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

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направлению подготовки)

Технология бизнес-анализа
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему "Технология кастомизации и развертывания в облаке программного продукта с открытым программным кодом (OTRS)"

Обучающийся Л. А. Веселова _____
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд. пед. наук, доцент, О. М. Гущина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Содержание

Введение

Актуальность представленного исследования проявляется в современной деловой среде, где эффективное использование облачных технологий и кастомизации программного обеспечения становится ключевым фактором для увеличения конкурентоспособности предприятий. В условиях быстрого технологического прогресса и стремительных изменений в бизнес-процессах существует срочная необходимость в разработке адаптивных и гибких программных продуктов, способных быстро и эффективно адаптироваться к изменениям внутренних и внешних условий.

Выбор данной темы исследования обусловлен реальной потребностью организаций, представленных в данном случае ООО ЦЗИ "Гриф", в оптимизации процессов кастомизации и развертывания программных продуктов с использованием облачных технологий. Успешная разработка оптимального алгоритма в данной области может существенно улучшить операционные процессы организации, повысить гибкость программных продуктов и, следовательно, способствовать более эффективной адаптации к изменяющимся требованиям рынка.

В настоящее время сфера облачных технологий и кастомизации программного обеспечения приобретает все большее значение в корпоративной среде. Проблема заключается в том, как эффективно интегрировать и использовать облачные технологии для достижения максимальной гибкости и адаптируемости программных продуктов, удовлетворяя уникальным потребностям конкретной организации, такой как ООО ЦЗИ «Гриф». Разработка оптимального алгоритма в этом контексте имеет большое значение для повышения эффективности и конкурентоспособности предприятий в современной цифровой экономике.

Научная разработанность проблемы была освещена в трудах специалистов, рассматривающих вопросы, тесно связанные с технологиями кастомизации и развертывания программных продуктов, проектированием,

реализацией и внедрением мобильных и веб-приложений с использованием облачных технологий, способствующих автоматизации бизнес-процессов предприятий. В их число входят исследования, связанные с вопросами разработки приложений (Дронов В. А, Ковалев С. В., Косарев А. В.), вопросы проектирования информационных систем (Гнеденко В. В., Прохоренок Н. А.), вопросы проектирования и реализации баз данных современных информационных систем с использованием облачных технологий (Симдянов И. К., Фрейн Б. А.).

Гипотеза заключается в том, что использование технологии по кастомизации и развертыванию программных продуктах, на примере мобильного приложения для ООО ЦЗИ «Гриф», позволит сформулировать оптимальный алгоритм действий по кастомизации и развертыванию программного продукта с использованием облачных технологий.

Цель данной работы заключается в разработке технологии кастомизации и развертывания программного продукта с использованием облачных технологий. В рамках этой работы необходимо решить следующие задачи:

– осуществить анализ предметной области;

– исследовать информацию об облачных платформах и провести их сравнение;

– исследовать инструменты управления зависимостями при развертывании программных продуктов, провести их сравнение;

– описать особенности мобильной разработки и развертывания на облачных платформах;

– обосновать выбор инструментов разработки мобильного приложения;

– сформировать техническое задание;

– спроектировать архитектуру исследуемой облачной платформы;

– разработать мобильное приложение с использованием облачной платформы для Центра защиты информации «Гриф»;

– сформировать алгоритм по развертыванию и кастомизации программных продуктов с использованием облачных технологий.

Объектом исследования является деятельность ООО ЦЗИ «Гриф». Предметом исследования является процесс проектирования облачного сервиса для ООО ЦЗИ «Гриф».

В ходе исследования применялись следующие методы: изучение соответствующей литературы и интернет-ресурсов, теоретический и сравнительный анализ, проектирование и программирование.

Практическая значимость работы заключается в том, что предложенные в рамках исследования подходы и рекомендации могут быть использованы организациями для оптимизации своей деятельности и улучшения конкурентоспособности. Внедрение облачных технологий может привести к снижению затрат на обслуживание и развитие информационной инфраструктуры, улучшению качества обслуживания клиентов и партнеров, а также повышению общей эффективности бизнеса.

1 Анализ предметной области

1.1 Анализ деятельности исследуемого предприятия

1.1.1 Технико-экономическая характеристика деятельности предприятия

Центр защиты информации «Гриф» представляет собой организацию, специализирующуюся на оказании услуг в области информационных технологий и защиты информации. Основанная в 2008 году, компания успешно функционирует на рынке, руководствуясь принципами комплексности, разумной достаточности и умеренной ценовой политики. Клиенты являются высшей ценностью для компании, что отражается в постоянном стремлении к высокому качеству выполняемых работ.

Центр защиты информации «Гриф» предоставляет услуги по организации защиты конфиденциальной информации в средствах автоматизации, а также в помещениях, предназначенных для конфиденциальных переговоров. Компания предоставляет аттестацию этих систем и помещений согласно требованиям безопасности информации.

С учетом значимости информации в современном бизнесе организация ориентирована на грамотное управление и обеспечение безопасности информации от внешних угроз. Учитывая использование информационных систем, основанных на различных технологиях, Центр защиты информации «Гриф» предоставляет услуги по организации защиты конфиденциальной информации в автоматизированных системах и помещениях организаций.

Организация предоставляет комплексные работы по оценке и обеспечению безопасности информации. Это включает в себя определение степени защищенности, выбор технических средств защиты, их установку и настройку. Аттестация производится в соответствии с требованиями нормативно-правовых актов по защите информации [36].

Центр защиты информации «Гриф» оказывает услуги как коммерческим организациям, работающим с конфиденциальной информацией, так и государственным учреждениям, для которых аттестация систем и помещений по безопасности информации обязательна. В своем стремлении к развитию технологий информатизации и обеспечению их безопасности, компания занимает активную позицию в соответствующей отрасли.

Основные технические и экономические характеристики компании представлены в таблице 1.

Таблица 1 – Основные технические и экономические характеристики компании

Наименование характеристики (показателя)	**Значение показателя**
Количество работников	45 работников - на середину 2023 года
Количество офисов в городе Рыбинск	1 - на середину 2023 года
Количество обращений в день	150 шт.
Среднее время регистрации заявки	4 мин. (по первым 2 кварталам 2023 года)
Среднее время, необходимое для согласования и направления заявки нужному исполнителю	25 мин. (по первым 2 кварталам 2023 года)

Данные показатели позволяют сделать вывод о том, что финансовые показатели компании за последний год относительно высокие, и важно поддерживать эти показатели и дальше.

1.1.2 Краткая характеристика подразделения или видов его деятельности

Организационную структуру управления предприятием можно представить так, как показано на рисунке 1. Высший руководящий орган – это директор.

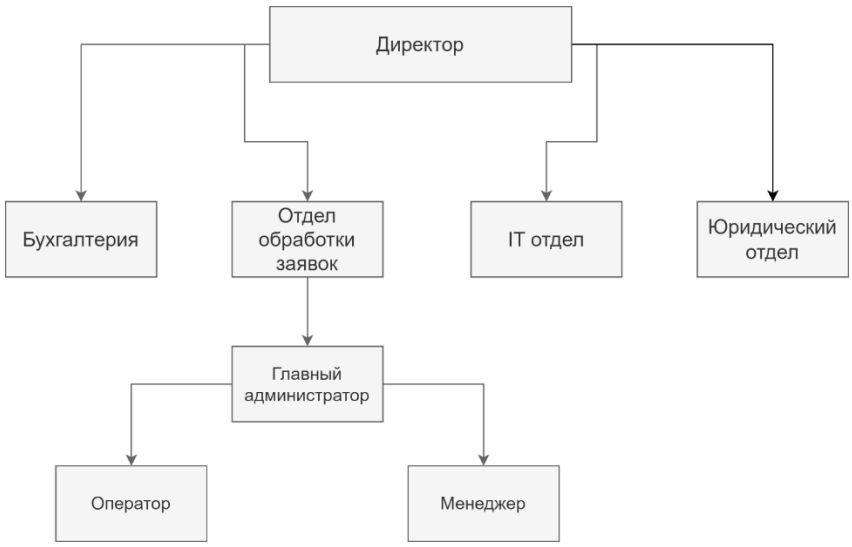

Рисунок 1 – Схема организационной структуры управления предприятием

Отдел финансов и бухгалтерии предприятия ответственен за финансовый учет и формирование отчетности. Его функционал включает в себя ведение бухгалтерских записей, мониторинг финансовых потоков, подготовку отчетов для финансовых органов и обеспечение прозрачности финансовой деятельности предприятия.

Отдел обработки заявок предприятия занимается приемом, обработкой и регистрацией запросов и заявлений от клиентов. Этот отдел эффективно взаимодействует с клиентами, собирая необходимую информацию и направляя запросы к соответствующим специалистам, обеспечивая высокий уровень обслуживания.

IT отдел нашего предприятия ответственен за обеспечение работоспособности информационной инфраструктуры и предоставление технической поддержки. Этот отдел следит за работой компьютерных систем, сетей и программного обеспечения, гарантируя бесперебойную деятельность предприятия и обеспечивая безопасность хранимых данных.

Юридический отдел предоставляет юридическую поддержку предприятия. Он занимается разрешением юридических вопросов, предоставляет консультации по законодательству, готовит документы,

участвует в правовых процедурах и обеспечивает соблюдение прав и интересов клиентов предприятия.

Предприятие обладает линейно-функциональной структурой управления, где функциональные подразделения не обладают правом принятия решений и прямого руководства подчиненными структурами. Вместо этого они участвуют в определении задач, подготовке решений и оказывают помощь линейным руководителям в выполнении отдельных управленческих функций.

1.1.3 Программная и техническая архитектура ИС предприятия

На предприятии используются различные системы управления. Например, автоматизированная система управления и планирования ресурсов "BAAN". Так же используется система управления задачами Trello и собственные разработки программного обеспечения, такие как АС "ПРОВЭД". Хранение информации осуществляется в СУБД PostgreSQL, а на рабочих станциях пользователей установлена операционная система Windows 10. Схема технической архитектуры показана на рисунке 2.

«BAAN» — это программная система планирования ресурсов предприятия (ERP). «BAAN» предоставил организациям комплексный набор интегрированных бизнес-приложений для управления различными аспектами их деятельности, включая производство, финансы, человеческие ресурсы, управление цепочками поставок, управление взаимоотношениями с клиентами и многое другое. Он был направлен на оптимизацию бизнес-процессов, повышение эффективности и облегчение принятия решений на основе данных.

Система «BAAN» состоит из различных модулей, которые можно настраивать в соответствии с конкретными потребностями организации. Он предлагает такие функции, как планирование производства, управление запасами, обработка заказов, финансовый учет, управление проектами, а также распределение.

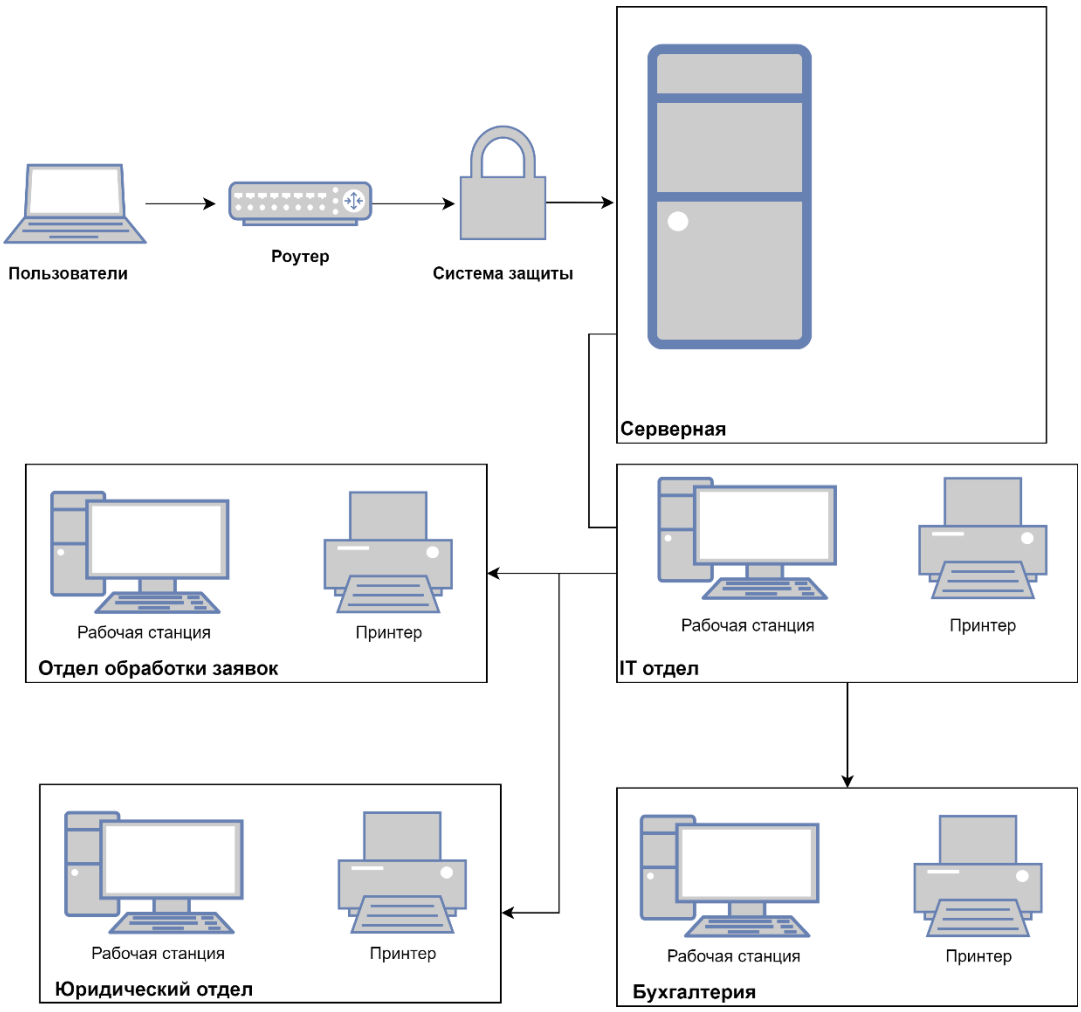

Рисунок 2 – Схема технической архитектуры

Одной из ключевых особенностей «BAAN» является многоуровневая клиент-серверная архитектура, которая позволяет пользователям взаимодействовать с системой через графический пользовательский интерфейс (GUI). Он поддерживает различные базы данных и операционные системы, обеспечивая гибкость для организаций с разнообразной ИТ-инфраструктурой.

Система «BAAN» применяется в организации для автоматизации управления производственными процессами, обеспечения поставок и финансовых операций. Она развернута на главном сервере и доступна всем сотрудникам, имеющим соответствующие разрешения на доступ. Схема программной архитектуры представлена на рисунке 3.

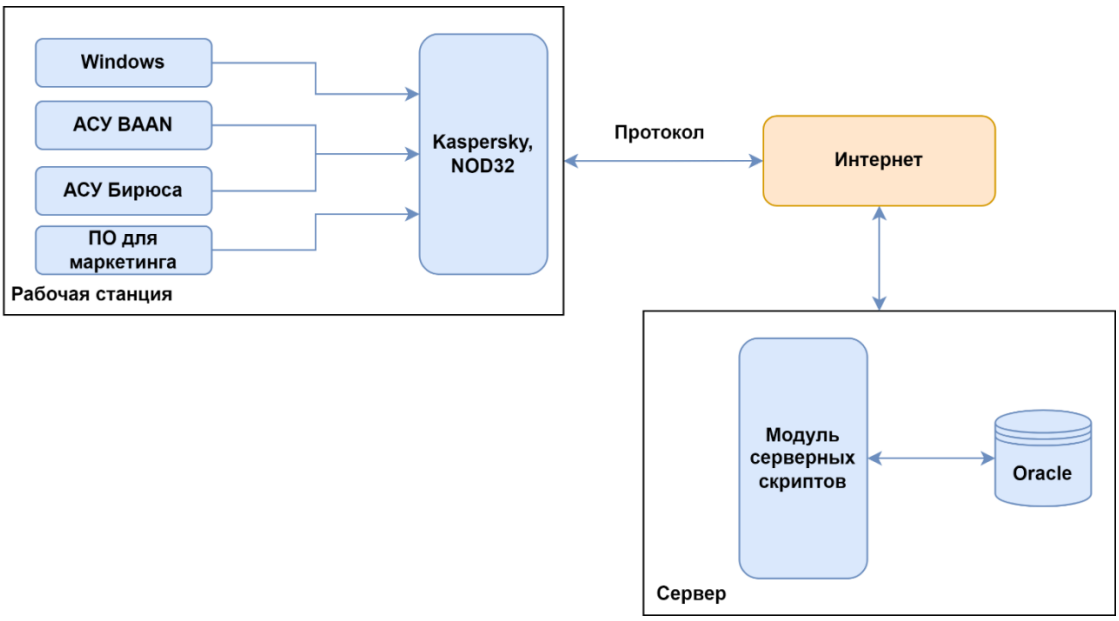

Рисунок 3 – Программная архитектура

На компьютерах работников установлено программное обеспечение, которое не полностью удовлетворяет их потребности, поэтому требуется расширение используемого программного обеспечения. Список программного обеспечения приведен в таблице 2.

Таблица 2 – Анализ программного обеспечения

Техническое/программное обеспечение	**Требует обновления (Да/Нет)**
Операционная система Windows 10	Нет
пакет Microsoft Office 365	Да
Антивирус Касперского	Да
Google Chrome	Да
Skype	Нет
Telegram	Нет
Adobe Illustrator CC 2022	Да
Яндекс Почта – почтовый клиент	Нет
CRM от Tilda.cc – система управления заявками новых клиентов	Да
Яндекс Директ – рекламный кабинет Яндекса	Да
Trello – менеджер задач	Нет

По результатам проведенного анализа выявлено, что программное обеспечение, используемое в компании, не способно полноценно охватить все аспекты работы предприятия. В частности, отсутствует эффективный инструмент для предоставления услуг клиентам.

1.1.4 Обоснование необходимости автоматизации задачи

Для формального описания бизнес-процессов мы построим функциональную модель IDEF0. «Основным элементом IDEF0-модели является диаграмма, которая представляет собой графическое описание модели предметной области или ее части. Основными компонентами IDEF0-диаграммы являются блоки» [34]. «Каждый блок диаграммы соответствует определенной функции, для которой необходимо определить входные данные, результат, управляющую функцию и механизм ее реализации. Описание взаимодействия функций с внешней средой и между собой осуществляется с помощью дуг (связей)» [31].

На рисунке 4 представлена IDEF0-модель, которая была декомпозирована на четыре подуровня.

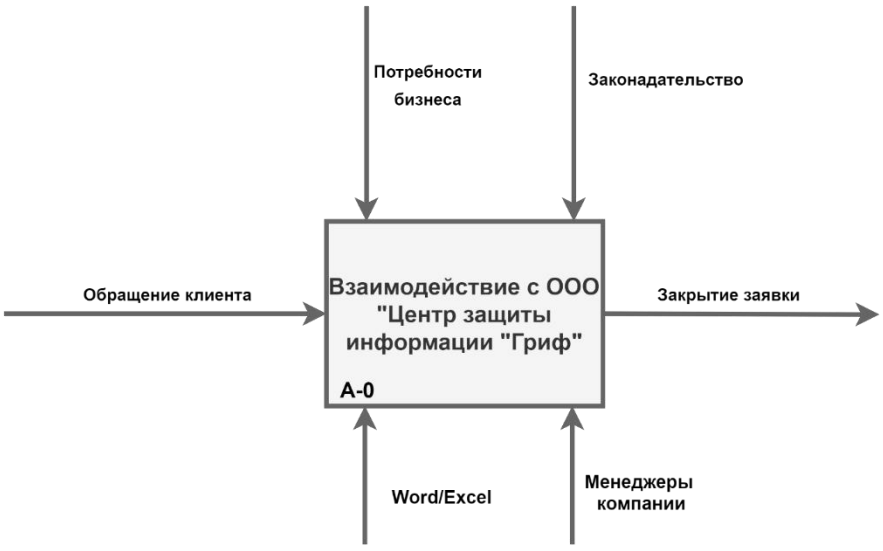

Рисунок 4 – Диаграмма верхнего уровня (A-0) для процесса КАК ЕСТЬ

На верхнем уровне блок A0 отвечает за взаимодействие на основе обращений клиентов. Результатом этого является закрытие заявки. Управлением служат законодательство и договор на оказание услуг. Механизмом реализации являются менеджеры компании.

Из представленного рисунка можно сделать вывод, что предприятие использует программы Microsoft Word и Excel для обслуживания своих клиентов. Более подробная информация о данном процессе представлена на рисунке 5, где изображена его декомпозиция. В ходе этого процесса были выделены четыре подзадачи, что является результатом детализации первого уровня.

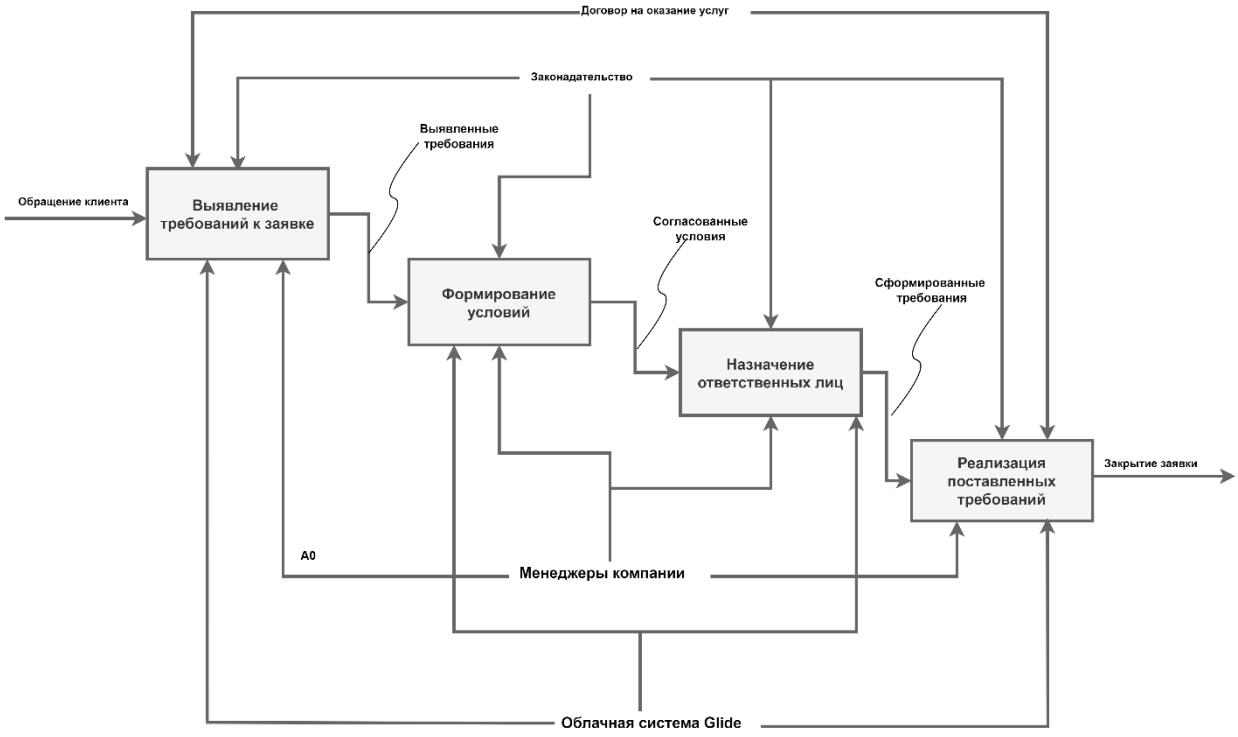

Рисунок 5 – Диаграмма декомпозиции A0

Действующий уровень управления предприятием характеризуется недостаточной степенью автоматизации, что негативно влияет на его способность адаптироваться к изменениям и увеличивает объем рутинной работы. Менеджеры для фиксации задач, которые ставятся перед

сотрудниками, используют файлы Excel. Однако функционал этих файлов ограничен и не способен эффективно решать многие задачи, стоящие перед предприятием.

1.2 Исследование технологий кастомизации и развертывания программных продуктов

1.2.1 Этапы кастомизации и развертывания программного продукта

«Процесс настройки и развертывания программного продукта с использованием облачных технологий состоит из нескольких этапов» [19]. Начальным шагом является проектирование и разработка программного приложения, в ходе которого разработчики создают базовые функциональные возможности и функции на основе требований и спецификаций, предоставленных заинтересованными сторонами. После разработки приложения его необходимо настроить в соответствии с конкретными потребностями и предпочтениями пользователей или организаций, которые будут его использовать.

Настройка обычно включает в себя изменение пользовательского интерфейса, добавление или удаление функций, интеграцию с другими системами или службами, а также настройку параметров для соответствия рабочим процессам. Облачные технологии предоставляют гибкую и масштабируемую среду для настройки программных приложений, позволяя разработчикам быстро вносить изменения и обновления по мере необходимости, не нарушая работу приложения [36].

После завершения настройки следующим шагом будет развертывание программного продукта в облачной среде. Развертывание включает настройку инфраструктуры и ресурсов, необходимых для запуска приложения, таких как виртуальные машины, базы данных, хранилище и сеть. Поставщики облачных услуг предлагают различные варианты

развертывания, включая общедоступные, частные и гибридные облака, а также модели контейнеризации и бессерверных вычислений.

Во время развертывания разработчикам необходимо убедиться, что приложение правильно настроено, оптимизировано по производительности и защищено от потенциальных угроз и уязвимостей. Это может включать настройку контроля доступа, реализацию шифрования и мониторинг производительности и доступности приложения. Поставщики облачных услуг предлагают ряд инструментов и услуг, которые помогают разработчикам управлять и контролировать свои приложения в облаке, например, облачный мониторинг, ведение журналов и службы безопасности.

После развертывания программного продукта в облачной среде он готов к доступу и использованию конечными пользователями или клиентами. Облачные технологии позволяют пользователям получать доступ к приложению из любой точки мира, используя любое устройство, подключенное к Интернету, что делает его доступным и удобным для широкого круга пользователей. Кроме того, облачное развертывание обеспечивает легкую масштабируемость и обновление, гарантируя, что программный продукт может расти и развиваться вместе с потребностями пользователей с течением времени.

Процесс настройки и развертывания программного продукта с использованием облачных технологий предполагает тщательное планирование, проектирование, разработку, настройку и развертывание. Облачные технологии обеспечивают гибкую, масштабируемую и безопасную среду для создания и запуска программных приложений, позволяя организациям предоставлять инновационные решения, отвечающие потребностям их пользователей и клиентов.

1.2.2 Особенности современных облачных платформ

Облачная платформа – это тип вычислительной службы, которая предоставляет пользователям доступ к различным ресурсам и услугам через Интернет. Эти ресурсы могут включать в себя вычислительную мощность,

хранилище, базы данных, сети и приложения, которые размещаются и управляются поставщиком облачных услуг в его центрах обработки данных.

Облачные платформы позволяют организациям получать доступ к вычислительным ресурсам и использовать их по требованию без необходимости инвестиций и обслуживания физической инфраструктуры. Облачные платформы позволяют пользователям легко регулировать количество используемых ресурсов в соответствии с их потребностями, оплачивая только за фактически использованные ресурсы. Такой подход, основанный на принципе оплаты по мере использования, обеспечивает высокую гибкость и масштабируемость, что делает облачные платформы привлекательными для предприятий любого размера, от начинающих компаний до крупных корпораций [41].

Поставщики облачных услуг, также известные как провайдеры облачных услуг (CSP) — это компании, которые предлагают услуги и решения облачных вычислений частным лицам, предприятиям и организациям. Эти провайдеры создают и управляют огромными центрами обработки данных по всему миру для размещения и управления инфраструктурой и услугами, необходимыми для облачных вычислений.

Примеры поставщиков облачных услуг включают Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), IBM Cloud и Alibaba Cloud и другие. Каждый поставщик облачных услуг предлагает типовой список услуг, включая: инфраструктуру как услугу (IaaS), платформу как услугу (PaaS) и программное обеспечение как услугу (SaaS), а также специализированные услуги, такие как машинное обучение, искусственный интеллект, Интернет. Вещи (IoT) и блокчейн [21].

Поставщики облачных услуг играют решающую роль в предоставлении организациям возможности использовать облачные вычисления для инноваций, масштабирования и преобразования своего бизнеса. Они предоставляют необходимую инфраструктуру, инструменты и услуги для создания, развертывания и управления приложениями и рабочими

нагрузками в облаке. Кроме того, поставщики облачных услуг вкладывают значительные средства в безопасность, соответствие требованиям и надежность, чтобы обеспечить целостность и доступность своих услуг, предлагая клиентам спокойствие при переходе в облако.

Amazon Web Services (AWS), Microsoft Azure и Google Cloud Platform (GCP) – три ведущие платформы облачных вычислений в отрасли. Эти платформы предлагают широкий спектр облачных услуг и решений для удовлетворения разнообразных потребностей предприятий и организаций.

AWS, запущенная Amazon в 2006 году, является одной из первых и наиболее широко распространенных платформ облачных вычислений. «Он предоставляет комплексный набор услуг, включая вычислительную мощность, хранилище, базы данных, машинное обучение, аналитику и многое другое. AWS известна своей масштабируемостью, надежностью и глобальной инфраструктурой, центры обработки данных расположены в регионах по всему миру» [45].

«Microsoft Azure, представленная Microsoft в 2010 году, является еще одной известной платформой облачных вычислений. Он предлагает разнообразный набор услуг, включая виртуальные машины, базы данных, искусственный интеллект и машинное обучение, Интернет вещей и инструменты для разработчиков. Azure тесно интегрирован с экосистемой продуктов и услуг Microsoft, что делает его популярным выбором для компаний, уже использующих технологии Microsoft» [48].

«Облачная платформа Google (GCP), запущенная Google в 2008 году, известна своими передовыми технологиями и инновациями. Он предоставляет широкий спектр услуг, включая вычисления, хранение, базы данных, машинное обучение и анализ данных. GCP известна своим опытом в таких областях, как искусственный интеллект, анализ данных и контейнеризация с такими инструментами, как TensorFlow и Kubernetes» [43].

Каждая облачная платформа предлагает уникальные функции и преимущества, соответствующие различным сценариям использования и отраслям. AWS обладает обширной экосистемой сервисов и сильным присутствием на рынке, что делает его подходящим для широкого спектра приложений. «Azure предпочитают предприятия, которые в значительной степени полагаются на продукты и услуги Microsoft, предлагающие бесшовную интеграцию и возможности гибридного облака. GCP известна своими передовыми технологиями и опытом в таких областях, как машинное обучение и анализ данных, которые подходят организациям со сложными требованиями» [24].

AWS, Azure и GCP являются ведущими платформами облачных вычислений, которые играют решающую роль в стимулировании инноваций и цифровой трансформации в различных отраслях. Постоянное развитие и инвестиции в новые технологии делают их ключевыми игроками в сфере облачных вычислений.

1.2.3 Инструменты управления зависимостями

Инструменты управления зависимостями – это программные утилиты, предназначенные для работы с внешними компонентами и библиотеками, используемыми в процессе разработки программного обеспечения. Эти инструменты играют решающую роль в управлении зависимостями программного проекта, обеспечивая доступность и совместимость необходимых компонентов и библиотек с требованиями проекта.

Инструменты управления зависимостями автоматизируют процесс получения, установки и управления зависимостями, упрощая рабочий процесс разработки и снижая риск ошибок и несоответствий. Эти инструменты обычно предоставляют такие функции, как разрешение зависимостей, управление версиями, разрешение конфликтов и изоляция зависимостей [27].

Общие примеры инструментов управления зависимостями включают менеджеры пакетов, такие как npm (менеджер пакетов узлов) для JavaScript,

pip для Python, Maven для Java и Composer для PHP. Эти инструменты позволяют разработчикам объявлять зависимости своих проектов в файле конфигурации, указывая необходимые пакеты и их версии.

Затем инструмент управления зависимостями извлекает указанные пакеты из репозиториев, разрешает зависимости и устанавливает их в среду проекта. Это также гарантирует актуальность и совместимость зависимостей друг с другом, автоматически обновляя их при необходимости.

Помимо менеджеров пакетов, инструменты автоматизации сборки, такие как Gradle и Apache Ant, также предоставляют возможности управления зависимостями, позволяя разработчикам указывать зависимости и включать их в процесс сборки.

Таким образом, инструменты управления зависимостями необходимы для современной разработки программного обеспечения, позволяя разработчикам эффективно управлять и контролировать зависимости своих проектов, обеспечивая стабильность, согласованность и совместимость на протяжении всего жизненного цикла разработки.

1.2.4 Анализ инструментов изоляции для работы на облачных платформах

Docker – широко используемая платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры — это легкие, переносимые и самодостаточные среды, содержащие все необходимое для запуска приложения, включая код, среду выполнения, библиотеки и зависимости. Docker предоставляет инструменты и сервисы для создания, управления и развертывания контейнеров, упрощая работу разработчиков и операторов с контейнерными приложениями.

Одной из ключевых особенностей Docker является его Docker Engine, который представляет собой среду выполнения контейнеров, запускающую контейнеры в операционной системе хоста. Docker Engine предоставляет простой и эффективный способ создания, запуска, остановки контейнеров и управления ими с помощью таких команд, как docker run, docker stop и docker

ps. Docker также включает инструменты для создания образов контейнеров, такие как Dockerfile и docker build, которые позволяют разработчикам определять среду и зависимости, необходимые для их приложений [46].

Еще одним важным компонентом Docker является Docker Hub, облачная служба реестра, в которой размещаются образы контейнеров. Docker Hub позволяет пользователям хранить, обмениваться и обнаруживать образы контейнеров, упрощая поиск и использование готовых образов для популярных программных стеков и приложений.

Помимо Docker, существует несколько альтернативных сред выполнения контейнеров и инструментов, каждый из которых имеет свои собственные функции и возможности. Podman – это инструмент управления контейнерами, разработанный Red Hat, который предоставляет совместимый с Docker интерфейс командной строки для управления контейнерами и образами. Podman спроектирован так, чтобы быть более легким и безопасным, чем Docker, с использованием архитектуры без демонов, которая устраняет необходимость в центральном процессе-демоне.

rkt, также известный как CoreOS rkt, — это еще одна среда выполнения контейнера, ориентированная на безопасность, простоту и возможность компоновки. rkt спроектирован как модульный и расширяемый, с минималистической архитектурой, которая уменьшает поверхность атаки и улучшает изоляцию. rkt поддерживает множество форматов контейнеров и сред выполнения, что делает его подходящим для широкого спектра случаев использования.

Containerd – это среда выполнения контейнера и демон, который используется в качестве основного механизма выполнения для Docker и других контейнерных платформ. Containerd предоставляет легкую и эффективную среду выполнения для запуска контейнеров с упором на стабильность, производительность и масштабируемость. Containerd спроектирован так, чтобы быть расширяемым и модульным, что позволяет

легко интегрировать его с другими инструментами и платформами управления контейнерами [38].

В таблице 3 приведен сравнительный анализ инструментов изоляции при работе на облачных платформах.

Таблица 3 – Сравнительный анализ инструментов изоляции

Критерий	**Docker**	**Podman**	**rkt**	**containerd**
Тип контейнера	Легковесный контейнер	Легковесный контейнер	Легковесный контейнер	Инфраструктурный контейнер
Демон необходим	Да	Нет	Нет	Да
Совместимость с Docker	Да	Да	Нет	Да
Изоляция	Да	Да	Да	Да
Активное сообщество	Да	Да	Нет	Да
Интеграция инструментов	Высокая	Средняя	Средняя	Средняя
Уровень безопасности	Средний	Высокий	Высокий	Высокий
Простота использования	Высокая	Высокая	Высокая	Высокая

Docker и его альтернативы, такие как Podman, rkt и Containerd, предоставляют разработчикам и командам эксплуатации гибкие и мощные инструменты для создания, управления и запуска контейнерных приложений. Эти инструменты позволяют организациям использовать преимущества контейнеризации, такие как переносимость, масштабируемость и эффективность, для модернизации процессов разработки и развертывания программного обеспечения.

Каждый из этих инструментов имеет свои особенности и применение в различных сценариях. Выбор между ними зависит от конкретных требований, привычек команды разработчиков и целей проекта.

1.3 Особенности современной мобильной разработки и развертывание на облачных платформах

1.3.1 Современные мобильные операционные системы

Мобильное приложение – это программное обеспечение, разработанное для установки и использования на мобильных устройствах, таких как смартфоны и планшеты. Оно предназначено для выполнения определенных функций, предоставления сервисов, обмена информацией и упрощения пользовательского опыта на мобильных платформах.

Android — мобильная операционная система, разработанная в компании Google. Ее история восходит к 2003 году, когда Энди Рубин, Рич Майнер, Ник Сирс и Крис Уайт основали Android Inc. Первоначальной целью была разработка передовой операционной системы для цифровых камер, но, осознав потенциал растущего рынка смартфонов, фокус сместился к созданию операционной системы для мобильных устройств. Google приобрела Android Inc. в 2005 году, ознаменовав начало ее пути к тому, чтобы стать доминирующим игроком на рынке мобильных ОС.

Первый коммерческий выпуск Android состоялся в сентябре 2008 года — Android 1.0. Далее последовали постоянные обновления и улучшения. Одной из определяющих особенностей Android является его открытый исходный код, который позволяет производителям настраивать и модифицировать операционную систему в соответствии со своими предпочтениями. Эта гибкость в значительной степени способствовала широкому распространению Android на различных устройствах разных производителей [17].

На протяжении многих лет Android быстро развивался, вводя новые функции, улучшения и оптимизации с каждым крупным обновлением. Платформа выросла и теперь поддерживает широкий спектр устройств, включая смартфоны, планшеты, умные часы, телевизоры и даже автомобильные системы. Универсальность и адаптивность сделали ее любимой операционной системой миллионов пользователей по всему миру.

Одним из ключевых преимуществ Android является обширная экосистема приложений. В магазине Google Play, официальном магазине

приложений для Android, размещены миллионы приложений, отвечающих практически любым потребностям и предпочтениям. Эта богатая экосистема предоставляет пользователям доступ к широкому спектру приложений: от инструментов повышения производительности и развлекательных приложений до игр и утилит.

Android предлагает бесшовную интеграцию со службами Google, такими как Gmail, Google Drive, Google Maps и Google Assistant, повышая производительность и удобство для пользователей, которые полагаются на эти службы. Интеграция облачных сервисов также гарантирует синхронизацию пользовательских данных на нескольких устройствах, обеспечивая единообразную и бесперебойную работу.

Настраиваемость Android — еще одно заметное преимущество. Пользователи имеют свободу персонализировать свои устройства, устанавливая собственные программы запуска, темы и виджеты, адаптируя пользовательский интерфейс по своему вкусу. Кроме того, Android поддерживает широкий спектр параметров настройки на уровне системы, позволяя пользователям настраивать параметры, разрешения и предпочтения в соответствии со своими потребностями.

Более того, Android способствует инновациям и конкуренции в мобильной индустрии, предоставляя производителям платформу для дифференциации своих продуктов посредством инноваций в аппаратном обеспечении и настройки программного обеспечения. Эта конкурентная среда приносит пользу потребителям, предлагая разнообразный набор устройств с различными характеристиками, дизайном и ценой.

Подводя итог, можно сделать вывод, что Android стал доминирующей силой в индустрии мобильных технологий благодаря своему открытому исходному коду, обширной экосистеме приложений, бесшовной интеграции со службами Google, настраиваемости и поддержке инноваций. Его продолжающееся развитие и широкое распространение подчеркивают его значение в формировании будущего мобильных компьютеров.

iOS — мобильная операционная система, разработанная Apple Inc. для линейки мобильных устройств, включая iPhone, iPad и iPod Touch. История iOS берет свое начало с появления первого iPhone в 2007 году, который произвел революцию в индустрии смартфонов благодаря интуитивно понятному интерфейсу с сенсорным экраном и бесшовной интеграции аппаратного и программного обеспечения.

iOS изначально возникла как закрытая экосистема, жестко контролируемая Apple, что отличало ее от Android с открытым исходным кодом. Такой подход позволил Apple обеспечить строгий контроль над пользовательским интерфейсом, гарантируя согласованность и надежность всех своих устройств. Закрытая экосистема также способствовала плавной интеграции с другими продуктами и сервисами Apple, такими как iCloud, iTunes и App Store [7].

Одной из определяющих особенностей iOS является удобный интерфейс, отличающийся простотой, элегантностью и логичностью. Интуитивно понятные мультитач-жесты, плавная анимация и визуально привлекательный дизайн способствуют плавному и приятному взаимодействию с пользователем. iOS известна своей стабильностью, безопасностью и производительностью, при этом Apple уделяет большое внимание оптимизации операционной системы для своего оборудования, чтобы обеспечить исключительную производительность и время автономной работы.

App Store, официальный магазин приложений Apple для iOS, может похвастаться огромным и разнообразным выбором приложений: от инструментов повышения производительности и приложений для социальных сетей до игр и мультимедийного контента. Строгий процесс проверки Apple гарантирует, что пользователям будут доступны только высококачественные и безопасные приложения, что повышает доверие к платформе.

Устройства iOS известны своей плавной интеграцией с другими продуктами Apple, такими как компьютеры Mac, Apple Watch и AirPods. Такие функции, как Handoff, Continuity и AirDrop, позволяют легко обмениваться контентом и синхронизировать его между устройствами, создавая целостную и взаимосвязанную экосистему для пользователей.

Приверженность Apple обеспечению конфиденциальности и безопасности — еще одно ключевое преимущество iOS. Компания использует надежные механизмы шифрования, строгую политику защиты данных и регулярные обновления безопасности для защиты пользовательских данных и защиты от вредоносных программ и киберугроз. Кроме того, такие функции, как Face ID и Touch ID, предлагают безопасные методы аутентификации, повышая общую безопасность устройств iOS [18].

Кроме того, iOS выигрывает от сильного сообщества разработчиков, которое постоянно расширяет границы разработки приложений и инноваций. Разработчики имеют доступ к мощным инструментам и ресурсам, таким как Xcode и SwiftUI, для создания захватывающих и многофункциональных приложений, использующих новейшие технологии, включая дополненную реальность, машинное обучение и дополненную реальность.

Итак, «основные преимущества iOS таковы:

– уделяет приоритетное внимание безопасности и конфиденциальности благодаря регулярным обновлениям и надежным механизмам шифрования для защиты пользовательских данных;

– предлагает интуитивно понятный и визуально привлекательный интерфейс, который помогает пользователям без труда ориентироваться в своих устройствах и взаимодействовать с ними;

– устройства iOS легко совмещаются с другими продуктами и услугами Apple, формируя целостную экосистему для пользователей;

– iOS оптимизирована для оборудования Apple, обеспечивая плавную работу и эффективное управление ресурсами» [46].

Ключевые недостатки iOS:

– iOS предоставляет меньше возможностей для персонализации по сравнению с Android, поскольку существуют ограничения на изменение внешнего вида интерфейса и настроек системы;

– обновления iOS часто приносят постепенные улучшения, а не радикальные инновации, что иногда создаёт ощущение отсутствия прогресса;

– устройства Apple обычно стоят дороже, чем аналогичные модели на Android, что делает их менее доступными для покупателей, которые стеснены в средствах;

– iOS функционирует в закрытой экосистеме, которую контролирует Apple, что ограничивает вариативность и выбор для пользователей;

– пользователи iOS сильно зависят от экосистемы Apple в отношении таких сервисов, как iCloud, iTunes и App Store.

Таким образом, iOS выделяется как мобильная операционная система премиум-класса, известная своим удобным интерфейсом, стабильностью, безопасностью, полной интеграцией с продуктами Apple, надежной экосистемой приложений и приверженностью конфиденциальности. Несмотря на то, что iOS может иметь закрытую экосистему по сравнению с Android, ориентация iOS на качество, надежность и инновации укрепила ее позицию ведущей платформы в сфере мобильных технологий.

1.3.2 Методы разработки мобильных приложений

Особенности разработки современных мобильных приложений включают следующие аспекты:

– Мобильная платформа. Современные мобильные приложения разрабатываются под конкретные операционные системы мобильных устройств, такие как iOS (для устройств Apple) и Android (для большинства смартфонов и планшетов). Разработчики должны учитывать различия в архитектуре, интерфейсах и ограничениях каждой платформы при создании приложений [43].

– Адаптивный дизайн. Современные мобильные приложения должны иметь адаптивный дизайн, который обеспечивает оптимальное отображение и взаимодействие с контентом на разных размерах экранов мобильных устройств. Разработчики должны учитывать различные разрешения экранов и ориентацию устройств.

– Пользовательский интерфейс и опыт. Один из ключевых аспектов разработки мобильных приложений – это создание интуитивного и привлекательного пользовательского интерфейса. Пользовательский опыт должен быть удобным, эффективным и соответствовать ожиданиям пользователей. Разработчики стремятся создать простой в использовании интерфейс с понятной навигацией и минимизацией лишних действий со стороны пользователя.

– Оптимизация производительности. Мобильные приложения должны быть оптимизированы для эффективного использования ресурсов мобильных устройств, таких как процессор, память и батарея. Разработчики стараются создать приложения, которые работают плавно, быстро загружаются и не приводят к излишнему потреблению энергии.

– Интеграция с устройствами и сервисами. Современные мобильные приложения часто требуют интеграции с различными функциями и сервисами мобильных устройств, такими как камера, геолокация, датчики, платежные системы и социальные сети. Разработчики должны уметь взаимодействовать с API и различными SDK для реализации таких функциональностей.

– Обновления и поддержка. Разработка мобильных приложений не заканчивается после выпуска. Важно предоставлять обновления приложений, внедрять исправления ошибок и улучшать функциональность на основе полученной обратной связи пользователей. Регулярные обновления и поддержка помогают поддерживать высокое качество и удовлетворенность пользователей.

Разработка современных мобильных приложений требует комбинации технических навыков, дизайнерской эстетики и понимания потребностей пользователей для создания удобных, функциональных и успешных приложений на мобильных платформах.

Разработка нативных мобильных приложений — это процесс создания приложений, специально предназначенных для работы на определенной платформе или операционной системе, с использованием собственных языков программирования, инфраструктур и инструментов, предоставленных разработчиком платформы. Например, приложения для iOS обычно разрабатываются с использованием языков программирования Swift или Objective-C и iOS SDK, а приложения для Android создаются с использованием Java или Kotlin и Android SDK. Такой подход позволяет разработчикам использовать все возможности платформы, включая доступ к собственным API, системным ресурсам и аппаратным функциям, для создания высокооптимизированных и производительных приложений [22].

Нативная разработка предлагает ряд преимуществ, включая оптимальную производительность, плавную интеграцию с экосистемой платформы и доступ к специфичным для платформы функциям и возможностям. Используя собственные API и платформы, разработчики могут создавать приложения, которые обеспечивают превосходную производительность, оперативность и удобство работы с пользователем по сравнению с кроссплатформенными. Кроме того, нативные приложения могут использовать специфичные для платформы функции, такие как push-уведомления, службы геолокации, доступ к камере и аппаратное ускорение.

Однако нативная разработка также сопряжена с некоторыми проблемами и ограничениями. Одним из основных недостатков является необходимость поддерживать отдельные базы кода для каждой платформы, что может увеличить время и стоимость разработки. Поскольку каждая платформа имеет свой собственный набор языков программирования, инструментов и сред разработки, разработчикам может потребоваться

изучить и овладеть несколькими технологиями, чтобы эффективно работать с разными платформами. Более того, обновления и изменения API-интерфейсов платформы или языков программирования могут потребовать внесения изменений в кодовую базу приложения, что приведет к дополнительным затратам на обслуживание. Напротив, кроссплатформенные подходы к разработке позволяют разработчикам написать код один раз и развернуть его на нескольких платформах, таких как iOS, Android, а иногда даже в веб-браузерах, используя единую базу кода. Технологии, поддерживающие работу на различных платформах, такие как React Native, Xamarin и Flutter, позволяют разработчикам создавать мобильные приложения с использованием привычных веб-технологий, таких как HTML, CSS и JavaScript. Эти языки программирования затем преобразуются в нативный код во время выполнения приложения или предварительно компилируются [29].

Кроссплатформенная разработка имеет ряд преимуществ, среди которых сокращение времени и затрат на создание приложений, упрощение поддержки и возможность охвата большей аудитории с помощью единого кода. Используя общую базу кода, разработчики могут оптимизировать процесс разработки и обеспечить согласованность на разных платформах, сохраняя при этом преимущества производительности и удобства взаимодействия с пользователями, сравнимые с нативными. Кроме того, кроссплатформенные платформы часто предоставляют инструменты и библиотеки для доступа к функциям и API, специфичным для платформы, что сводит к минимуму потребность в коде, специфичном для платформы.

Однако такая разработка имеет некоторые недостатки, такие как потенциальные узкие места в производительности, ограничения в доступе к определенным функциям, специфичным для платформы, а также зависимость от сторонних платформ и инструментов. Хотя кроссплатформенные технологии стремятся сократить разрыв между нативной и ненативной разработкой, все же могут быть случаи, когда

нативная разработка предпочтительна или необходима для достижения оптимальной производительности, гибкости или интеграции с экосистемой платформы. В конечном счете выбор между нативной и не нативной разработкой зависит от таких факторов, как требования проекта, бюджетные ограничения, ресурсы разработки и целевая аудитория.

Прогрессивные веб-приложения (PWA) – это веб-приложения, которые используют современные веб-технологии, чтобы обеспечить взаимодействие с пользователем, аналогичное нативным мобильным приложениям. PWA созданы для того, чтобы быть быстрыми, надежными и привлекательными, предлагая такие функции, как автономная поддержка, push-уведомления и установка на главный экран.

Одним из ключевых преимуществ PWA является их кроссплатформенная совместимость, поскольку они могут работать на любом устройстве с современным веб-браузером, включая настольные компьютеры, смартфоны и планшеты. Это устраняет необходимость отдельной разработки для разных платформ, сокращая время и стоимость разработки.

Производительность – еще один ключевой аспект PWA. Благодаря таким методам, как кэширование, отложенная загрузка и предварительная выборка, PWA обеспечивают более быструю загрузку и более плавную навигацию, способствуя улучшению пользовательского опыта. Кроме того, сервисы позволяют PWA работать в автономном режиме, кэшируя ресурсы и обеспечивая доступ к ранее посещенным страницам, что еще больше повышает вовлеченность пользователей [32].

Еще одним преимуществом PWA являются их возможности в автономном режиме, позволяющие пользователям получать доступ к контенту и функциям, даже когда они не в сети или имеют нестабильное подключение к Интернету. Это достигается за счет сервисов, которые кэшируют ресурсы и обеспечивают автономный доступ к ранее посещенным страницам.

PWA также можно установить непосредственно на устройство пользователя, при этом они будут выглядеть и функционировать как собственные приложения. Это улучшает видимость и вовлеченность, поскольку пользователи могут получить доступ к приложению со своего главного экрана без необходимости посещать магазин приложений. Кроме того, PWA поддерживают push-уведомления, что позволяет разработчикам повторно привлекать пользователей своевременными и актуальными обновлениями, даже если приложение не используется активно. Это помогает повысить удержание пользователей и увеличить количество повторных посещений.

Несмотря на свои преимущества, PWA также имеют некоторые ограничения. Например, у них может не быть доступа к определенным функциям и API-интерфейсам устройства, что ограничивает их функциональность по сравнению с нативными приложениями. Кроме того, поддержка PWA различается в разных браузерах и платформах, что может повлиять на единообразие взаимодействия с пользователем.

Таким образом, можно выделить «следующие ключевые особенности PWA: кроссплатформенная совместимость; улучшенная производительность; автономные возможности; установка домашнего экрана; всплывающее уведомление; ограниченный доступ к собственным функциям устройства и API; различная поддержка в разных браузерах и платформах» [14].

PWA используются для различных целей, включая электронную коммерцию, средства массовой информации, инструменты повышения производительности и социальные сети. Они особенно популярны среди компаний, стремящихся охватить более широкую аудиторию на разных устройствах и платформах без необходимости разработки отдельных собственных приложений. PWA предлагают привлекательную альтернативу традиционным разработкам нативных приложений, сочетая в себе охват и доступность Интернета с возможностями и пользовательским интерфейсом нативных приложений.

PWA стали важным компонентом для многих компаний, особенно тех, у которых нет собственных мобильных приложений. В первую очередь это связано с тем, что PWA предлагают простой способ расширить свое присутствие в Интернете на мобильных устройствах без необходимости разработки отдельных собственных приложений. Для компаний, которые в значительной степени полагаются на свои веб-сайты для взаимодействия с клиентами и предоставления своих услуг, PWA предоставляют эффективное решение. Используя те же веб-технологии, которые используются для создания своих веб-сайтов, компании могут легко преобразовать свои сайты в PWA, улучшая удобство использования и доступность на различных устройствах.

1.3.3 Развертывание мобильных приложений на облачных платформах

В рамках рассматриваемого исследования осуществляется анализ процессов развертывания мобильных приложений на облачных платформах. В первую очередь, производится выбор подходящей облачной платформы в соответствии с установленными требованиями. Среди популярных облачных провайдеров выделяются Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), а также другие.

После регистрации и конфигурации среды разработки в рамках выбранной облачной платформы производится загрузка и хранение ресурсов мобильного приложения. Эти ресурсы, такие как изображения, видео и другие файлы, хранятся в облачных хранилищах, таких как Amazon S3, Azure Blob Storage, Google Cloud Storage [52].

При необходимости развертывания серверной части приложения, например backend-сервера, осуществляется ее развертывание с использованием ресурсов облачной платформы. Это может включать в себя создание и развертывание серверных функций, виртуальных машин или контейнеров в соответствии с архитектурой приложения.

Дополнительно, настраивается база данных с использованием облачных баз данных, таких как Amazon RDS, Azure Database, Google Cloud SQL, с учетом требований и характеристик приложения. Производится управление конфигурациями приложения, а также принятие мер безопасности, таких как управление доступом и шифрование данных [23, 44].

После завершения развертывания осуществляется тестирование и мониторинг приложения в облаке. При необходимости проводятся обновления приложения с использованием предоставляемых облачным провайдером механизмов, а также масштабирование ресурсов в зависимости от динамически меняющихся требований. Также обеспечивается высокая доступность приложения с применением балансировки нагрузки и механизмов автомасштабирования, что является важным аспектом в контексте облачных вычислений.

В результате проведенного исследования, выполнены ключевые задачи по анализу предметной области, изучению современных облачных платформ, рассмотрению вопросов использования инструментов управления зависимостями при развертывании программных продуктов, а также освещены особенности мобильной разработки и развертывания на облачных платформах. Анализ предметной области позволил углубленно рассмотреть актуальные аспекты технологий кастомизации и развертывания программного продукта с использованием облачных технологий. Исследование современных облачных платформ дало представление о спектре возможностей и особенностях каждого провайдера.

Рассмотрение инструментов управления зависимостями при развертывании программных продуктов помогло выделить ключевые аспекты в обеспечении стабильности и эффективности процесса разработки. Описанные особенности мобильной разработки и развертывания на облачных платформах позволяют сделать вывод о том, что интеграция облачных технологий с мобильной разработкой является важным

направлением, обеспечивающим гибкость, масштабируемость и высокий уровень доступности для программных продуктов.

При выборе облачной платформы следует учитывать множество факторов, помимо инструментов и сервисов. Важными критериями являются интеграция с другими сервисами, гибкость и масштабируемость, безопасность и возможность совместной работы. Стоит отметить, что все рассматриваемые платформы предоставляют широкий спектр инструментов и сервисов для создания моделей машинного обучения. Таким образом, выполнение поставленных задач дало полную картину о взаимосвязи облачных технологий и процессов разработки программного обеспечения, а также выявило ключевые факторы успешного развертывания на облачных платформах.

2 Обоснование выбора средств разработки и формирование технического задания

2.1 Обоснование инструментов разработки мобильного приложения для ООО ЦЗИ «Гриф»

2.1.1 Нативные инструменты для создания мобильных приложений

Чтобы выбрать оптимальный инструмент для создания мобильного приложения, был проведён сопоставительный анализ инструментов разработки мобильных приложений.

Apache Cordova - это платформа для создания мобильных приложений, которая позволяет разрабатывать кроссплатформенные приложения, используя стандартные веб-технологии, такие как HTML, CSS и JavaScript. Фреймворк был создан в 2009 году под названием PhoneGap, а в 2011 году был перенесен в Apache Software Foundation и переименован в Apache Cordova.

История появления Apache Cordova связана с необходимостью создания мобильных приложений, которые могут работать на разных операционных системах, таких как iOS, Android и Windows. Ранее разработчики должны были создавать отдельные приложения для каждой платформы, что было дорогостоящим и затрудняло поддержку приложений. PhoneGap позволил создавать кроссплатформенные приложения, используя общий код для всех платформ [33, 49].

Основные особенности Apache Cordova включают в себя следующие пункты:

– Кроссплатформенность. Приложения, созданные с помощью Apache Cordova, могут работать на разных операционных системах, используя один и тот же код.

– Использование веб-технологий. Apache Cordova позволяет использовать знакомые веб-технологии, такие как HTML, CSS и JavaScript, что упрощает разработку мобильных приложений для веб-разработчиков.

– Доступ к нативным API. Apache Cordova предоставляет доступ к нативным API устройства, таким как камера, геолокация и контакты, через JavaScript.

– Поддержка множества плагинов. Apache Cordova имеет множество плагинов, которые расширяют его функциональность, такие как плагин для работы с сетью, плагин для работы с базой данных и многие другие [27].

Однако Apache Cordova также имеет свои недостатки. Приложения, созданные с помощью Apache Cordova, могут быть менее производительными, чем нативные приложения, поскольку они работают в веб-контейнере. Также хотя Apache Cordova предоставляет доступ к нативным API, он может не иметь доступа ко всем возможностям устройства, что может быть ограничивающим для некоторых приложений.

Для расширения функциональности Apache Cordova требуется использование сторонних плагинов, что может привести к зависимости от сторонних разработчиков и необходимости обновления плагинов при обновлении платформы. Для работы с Cordova необходимо устанавливать программные development kits (SDK) соответствующих мобильных платформ. Эта проблема решается с помощью сервиса PhoneGap Build, который представляет собой облачное решение. Он компилирует HTML5-код в готовые приложения, избавляя разработчиков от необходимости поддерживать нативные SDK.

Xamarin – среда разработки мобильных приложений, созданная Microsoft. Она кроссплатформенная и позволяет разработчикам создавать собственные мобильные приложения для iOS, Android и Windows, используя единый код на языке C #. Xamarin использует платформу .NET, предоставляя разработчикам знакомые инструменты и библиотеки для создания мобильных приложений [43]. Одним из ключевых преимуществ

Xamarin является его способность совместно использовать значительную часть кода на разных платформах, тем самым сокращая время и усилия на разработку. С помощью Xamarin разработчики могут один раз написать бизнес-логику, код доступа к данным и основные функции и повторно использовать их на нескольких платформах, сохраняя при этом отдельные компоненты пользовательского интерфейса (UI) для каждой платформы.

Еще одним преимуществом Xamarin является его производительность и удобство работы с пользователем. В отличие от некоторых кроссплатформенных платформ, которые полагаются на веб-технологии или промежуточные уровни, Xamarin компилирует код в собственные двоичные файлы, обеспечивая оптимальную производительность и полную интеграцию с API и функциями, специфичными для платформы. Это позволяет разработчикам создавать приложения, которые выглядят и работают как собственные приложения, обеспечивая единообразный и совершенный пользовательский интерфейс на разных платформах.

Xamarin состоит из нескольких компонентов, в том числе:

– Xamarin.Forms, набор инструментов пользовательского интерфейса, который позволяет разработчикам создавать общий код пользовательского интерфейса с помощью единого кроссплатформенного API. Xamarin.Forms упрощает процесс создания макетов и элементов управления пользовательского интерфейса, предоставляя набор общих элементов пользовательского интерфейса, которые можно настроить в соответствии с собственным внешним видом каждой платформы.

– Xamarin.iOS, набор инструментов и библиотек для создания приложений iOS с использованием C# и платформы .NET. Xamarin.iOS предоставляет разработчикам возможность применять привычные инструменты разработки, такие как Visual Studio или Visual Studio для Mac, для создания, тестирования и отладки приложений iOS, а так же использовать специфические API-интерфейсы iOS.

– Xamarin.Android. предоставляет инструменты и библиотеки для создания приложений Android с использованием C# и платформы .NET. Разработчики могут получить доступ к собственным API-интерфейсам Android и использовать такие функции, как макеты XML, намерения и действия, для создания приложений Android, которые легко интегрируются с платформой.

Средство предварительного просмотра Xamarin.Forms – это визуальный инструмент, интегрированный в Visual Studio, «который позволяет разработчикам просматривать макеты пользовательского интерфейса Xamarin.Forms в режиме реального времени, что упрощает проектирование и настройку элементов пользовательского интерфейса для различных устройств и размеров экрана» [17]. «Xamarin предлагает мощное и гибкое решение для разработки мобильных приложений для любых платформ, позволяющее разработчикам использовать имеющиеся у них навыки и инструменты для создания высококачественных собственных мобильных приложений для платформ iOS, Android и Windows» [19].

Flutter – это комплект разработки программного обеспечения пользовательского интерфейса (SDK) с открытым исходным кодом, разработанный Google. Он позволяет разработчикам создавать скомпилированные в собственном коде приложения для мобильных устройств, Интернета и настольных компьютеров из единой базы кода. Flutter использует язык программирования Dart и предоставляет богатый набор настраиваемых виджетов и инструментов для создания современных, адаптивных пользовательских интерфейсов. «Ключевые особенности и преимущества Flutter включают единую базу кода, быструю разработку с горячей перезагрузкой, богатые компоненты пользовательского интерфейса, высокую производительность, а также активное сообщество и экосистему» [3].

Ionic, с другой стороны, представляет собой платформу с открытым исходным кодом для создания кроссплатформенных мобильных, веб- и

настольных приложений с использованием таких веб-технологий, как HTML, CSS и JavaScript. Он построен на основе Angular и предоставляет набор компонентов пользовательского интерфейса, инструментов и сервисов для создания современных и интерактивных приложений. Ключевые особенности и преимущества Ionic включают использование веб-технологий, независимость от платформ, богатые компоненты пользовательского интерфейса, интеграцию с Angular, а также большое сообщество и экосистему плагинов.

React Native - это открытая платформа, созданная Facebook, для разработки кроссплатформенных мобильных приложений с использованием JavaScript и React, декларативной библиотеки пользовательского интерфейса. Этот инструмент позволяет писать код один раз и запускать его на разных платформах, включая iOS, Android и веб-приложения. Ключевые особенности и преимущества React Native включают JavaScript и React, собственную производительность, горячую перезагрузку, большое сообщество и экосистему, а также компоненты, специфичные для платформы.

На основании представленной информации были проведены оценки фреймворков по определенным критериям, используя 10-балльную шкалу. В таблице 4 представлен краткий обзор сравнительного анализа фреймворков.

Таблица 4 – Сравнительный анализ платформ

Показатель	**ApacheCordova**	**ReactNative**	**Xamarin**	**Ionic**	**Flutter**
Поддержка Android	9	9	10	9	10
Поддержка iOS	9	9	9	9	5
Легкость обучения	10	7	8	8	6
Скорость работы	6	6	8	7	7
Поддержка сообществом	7	9	9	7	9

На основе проведенного анализа можно заключить, что для каждой конкретной задачи, в зависимости от поставленных требований, необходимо выбирать соответствующую технологию [39].

Xamarin, Flutter, Ionic и React Native – это мощные платформы для создания кроссплатформенных мобильных приложений. Выбор между ними зависит от таких факторов, как навыки разработки, требования проекта, соображения производительности и предпочтения экосистемы. У каждого фреймворка есть свои сильные и слабые стороны, поэтому разработчикам следует тщательно оценить варианты и выбрать тот, который лучше всего соответствует их потребностям.

React Native хорошо подходит для опытных JavaScript-разработчиков и команд, но требует хорошего знания iOS Objective C API и Android Java API. Создаваемые приложения выглядят и работают как нативные, однако следует учитывать молодость данного фреймворка. Доля общего кода может достигать 90%. Во всех случаях, когда требуется более глубокий контроль и прямой доступ к аппаратным ресурсам, предпочтительнее использовать нативные инструменты разработки.

2.1.2 No-code решения для разработки мобильных приложений

FlutterFlow – это визуальный конструктор приложений для Flutter, популярного пакета разработки программного обеспечения (SDK) с открытым исходным кодом, разработанного Google. Он позволяет пользователям создавать приложения Flutter с использованием визуального интерфейса без написания кода вручную. «FlutterFlow упрощает процесс создания мобильных и веб-приложений, предоставляя интерфейс перетаскивания для разработки макетов пользовательского интерфейса и добавления интерактивных компонентов» [27]. Интерфейс главной страницы FlutterFlow показана на рисунке 6.

FlutterFlow появился как ответ на растущий спрос на инструменты, упрощающие процесс разработки приложений Flutter, особенно для разработчиков и дизайнеров, которые могут не иметь большого опыта

программирования. Платформа была создана командой разработчиков, которые стремились демократизировать разработку приложений, сделав ее более доступной для более широкой аудитории.

Основная цель FlutterFlow — ускорить разработку приложений Flutter за счет быстрого создания прототипов, итераций и совместной работы. С помощью FlutterFlow пользователи могут быстро создавать макеты пользовательского интерфейса, добавлять виджеты и компоненты, определять потоки навигации и просматривать свое приложение в режиме реального времени. Этот визуальный подход к разработке приложений помогает сократить время разработки и устраняет необходимость в написании повторяющегося кода, позволяя разработчикам сосредоточиться на создании удобного пользовательского интерфейса.

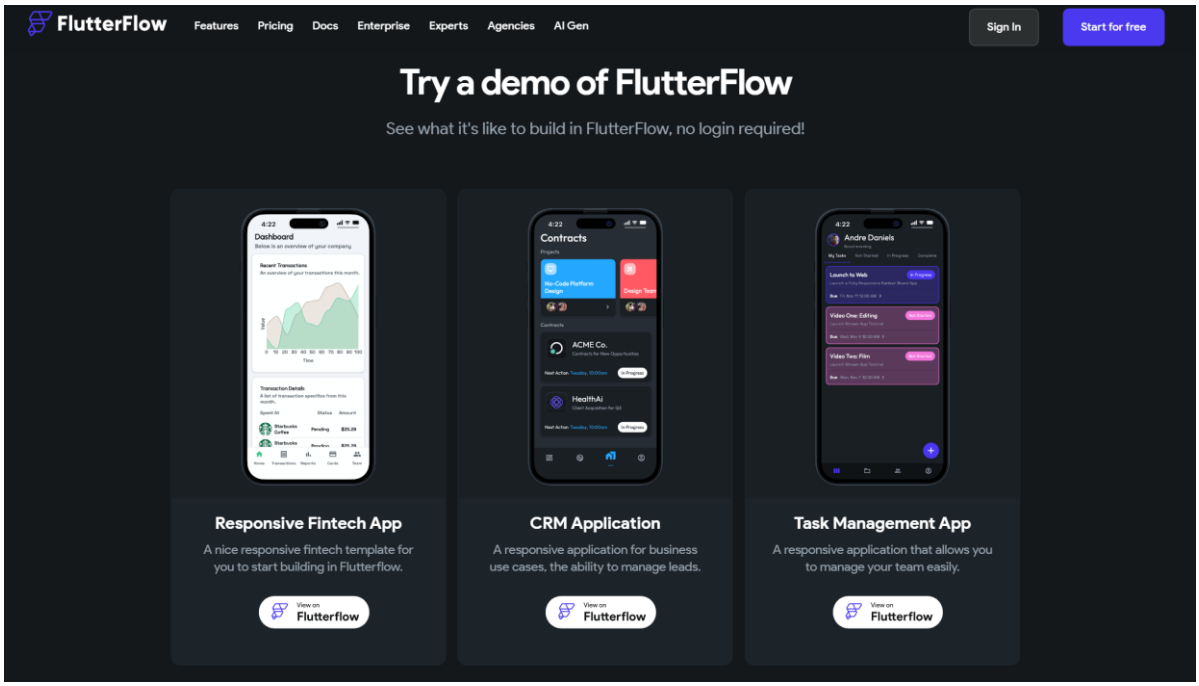

Рисунок 6 – Интерфейс главной страницы FlutterFlow

Одной из ключевых особенностей FlutterFlow является интуитивно понятный интерфейс перетаскивания, который позволяет пользователям создавать макеты пользовательского интерфейса, просто перетаскивая

виджеты на холст и располагая их по желанию. Платформа предоставляет библиотеку готовых компонентов пользовательского интерфейса, включая кнопки, текстовые поля, изображения и элементы навигации, что позволяет легко создавать сложные макеты без написания кода. FlutterFlow также предлагает бесшовную интеграцию с редактором кода Flutter, позволяя пользователям легко переключаться между визуальным редактором и редактором кода. Пользователи могут настраивать сгенерированный код, добавлять бизнес-логику и расширять функциональность своего приложения, используя богатый набор API и библиотек Flutter. Кроме того, FlutterFlow поддерживает функции совместной работы, позволяя нескольким членам команды одновременно работать над одним проектом. Пользователи могут делиться прототипами, собирать отзывы и работать над дизайном в режиме реального времени, что способствует более эффективному и совместному процессу разработки приложений [9].

FlutterFlow – это мощный инструмент для визуального создания приложений Flutter, позволяющий разработчикам и дизайнерам быстро и легко создавать высококачественные кроссплатформенные приложения. Его удобный интерфейс, интеграция с экосистемой Flutter и функции совместной работы делают его ценным активом для команд, стремящихся ускорить рабочий процесс разработки приложений Flutter.

Adalo – это платформа, которая позволяет пользователям создавать мобильные и веб-приложения без написания кода. Он предоставляет визуальный интерфейс для проектирования и создания приложений с использованием компонентов перетаскивания, что делает разработку приложений доступной для пользователей без навыков программирования.

Платформа работает на основе подхода к разработке без кода или с минимальным использованием кода, что означает, что пользователи могут создавать функциональные приложения, просто размещая предварительно созданные компоненты и настраивая их свойства. Adalo предлагает широкий спектр настраиваемых компонентов, включая элементы пользовательского

интерфейса, такие как кнопки, текстовые поля, списки и меню навигации, а также более продвинутые функции, такие как аутентификация пользователей, базы данных и интеграция со сторонними организациями.

Особенности работы с Adalo следующие:

– Визуальный интерфейс, пользователи начинают с входа на платформу Adalo и доступа к визуальному редактору. Редактор предоставляет холст, на котором пользователи могут создавать макет пользовательского интерфейса своего приложения, перетаскивая компоненты на холст и располагая их по своему усмотрению.

– Настройка компонентов, пользователи могут настраивать внешний вид и поведение каждого компонента, регулируя его свойства и настройки. Например, они могут изменить текст, цвет, размер и стиль кнопки или настроить источник данных для компонента списка.

– Управление данными, Adalo включает встроенные функции базы данных, позволяющие пользователям определять структуры данных, создавать коллекции и управлять данными в своих приложениях. Пользователи могут подключать компоненты пользовательского интерфейса к источникам данных и отображать динамический контент на основе пользовательского ввода или запросов к базе данных.

– Логика и рабочий процесс, пользователи могут определять логику и рабочий процесс в своих приложениях, используя визуальные действия и условия Adalo. Например, «они могут настроить навигацию между экранами, инициировать действия в ответ на взаимодействие с пользователем или реализовать правила аутентификации и авторизации» [18].

– Предварительный просмотр и тестирование. После того как приложение разработано и настроено, пользователи могут просмотреть его на платформе Adalo, чтобы увидеть, как оно выглядит и работает на разных устройствах. Они также могут протестировать поведение и удобство использования приложения, чтобы выявить любые проблемы или улучшения.

Если приложение удовлетворено, пользователи могут опубликовать его в Интернете или магазинах приложений непосредственно с платформы Adalo. Adalo предоставляет варианты развертывания как веб-приложений, так и собственных мобильных приложений для iOS и Android, что позволяет пользователям охватить свою целевую аудиторию на различных платформах.

Adalo упрощает процесс разработки приложений, предоставляя удобный интерфейс, предварительно созданные компоненты и встроенные функции для управления данными, логики и развертывания. Он дает возможность частным лицам и предприятиям быстро и эффективно создавать собственные приложения без необходимости обширных знаний или опыта в области кодирования.

Таблица 5 содержит анализ ранее изученных инструментов. Как видно из таблицы, «каждый инструмент обладает своими достоинствами и недостатками. React Native и Kotlin являются более сложными в использовании, но они также обладают большим потенциалом и мощными инструментами разработки. FlutterFlow и Adalo, с другой стороны, предлагают более простой интерфейс для создания приложений без необходимости написания кода, хотя они могут быть ограничены в возможностях и по функциональности» [11].

Таблица 5 – Сравнительный анализ инструментов разработки

Критерий	**FlutterFlow**	**Adalo**
Язык программирования	Dart	Не требуется
Поддерживаемые платформы	Android, iOS, веб	Android, iOS
Стоимость	Бесплатно (с ограничениями) / Платно	Бесплатно (с ограничениями) / Платно
Уровень сложности	Низкий	Низкий
Удобство использования	Простой	Простой
Качество графики и дизайна	Отличное	Хорошее
Наличие библиотек и плагинов	Ограничено	Множество
Сообщество разработчиков	Небольшое	Большое

В конечном счете «выбор инструмента для разработки мобильных приложений зависит от конкретных потребностей и задач, которые необходимо решить. Если требуется создать более сложное приложение, то React Native или Kotlin могут быть более подходящими вариантами. Если же нужно быстро создать простое приложение, то можно выбрать FlutterFlow или Adalo» [7].

2.2 Особенности использования облачной платформы Glide для разработки программного продукта ООО ЦЗИ «Гриф»

Glide – это облачная платформа, позволяющая разрабатывать мобильные приложения.

Glide предоставляет разработчикам широкий спектр компонентов и утилит для оптимизации процесса разработки мобильных приложений. Он предлагает решения для таких задач, как создание пользовательских интерфейсов, обработка навигации, управление данными и реализация анимации. Главная страница сервиса показана на рисунке 7.

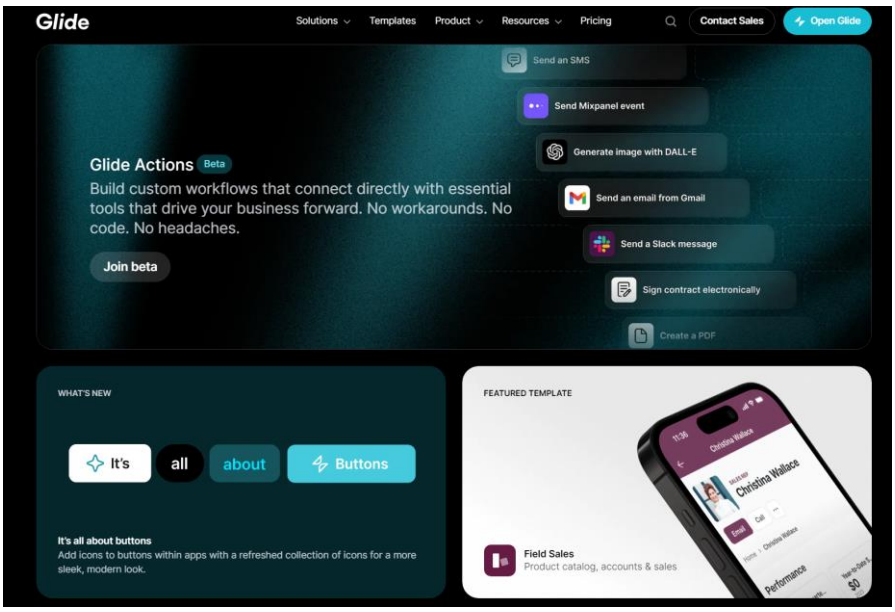

Рисунок 7 – Главная страница GlideApps

Одним из главных преимуществ Glide является его простота и удобство использования. Он поставляется с простым API, который позволяет разработчикам быстро создавать сложные и интерактивные пользовательские интерфейсы без написания лишнего кода. Интуитивный дизайн Glide позволяет разработчикам больше сосредоточиться на создании функциональности приложения, а не увязнуть в деталях реализации.

Glide предлагает широкие возможности настройки, позволяющие разработчикам адаптировать внешний вид и поведение своих приложений в соответствии с конкретными требованиями дизайна. Он предоставляет множество готовых компонентов пользовательского интерфейса, таких как кнопки, текстовые поля, списки и панели навигации, которые можно легко настроить и интегрировать в макет приложения.

Glide также упрощает процесс навигации в мобильных приложениях. Он предлагает компоненты и шаблоны навигации, которые облегчают плавный переход между различными экранами и представлениями, улучшая общий пользовательский опыт. Glide включает функции для эффективного управления данными приложений. Он предоставляет инструменты для кэширования данных, локального хранения и синхронизации с удаленными серверами, гарантируя, что приложения смогут быстро и надежно получать доступ к данным и манипулировать ими.

Еще одним примечательным аспектом Glide является поддержка анимации и визуальных эффектов. Он предлагает встроенные библиотеки анимации и инструменты для создания плавных переходов, эффектов и взаимодействий, добавляя мобильным приложениям изящный и динамичный вид.

Glide – это универсальный и мощный набор инструментов для разработки мобильных приложений, предлагающий ряд функций и утилит, упрощающих процесс разработки и улучшающих взаимодействие с пользователем. Его простота, гибкость и широкие возможности настройки

делают его популярным выбором среди разработчиков для создания высококачественных мобильных приложений для платформ Android и iOS.

Помимо простоты и возможностей настройки, Glide выделяется своей производительностью и эффективностью. Он оптимизирован для обеспечения плавного и быстрого реагирования приложений даже на устройствах с ограниченными ресурсами. Эффективное управление памятью и методы рендеринга Glide способствуют ускорению загрузки и снижению расхода заряда батареи, что является решающим фактором при разработке мобильных приложений. Более того, у Glide есть активное и активное сообщество разработчиков, которые способствуют его постоянному совершенствованию и расширению. Сообщество предоставляет ресурсы, документацию и форумы поддержки, где разработчики могут обмениваться идеями, обращаться за помощью и делиться передовым опытом. Такая среда совместной работы способствует инновациям и позволяет разработчикам использовать опыт друг друга для более быстрого создания более качественных приложений.

Glide предлагает комплексное решение для разработки мобильных приложений, позволяющее разработчикам создавать многофункциональные, визуально привлекательные и высокопроизводительные приложения для платформ Android и iOS. Сочетание простоты, гибкости, производительности и поддержки сообщества делает его ценным инструментом для создания передового мобильного опыта.

В данном разделе большое внимание уделено выбору инструментов разработки мобильного приложения, что имеет важное значение для обеспечения эффективности и гибкости процесса разработки. Этот этап работы включал в себя анализ требований проекта, оценку возможностей различных инструментов и формирование оптимального стека технологий.

Сформулированное техническое задание стало ключевым документом, определяющим параметры и характеристики программного продукта, а также критерии успешного его развертывания на облачной платформе. Этот этап

работы направлен на обеспечение четкости и понимания всех этапов процесса разработки, что в свою очередь способствует более эффективной координации усилий команды разработчиков.

Таким образом, результаты научно-исследовательской работы предоставляют обширный обзор и обоснование стратегических решений, необходимых для успешной разработки программного продукта, а также создают основу для последующих этапов проекта, включая разработку, тестирование и развертывание.

3 Разработка мобильного приложения с помощью облачных технологий и формирование алгоритма по развертыванию и кастомизации программных продуктов

3.1 Проектирование элементов модели

3.1.1 Концептуальная модель

Концептуальная модель позволяет создать абстрактное представление о системе, явлении или процессе на основе концепций, идей, исследований и предположений. Она служит основой для дальнейшего проектирования, анализа и понимания объекта исследования. На рисунке 8 показана контекстная диаграмма для модели КАК ДОЛЖНО БЫТЬ. На рисунке 9 показана декомпозиция этой модели.

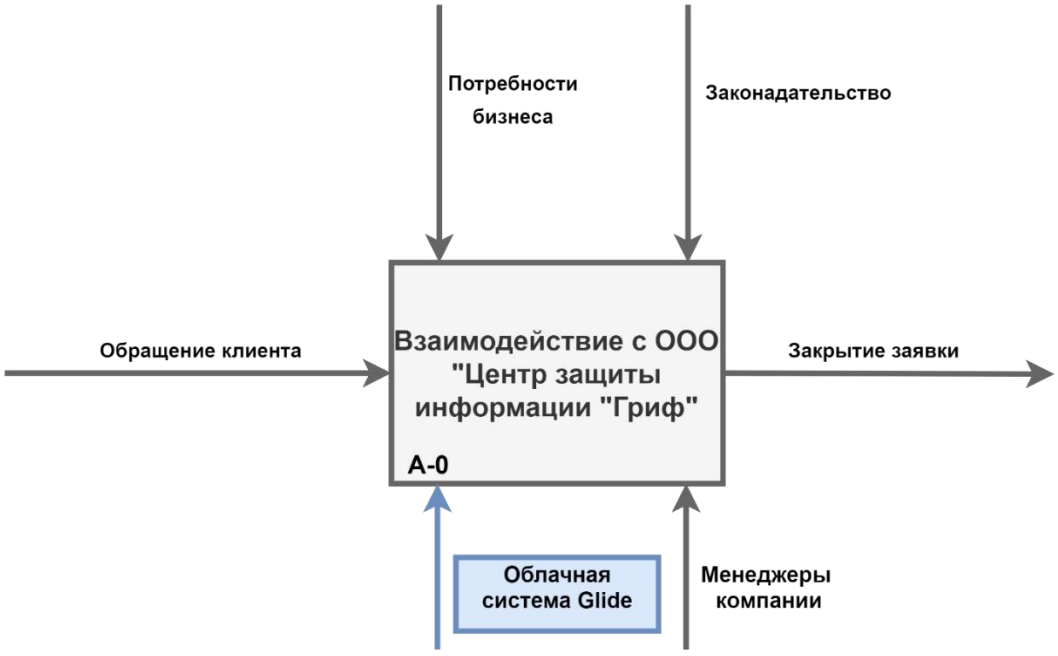

Рисунок 8 – Диаграмма верхнего уровня (A-0) для процесса КАК ДОЛЖНО БЫТЬ

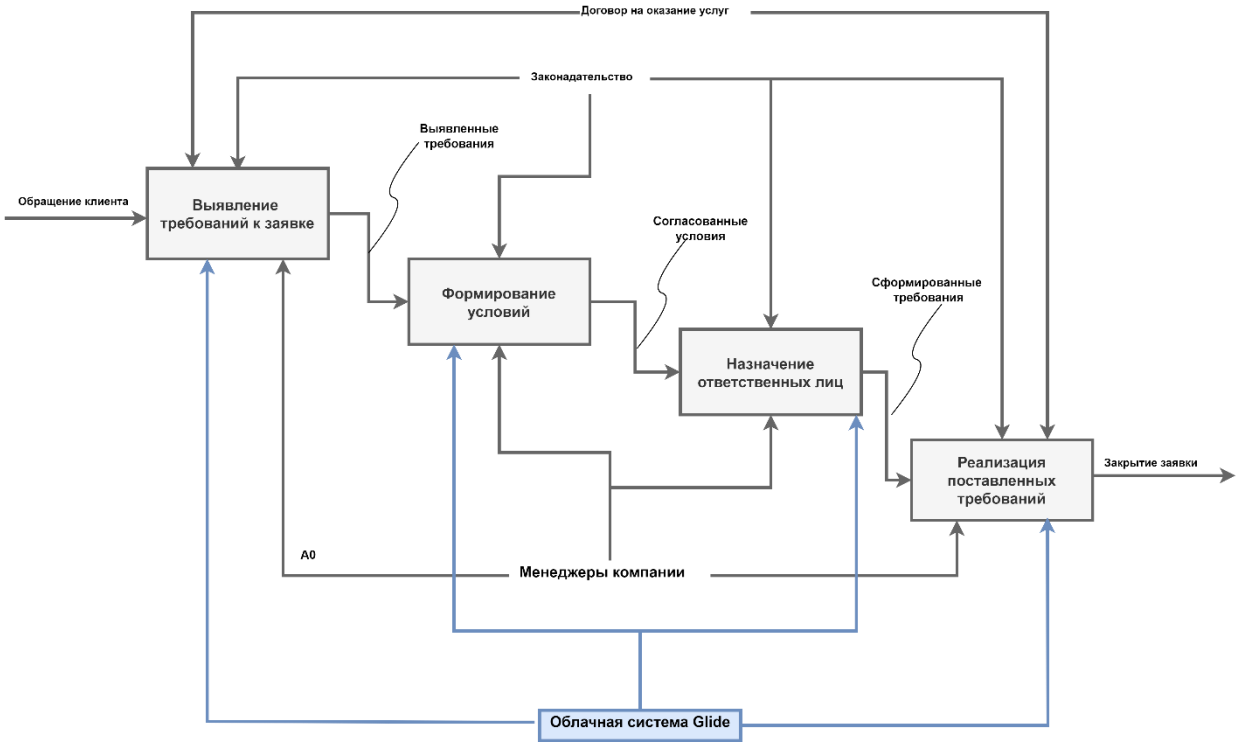

Рисунок 9 – Диаграмма декомпозиции A0 для процесса КАК ДОЛЖНО БЫТЬ

Как видно из диаграммы внедрение облачной системы для управления процессом обслуживания позволит улучшить процессы внутри предприятия. Так как ранее на предприятии использовали текстовые и табличные редакторы для обработки заявок. После внедрения облачной системы Glide этот процесс будет автоматизирован.

3.1.2 Диаграммы прецедентов

Для успешной кастомизации и развертывания программного продукта для ООО ЦЗИ «Гриф» с использованием облачных технологий необходимо предварительно провести проектирование элементов модели. Диаграмма прецедентов для разрабатываемого программного продукта показана на рисунке 10.

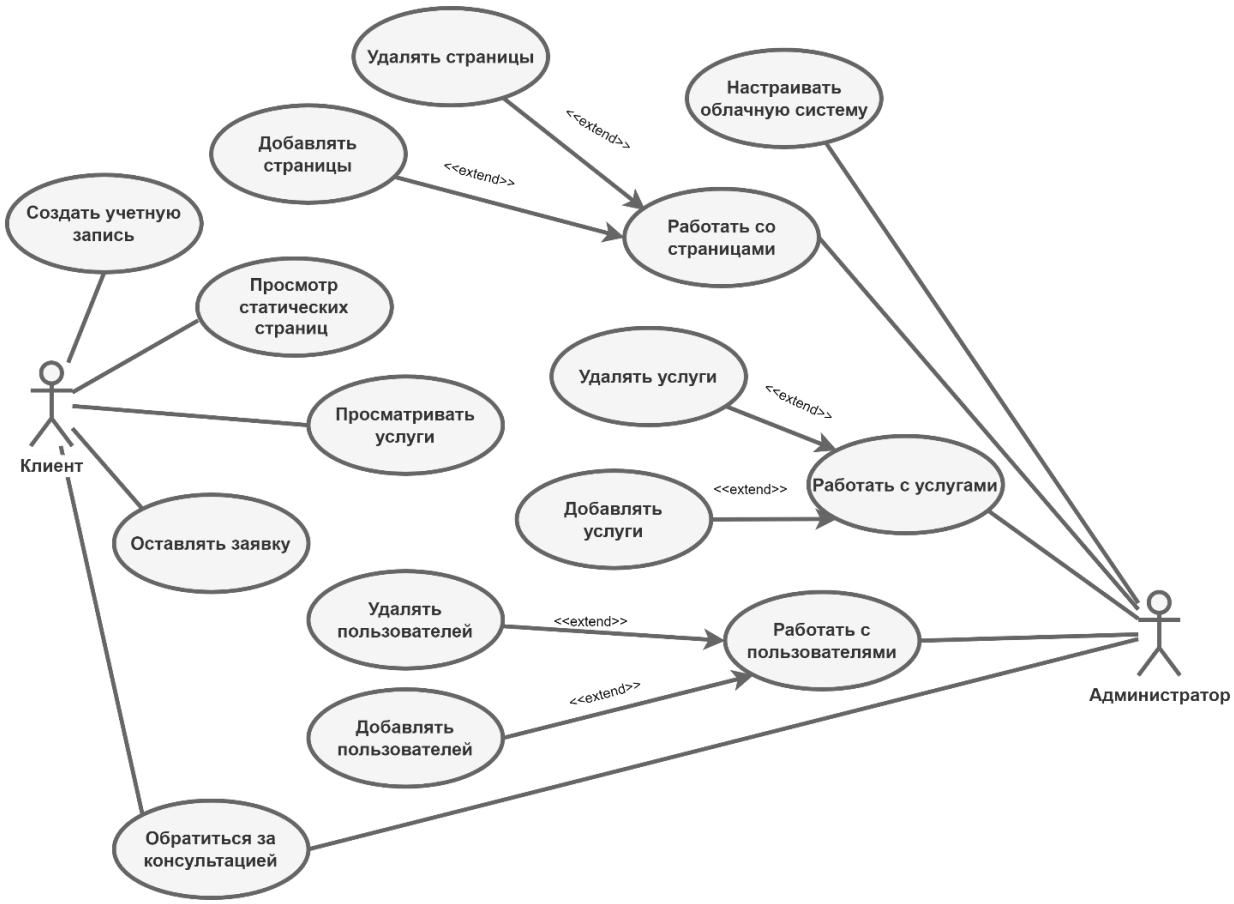

Рисунок 10 – Диаграмма прецедентов

Пользователи системы смогут получать информацию касательно деятельности ООО ЦЗИ «Гриф», создавать учетные записи, обращаться за консультацией, оставлять заявки. Администраторы системы смогут обновлять контент программного продукта, обрабатывать заявки клиентов, настраивать облачную систему.

3.1.3 Дерево функций и алгоритм работы приложения

Сценарий диалога показан на рисунке 11. Основной сценарий работы с программным продуктом заключается в нескольких шагах, показанных на рисунке. Пользователь системы ищет информацию касательно деятельности, оставляет заявку на получение услуги. Администраторы системы в свою очередь могут определять ответственное лицо для выполнения полученной заявки. По итогу результаты заявки отправляются пользователю. Дерево функций показано на рисунке 12.

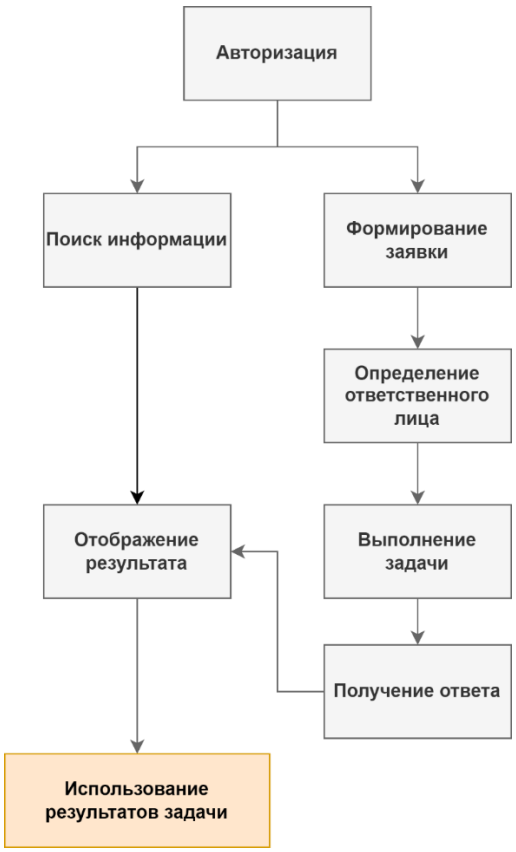

Рисунок 11 – Сценарий диалога

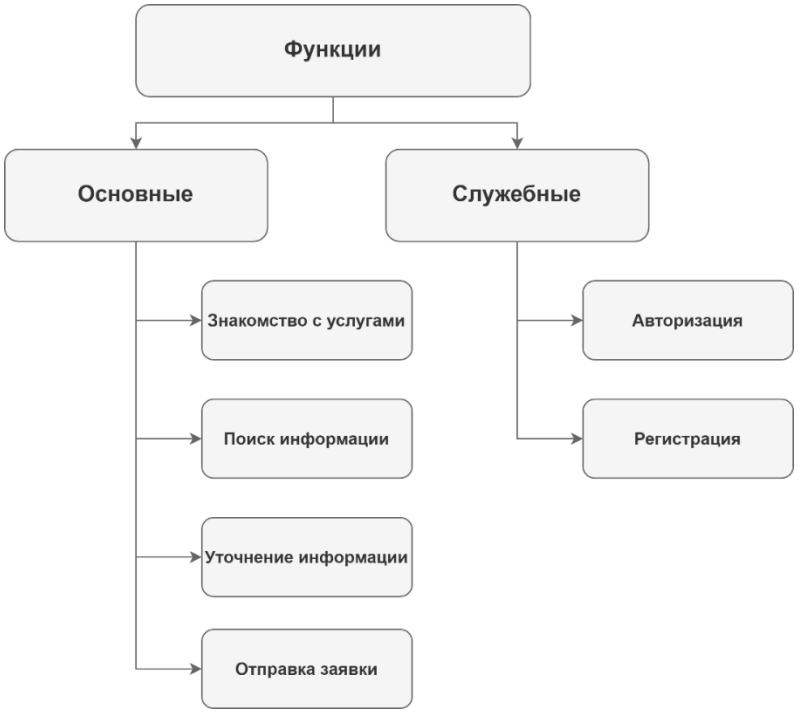

Рисунок 12 – Дерево функций

Основными функциями системы являются базовые функции, необходимые для правильной работы согласно техническому требованию на разработку программного продукта для ООО ЦЗИ «Гриф». Служебные требования включают в себя авторизацию и регистрацию.

3.1.4 Структура приложения

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это средство документирования проекта с точки зрения сценариев использования [3]. Диаграмма последовательности показана на рисунке 13.

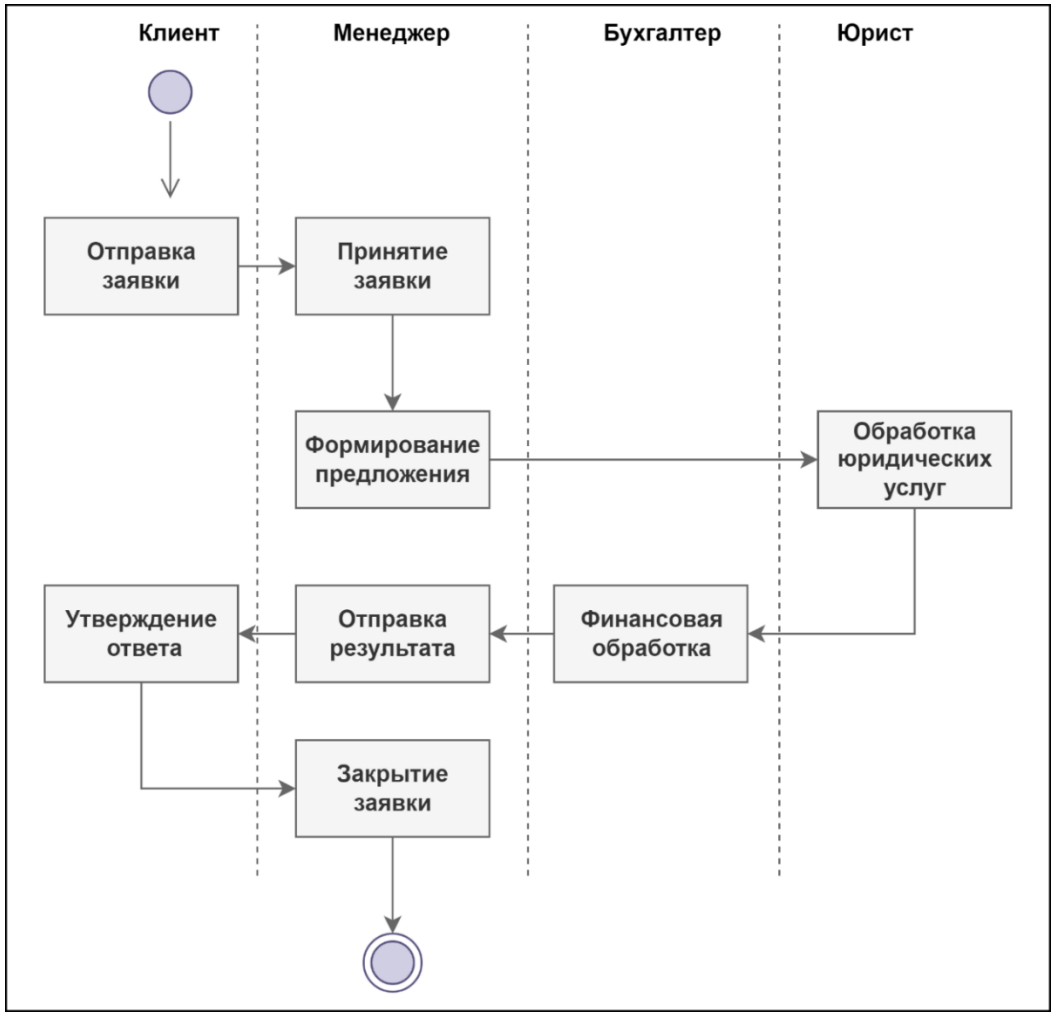

Рисунок 13 – Диаграмма последовательности

Диаграмма кооперации предназначена для описания поведения системы на уровне отдельных объектов, которые обмениваются между собой сообщениями, чтобы достичь нужной цели или реализовать некоторый вариант использования. Такое представление структуры модели как совокупности взаимодействующих объектов и обеспечивает диаграмма кооперации. Диаграмма кооперации показана на рисунке 14.

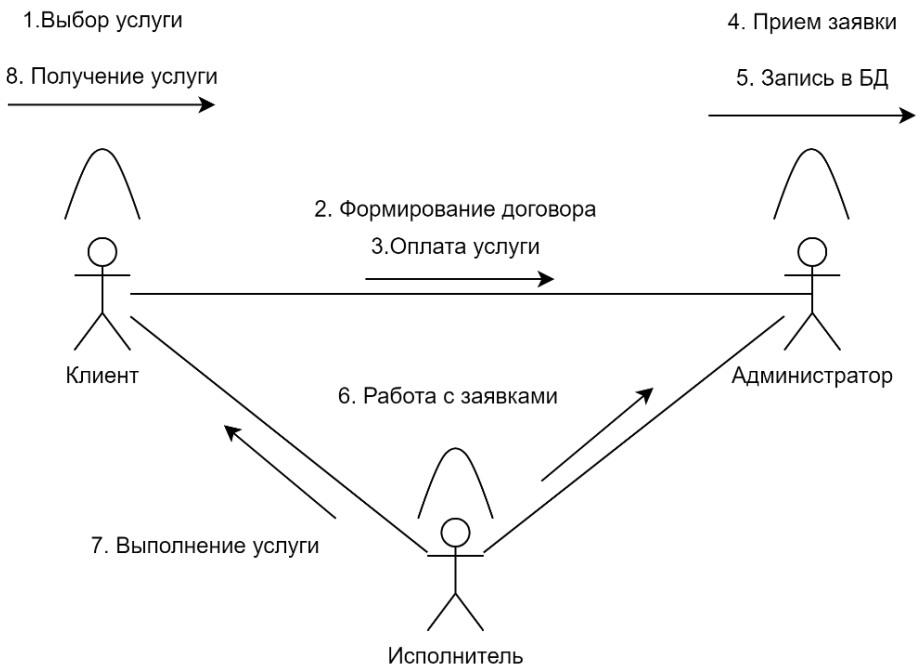

Рисунок 14 – Диаграмма кооперации, отображающая процесс работы после внедрения облачной системы

Такое представление структуры модели как совокупности взаимодействующих объектов и обеспечивает диаграмма кооперации.

3.2 Развертывание облачной платформы для ООО ЦЗИ «Гриф» посредством технологии Glide

Перед началом разработки потребуется зарегистрироваться в Glide и создать проект (рисунок 15).

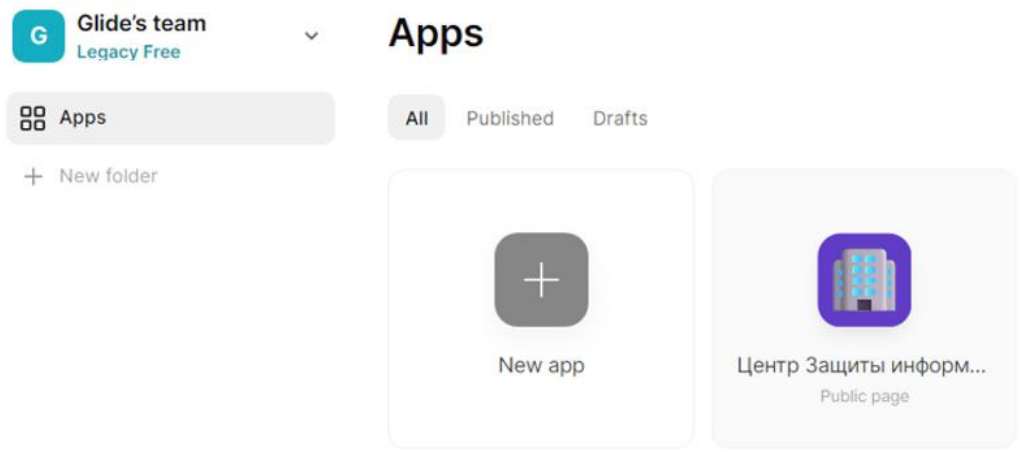

Рисунок 15 – Создание проекта

Создание приложения происходит в рамках рабочей зоны, которая включает несколько фреймов, предоставляющих возможность добавления необходимого контента и функционала в приложение (рисунок 16).

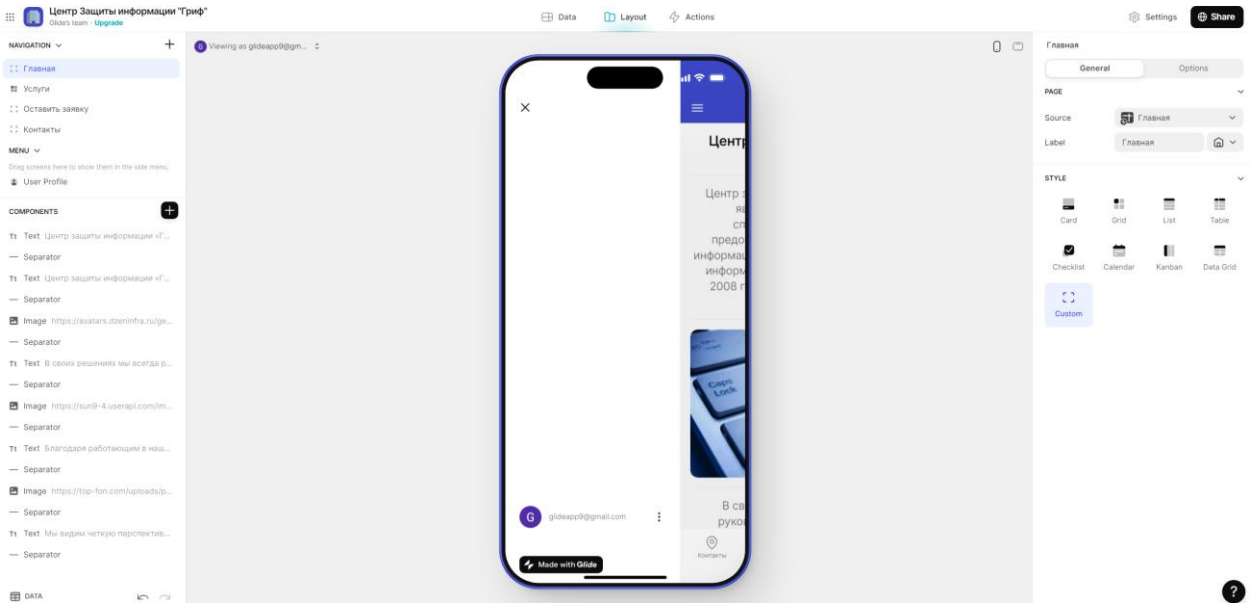

Рисунок 16 – Рабочая зона

Работа над приложением начинается с создания базы данных, которую представляют собой Google Таблицы. Они служат хранилищем для всего содержимого приложения. Сначала создается таблица Google, затем

определяются ее поля, после чего они подключаются к соответствующим страницам приложения. Рабочая область GlideApps содержит несколько ключевых элементов. Конструктор приложений является основным окном, где можно создавать и настраивать мобильные приложения. Он состоит из нескольких вкладок, включая «Дизайн», «Данные», «Компоненты» и «Действия» (рисунок 17).

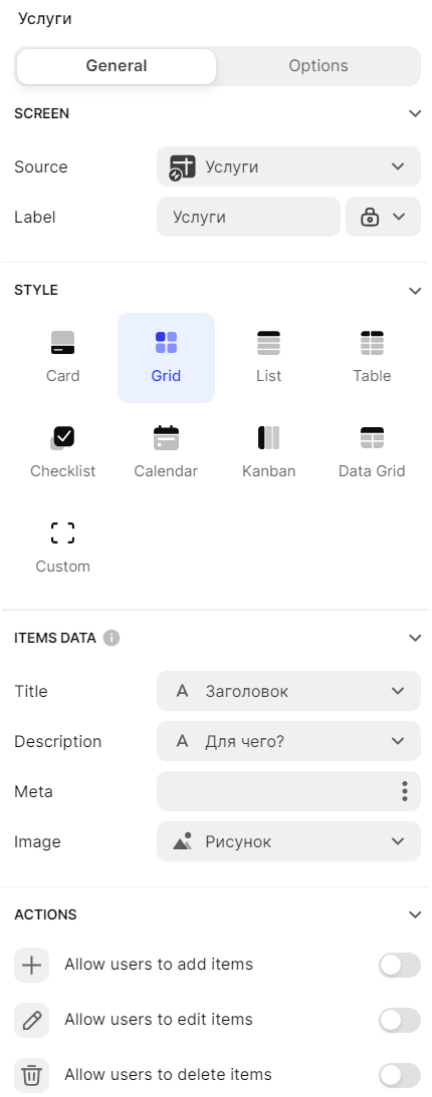

Рисунок 17 – Раздел с функциональными блоками

Рабочее пространство Glide состоит из нескольких основных компонентов, которые обеспечивают удобную и эффективную среду для разработки мобильных приложений:

– Визуальный редактор – это основной инструмент для создания пользовательского интерфейса приложения. В визуальном редакторе пользователи могут перетаскивать на экран мобильного устройства графические элементы, такие как кнопки, текстовые поля, изображения, списки и другие компоненты. Этот редактор обычно предоставляет макеты для экранов разных размеров и ориентаций устройств, что позволяет создавать адаптивные интерфейсы.

– Компоненты и элементы управления. Рабочая среда Glide предлагает богатую библиотеку готовых компонентов и элементов управления, которые можно легко добавить на экран приложения. Эти компоненты включают кнопки, текстовые поля, флажки, переключатели, изображения, списки, табличные представления и многое другое. Пользователи могут настроить внешний вид и поведение каждого компонента в соответствии с требованиями проекта.

– Конфигурация свойств и событий. Каждый компонент и элемент управления в рабочей области Glide имеет набор свойств и событий, которые можно настроить для определения его поведения. Например, пользователи могут установить текст кнопки, цвет фона текстового поля или определить действие, которое будет выполняться при нажатии на изображение. Это позволяет создавать интерактивные и адаптивные приложения с разнообразным функционалом.

– Панели инструментов и боковые панели. Рабочее пространство обычно включает панели инструментов и боковые панели, предоставляющие дополнительные функции и возможности. Панели инструментов могут содержать инструменты для форматирования текста, управления макетом, вставки изображений и других операций. Боковые панели могут содержать инструменты для управления проектами, настройки свойств элементов, предварительного просмотра приложений и т. д.

– Редактор кода. Для более опытных пользователей и разработчиков рабочая область Glide также включает редактор кода. Он позволяет

напрямую редактировать исходный код приложения, добавлять собственные функции, реализовывать сложную логику и взаимодействия, а также интегрировать сторонние библиотеки и инструменты.

Вместе эти компоненты образуют удобную и интуитивно понятную среду разработки в Glide, помогая пользователям создавать высококачественные и привлекательные мобильные приложения для различных платформ.

В Glide навигация между экранами организуется через список или меню, содержащее ссылки или кнопки для перемещения между различными экранами приложения. Было разработано четыре основных экрана приложения (рисунок 18).

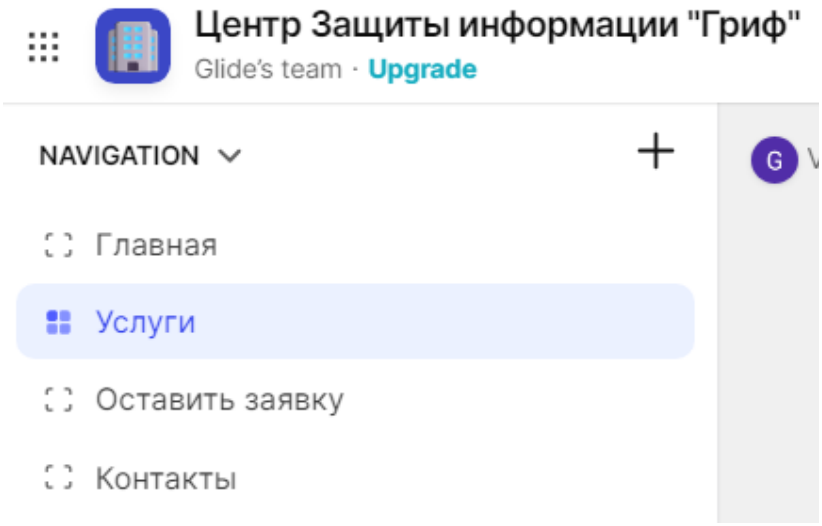

Рисунок 18 –Навигация по экранам

Некоторые элементы приложения могут быть изменены напрямую в самом приложении, без необходимости редактирования таблиц. Например, небольшие текстовые фрагменты можно ввести непосредственно через пользовательский интерфейс программы (рисунок 19).

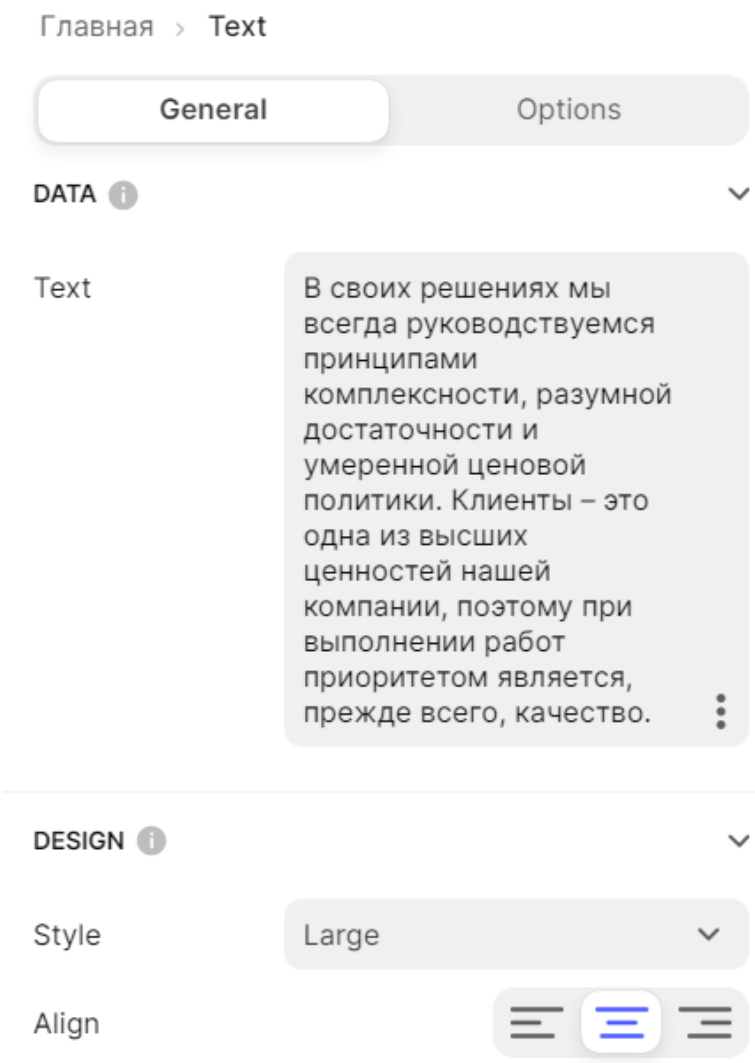

Рисунок 19 – Текстовый блок

После того как весь контент приложения был сформирован, требовалось настроить иконку и информацию о публикации. После этого приложение было готово к использованию (рисунок 20).

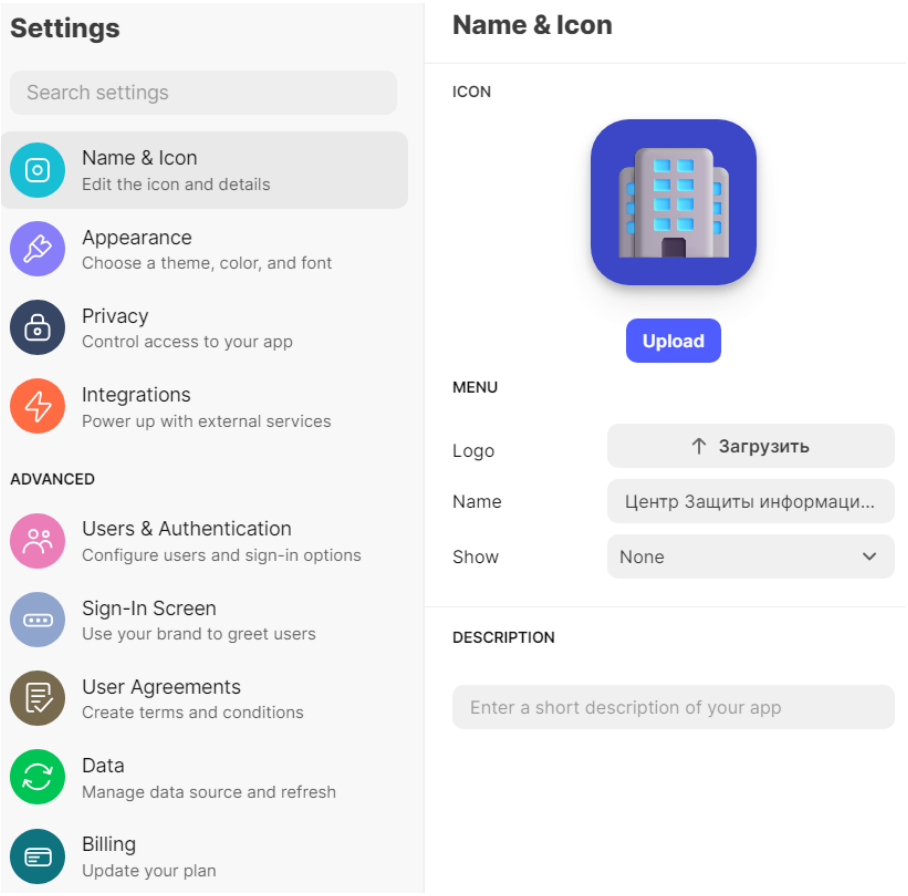

Рисунок 20 – Настройки публикации

После завершения процесса разработки приложение готово к публикации и интеграции в рабочие процессы организации.

3.3 Результаты разработки мобильного приложения с использование облачных технологий

Созданное приложение включает в себя четыре раздела, представленных вкладками. Первый раздел, называемый «Главная», представляет собой стартовую страницу, которая содержит основную информацию о компании и изображения (рисунок 21).

Раздел «Услуги» содержит разделы с описанием каждой услуги, которые предоставляются Центром защиты информации «Гриф» (рисунок 22).

Рисунок 21 – Главная страница

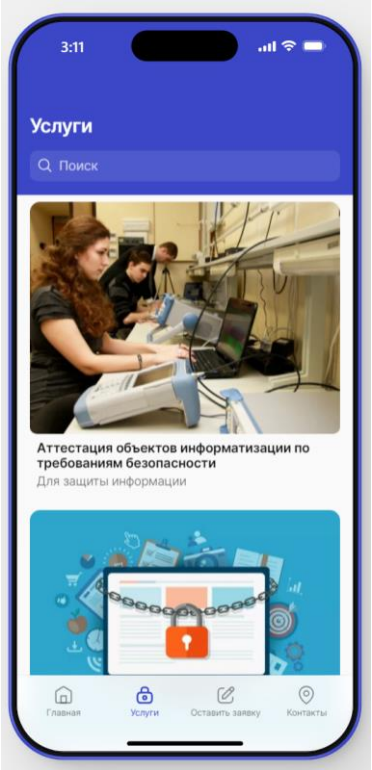

Рисунок 22 – Раздел «Услуги»

Второй раздел приложения представлен формой для приема заявок (рисунок 23). Эта форма становится доступной после нажатия на кнопку «Оставить заявку» (рисунок 24).

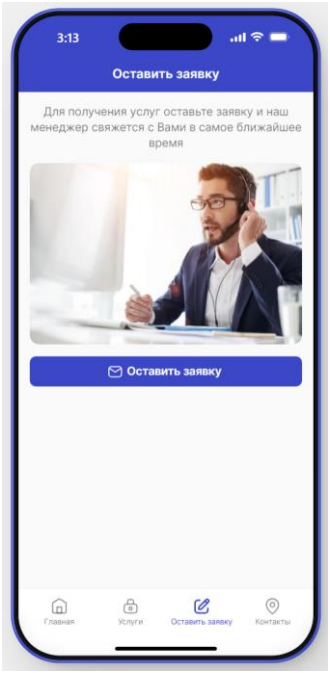

Рисунок 23 – Форма приёма заявок

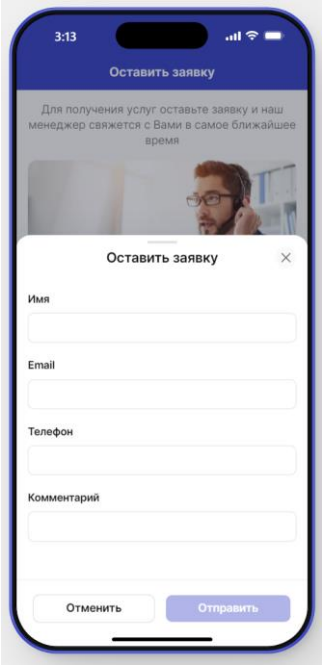

Рисунок 24 – Форма подачи заявки

При нажатии на кнопку открывается форма, которую пользователь должен заполнить, указав необходимые данные. В этой форме есть обязательные поля, которые нельзя оставить пустыми.

В указанной форме, в графе «комментарий», пользователь вводит всю дополнительную информацию.

Последний раздел приложения предоставляет пользователю контактную информацию, включая местоположение предприятия, что помогает пользователям легко его найти. (рисунок 25).

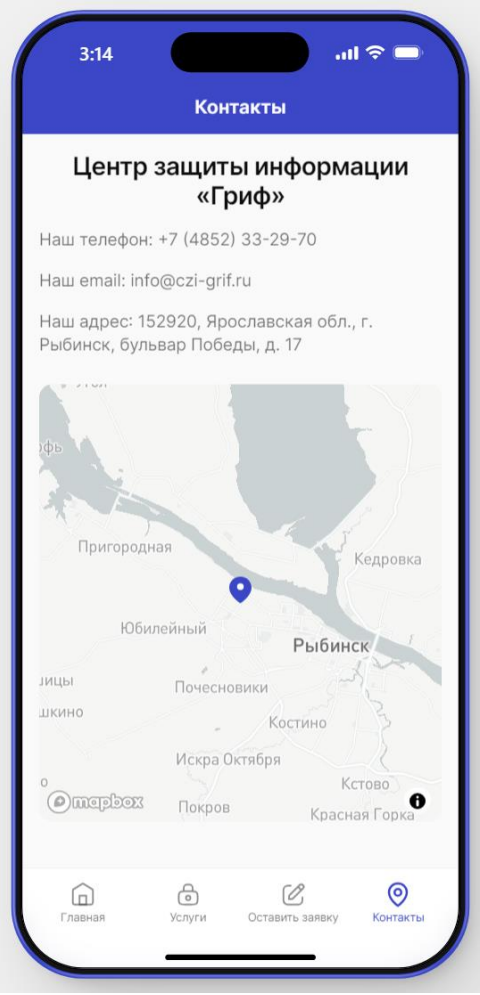

Рисунок 25 – Раздел «Контакты»

При нажатии на карту открывается изображение карты, на которой отмечено местоположение объекта.

3.4 Интеграция разработанного программного обеспечения в рабочий процесс

Полученные заявки отображаются в личном кабинете в соответствующей таблице Google, связанном с аккаунтом Glide (рисунок 26).

После завершения работы над приложением его необходимо опубликовать и сделать доступным для всех. После этого, при переходе по специальной ссылке, каждый пользователь сможет установить созданное веб-приложение (PWA) на свое мобильное устройство (рисунок 27).

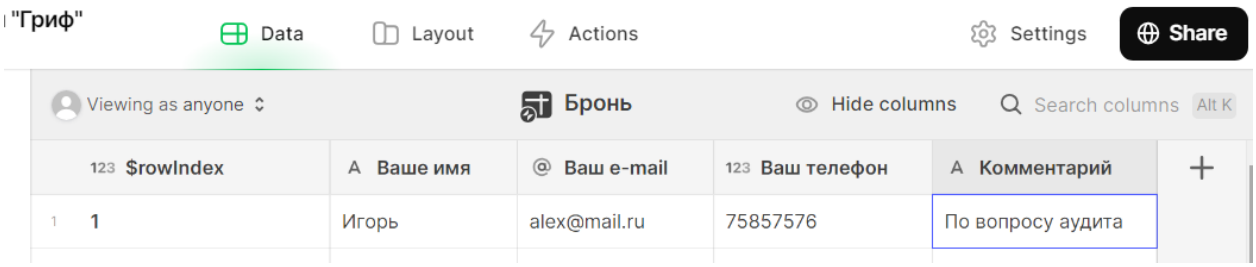

Рисунок 26 – Таблица Google с заявками

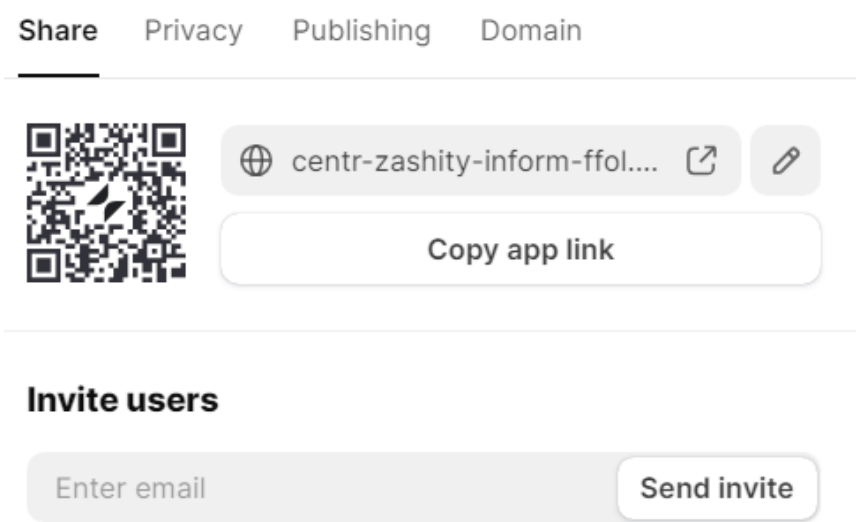

Рисунок 27 – Форма для установки приложения

Важно отметить, что процесс установки PWA приложения на мобильное устройство зависит от операционной системы и браузера, используемого на устройстве.

Процесс установки PWA приложения на мобильное устройство довольно прост. Пользователи начинают с посещения веб-приложения с помощью совместимого веб-браузера на своем мобильном устройстве. PWA – это веб-приложения, поэтому доступ к ним можно получить через URL-адрес, как и к любому другому веб-сайту.

Получив доступ к PWA, пользователи взаимодействуют с его интерфейсом и содержимым так же, как с любым другим веб-сайтом или веб-приложением. Они могут перемещаться по различным страницам, выполнять действия и получать доступ к различным функциям, предоставляемым PWA.

Чтобы установить PWA на свое мобильное устройство, пользователям обычно необходимо добавить его на главный экран. Обычно это можно сделать, открыв параметры меню браузера или нажав на значок общего доступа в браузере. В меню браузера или параметрах общего доступа пользователи должны искать опцию с надписью: «Добавить на главный экран».

Как только опция «Добавить на главный экран» будет найдена, пользователи смогут нажать на нее, чтобы начать процесс установки. Это действие обычно вызывает диалоговое окно или экран подтверждения, на котором пользователи могут подтвердить свое намерение добавить PWA на свой главный экран.

Пользователи имеют возможность настроить имя или значок PWA перед добавлением его на свой главный экран. После внесения любых желаемых изменений они могут подтвердить свой выбор, и PWA будет добавлено на главный экран их устройства в виде значка приложения.

После установки пользователи получают доступ к PWA прямо с главного экрана своего устройства, как и к любому другому собственному мобильному приложению. Нажатие на значок PWA запускает приложение в полноэкранном режиме, обеспечивая плавную работу, аналогичную приложению.

Следуя этим шагам, пользователи смогут легко устанавливать прогрессивные веб-приложения на свои мобильные устройства, что позволит им пользоваться преимуществами приложений, аналогичных нативным, непосредственно из Интернета.

Благодаря своей кроссплатформенности приложение может использоваться одновременно как пользователями Android, так и iOS.

3.5 Формирование алгоритма по развертыванию и кастомизации программных продуктов с использованием облачных технологий

Алгоритм развертывания и кастомизации программных продуктов с применением облачных технологий включает в себя несколько этапов, начиная с подготовки инфраструктуры и заканчивая настройкой и оптимизацией работы приложения. На рисунке 28 показана схема работы алгоритма.

Рисунок 28 – Схема работы алгоритма

Алгоритм можно сформулировать следующим образом.

Подготовка инфраструктуры:

– определение требований к облачному окружению, включая масштабируемость, безопасность, доступность и производительность;

– выбор облачного провайдера, учитывая его возможности, стоимость и соответствие требованиям проекта;

– создание учетных записей и настройка прав доступа для членов команды, ответственных за развертывание и кастомизацию программных продуктов.

Разработка и настройка приложения:

– создание основного кодового базиса приложения, используя средства разработки и фреймворки, подходящие для конкретного проекта (например, Glide);

– интеграция необходимых библиотек и инструментов для обработки данных, управления бизнес-логикой и взаимодействия с облачными сервисами;

– настройка параметров приложения, таких как интерфейс пользователя, функциональные возможности и безопасность.

Развертывание в облаке:

– загрузка приложения в облачное окружение с помощью инструментов, предоставляемых облачным провайдером (например, AWS, Azure, Google Cloud);

– конфигурация серверных ресурсов, включая выделение вычислительных мощностей, сетевых настроек и настройку мониторинга производительности.

Тестирование и отладка:

– проведение тестирования приложения в облачном окружении для проверки его работоспособности, производительности и безопасности;

– исправление выявленных ошибок и улучшение работы приложения в соответствии с результатами тестирования.

Кастомизация приложения:

– идентификация потребностей пользователей и бизнес-задач, определяющих требования к кастомизации приложения;

– использование инструментов для настройки интерфейса пользователя, добавления новых функциональных возможностей и адаптации приложения под конкретные потребности.

Управление версиями и обновлениями:

– разработка стратегии управления версиями приложения, включая механизмы автоматического обновления и отката изменений;

– регулярное внедрение обновлений с учетом обратной связи от пользователей и изменений в требованиях к приложению.

Мониторинг и оптимизация:

– установка систем мониторинга производительности приложения и инфраструктуры облачного окружения для выявления узких мест и проблем;

– анализ данных мониторинга и принятие мер по оптимизации работы приложения, включая масштабирование ресурсов, оптимизацию кода и улучшение пользовательского опыта.

Обучение и поддержка:

– проведение обучения для пользователей и администраторов системы, чтобы обеспечить эффективное использование и понимание функционала приложения;

– предоставление технической поддержки пользователей и решение возникающих проблем или запросов.

Таким образом, алгоритм развертывания и кастомизации программных продуктов с использованием облачных технологий включает в себя ряд шагов, начиная от подготовки инфраструктуры и разработки приложения до тестирования, кастомизации, управления версиями, мониторинга и обновлений, а также обучения и поддержки пользователей. Каждый этап требует внимания к деталям и соблюдения передовых практик разработки и управления проектами для успешной реализации программного продукта.

3.6 Апробация решения

Апробация решения проводилась с целью подтверждения или опровержения гипотезы исследования, согласно которой использование технологии по кастомизации и развертыванию программных продуктов с применением облачных технологий позволит сформулировать оптимальный алгоритм действий. В рамках апробации была проведена доказательная база, основанная на результате разработки мобильного приложения для ООО ЦЗИ «Гриф» и сформированном алгоритме развертывания и кастомизации программных продуктов.

Разработанное мобильное приложение было представлено представителям ООО ЦЗИ «Гриф» для оценки и обратной связи. В процессе апробации пользователи приложения оценивали его удобство,

функциональность и соответствие их потребностям. Также были собраны данные о производительности приложения, его надежности и безопасности.

Полученные результаты апробации свидетельствуют о том, что разработанное приложение полностью соответствует требованиям пользователей и обеспечивает удобный доступ к информации о деятельности ООО ЦЗИ «Гриф». Пользователи выразили высокую удовлетворенность функциональностью и интерфейсом приложения, а также отметили его высокую производительность и надежность.

Сформированный алгоритм развертывания и кастомизации программных продуктов также был апробирован и оценен представителями организации. Они подтвердили его эффективность и применимость для реализации облачного приложения. Результаты апробации подтвердили гипотезу исследования, демонстрируя успешное применение технологии кастомизации и развертывания программных продуктов с использованием облачных технологий для создания оптимального алгоритма действий.

Исследование деятельности предприятия выявило ключевые аспекты, которые подлежат оптимизации и улучшению, особенно в контексте внедрения облачных технологий.

Спроектированная архитектура облачного сервиса представляет собой целостное решение, направленное на совершенствование процессов и обеспечение более эффективного внутреннего взаимодействия, а также взаимодействия с клиентами и партнерами предприятия. Принятые технологические решения предоставляют надежную основу для будущего развития и успешной адаптации к динамичным изменениям в среде бизнеса.

В результате работы была разработана облачная платформа для Центра защиты информации «Гриф», которая позволит предприятию в удобной форме получать и обрабатывать входящие заявки.

Результаты данного исследования формируют твердую основу для разработки и внедрения инновационных подходов в деятельности предприятия, что в свою очередь способствует укреплению его позиций на

рынке и обеспечивает устойчивое развитие в условиях современной бизнес-среды.

Заключение

В результате работы был сформирован алгоритм развертывания и кастомизации программных продуктов с применением облачных технологий на примере Центре защиты информации «Гриф».

В рамках данной работы были решены следующие задачи:

– осуществлён анализ предметной области;

– исследованы современные облачные платформы;

– рассмотрены вопросы использования инструментов управления зависимостями при развертывании программных продуктов;

– описаны особенности мобильной разработки и развертывания на облачных платформах;

– обоснован выбор инструментов разработки мобильного приложения;

– сформировано техническое задание;

– спроектирована архитектура исследуемой облачной платформы;

– разработано мобильное приложение с использованием облачной платформы для Центра защиты информации «Гриф»;

– сформирован алгоритм по развертыванию и кастомизации программных продуктов с использованием облачных технологий.

В ходе работы были проведены аналитические исследования, что позволило внимательно рассмотреть актуальные аспекты технологий кастомизации и развертывания программного продукта с использованием облачных технологий. Были рассмотрены современные облачные платформы с целью выявления их особенностей и спектра возможностей, что дало полное представление о потенциале каждого провайдера. Важным этапом стал анализ инструментов управления зависимостями при развертывании программных продуктов, что помогло выделить ключевые аспекты в обеспечении стабильности и эффективности процесса разработки.

С учетом особенностей мобильной разработки и развертывания на облачных платформах был сформулирован ряд выводов. В частности, выявлено, что интеграция облачных технологий с мобильной разработкой представляет собой важное направление, обеспечивающее гибкость, масштабируемость и высокий уровень доступности для программных продуктов. При выборе облачной платформы следует учитывать множество факторов, помимо инструментов и сервисов, таких как интеграция с другими сервисами, гибкость и масштабируемость, безопасность и возможности для совместной работы. Стоит отметить, что все рассматриваемые платформы предоставляют широкий спектр инструментов и сервисов для разработки моделей машинного обучения.

Основываясь на результатах анализа инструментов разработки мобильного приложения, были сформированы ключевые аспекты технического задания, которое стало основой для определения параметров и характеристик программного продукта. Это позволило четко определить этапы процесса разработки и обеспечило более эффективную координацию усилий команды разработчиков.

С учетом результатов анализа деятельности предприятия и спроектированной архитектуры облачного сервиса можно сделать вывод о целостности и целенаправленности решений, направленных на совершенствование процессов и обеспечение более эффективного внутреннего и внешнего взаимодействия предприятия. Разработанная облачная платформа создает надежную основу для будущего развития и успешной адаптации к динамичным изменениям в среде бизнеса.

Таким образом, выполнение поставленных задач дало полную картину о взаимосвязи облачных технологий и процессов разработки программного обеспечения, а также выявило ключевые факторы успешного развертывания на облачных платформах. Полученные результаты предоставляют обширный обзор и обоснование стратегических решений, необходимых для успешной

разработки программного продукта, и создают основу для последующих этапов проекта, включая разработку, тестирование и развертывание.

Для предприятия была успешно развернута система по приему заявок с использованием облачной платформы Glide. Это позволило значительно упростить и оптимизировать процесс обработки входящих запросов со стороны клиентов и партнеров компании. Главным преимуществом такого подхода является возможность быстрого и гибкого масштабирования системы в зависимости от объема поступающих заявок, а также улучшение качества обслуживания за счет автоматизации процессов и повышения оперативности реагирования на запросы.

Сформированный алгоритм развертывания и кастомизации программных продуктов с использованием облачных технологий представляет собой ценный инструмент для дальнейшего развития компании. Он включает в себя шаги по выбору облачной платформы, анализу требований проекта, формированию технического задания, выбору инструментов разработки и непосредственному развертыванию и кастомизации программного продукта. Этот алгоритм обеспечивает систематизацию и структурирование процесса разработки, что способствует более эффективному использованию ресурсов и сокращению времени на внедрение новых решений.

Эта работа может быть продолжена путем дальнейшего изучения возможностей применения облачных технологий в других областях деятельности предприятия. Например, это может включать в себя анализ потенциала применения машинного обучения и аналитики данных на основе облачных сервисов для оптимизации бизнес-процессов и повышения конкурентоспособности компании. Также можно дополнительно исследовать возможности интеграции облачных технологий с другими информационными системами предприятия с целью создания единой цифровой экосистемы, способствующей более эффективному

взаимодействию между различными подразделениями и улучшению общей производительности предприятия.

Список используемой литературы

1. Бостром Н. Искусственный интеллект. Этапы. Угрозы. Стратегии. М. : МИФ, 2014. 760 с.
2. Владстон Ф. Ф. Теоретический минимум по Computer Science. Все что нужно программисту и разработчику. С. : Питер, 2018. 224 с.
3. Гудфеллоу Я., Курвилль А., Бенджио И. Глубокое обучение. М. : ДМК Пресс, 2017. 654 с.
4. Джулли А., Пал С. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow. М. : ДМК Пресс, 2017. 296 с.
5. Иерузалимски Р. Программирование на языке Lua. М : ДМК Пресс, 2018. 384 с.
6. Клепманн М., Высоконагруженные приложения. Программирование, масштабирование, поддержка. С. : Питер, 2019. 640 с.
7. Лащевски Т., Облачные архитектуры: разработка устойчивых и экономичных облачных приложений. С. : Питер, 2022. 320 с.
8. Мартин Р. Идеальный программист. Как стать профессионалом разработки ПО. С. : Питер, 2019. 240 с.
9. Мартин Р. Чистый код: создание, анализ и рефакторинг. С. : Питер, 2018. 464 с.
10. Мелькин Н., Горяев К. Искусство продвижения сайта. Полный курс SEO: от идеи до первых клиентов. М : Инфра-Инженерия, 2018. 280 с.
11. Нахавандипур В. iOS. Приемы программирования. С : Питер, 2017. 1000 с.
12. Нефедов Ю. В. Разработка кроссплатформенных мобильных приложений – перспективные методы и стандартные практики. С : Питер, 2022. 215 с.
13. Николенко С., Кадурин А., Архангельская Е. В. Глубокое обучение. Погружение в мир нейронных сетей. С. : Питер, 2018. 481 с.

14. Ньюмен С, От монолита к микросервисам. С : БХВ-Петербург, 2021. 266 с.

15. Орлов С. А. Теория и практика языков программирования. С : Питер, 2018. 688 с.

16. Пилипенко Е.В., Редькин В.А., Каракотов Р.В. Современные тенденции в мобильной разработке // Форум молодых ученых. 2021. №12 (64). URL: https://cyberleninka.ru/article/n/sovremennye-tendentsii-v-mobilnoy-razrabotke (дата обращения: 10.04.2023).

17. Прохоренок Н. Основы Java. С : БХВ-Петербург, 2017. 688 с.

18. Пьюривал С. Основы разработки веб-приложений. С : Питер, 2015. 272 с.

19. Робисон У. C# без лишних слов. М : ДМК Пресс, 2017. 344 с.

20. Рындина А.С. Выбор платформы для разработки web-сайта. Вестник современных исследований. 2017 С. 103-107

21. Семенчук В. Мобильное приложение как инструмент бизнеса. М. : Альпина Диджитал, 2017. 270 с.

22. Сенько А., Работа с BigData в облаках. Обработка и хранение данных с примерами из Microsoft Azure. С : Питер, 2019. 448 с.

23. Тарасов С. СУБД для программиста. Базы данных изнутри. М : СОЛОН-Пресс, 2015. 322 с.

24. Филимонова Е. Информационные технологии в профессиональной деятельности. М : КноРус, 2017. 483 с.

25. Форд Н., Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. С : Питер, 2020. 448 с.

26. Хабибуллин И. Самоучитель Java (3-е издание). С : БХВ-Петербург, 2008. 759 с.

27. Хабр. Обзор кросс-платформенных фреймворков мобильной разработки. URL: https://habr.com/ru/post/421227/ (дата обращения 05.11.2023).

28. Шолле Ф. Глубокое обучение на R. С : Питер, 2018. 400 с.

29. Aurelien G. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2017. 572 p.

30. Aws [Электронный ресурс] Amazon EC2. Режим доступа: https://aws.amazon.com/ru/ec2/ Дата обращения (05.11.2023)

31. Aws [Электронный ресурс] Amazon S3 Режим доступа: https://aws.amazon.com/ru/s3/ Дата обращения (05.11.2023)

32. Aws [Электронный ресурс] Amazon SageMaker. Режим доступа: https://aws.amazon.com/ru/sagemaker/ Дата обращения (05.11.2023)

33. Aws [Электронный ресурс] AWS Amazon. Режим доступа: https://aws.amazon.com/ru/ Дата обращения (05.11.2023)

34. AzeriCMS [Электронный ресурс] Топ-10 Операционные системы мобильных телефонов. Режим доступа: http://azericms.com/mobilnyie-prilozheniya/top-10-mobilnyih-telefonov-operatsionnyie-sistemyi.html Дата обращения (05.11.2023)

35. Ford M. Architects of Intelligence: The truth about AI from the people building it. Packt Publishingб 2018. 554 p.

36. GitBook [Электронный ресурс] Архитектуры ReactNative, Xamarin, PhoneGap и Qt. Режим доступа: https://slavachernikoff.gitbooks.io/architectures-of-reactnative-xamarin-phonegap-qt/ Дата обращения (05.11.2023)

37. Google Cloud [Электронный ресурс] AI and machine learning products. Режим доступа: https://cloud.google.com/products/ai Дата обращения (05.11.2023)

38. Google Cloud [Электронный ресурс] AutoML. Режим доступа: https://cloud.google.com/automl Дата обращения (05.11.2023)

39. Google Cloud [Электронный ресурс] Colab. Режим доступа: https://colab.google/ Дата обращения (05.11.2023)

40. Google Cloud [Электронный ресурс] Introduction to AI Platform. Режим доступа: https://cloud.google.com/ai-platform/docs/technical-overview Дата обращения (05.11.2023)

41. Learn React.js [Электронный ресурс] Учебник: введение в React. Режим доступа: https://learn-reactjs.ru/tutorial Дата обращения (05.11.2023)

42. Michelle M. F. Corona SDK Mobile Game Development Beginner's Guide. Gardners Books, 2018. 150 с.

43. Microsoft [Электронный ресурс] Что такое служба «Машинное обучение Microsoft Azure»? Режим доступа: https://learn.microsoft.com/ru-ru/azure/machine-learning/overview-what-is-azure-machine-learning?view=azureml-api-2 Дата обращения (05.11.2023)

44. Microsoft [Электронный ресурс] Azure Cognitive Services. Режим доступа: https://azure.microsoft.com/en-us/products/cognitive-services Дата обращения (05.11.2023)

45. Microsoft [Электронный ресурс] Azure Databricks. Режим доступа: https://azure.microsoft.com/en-us/products/databricks Дата обращения (05.11.2023)

46. Microsoft [Электронный ресурс] Azure Machine Learning. Режим доступа: https://azure.microsoft.com/en-us/products/machine-learning Дата обращения (05.11.2023)

47. ML Space [Электронный ресурс] ML Space. Режим доступа: https://cloud.ru/ru/aicloud/mlspace Дата обращения (05.11.2023)

48. React [Электронный ресурс] React Tutorial. Режим доступа: https://reactjs.org/ Дата обращения (05.11.2023)

49. SEORU [Электронный ресурс] Что такое PWA? Внедрение и преимущества технологии для владельцев сайтов. Режим доступа: https://seo.ru/blog/chto-takoe-pwa-prosto-o-tehnologii/ Дата обращения (05.11.2023)

50. SmartumPro [Электронный ресурс] Сравнение решений для кроссплатформенной разработки: PhoneGap, Xamarin, Flutter, React Native.

Режим доступа: https://smartum.pro/ru/blog-ru/sravneniye-resheniy-krossplatformennoy-razrabotki-phonegap-xamarin-flutter-react-native/ Дата обращения (05.11.2023)

51. Topol E. Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again. Basic Books, 2019. 400 p.

52. Yandex Cloud [Электронный ресурс] Yandex DataSphere. Режим доступа: https://cloud.yandex.ru/services/datasphere Дата обращения (05.11.2023)

Приложение А

Техническое задание на разработку мобильного приложения для ООО ЦЗИ «Гриф»

1 Общие сведения

1.1 Полное наименование системы и её условное обозначение

Полное наименование системы: Разработка мобильного приложения для ООО ЦЗИ «Гриф».

Краткое наименование системы: «Мобильное приложение».

1.2 Перечень документов, на основании которых создается система

Система создается на основании договора между заказчиком и разработчиком, подписанного 14.02.2023.

1.3 Сведения об источниках и порядке финансирования работ

Источником финансирования являются средства Заказчика. Порядок финансирования определяется Договором.

2 Цели и назначение создания системы

2.1 Цели создания системы

Веб-проект: Мобильное приложение создаётся с целью получить удобный инструмент, с помощью которого клиенты ООО ЦЗИ «Гриф» смогут получать актуальную информацию о деятельности предприятия, особенностях его работы.

2.2 Назначение системы

Система предназначена для выполнения следующих основных функций:

мобильная система для отслеживания деятельности ООО ЦЗИ «Гриф»;

обеспечения удобной системы приема заявок.

Выполнение основных функций должно достигаться за счет автоматизации следующих задач:

авторизация и регистрация пользователей системы;

формирования контента для приложения;

управления формирования конечных документов для администратора;

отслеживания процесса полученных заявок.

3 Требования к системе

3.1 Требования к структуре системы в целом

Система состоит из мобильного приложения, с которым взаимодействуют клиенты компании и серверной части приложения, с которым взаимодействуют менеджеры компании, осуществляющие наполнение приложения контентом.

3.2 Требования к способам и средствам обеспечения информационного взаимодействия компонентов системы

3.2.1 Информационное взаимодействие подсистем должно осуществляться через единое информационное пространство и посредством использования стандартизированных протоколов и форматов обмена данными.

3.2.2 Все компоненты подсистем должны функционировать в пределах единого логического пространства, обеспеченного интегрированными средствами серверов данных и серверов приложений.

3.3 Требования к режимам функционирования системы

3.3.1 Должна обеспечиваться работа в штатном режиме.

3.3.2 Должна обеспечиваться работа в режиме системного администрирования.

3.4 Требования по диагностированию системы

Система должна иметь подсистему диагностики ошибок. Если в процессе проверки выявлены ошибки, то подсистема диагностики должна сделать попытку исправления найденной ошибки. После завершения процесса проверки необходимо сообщать пользователю о результатах.

3.5 Перспективы развития, модернизации системы

3.5.1 При разработке приложения должна быть предусмотрена возможность масштабирования системы, а именно:

расширение основных и базовых функций;

изменение дизайна приложения;

легкое изменение формы заявки (полей формы).

3.5.2 Вследствие требований п. 2.1.5.1, система должна быть разработана на технологиях, которые позволяют легко масштабировать систему. Предлагается использование системы Glide.

3.6 Требования к функциям, выполняемым системой

Основные функции системы:

продуманный дизайн и информационная структура приложения;

обеспечение авторизации пользователей;

разработка панели управления;

возможность просмотра услуг;

возможность оставить заявку через форму обратной связи;

просмотр полученных заявок;

обеспечение защиты информации приложения.

3.7 Требования одновременности выполнения группы функций

3.7.1 Система должна выдерживать одновременное количество пользователей до 2 тысяч.

3.7.1 Ограничение на количество пользователей должно зависеть только от технических характеристик хостинг сервиса, на котором будет развернута серверная часть проекта.

3.8 Требования к видам обеспечения

3.8.1 Требования к математическому обеспечению

Требования к математическому обеспечению не предъявляются.

3.8.2 Требования к информационному обеспечению

Состав, структура и способы организации данных в системе должны быть определены на этапе технического проектирования.

Технические средства, обеспечивающие хранение информации, должны использовать современные технологии, позволяющие обеспечить повышенную надежность хранения данных и оперативную замену оборудования:

распределенная избыточная запись/считывание данных;

копирование;

независимые дисковые массивы;

кластеризация.

В состав системы должна входить специализированная подсистема резервного копирования и восстановления данных.

3.8.3 Требования к лингвистическому обеспечению

Все прикладное программное обеспечение системы для организации взаимодействия с пользователем должно использовать русский язык.

3.8.4 Требования к программному обеспечению (ПО)

Программное обеспечение будет разработано на Glide. Функциональность должна обеспечиваться выполнением подсистемами всех их функций. Надежность должна обеспечиваться за счет предупреждения ошибок недопущения ошибок в готовом приложении.

3.9 Общие технические требования к системе

3.9.1 Требования к численности и квалификации персонала и пользователей системы

3.9.1.1 Пользователями системы должны выступать: клиенты и менеджеры компании.

3.9.1.2 В состав персонала, выполняющего техническую эксплуатацию системы, должны входить:

менеджеры приложения, назначенные руководством компании;

IT-специалисты.

3.9.2 Требования к показателям назначения

3.9.2.1 В приложении должны быть предусмотрены возможности масштабирования системы в зависимости от необходимости внедрения новых функций.

3.9.2.2 Целевое назначение системы должно сохраняться на протяжении всего срока эксплуатации системы.

3.9.3 Требования к надежности

Требования к обеспечению надежного функционирования программы. Надежное (устойчивое) функционирование системы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

организацией бесперебойного питания технических средств;

использованием лицензионного программного обеспечения;

регулярным выполнением требований ГОСТ 51188-98.

Защита информации. Испытания программных средств на наличие компьютерных вирусов

3.9.4 Требования по безопасности

Все внешние элементы технических средств системы, находящиеся под напряжением, должны иметь защиту от случайного прикосновения, а сами технические средства иметь зануление или защитное заземление в соответствии с ГОСТ 12.1.030-81 и ПУЭ.

3.9.5 Требования к эргономике и технической эстетике

Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав системы должно осуществляться посредством визуального графического интерфейса (GUI). Интерфейс

системы должен быть понятным и удобным, не должен быть перегружен

графическими элементами и должен обеспечивать быстрое отображение экранных форм. Средства редактирования информации должны удовлетворять принятым соглашениям в части использования функциональных клавиш, режимов работы, поиска, использования оконной системы. Ввод-вывод данных системы, прием управляющих команд и отображение результатов их исполнения должны выполняться в интерактивном режиме. Все надписи экранных форм, а также сообщения, выдаваемые пользователю (кроме системных сообщений) должны быть на русском языке.

Экранные формы должны проектироваться с учетом требований унификации: все экранные формы пользовательского интерфейса должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации.

4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

4.1 Работа системы должна обеспечиваться только в штатном режиме.

4.2 При выходе системы из строя, самой системой, а в случае ее полного отказа IT администратором должны быть приняты меры по блокировке системы для доступа пользователей, с последующим немедленным резервным копированием информации всего веб-приложения.

4.3 Авторские права на программное обеспечение, на архитектуру системы и модели данных принадлежат разработчику и защищаются действующим законодательством.

4.4 Пользователь не имеет права передавать программное обеспечение третьим лицам, самостоятельно или через третьих лиц использовать систему в качестве образца для разработки аналогичного программного обеспечения.

Продолжение Приложения А

5 Требования к защите информации от несанкционированного доступа

5.1. Требования к информационной безопасности

Полноценная защита конфиденциальной информации, обеспечение ее целостности при полном отсутствии риска нанести ущерб работе предприятия.

5.2. Требования к антивирусной защите

За средства антивирусной защиты отвечает хостинг. Выбранный хостинг сервис должен отвечать всем современным стандартам антивирусной защиты.

5.3 Требования по сохранности информации при авариях

В Системе должно быть обеспечено резервное копирование данных.

Выход из строя трех жестких дисков дискового массива хостинга не должен сказываться на работоспособности подсистемы хранения данных.

6 Порядок разработки системы

6.1 Порядок организации разработки системы

Порядок организации разработки системы определяется стадиями и этапами работ (см. п. 5), которые установлены в соответствии с ГОСТ Р 59793-2021.

Целью стадии «Разработка технического задания» является разработка проектных решений и документации по системе и её частям. Проектные решения должны соответствовать заданным в ТЗ требованиям, а также требованиям действующих стандартов, применение которых предусмотрено в договоре или ТЗ. Проектные решения должны быть описаны в

документации в соответствии с требованиями, установленными ГОСТ Р 59795-2021.

Разработку документации выполняет разработчик. Эксплуатационная документация должна быть согласована с заказчиком.

Целью стадии «Тестирование и отладка системы» является реализация

проектных решений и плана мероприятий по вводу системы в действие, а также подготовка к постоянной эксплуатации системы. За организацию работ на стадии ответственность несёт заказчик. За результаты работ на стадии ответственность несёт разработчик.

Для проведения испытаний системы необходимо создать комиссии из специально привлекаемых специалистов. Целесообразно в состав таких комиссий включать специалистов из незаинтересованных организаций. Комиссию, производящую приемочные испытания, формирует заказчик.

6.2 Перечень документов и исходных данных для разработки

При разработке системы должны быть обеспечены требования п. 4 настоящего ТЗ. В процессе разработки системы перечень документов может быть скорректирован по согласованию сторон.

7 Порядок контроля и приемки системы

7.1 Виды испытаний: приемо-сдаточные испытания должны проводиться на хостинге Заказчика в оговоренные сроки. Приемо-сдаточные испытания веб-приложения должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком Программы и методик испытаний. Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний.

Ответственность за организацию и проведение приемки системы должен нести заказчик. Приемка системы должна производиться по

завершению приемки всех задач системы. При этом необходимо предоставить обеспечение материальной частью (технические средства), проектной документацией и специально выделенным персоналом.

7.2 Общие требования к приемке работы: на основании протокола проведения испытаний исполнитель совместно с заказчиком подписывает акт приемки-сдачи веб-приложения в эксплуатацию.

Продолжение Приложения А

Приемка этапа заключается в рассмотрении и оценке проведенного объема работ и предъявленной технической документации в соответствии с требованиями настоящего технического задания.