощью выпадающего меню действий при нажатии на заказ в этом списке.

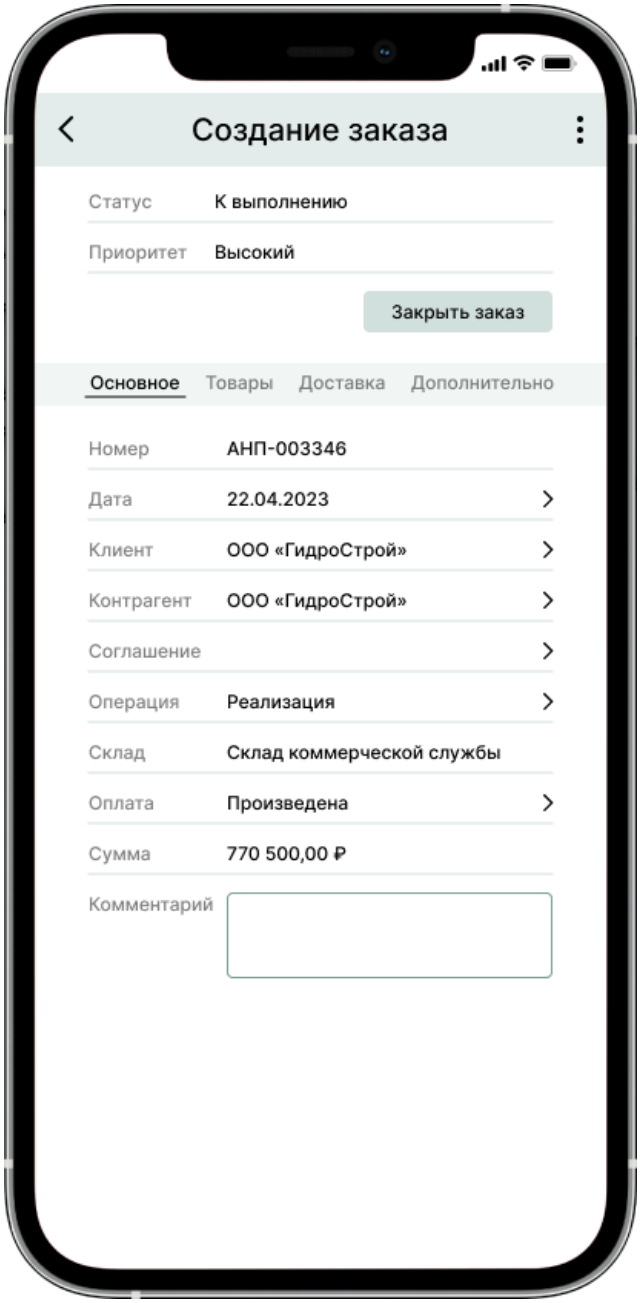

Рисунок 16 - Создание заказа

Инструментом для наглядного отслеживания данных является создание отчетов. Для их просмотра необходимо перейти с помощью меню в соответствующий раздел и открыть список доступных отчетов. Пример списка представлен на рисунке 17.

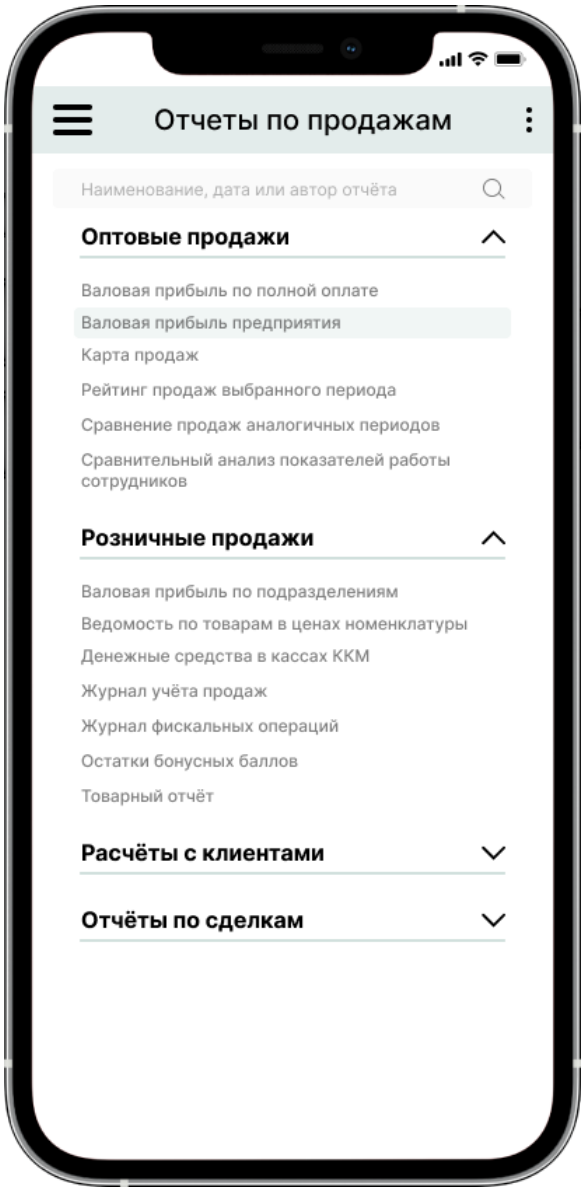

Рисунок 17 - Список отчетов

Отчет может включать в себя как текстовую информацию, так и различные диаграммы. Пример отчета с инфографикой изображен на рисунке 18.

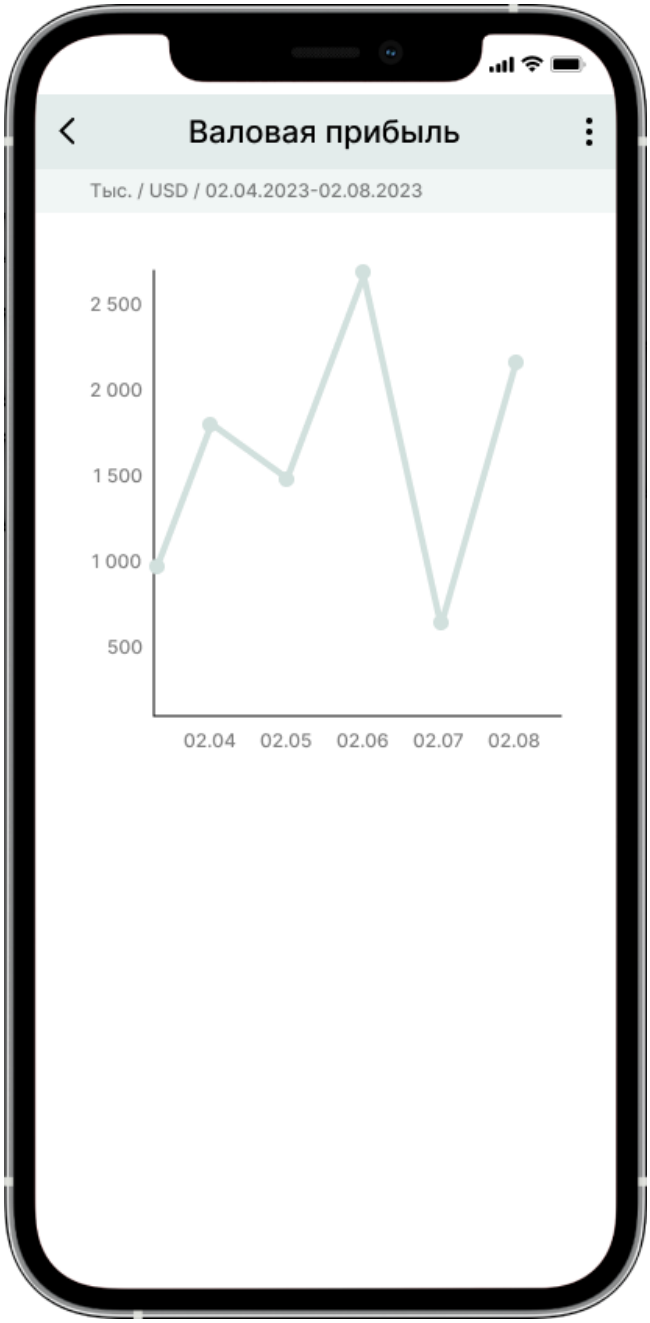

Рисунок 18 - Пример отчета

Кроме того, важной функцией ERP-системы является автоматизация процессов и документооборота, связанного с сотрудниками. На рисунке 19 представлена операция отправки запроса на отпуск одним из сотрудников.

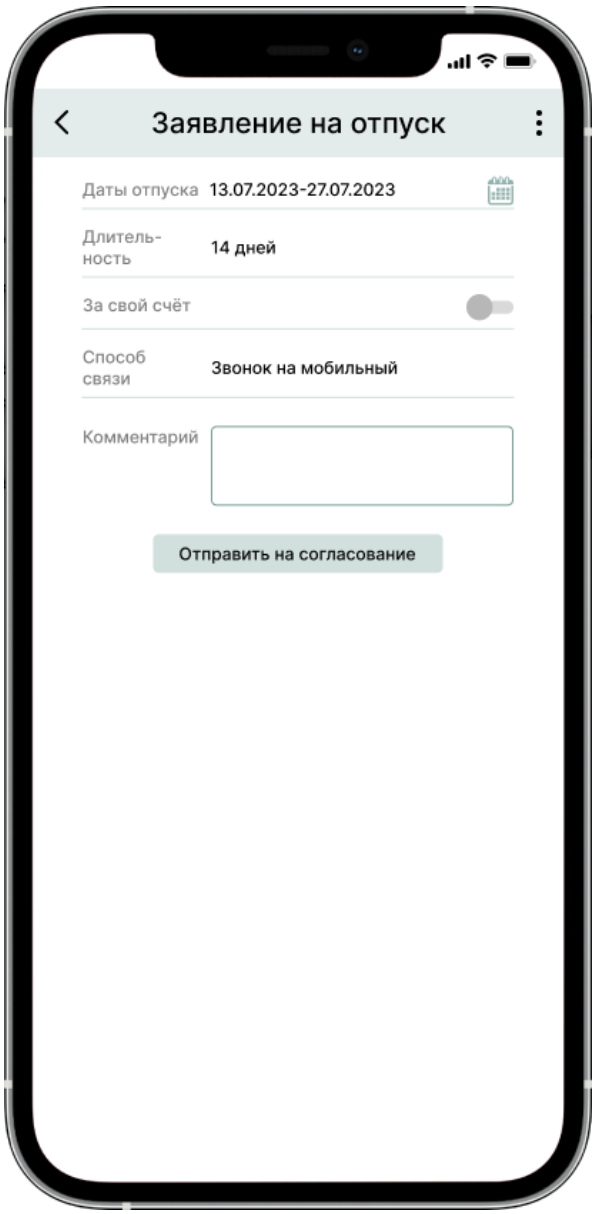

Рисунок 19 - Отправка запроса на отпуск

После отправки данный запрос отобразится у руководителя, который может одобрить или отклонить заявку. Пример того, как руководитель видит на своем экране данный запрос, представлен на рисунке 20.

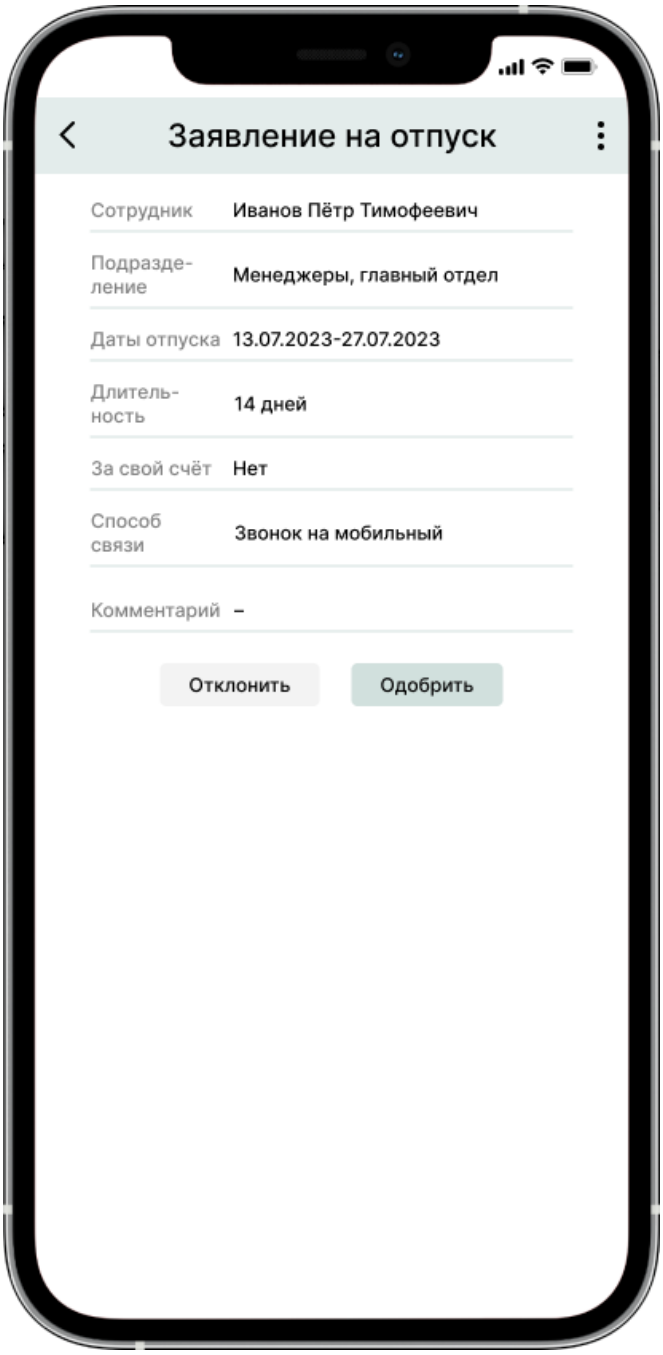

Рисунок 20 - Запрос на отпуск с экрана руководителя

Помимо функций ERP-системы, необходимо реализовывать те изменения модели, которые должны её улучшить. Улучшения, которые могут быть добавлены, следующие:

– добавление асинхронного взаимодействия,

– упрощение построения модели бизнес-процесса,

– добавление валидации запросов в адаптерах,

– улучшение логики работы с мобильными приложениями.

В первую очередь, необходимо добавить асинхронное исполнение процессов в платформу. Асинхронное выполнение процессов и команд позволит значительно сократить потребление ресурсов и ускорить работу приложения. Для реализации асинхронного взаимодействия будут применяться специальные интеграционные адаптеры (для приема запросов) и асинхронные коллы (для отправки запросов). Пример блока для вызова внешнего сервиса в асинхронном режиме представлен на рисунке 21.

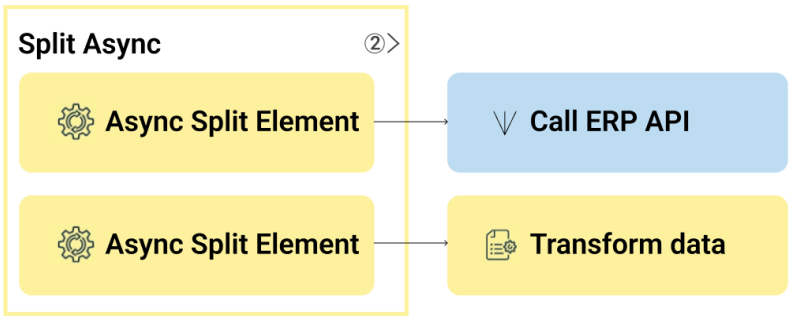

Рисунок 21 - Асинхронный вызов внешней системы

Данное изменение покажет наибольшую эффективность при большой нагрузке и большом количестве запросов пользователей в единицу времени. Для сравнения скорости работы необходимо при одинаковой нагрузке замерить скорость выполнения операций в синхронном режиме и скорость выполнения операций после добавления асинхронных способов работы.

Следующим улучшением является упрощение low-code настройки. BPMN-модель является довольно громоздкой схемой, на которой сложно отследить весь процесс. Для упрощения предлагается использовать блоки, внутри которых может содержаться необходимая логика. Пример такой модели представлен на рисунке 22.

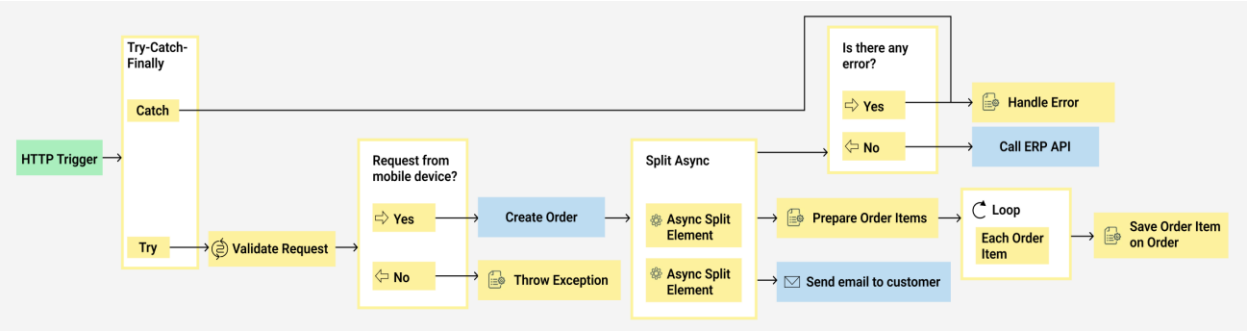

Рисунок 22 - Общий вид модели

На рисунке 23 представлен пример обработки ответа внешней системы на платформе.

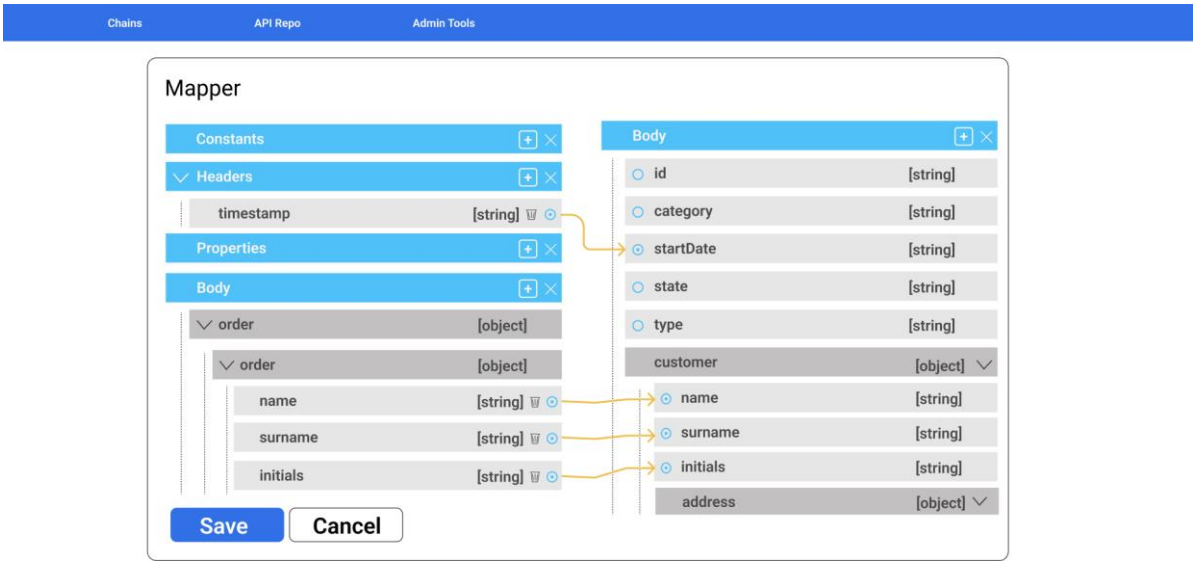

Рисунок 23 - Обработка ответа

Также обработку можно реализовать с помощью скрипта, написанного на языке программирования Groovy. Пример обработки ответа от системы с помощью скрипта представлен на рисунке 24.

```groovy
import groovy.json.JsonSlurper
import groovy.json.JsonOutput

def responseBody = exchange.getMessage().getBody(String.class)
def jsonSlurper = new JsonSlurper()
def response = jsonSlurper.parseText(responseBody)

def result = processResponse(response)

exchange.getMessage().setBody(JsonOutput.toJson(result))

// Functions
def processResponse(def response) {
    validateData(response.startDate)
    transformOrder(response)
}

def validateData(def date) {
```

Рисунок 24 - Обработка ответа с помощью скрипта

Специфика работы с мобильным приложением заключается в следующем: мобильное приложение может работать в оффлайн-режиме или иметь нестабильное подключение к сети, поэтому необходимо предусмотреть такую возможность работы, когда пользователь работает с системой оффлайн. Для решения этой проблемы был создан алгоритм, представленный на рисунке 25. Данный алгоритм учитывает работу мобильного приложения как в онлайн, так и в оффлайн режимах, и на основе режима работы выбирает способ отправки запроса в корпоративную информационную систему.

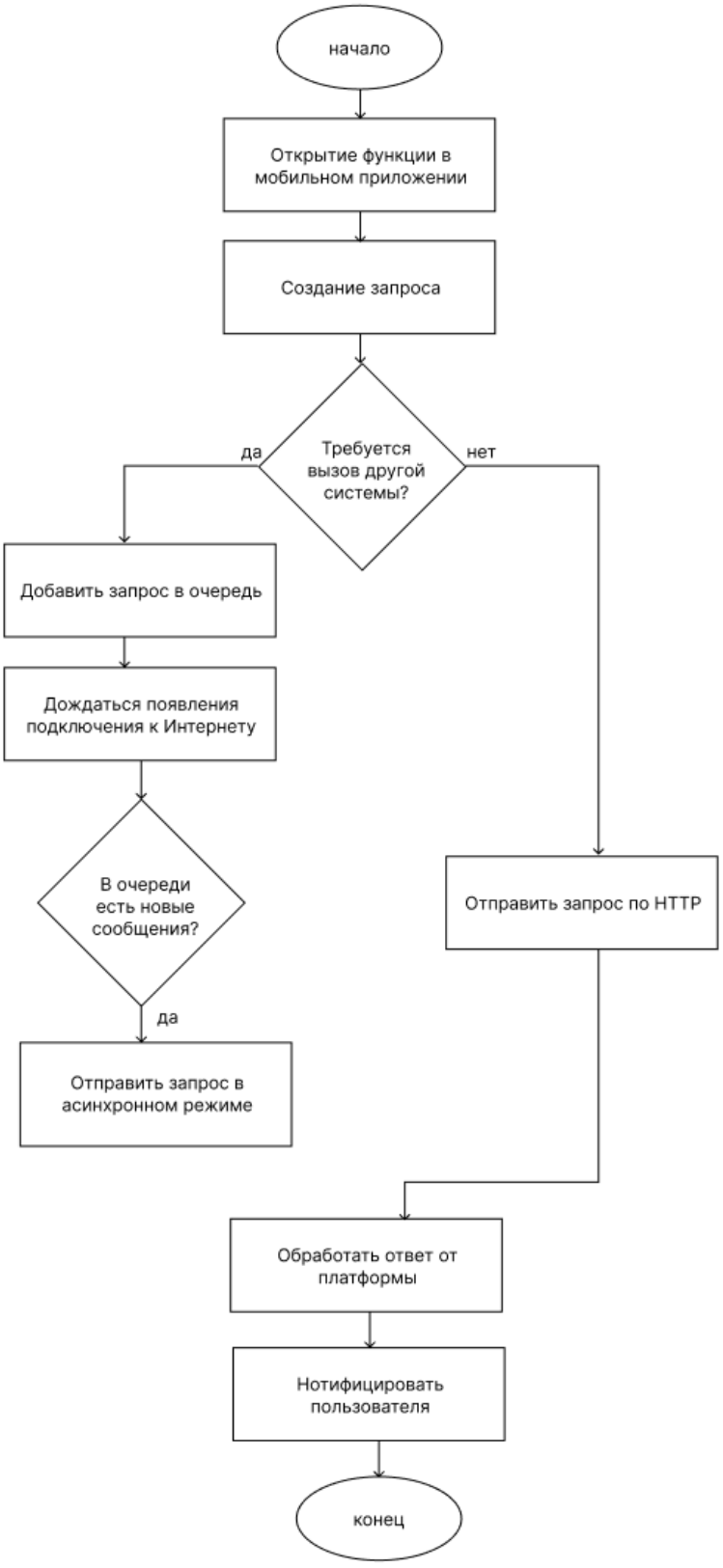

Рисунок 25 - Алгоритм работы платформы с мобильным приложением и ERP-системой

Данный алгоритм позволяет системам взаимодействовать вне зависимости от подключения к сети, с той разницей, что запросы, созданные без подключения, будут отправлены по очереди в асинхронном режиме после появления на мобильном устройстве подключения к Интернету.

3.2 Интеграция мобильного приложения с ERP-системой

Ключевым этапом работы является применение измененной интеграционной модели к мобильному приложению и ERP-системе.

Разрабатываемое решение должно уметь отправлять свои запросы так же, как внешние системы обращаются к решению. Однако, тут есть свои особенности. Так как внешние системы будут использовать только один адрес для отправки запроса на решение, изнутри мы не можем также отправлять запросы на какой-то уникальный адрес или складывать все наши запросы в один топик в Kafka. Решением этой проблемы может стать использование сервисов - спецификаций, в которых будет заранее настроен адрес сервиса и указаны API, которые данный сервис предоставляет. Таким образом, список внешних систем, с которыми может взаимодействовать решение, может расширяться и настраиваться. Точно так же можно настроить спецификации для топиков Kafka и RabbitMQ. Примеры таких сервисов представлены на рисунках 26-27.

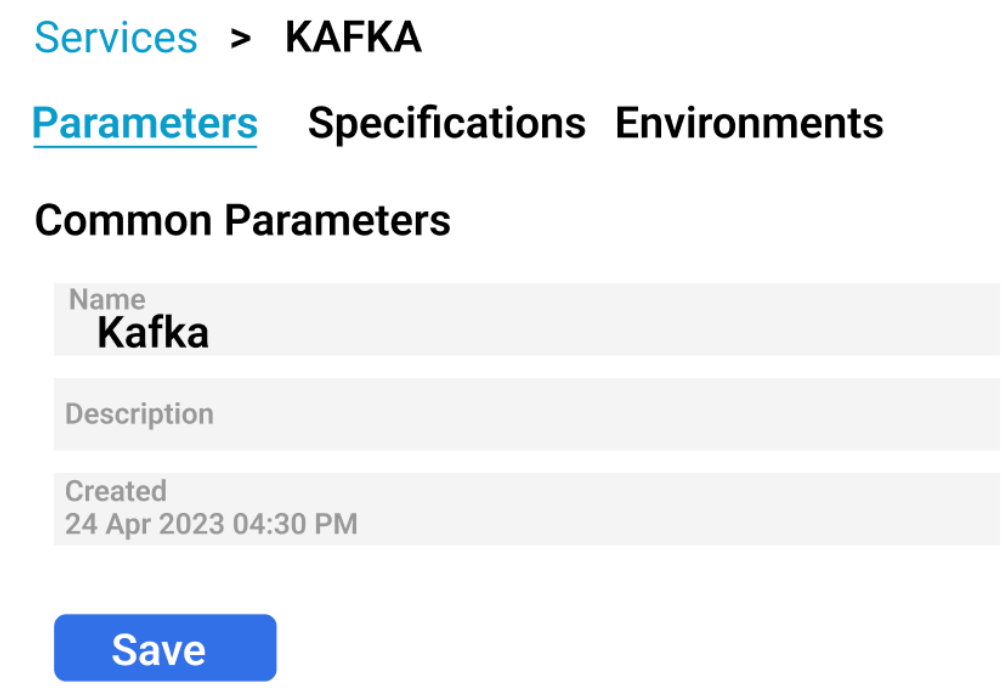

Рисунок 26 - Сервис для работы с Kafka

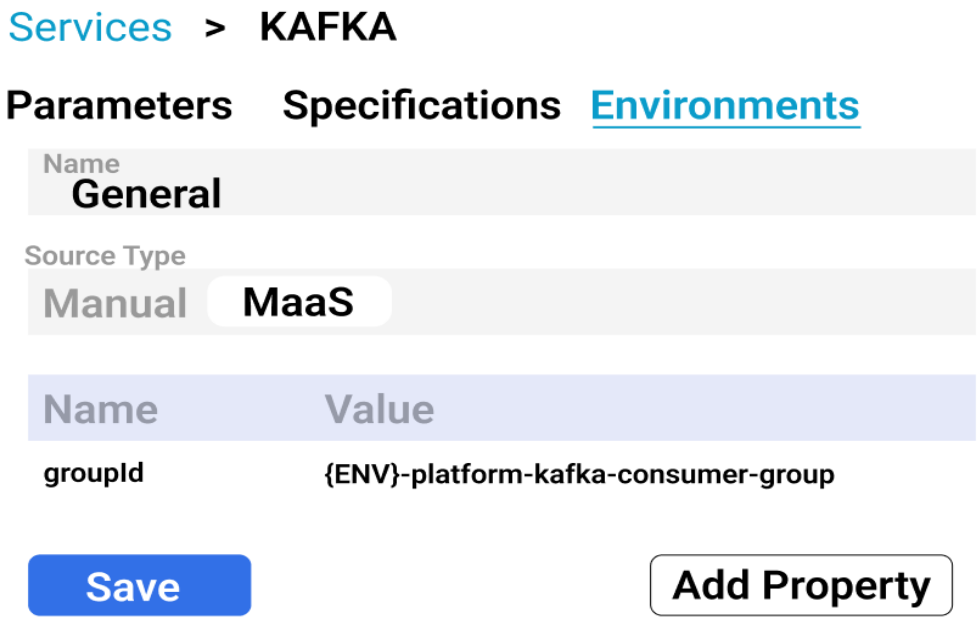

Рисунок 27 - Настройка сервиса

Сервисы, которые общаются с помощью REST API, могут быть настроены с помощью Swagger Editor, являющимся бесплатным средством для

создания API-документации, настройки и проверки запросов. Пример такой настройки представлен на рисунке 28.

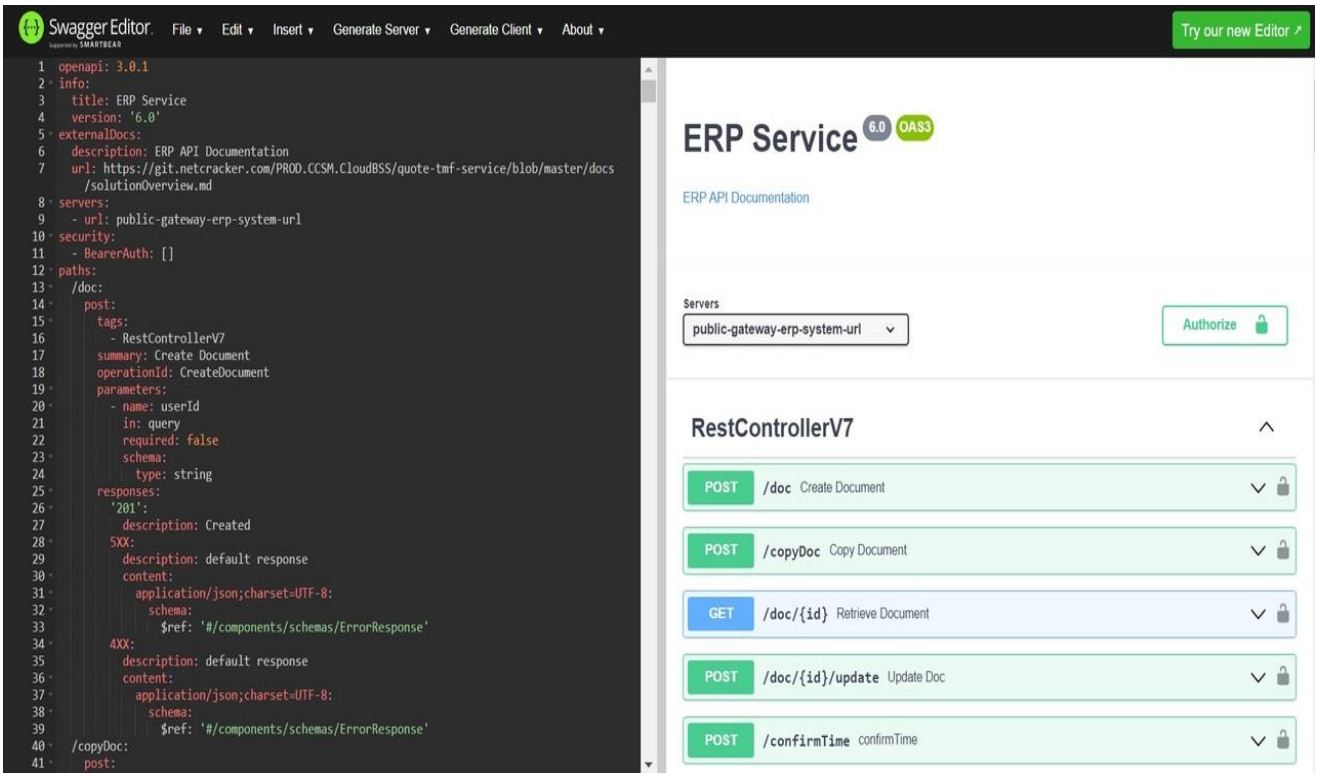

Рисунок 28 - Спецификация сервиса в Swagger Editor

На рисунке 29 представлен пример одного из API, который можно использовать для создания документа.

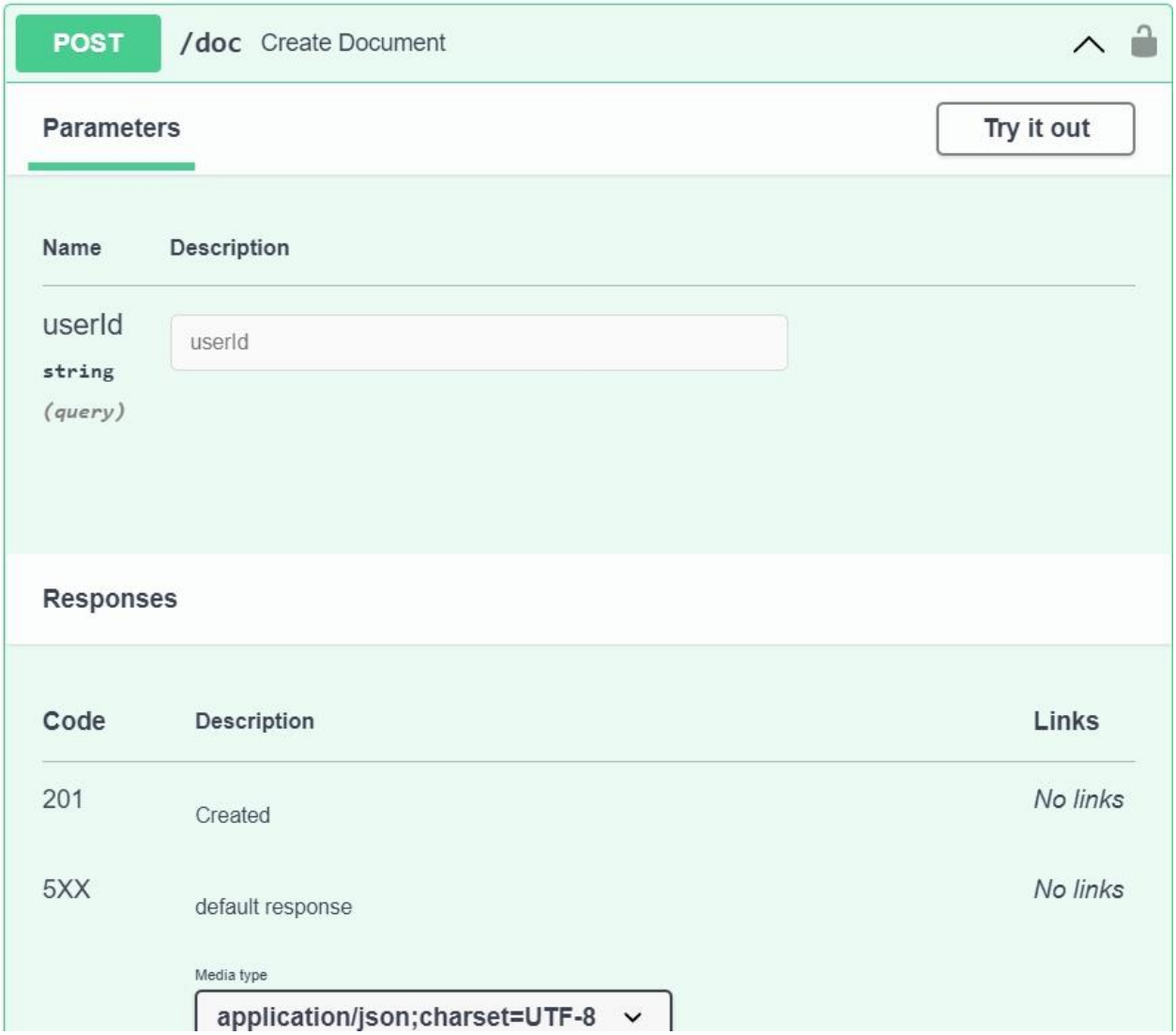

Рисунок 29 - Пример API

Для развертывания на разных серверах и окружениях должна существовать возможность экспорта и импорта всех блоков, сервисов и конфигураций. Для большего удобства все операции экспорта и импорта должны происходить в одном формате, например, zip-архива. На рисунке 30 представлен алгоритм развертывания решения на новом сервере.

Рисунок 30 - Алгоритм развертывания решения на новом сервере

Последним пунктом интеграции мобильного приложения и ERP-системы является настройка их подключения и проверка работоспособности.

Первым шагом настройки будет создание контейнера для работы платформы. Это необходимо для управления жизненным циклом платформы и развертывания на окружении. Для контейнеризации будет использоваться платформа Docker.

Преимущества Docker:

– возможность отката - в любой момент контейнер можно «сбросить», откатить до изначальной версии;

– готовая среда для запуска - контейнер вместе с приложением сразу содержит среду для работы;

– небольшой вес - Docker-контейнер чаще всего весит не больше пары сотен мегабайтов, иногда значительно меньше;

– изолированность от внешней среды - у контейнера нет доступа к информации на хосте;

– удобные инструменты управления - образы Docker хранятся в Docker-реестре, откуда можно запускать готовые среды и на их основе настраивать свои.

Структура Docker представлена на рисунке 31.

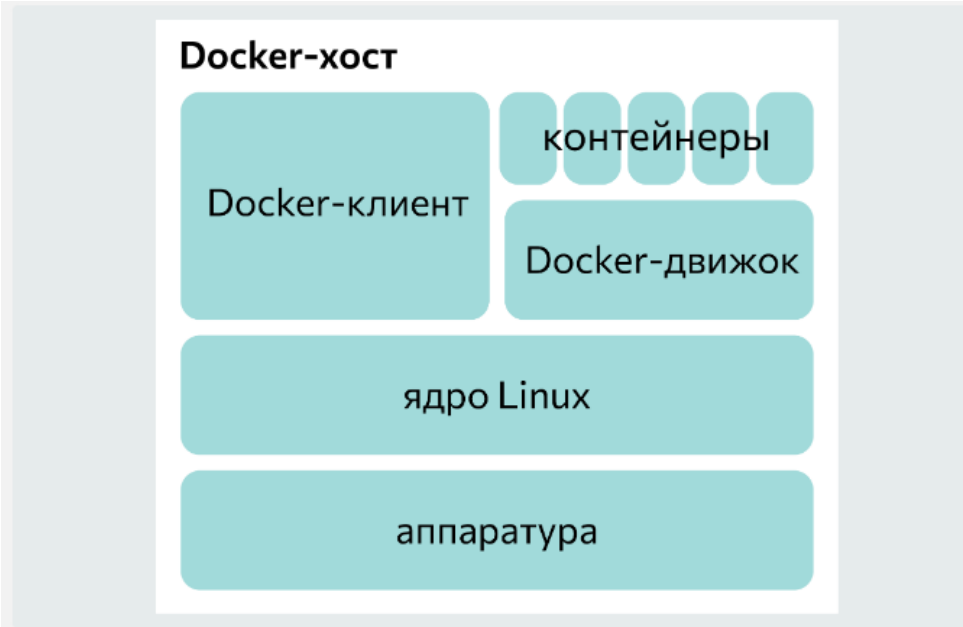

Рисунок 31 - Структура Docker

Следующим шагом является развертывание контейнера на сервере. Схема развертывания представлена на рисунке 32.

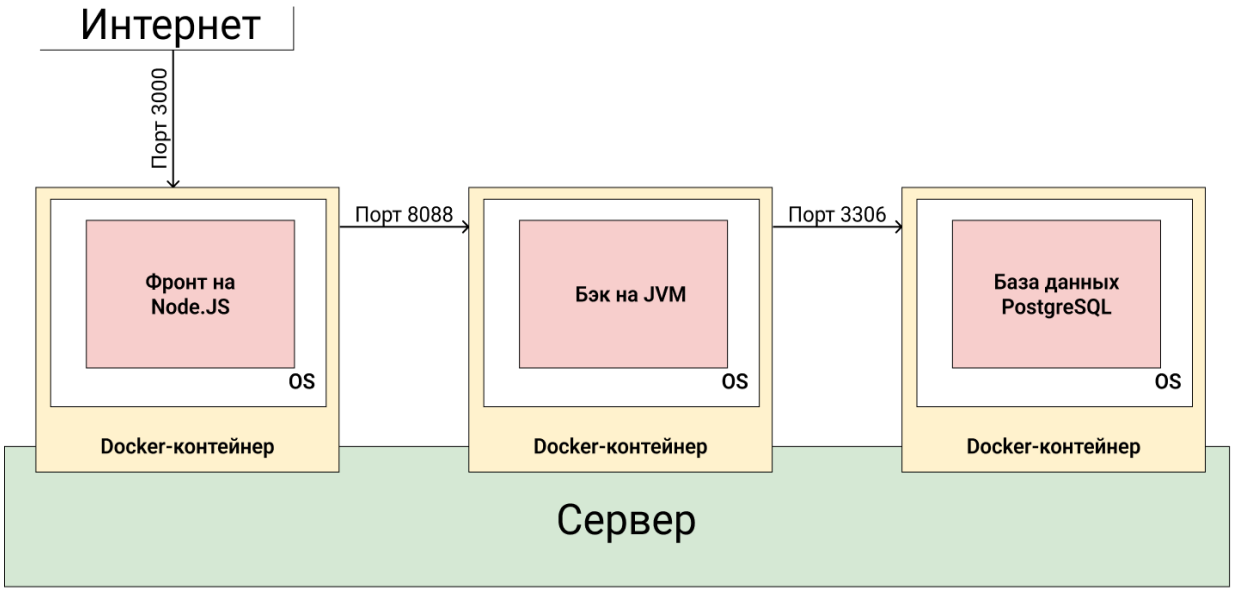

Рисунок 32 - Схема развертывания

Далее необходимо настроить подключение между платформой и ERP-

системой. Для этого необходимо в настройках сервисов указать правильные адреса. Пример такой настройки представлен на рисунке 33.

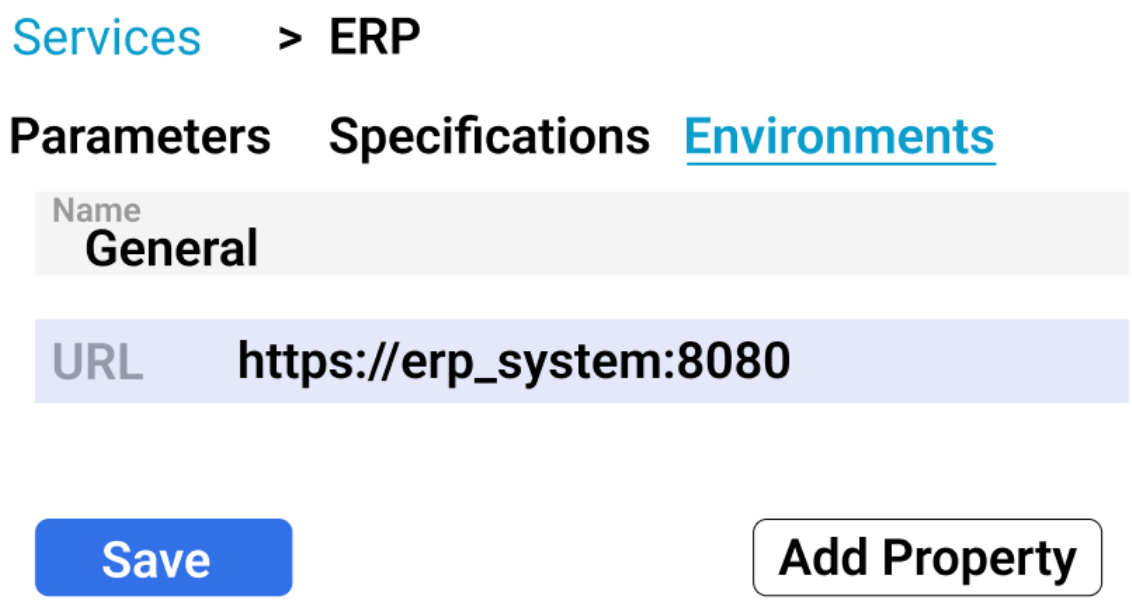

Рисунок 33 - Настройка подключения к ERP-системе

Для проверки работы платформы необходимо отправить запросы с мобильного приложения на интеграционную платформу и проверить результат.

В качестве проверки была выбрана операция создания заказа. Отправка запроса представлена на рисунке 34.

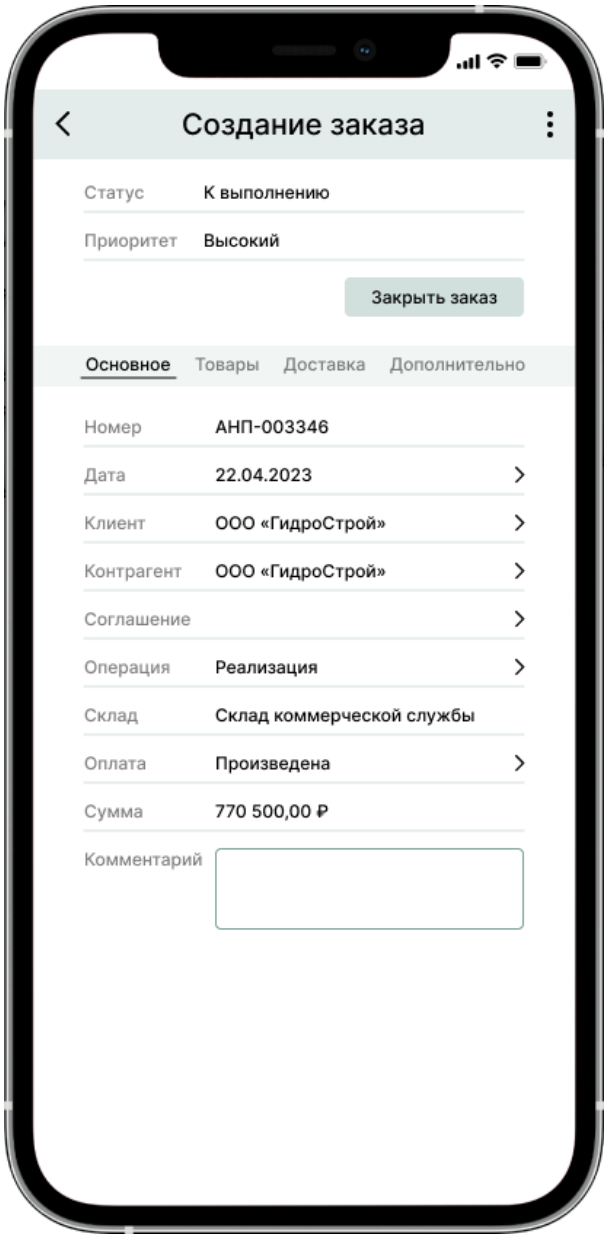

Рисунок 34 - Отправка запроса с мобильного приложения

На рисунках 35-36 представлены логи обработки запроса на платформе. Логи включают в себя дерево логов, в котом можно увидеть, какие элементы отработали в запросе, какое количество времени заняло их выполнение, а также успешность выполнения каждого блока.

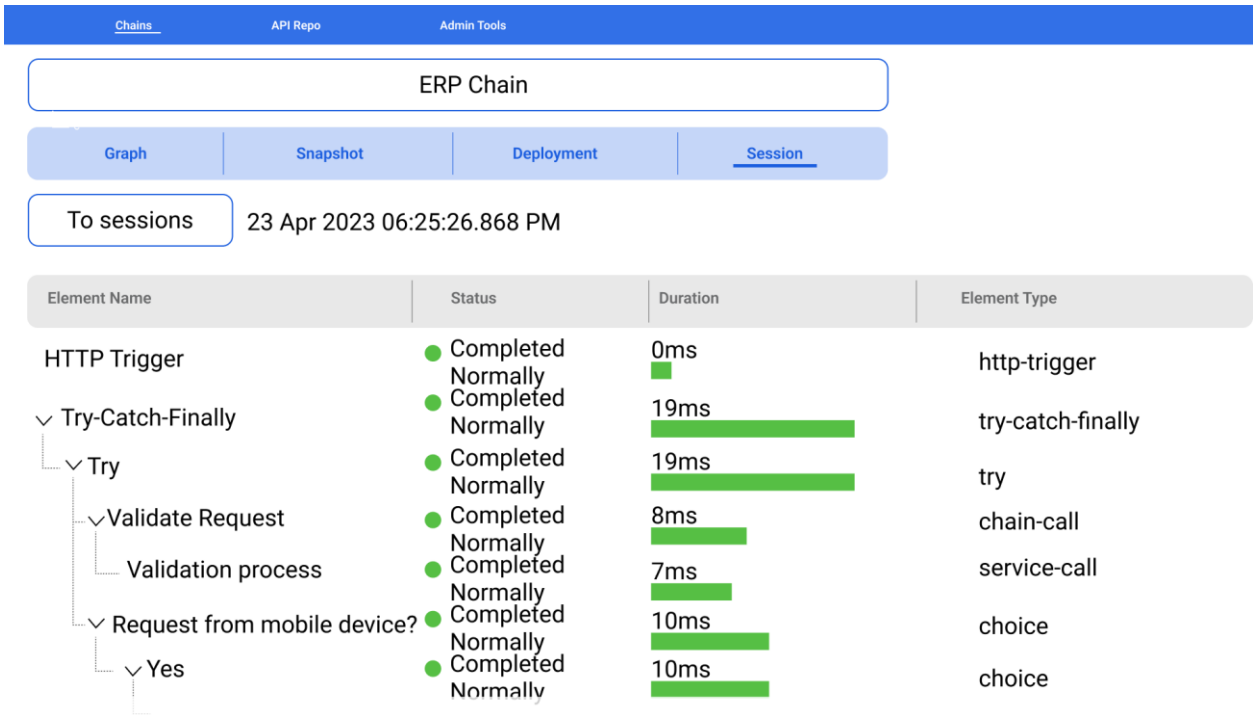

Рисунок 35 - Дерево логов

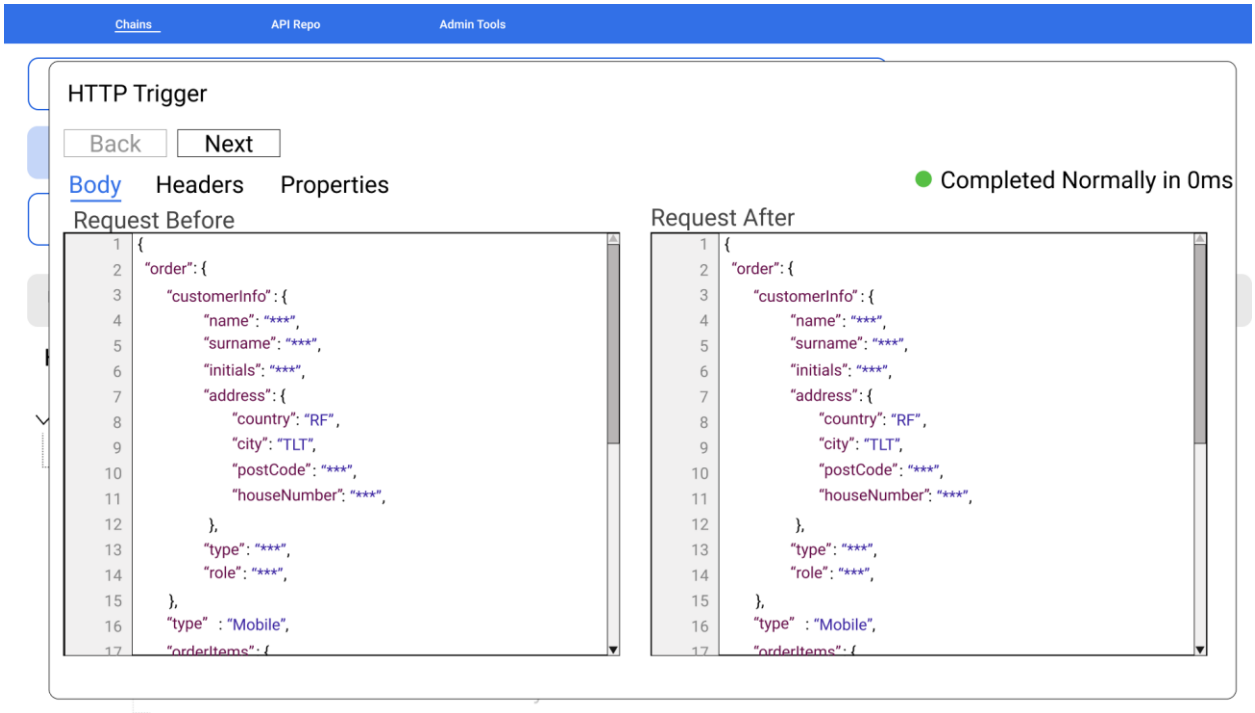

Рисунок 36 - Проверка запроса

Далее запрос был отправлен в ERP-систему, и мы можем в этом убедиться в логах самой ERP-системы. Логи представлены на рисунке 37.

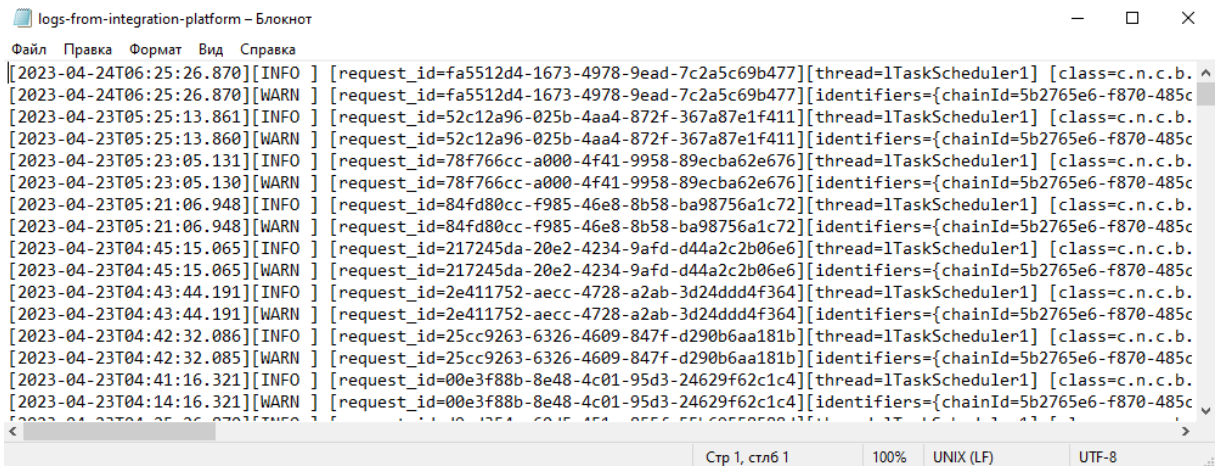

Рисунок 37 - Логи ERP-системы

На рисунке 38 представлен ответ ERP-системы, пришедший на платформу в ответ на запрос.

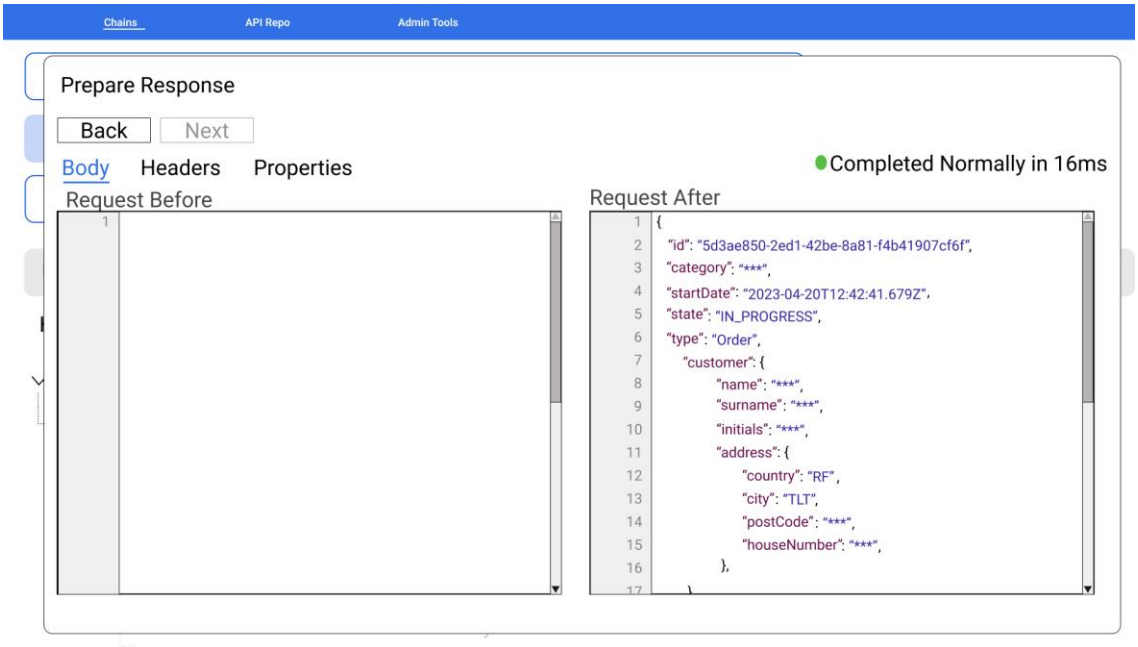

Рисунок 38 - Ответ ERP-системы

В мобильном приложении мы можем увидеть, что запрос был обработан и статус изменился. На рисунке 39 мы можем увидеть, что заказ был создан, отображается в списке заказов и для него доступны все действия, которые

можно провести с заказами.

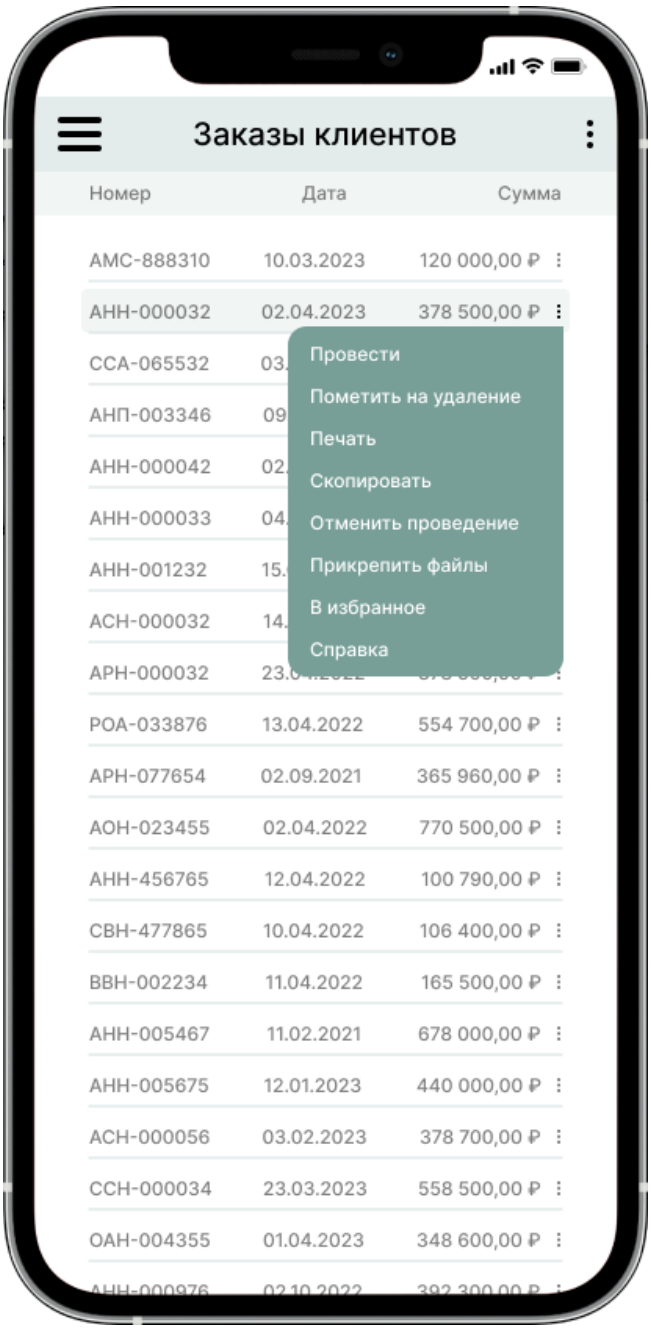

Рисунок 39 - Выполнение работы функции

Одним из улучшений является изменение алгоритма работы мобильного приложения и платформы. Для проверки этого улучшения необходимо отключить Интернет и отправить несколько запросов. Запросы представлены на рисунке 40.

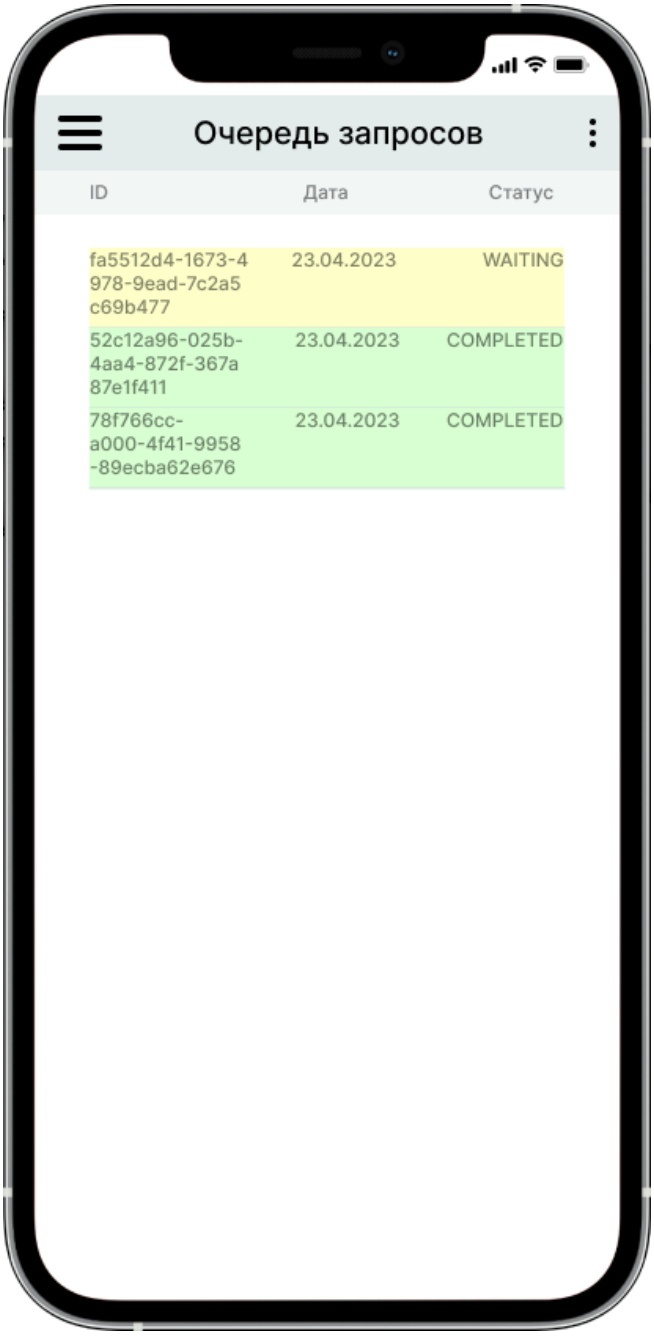

Рисунок 40 - Отправка нескольких запросов

При включении Интернета данные запросы будут отправлены в очередь сообщений и обработаны в асинхронном режиме. Запрос, попавший в очередь Kafka, представлен на рисунке 41.

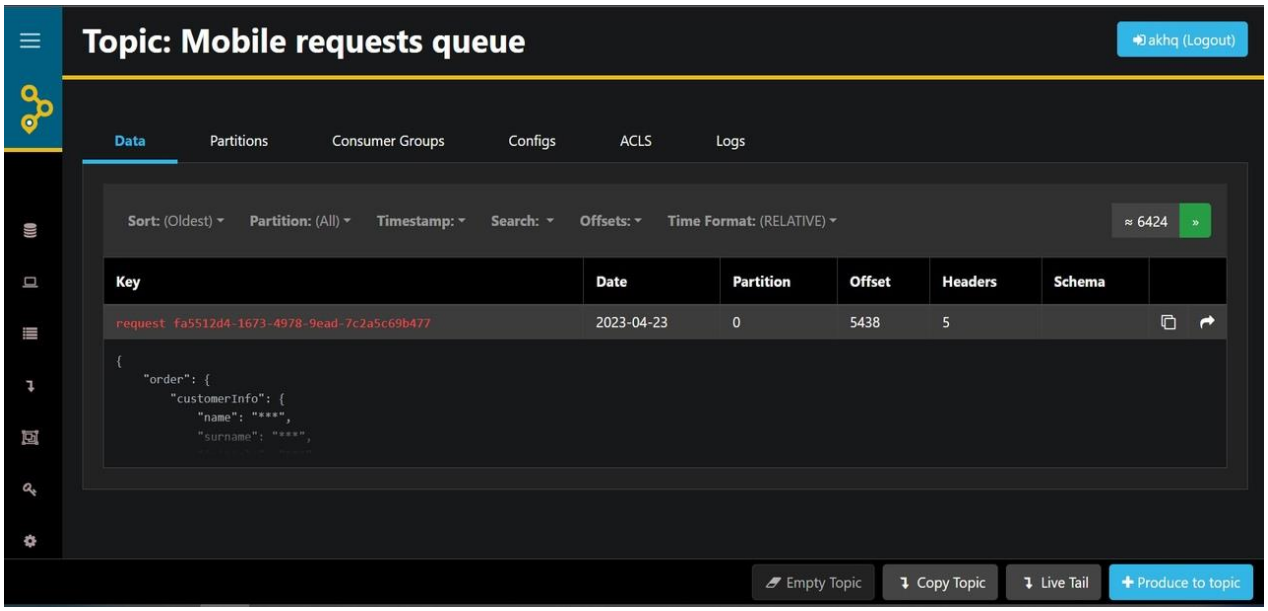

Рисунок 41 - Сообщение в очереди

Данный запрос будет обработан после получения доступа к Интернету. На рисунке 42 представлен запрос, попавший на интеграционную платформу после чтения очереди из очереди сообщений.

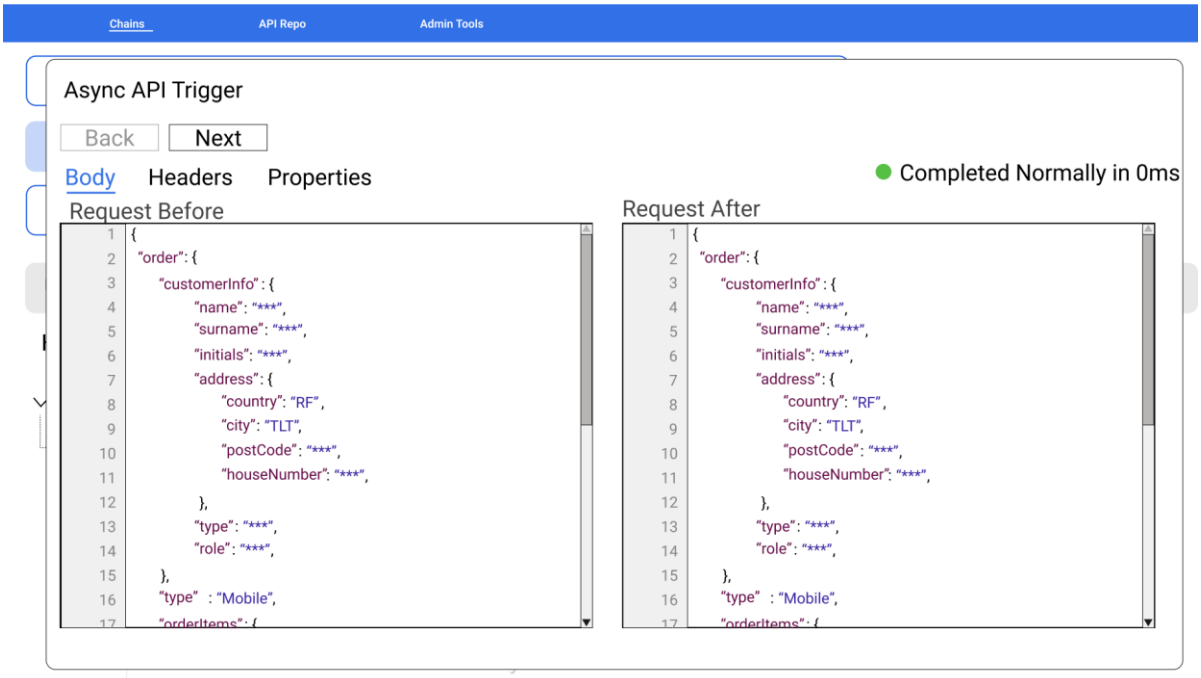

Рисунок 42 - Асинхронный запрос на платформе

Для обработки ошибок, в том числе в качестве ответа от другой системы, можно использовать обработчик ответов в блоке «Service Call», добавить скрипт после вызова сервиса или отлавливать ошибки с помощью блока «Try-Catch-Finally». В случае ошибки лог сессии будет выглядеть так, как представлен на рисунке 43.

Рисунок 43 - Пример ошибки в сессии

Для реализации построения моделей бизнес-процессов могут использоваться следующие блоки:

– Triggers:

1) HTTP Trigger,

2) Async API Trigger,

3) Chain Trigger,

4) Kafka Trigger,

5) RabbitMQ Trigger;

– Transformation:

1) Mapper,

2) Script;

– Senders:

1) HTTP Sender,

2) Kafka Sender,

3) Mail Sender,

4) RabbitMQ Sender,

5) Service Call;

– Routing:

1) Chain Call,

2) Choice,

3) Loop,

4) Split,

5) Split Async,

6) Try-Catch-Finally.

Список всех доступных элементов представлен на рисунках 44-46.

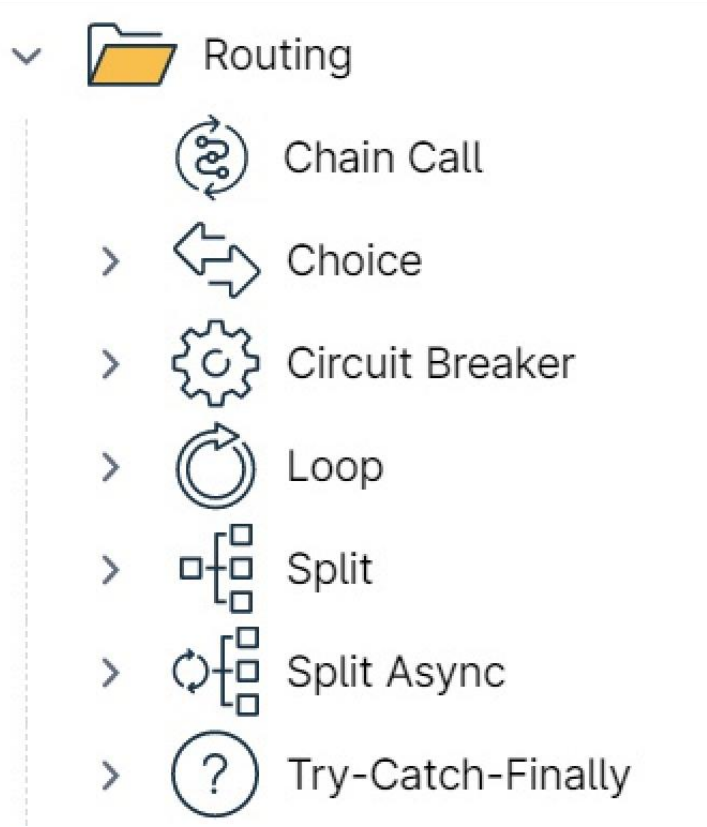

Рисунок 44 - Блоки для построения модели в разделе «Routing»

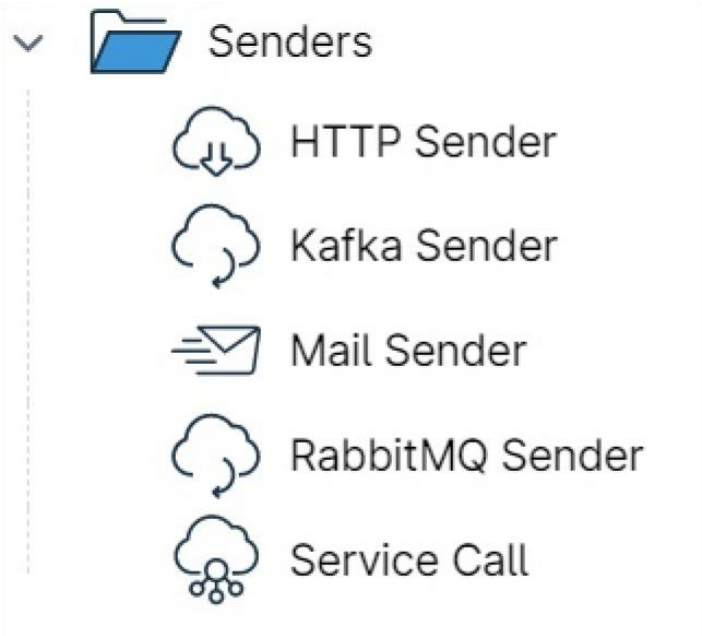

Рисунок 45 - Блоки для построения модели в разделе «Routing»

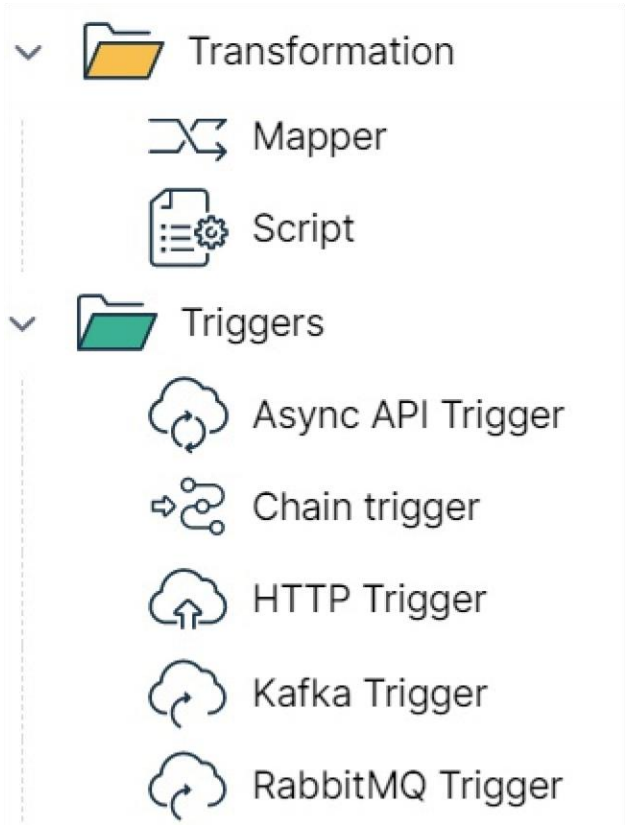

Рисунок 46 - Блоки для построения модели в разделе «Routing»

Все элементы в модели обладают уникальным идентификатором, из-за

чего можно однозначно определить элементы и определить связи между ними. В качестве идентификатора используется UUID. «UUID (universally unique IDentifier) - стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation (OSF) как часть DCE - среды распределенных вычислений» [28]. Идентификатор представляет собой 128-битное число, в текстовом представлении отображающееся как серия из 32 шестнадцатеричных символов.

В самом простом виде модель будет выглядеть так, как изображено на рисунке 47.

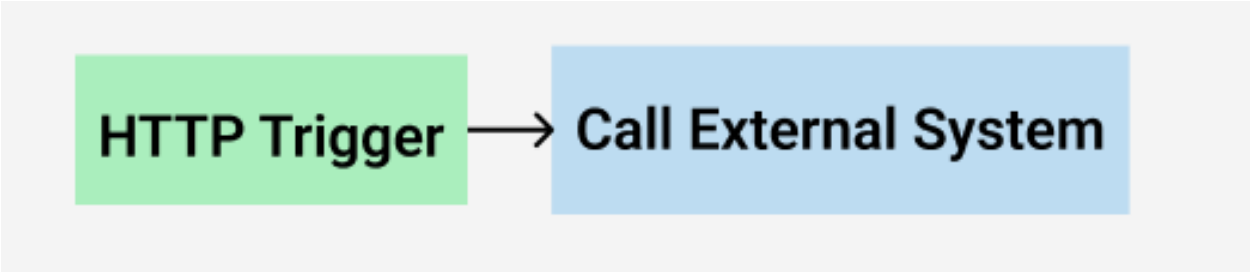

Рисунок 47 - Простейшая интеграционная цепочка

При импорте данной интеграционной цепочки она преобразуется в формат yaml. Пример описания интеграционной цепочки в файле представлен на рисунке 48.

Рисунок 48 - Конфигурация интеграционной цепочки

«YAML - это язык для сериализации данных, который отличается простым синтаксисом и позволяет хранить сложноорганизованные данные в компактном и читаемом виде» [26]. Данный язык позволяет увидеть следующие важные данные об интеграционной цепочке:

– идентификатор,

– имя цепочки,

– время последней модификации,

– список элементов.

Каждый элемент в списке элементов содержит определенные данные, но общими для всех являются идентификаторов, имя элемента и его тип.

После описания всех элементов следует блок, описывающий

зависимости между элементами. Пример такого блока представлен на рисунке 49.

Рисунок 49 - Блок зависимостей элементов

Таким образом, все сообщения и запросы, отправляемые с мобильного устройства, будут обработаны, и пользователь получит уведомление в приложении, когда его запрос будет завершен. Кроме того, такой подход позволяет передавать данные сразу в несколько систем, поскольку очередь сообщений является общедоступной в рамках одной инфраструктуры.

В рамках третьей главы была применена улучшенная интеграционная модель, разработано нативное мобильное приложение для операционной системы Android, разработана платформа, соединяющая мобильное приложение и ERP-систему, настроено соединение между всеми системами, проведен тест функциональности систем, разработан алгоритм взаимодействия мобильного приложения с платформой.

4. Оценка модели и анализ результатов

4.1 Оценка возможностей и применимости полученного решения

«Перед тем как приступить к анализу решения, стоит оценить его применимость, удобство и функциональную пользу» [18]. Взаимодействие систем с использованием нового решения представлено на рисунке 50.

Рисунок 50 - Взаимодействие систем

Полученное решение имеет широкий круг применения. Оно может быть использовано для связи двух и более различных систем, перестраивая форматы данных внутри себя и выполняя дополнительные операции и валидации. Использование визуального программирования с перетаскиванием и подключением блоков сильно упрощает процесс настройки и создания новых подключений, нет возможности написать неправильный код вне трансформационных скриптов. Функционально такое решение способно заменить использование нескольких сервисов, что упрощает ведение проекта

в целом.

Алгоритм работы полученного решения представлен на рисунке 51.

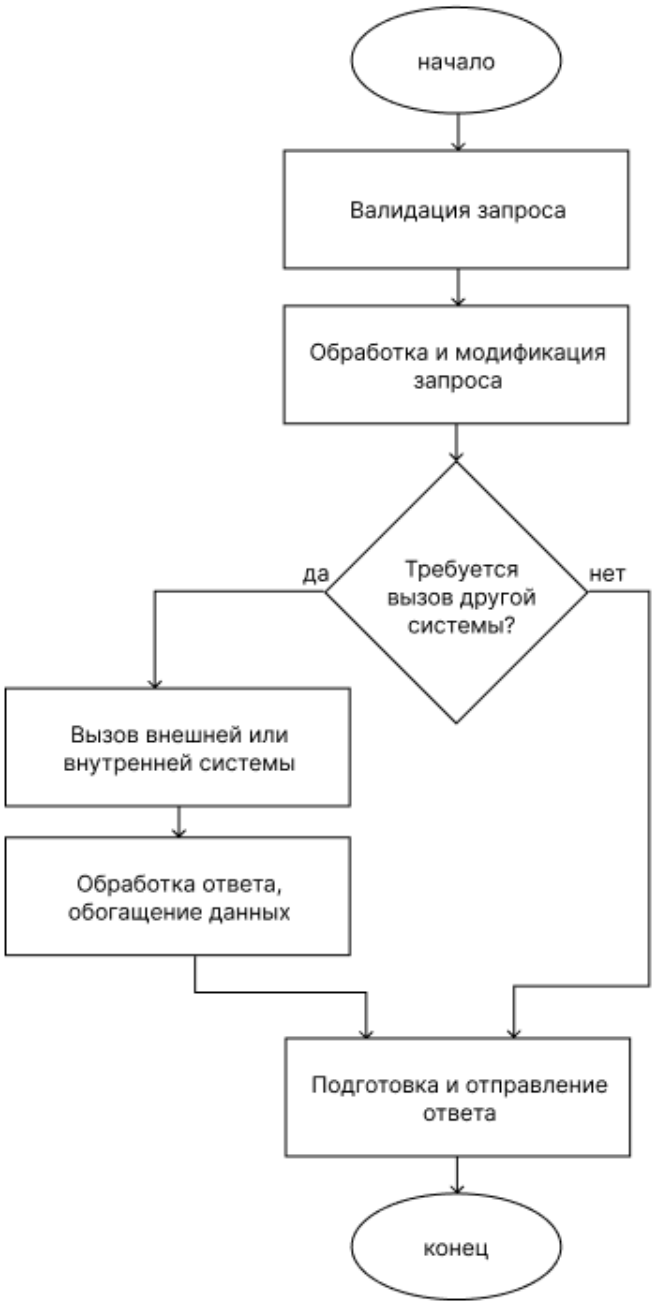

Рисунок 51 - Алгоритм работы сервисов в общем виде

В UI данный алгоритм будет выглядеть похожим на модель в нотации BPMN, но с более наглядными блоками для построения. Пример модели был представлен на рисунке 22. Данный алгоритм может быть модернизирован и

улучшен для работы по любому сценарию.

4.2 Тестирование интеграционной платформы

Для оценки эффективности внесенных изменений необходимо провести сравнительный анализ по каждому из улучшений, выделяя те параметры, которые наиболее наглядно отображают разницу между старой и обновленной интеграционной платформой. Сравнение по каждому пункту представлено в таблицах 1-4.

Таблица 1 - Асинхронное взаимодействие

Асинхронный режим работы	Объем данных			
	4 КБ	9 КБ	24 КБ	48 КБ
Скорость обработки запроса	1 мс	2 мс	4 мс	5 мс
Время выполнения от отправки запроса до получения ответа	224 мс	304 мс	554 мс	707 мс

Таблица 2 - Синхронное взаимодействие

Синхронный режим работы	Объем данных			
	4 КБ	9 КБ	24 КБ	48 КБ
Скорость обработки запроса	1 мс	2 мс	4 мс	5 мс
Время выполнения от отправки запроса до получения ответа	235 мс	340 мс	650 мс	860 мс

Таблица 3 - Упрощение построения BPMN-модели

Параметры	Обновленная платформа	Интеграционная платформа
Количество элементов для построения модели	18	15
Минимальное количество элементов для построения модели	2 - интеграционный адаптер и исполнительный блок	3 - точка входа, блок действия, точка выхода

Таблица 4 - Интеграционные адаптеры

Параметры	Обновленная платформа	Интеграционная платформа
Виды адаптеров	HTTP, Kafka, RabbitMQ, JMS, chain trigger, scheduler	HTTP
Режимы работы адаптеров	Синхронный и асинхронный	Синхронный
Валидация запросов в адаптерах	Есть	Нет

Исходя из полученных результатов, можно заключить, что интеграционная модель была улучшена, скорость и эффективность работы платформы возросла, сложность реализации при этом выросла слабо.

В рамках четвертой главы был проведен сравнительный анализ старой и улучшенной интеграционных моделей, из которого можно заключить, что улучшенная интеграционная модель обладает более высокой эффективностью и является более простой в использовании, чем уже существующее решение.

Заключение

В рамках выпускной квалификационной работы было проведено исследование моделей и методов интеграции мобильных приложений с корпоративными информационными системами, выявлены их преимущества и недостатки.

Наиболее оптимальным способом взаимодействия мобильного приложения с другой системой является использование промежуточного слоя. Такой подход обладает следующими преимуществами:

– упрощенное подключение сложной инфраструктуры,

– легкость масштабирования,

– низкая нагрузка на мобильного клиента.

В рамках первой главы был проведен анализ существующих подходов к интеграциям различных систем, рассмотрены ERP-системы, выявлены отличия мобильных приложений от остальных систем.

В рамках второй главы были рассмотрены способы передачи сообщений между системами, выбран пошаговый метод создания модели, проанализировано существующее решение с использованием интеграционной платформы и предложены способы решения найденных недостатков.

В рамках третьей главы было разработано мобильное приложение, поддерживающее функции ERP-системы, была реализована интеграционная платформа с применением улучшений, указанных во второй главе, настроено соединение между ними и между интеграционной платформой и корпоративной информационной системой.

В рамках четвертой главы был проведен сравнительный анализ интеграционных платформ, в результате которого было установлено, что принятые улучшения действительно улучшили работу платформы и не слишком сильно усложнили ее разработку и сопровождение.

Список используемых источников

1. Arquillian documentation // Arquillian [Электронный ресурс]. - Режим доступа: http://arquillian.org/docs/.

2. Functional Testing // Software Testing Fundamentals [Электронный ресурс]. - Режим доступа: http://softwaretestingfundamentals.com/functionaltesting/.

3. HTTP basic authentication // IBM [Электронный ресурс]. - Режим доступа: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.1.0/com.ibm.cics.ts.i nternet.doc/topics/dfhtl2a.html.

4. IEEE Recommended Practice on Software Reliability // IEEEXplore [Электронный ресурс]. - Режим доступа: https://ieeexplore.ieee.org/document/7827907.

5. Send notifications across platforms for free // Firebase [Электронный ресурс]. - Режим доступа: https://firebase.google.com/products/cloudmessaging/?hl=ru (дата обращения 01.03.2019).

6. What are the popular types and categories of apps // ThinkMobiles [Электронный ресурс]. – Режим доступа: https://thinkmobiles.com/blog/bestaugmented-reality-apps/.

7. Автоматизация тестирования Java EE веб-сервисов с помощью SoapUI и Arquillian // Хабр [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/267301/.

8. Белик, А.Г. Качество и надежность программных систем / Белик А.Г., Цыганенко В.Н.; Учеб.пособие. - Омск 2018. - с. 80.

9. Брюс Эккель, Философия Java / Брюс Эккель; Питер. – Классика Computer Science., 2019. − 1168 с.

10. Василенко Н. В., Макаров В. А. Оценка надежности программного обеспечения // Вестник НовГУ. 2005. №30. [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/otsenka-nadezhnosti-programmnogoobespecheniya.

11. Всегда ли нужны Docker, микросервисы и реактивное программирование // ITnan [Электронный ресурс]. – Режим доступа: http://itnan.ru/post.php?c=1&p=436346.

12. Гради Буч, Язык UML Руководство пользователя / Гради Буч, Джеймс Рамбо, Ивар Якобсон; Москва. – ДМК., 2006. − 483 с.

13. Дьяченко Д.Г. Унификация системного программного обеспечения на основе технологии виртуализации / Д.Г. Дьяченко // Известия ТулГУ. Технические науки. 2012. №5. [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/unifikatsiya-sistemnogo-programmnogoobespecheniya-na-osnove-tehnologii-virtualizatsii.

14. Интеграция бизнес-процессов и сервис-ориентированная архитектура // Решения и технологии [Электронный ресурс]. – Режим доступа: https://www.osp.ru/data/134/081/1237/soa2.pdf.

15. История мобильного Интернета // Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/yota/blog/352450/.

16. Как и для чего использовать Docker // Hexlet Guides [Электронный ресурс]. - Режим доступа: https://guides.hexlet.io/docker/.

17. Колчанов, В.Д. Экономическая эффективность внедрения информационных технологий / Колчанов В.Д., Кобко Л.И.; Учеб.пособие. - Москва 2006. - с. 177.

18. Корпоративная информационная система: определение и структура. Современные подходы к построению корпоративных информационных систем. // Корпоративные информационные системы [Электронный ресурс]: – Режим доступа: http://e-educ.ru/ism14.html.

19. Машинное обучение и мобильная разработка // Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/oleg-bunin/blog/416477/.

20. Методы и средства интеграции информационных систем в рамках единого информационного пространства. [Электронный ресурс]. – Режим доступа: http://lab18.ipu.ru/projects/conf2012/1/7.html.

21. Оценка надежности программного обеспечения // Dev Harmony

[Электронный ресурс]. - Режим доступа: http://kirnosenko.com/2011/03/07/software-reliability-estimation/.

22. Пол Дейтел, Android для разработчиков / Пол Дейтел, Харви Дейтел, Александр Уолд; Санкт-Петербург. – Питер., 2016. − 512 с.

23. Сервис-ориентированная архитектура (SOA) // Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/mailru/blog/342526/.

24. Чистый код с Google Guava // Хабр [Электронный ресурс]. - Режим доступа: https://habr.com/ru/post/244347/.

25. Яковлев В.П. Корпоративные информационные системы: конспект лекций / В.П. Яковлев; СПбГТУРП. – СПб., 2015. − 117 с.

26. "Software Development: A Manager's Guide" // Harvard Business Review [Электронный ресурс] - Режим доступа: https://hbr.org/product/software-development-a-managers-guide/an/R0012A-PDF-ENG

27. "Integration of computer networks and systems" // International Journal of Computer Networks [Электронный ресурс] - Режим доступа: https://airccj.org/ijcnc/CNCO2019.html

28. "Java Programming Language" // Oracle [Электронный ресурс] - Режим доступа: https://www.oracle.com/java/technologies/javase-jdk16-downloads.html

29. "Architecture of Information Systems: Design, implementation, and management" // John Wiley & Sons [Электронный ресурс] - Режим доступа: https://www.wiley.com/en-us/Architecture+of+Information+Systems%3A+Design%2C+Implementation%2C+and+Management

30. "Corporate Information Systems: Challenges and Opportunities" // Journal of Information Systems [Электронный ресурс] - Режим доступа: https://aisel.aisnet.org/jis/vol20/iss1/6/