РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Допустить к защите в ГЭК
И.о.зав. кафедрой «ВТ и АСУ»

_____ О.В. Игнатьева

«_____»_____2021 г.

Разработка микросервисного приложения для организации работы контакт-центра

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной (бакалаврской) работе
АВБ 12.02.13

Направление подготовки «Информационные системы и технологии»,
профиль «Информационные системы и технологии на транспорте»

Обучающийся	_____	И.А. Мазуров
Руководитель работы д.п.н., профессор	_____	О.И. Соколова
Нормоконтроль ст. преподаватель	_____	Н.Р. Осипова
Научный консультант к.т.н., научный сотрудник	_____	О.И. Соколова

2021

РОСЖЕЛДОР
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Ростовский государственный университет путей сообщения»
(ФГБОУ ВО РГУПС)

Кафедра «ВТ и АСУ»

УТВЕРЖДАЮ

И.о.зав.кафедрой
_____ О.В.Игнатьева

«_____» _____ 2021 г.

ЗАДАНИЕ

на выпускную квалификационную(бакалаврскую) работу

Студенту Мазурову Илье Андреевичу
(фамилия, имя, отчество)

Группа АВБ-4-032

1. Тема работы: Разработка микросервисного приложения для организации работы контакт-центра

Утверждена приказом по университету № 95/ос от 26.01.2021 г.
Срок сдачи студентом законченной работы «9» июня 2021 г.

2. Исходные данные к работе _____

3. Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих разработке)

4. Перечень графического материала (с точным указанием названий слайдов презентации)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Дата выдачи задания « 8 » февраля 2021 г.

Руководитель работы _____
(подпись) (инициалы, фамилия)

Задание принял к исполнению « 8 » февраля 2021 г.

Студент_____
(подпись) (инициалы, фамилия)

Реферат

Бакалаврская работа содержит 75 листов пояснительной записки, включающей 12 рисунков, 8 таблиц, 14 источника и 5 приложений.

Объектом исследования является разработка микросервисного приложения для организации работы контакт-центра.

Цель работы – разработка микросервисного приложения для организации работы контакт-центра.

Введение

Использование информационных технологий больше не сводится только к установке оборудования или программного обеспечения, решению компьютерных проблем или контролю за тем, кто может получить доступ к конкретной системе. Сегодня IT-сфера востребована также в:

1 поддержке сети и устройства для максимального времени безотказной работы;

2 автоматизации процессов для повышения эффективности бизнеса;

3 исследовании, внедрении и управлении новыми технологиями для удовлетворения меняющихся потребностей бизнеса;

4 поддержке уровня обслуживания, безопасности и возможности подключения для обеспечения непрерывности и долговечности бизнеса;

Из всего вышесказанного можно сделать вывод, что IT-сфера все больше становится популярна в области бизнеса.

Благодаря информационным технологиям многие традиционные бизнес-модели и концепции претерпели изменение. Информационные технологии дали новые возможности и перспективы, а также позволили повысить эффективность ведения бизнеса. Множество повседневных рутинных и ручных задач были автоматизированы и оптимизированы.

Одним из главных нововведений со стороны информационных технологий стало обеспечение непрерывной связи с клиентами. Каждый человек, имеющий доступ в интернет способен получить обратную связь или ответ на интересующий его вопрос различными способами, например, по электронной почте или через чат на веб-сайте. Но остаются люди, которые не готовы долго ждать ответов на свои вопросы, или которые предпочитают общаться с агентами по обслуживанию клиентов в реальном времени, для всех этих вариантов связи было создано единое решение – контакт-центр.

На сегодняшний день любое дело, основывающееся на продаже товаров и услуг обязано иметь постоянно работающую техническую поддержку и связь с

клиентом – это стандарт современного бизнеса. Ранее с этим успешно справлялись колл-центры, основными задачами которых были: обработка звонков и информирование клиентов в интересах бизнеса. В современном мире потребности клиентов меняются, и компаниям приходится искать все больше способов для контакта с конечными пользователями. Выходом из такой ситуации является – создание или организация контакт-центра, который представляет собой следующий эволюционный этап развития колл-центров.

Рассматриваемая тема выпускной квалификационной работы является актуальной, так как посвящена разработке микросервисного приложения для организации работы контакт-центра. Использование контакт-центра сейчас актуально для самых разных отраслей, и большинство компаний, которые начинают заниматься предоставлением товаров и услуг, являются крупными финансовыми организациями и IT-компаниями, обязаны иметь средства высокоэффективной службы поддержки.

Целью выпускной квалификационной работы является создание микросервисного приложения для организации работы контакт-центра, которое позволит организации управлять всеми взаимодействиями с клиентами по различным каналам связи.

Для достижения поставленной цели необходимо выполнить следующие взаимосвязанные задачи:

1 Анализ предметной области. Изучить и проанализировать существующие разработки в предметной области. Разработать техническое задание согласно требованиям;

2 Спроектировать приложение на основе диаграмм прецедентов, классов и последовательности. Рассчитать время разработки приложения на основе функционально-ориентированных метрик и модели издержек разработки;

3 Реализовать приложение на основе поставленного технического задания и проекта. Применить спроектированные UML диаграммы и полученные при изучении предметной области теоретические знания при

разработке приложения.

Комплексный характер темы выпускной квалификационной работы определил необходимость изучения и практического применения трудов национальных и зарубежных разработчиков программного обеспечения: Джулия Коннелл - профессор Высшей школы бизнеса Ньюкасла, Джон Берджесс - профессор Высшей школы бизнеса Ньюкасла, Джеффри Рихтер – IT-специалист и разработчик, автор нескольких книг по технологиям Microsoft, Эндрю В. Троелсен - является менеджером по технологиям в Thomson Reuters в подразделении Enterprise Content Platform и другие, которые изучали и проектировали информационные системы контакт-центров. Их труды были использованы при разработке и создании микросервисного приложения для организации работы контакт-центра.

В первой главе был осуществлен анализ предметной области для разрабатываемого приложения, рассмотрены программы-аналоги, разработано техническое задание для дальнейшего проектирования и разработки.

Во второй главе при проектировании микросервисного приложения для организации работы контакт-центра был выполнен анализ требований к разрабатываемой системе на основе построения диаграмм UML. На основе построения диаграмм прецедентов спроектированы основные функциональные возможности приложения, построение диаграммы классов позволит сформировать основу программной реализации, расчет функционально-ориентированных метрик и модели издержек даст представление о примерных сроках разработки приложения.

В третьей главе описан процесс разработки программного кода, дано описание взаимодействия частей приложения, а также содержится описание интерфейса готового приложения и его функционал.

1 Анализ предметной области разрабатываемого приложения

1.1 Актуальность разработки микросервисного приложения для организации работы контакт-центра

В настоящее время информационные технологии стали активно применяться в различных областях нашей жизни, исключением не стала и такая сложная тема, как взаимодействие клиентов и бизнеса. Широкое внедрение цифровых устройств и постоянное подключение к Интернету меняют то, как клиенты желают общаться с организациями. Разговоры больше не ограничиваются традиционным голосовым каналом, вместо этого клиенты ожидают беспрепятственного взаимодействия с компаниями по нескольким каналам, такими как чат, мобильные устройства, видео и социальные сети.

Правильным решением в такой ситуации будет - интеграция в организацию работы компании новых технологий, а именно контакт-центра. Контакт-центр — это бизнес-подразделение внутри организации, которое управляет взаимодействием с клиентами. В отличие от колл-центра, который получает запросы только по телефону, контакт-центр обрабатывает входящие и исходящие сообщения клиентов по нескольким каналам, таким как телефон, Интернет, чат, электронная почта, приложения для обмена сообщениями, социальные сети, текстовые сообщения, факс и традиционная почта. Контакт-центры используют различные типы передовых технологий, чтобы помочь быстро решать проблемы клиентов, отслеживать и собирать данные о взаимодействие с клиентами с целью улучшения производительности работы бизнеса.

Контакт-центр обладает рядом достоинств, которые помогут любой организации стать лучше:

1 Улучшенный клиентский опыт. Контакт-центр часто является основным каналом для большинства взаимодействий клиентов с компанией. Через контакт-центр можно обеспечить отличный опыт взаимодействия несколькими способами. Например, клиенты могут воспользоваться возможностями самообслуживания, предоставляемыми контакт-центром, для

быстрого выполнения общих задач. Контакт-центры могут обеспечить подключение клиентов к подходящему агенту для более быстрого решения их проблемы. Кроме того, поскольку контакт-центр поддерживает несколько видов связи, клиенты могут взаимодействовать по выбранному ими каналу.

2 Повышение эффективности. Поскольку контакт-центры являются центральной точкой взаимодействия с клиентами, они могут повысить производительность бизнеса, отвечая на многочисленные запросы клиентов. Контакт-центры могут предоставить организации возможность автоматически направлять запросы на наиболее подходящий ресурс, что позволяет операторам реагировать быстро и эффективно.

3 Расширенное понимание и видимость. Контакт-центр объединяет взаимодействие с клиентами по всем каналам связи. Данные, собранные в результате этих взаимодействий, дают ценную информацию, которая может помочь в принятии важных бизнес-решений. Эта информация может помочь с усовершенствованием дизайна продукта, вопросами качества продукта, а также с выявлением шаблонов и проблемных областей на пути клиента к улучшенному взаимодействию с компанией.

Разрабатываемое микросервисное приложение для организации работы контакт-центра станет доступной возможностью любым компаниям организовать эффективную работу. Такое приложение подойдет любой отрасли, где важна коммуникация с клиентом: онлайн-магазинам, банкам, сфере грузоперевозок, туристическим агентствам и другим компаниям. С помощью организации контакт-центров можно достичь высокой эффективности в работе службы поддержки, которая необходима Интернет-провайдерам и операторам сотовой связи, логистическим и транспортным компаниям.

Использование для реализации микросервисного приложения вполне логично, ведь микросервисы, обычно называемые «микросервисной архитектурой», представляют собой способ структурирования приложения таким образом, что бизнес-возможности, которые должны быть предоставлены в приложении, могут быть отделены, созданы и развернуты как независимые служ-

бы. Вместо того, чтобы разрабатывать приложение как единое целое (так называемая монолитная архитектура приложения), вся бизнес-функциональность приложения разбивается на уникальные процессы. Каждый процесс проектируется и разрабатывается как самостоятельный сервис, содержащий свою бизнес-логику и имеющий свой набор обособленных данных.

Микросервисная архитектура помогает снизить сложность разработки. Большие или сложные приложения можно разбить на более простые, легко разрабатываемые и поддерживаемые сервисы, которые можно обновлять по мере изменения динамики бизнеса и необходимости переделывать рабочие процессы, чтобы они отражали реальную работу.

Команде разработчиков также легче практиковать непрерывную доставку и развертывание функций кода для поддержки динамических процессов, включающих частые или динамически изменяющиеся бизнес-среды. Кроме того, это помогает бизнесу постоянно развивать свой технологический стек и внедрять новые процессы и методы, чтобы оставаться конкурентоспособными на рынке.

Одной из главных причин использования микросервисного приложения является необходимость возможности масштабирования в зависимости от бизнес-требований. Крупные организации имеют различные бизнес-процессы и требования. Их необходимо автоматизировать и переделывать по мере изменения масштаба. Монолитные архитектуры требуют времени для создания, изменения и развертывания. С помощью микросервисов конкретную службу можно перепроектировать и развернуть за короткое время, не влияя на объем других процессов или сервисов. Новые сервисы можно создавать в короткие сроки и развертывать независимо.

Архитектура микросервисов не определяет и не ссылается на конкретный набор технологий, процессов или инструментов. Скорее, она сосредотачивается на целях. Для любого бизнеса чем раньше будут автоматизированы рабочие процессы, тем быстрее будет доставка. Кроме того, автоматизированные процессы должны быть надежными и согласованными, с точки зрения обеспечивания. Реальная ценность микросервисов для бизнеса может быть реализована

путем сосредоточения внимания на двух ключевых аспектах — скорости и надежности — путем их эффективного балансирования в соответствии с потребностями.

Таким образом, в данном подразделе была рассмотрена и обоснована актуальность разработки микросервисного приложения для организации работы контакт-центра.

1.2 Аналитический обзор существующих приложений для организации работы контакт-центра

Для более полного понимания о том, что представляет из себя контакт-центр и для составления более четкой структуры о функциональных возможностях разрабатываемого программного средства, необходимо изучить рынок аналогичных приложений или приложений, предоставляющих близкий к разрабатываемому приложению функционал.

Контакт-центры делятся на несколько типов в зависимости от их реализации:

1 Аппаратные контакт-центры. Организации могут устанавливать и размещать аппаратные контакт-центры на физических локальных серверах. Следовательно, аппаратные варианты требуют от организаций достаточного пространства для размещения и мощности в обслуживание серверов, эффективных процедур аварийного восстановления и компетентных процессов обновления оборудования;

2 Облачные контакт-центры. В данном случае контакт-центры размещаются на Интернет-серверах облачных провайдеров и фильтруют все входящие и исходящие сообщения. Агенты могут получить доступ к облачным контакт-центрам из любой точки сети Интернет. Эти центры функционируют так же, как и другие;

3 Hosted контакт-центры. Для этого варианта организация передает инфраструктуру другой компании, которая управляет системами извне. Такой

подход может минимизировать первоначальные затраты на техническое обслуживание, что часто приводит к повышению эффективности инвестиций в проекты;

4 Виртуальные контакт-центры. Они позволяют агентам работать удаленно. Виртуальные контакт-центры, обеспечивая гибкость и комфорт для операторов, одновременно снижают расходы компаний.

Но несмотря на разную реализацию, все они выполняют схожие функции, имея одну суть разработки. Чтобы более детально разобраться в функционале каждого вида контакта-центра, необходимо рассмотреть несколько существующих аналогов.

На первом месте в России по популярности находится компания Mango Office — это одна из крупнейших телекоммуникационных компаний страны, является абсолютным лидером российского рынка виртуальных АТС и одним из ведущих поставщиков SaaS-решений.

Рассматриваемая компания предоставляет множество сервисов, которые включают в себя: виртуальную АТС, коллтрекинг, интеграции с различными CRM-системами, сквозную аналитику, речевую аналитику, бизнес-аналитику и контакт-центр.

В данном случае клиенту предоставляют услуги облачного контакт-центра, в число которых входит:

1 Омниканальное общение - работа со всеми заявками из единого окна: звонки, письма, сообщения из чата на сайте, социальные сети, мессенджеры, формы обратной связи и заказы на обратный звонок. Вся история коммуникаций из любых каналов связи с клиентом сохраняется в карточке-сделки Контакт-центра;

2 Умная маршрутизация звонков - качественный прием обращений: голосовое меню, автоинформатор о времени ожидания. Выстраивание индивидуального маршрута звонков специально под бизнес-процессы: гибкие алгоритмы распределения звонков;

3 Голосовые роботы - прием и обработка звонков

роботизированными сервисами по индивидуальным заданным сценариям. Массовые исходящие обзвоны клиентских баз;

4 Чат-боты – прием и обработка роботом сообщений от клиентов из соцсетей, мессенджеров, чата на сайте и диалоги с ними по заданным сценариям. Адресация текстового обращения на сотрудника при необходимости. Автоматическое создание карточки нового клиента или занесение заявки от текущего клиента в CRM;

5 Анализ работы сотрудников - контроль работы операторов по любым заданным параметрам в динамике: количество пропущенных, количество перезвонов, время на линии, выполнение задач сотрудниками, время на обработку текстового обращения, успешные/неуспешные звонки и многое другое;

6 Интеграции - Контакт-центр MANGO OFFICE уже интегрирован с основными CRM-системами и множеством других бизнес-приложений. Благодаря готовым интеграциям внедрение проходит максимально быстро и бесшовно.

По данному решению можно сделать вывод, что оно применимо в разных бизнес-подразделениях для решения разных задач: организация отделов продаж, отделов логистики, построение и эксплуатация колл-центров, и контакт-центров.

IPT Call Center – один из поставщиков услуг в сфере аутсорсинговых цифровых технологий, ориентированных на обслуживание быстрорастущих, революционных, рыночных и технологических компаний, помогающих контролировать взаимодействие, выполнять сложные задачи и развивать свои бренды.

Данная компания предлагает полный аутсорсинг бизнес-процессов, который повышает лояльность к бренду, включая колл-центр, автоответчик, чат, техническую поддержку и поддержку по электронной почте.

В функции контакт-центра включены:

1 Запуск и сопровождение центра бесперебойной связи с клиентами с

использованием инструментов call/контакт-центра: телефон, электронная почта, онлайн чат;

2 100% записанных звонков и хранение записей до 3-х месяцев;

3 Поддержание стандарта обеспечения параметра SLA 80/20, т.е. 80% поступивших звонков в течение 20 секунд;

4 Запуск IVR - приветственное голосовое объявление, позволяющее выбирать пункты меню с помощью клавиатуры телефона;

5 Юридическое сопровождение: получение согласия на обработку персональных данных и маркетингового согласия, а также выполнение информационного обязательства GDPR;

6 Индивидуальная оценка в зависимости от масштаба проекта.

По данному варианту контакт-центра можно сделать вывод, что данная компания предоставляет минимальный набор услуг, которые увеличивают ценность и количество клиентов, а также уровень обслуживания.

Также рассмотрим американскую компанию Concetrix. Concetrix - это американская компания, предоставляющая бизнес-услуги , специализирующаяся на привлечении клиентов и повышении эффективности бизнеса.

Компания создает инновационные решения, сочетая таланты с технологиями, чтобы помочь установить глубокие связи с клиентами, которые повышают лояльность к бренду и оптимизируют результаты бизнеса. Одной из услуг, предоставляемой данной компанией является контакт-центр.

Контакт-центр Concetrix – это облачное решение, которое обеспечивает последовательное, насыщенное и персонализированное взаимодействие с клиентами, укрепляющее доверие и лояльность к бренду.

Многоуровневые управляемые услуги устраняют головную боль, связанную с управлением решением самостоятельно. Concetrix позаботится об инфраструктуре и операциях контакт-центра, чтобы организация могла сосредоточиться на своем бизнесе.

Список функций, включающих в себя контакт-центр:

1 Мониторинг;

2 Оповещения;

3 Продажа билетов;

4 Управление инцидентами;

5 Проактивные коммуникации с клиентами;

6 Управление хранилищем;

7 Отчетность по управлению эффективностью;

8 Интегрированная аналитика данных;

9 Оптимизация решения.

Таким образом в данном разделе был изучен рынок приложений для организации работы контакт-центра. Были рассмотрены аналоги разрабатываемого программного средства и их основные функции.

1.3 Техническое задание на создание микросервисного приложения для организации работы контакт-центра

1.3.1 Общие сведения

Полное наименование системы: микросервисное приложение для организации работы контакт-центра.

Краткое наименование: контакт-центр.

1.3.2 Назначение системы

Контакт-центр — это центральная точка, из которой организации управляют всеми взаимодействиями с клиентами по различным каналам связи.

Их главное предназначение состоит в том, чтобы предложить клиентам эффективную и действенную техническую поддержку, наладить обслуживание клиентов и помочь бизнесу в продажах.

Контакт-центр должен улучшить клиентский опыт, например, клиенты могут воспользоваться возможностями самообслуживания, предоставляемыми контакт-центром, для быстрого выполнения общих задач.

Основным назначением контакт-центра является также повышение эффективности, поскольку контакт-центры являются центральной точкой взаимодействия с клиентами, они могут повысить производительность бизнеса, отвечая на многочисленные запросы клиентов.

Также контакт-центр предназначен для расширения понимания и видимости, другими словами контакт-центр должен позволить воспользоваться данными, собранными в ходе взаимодействий с клиентами для принятия важных бизнес решений. Эта информация может помочь с дизайном продукта, вопросами качества, а также с выявлением шаблонов и проблемных областей при взаимодействии клиента с компанией.

Таким образом, было рассмотрено основное назначение разрабатываемого по.

1.3.3 Цели создания системы

Микросервисное приложение для организации работы контакт-центра создается с целью:

– Улучшить клиентский опыт - программное обеспечение контакт-центра собирает данные о клиентах из каждого используемого канала и объединяет их в единый профиль клиента. Поскольку большинство клиентов контакт-центра взаимодействуют по нескольким каналам, центр собирает больше данных. Больше данных может позволить контакт-центру адаптировать клиентский опыт для конкретных абонентов и лучше маршрутизировать вызовы и другие виды связи;

– Улучшить информацию о клиентах – контакт-центры могут улучшить профилирование клиентов. Когда клиенты взаимодействуют с колл-центрами или контакт-центрами, они делятся информацией о своих личных предпочтениях и поведении, которую агенты могут собирать и использовать для улучшения клиентского опыта при будущих взаимодействиях. Организации также могут интегрировать программное обеспечение CRM с контакт-центром,

чтобы собирать больше данных о клиентах и эффективно их анализировать;

– Увеличить экономию времени и денег - контакт-центры позволяют клиентам самостоятельно обслуживать и решать свои проблемы с помощью двустороннего обмена мгновенными сообщениями на основе ключевых слов, обмена текстовыми сообщениями или общения с чат- ботом . Такое самообслуживание сокращает время, которое операторы проводят у телефона, сокращая время ожидания клиентов и снижая общие расходы.

Таким образом, были рассмотрены основные цели создания разрабатываемого программного обеспечения.

1.3.4 Требования к системе

Приложение в виду относительно большого количества функционального обеспечения будет представлять собой микросервисную систему. Каждая отдельная служба будет создана в виде отдельного сервиса, абсолютно изолированного от других сервисов.

Система должна поддерживать следующие режимы функционирования:

– основной режим работы операциониста;

– режим работы администратора.

В основном режиме работы операциониста система предоставляет доступ к взаимодействию с входящими и исходящими взаимодействиями.

Режим работы администратора расширяет возможности основного режима работы операциониста посредством открытие дополнительных разделов работы со статистикой. При этом функционал доступный в основном режиме сохраняется.

Таким образом, в данном разделе, согласно техническим требованиям предметной области и особенностям разработки микросервисного программного обеспечения, было разработано техническое задание для разработки микросервисного приложения для организации работы контакт-центра.

2 Проектирование микросервисного приложения для организации работы контакт-центра

2.1 Анализ требований на основе диаграммы прецедентов и сценария использования микросервисного приложения для организации работы контакт-центра

Для более детального проектирования разрабатываемой системы необходимо обратиться к UML–языку проектирования, благодаря которому в минимальные сроки будет возможным спланировать все нюансы будущей информационной системы.

Моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее обшей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели, соответствующей программной системы.

Одной из основных диаграмм в языке UML является диаграмм прецедентов. Эта диаграмма отображает отношения между различными пользователями системами(актерами) назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования.

Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

В большинстве случаев построение диаграммы прецедентов преследует следующие цели:

– формирование общих требований к функциональным процессам системы;

– разработка базовой концептуальной модели информационной системы для ее последующей детализации;

– формирование исходной документации для налаживания взаимодействия между разработчиками системы и заказчиками, а также пользователями.

Для отображения основных функциональных возможностей системы, благодаря которым пользователей может получить необходимый результат была разработана диаграмма прецедентов, изображенная ресурсы на история рисунке 2.1 и фиксирует демонстрирующая любом основные конструктивных прецеденты и их взаимосвязи между собой и пользователями системы в понять рамках таблица данной выполняет программы.

Рисунок 2.1 – Диаграмма прецедентов

Благодаря проектированию данной UML-диаграммы было сформировано представление о структуре будущей информационной системы, а также отоб-

ражены основные связи между основными функциональными прецедентами системы. Информация, полученная при построении данной диаграммы, упростит и ускорит процесс дальнейшей разработки проекта приложения.

На основе существующей диаграммы прецедентов разрабатываются сценарии вариантов использования, то есть детальное описание каждого возможного прецедента. Данная процедура помогает предоставить удобное и понятное техническое задание разработчику, который, зачастую, не должен вникать в бизнес-логику информационной системы.

Сценарий: Отправка сообщения.

Актеры: Пользователь, Система.

Цель: отправить сообщение.

Предусловие: Профиль пользователя не авторизован.

Краткое описание: Пользователь пользуется своим профилем для отправки сообщения. Пользователь проходит авторизацию в системе, далее запускает необходимую процедуру после чего система совершает отправку письма по указанному номеру.

Тип: Базовый

Ссылки на другие варианты использования:

Включает в себя ВИ авторизации профиля пользователя.

Раздел – «Типичный ход событий»:

Действия актеров:

1. Пользователь попадает в Окно Авторизации.
2. Пользователь вводит свои данные для авторизации (логирования) в соответствующие поля приложения.
3. Система создает подключение к БД для того, чтобы сверить введённые данные с имеющимися в БД.

Исключение №1: Ошибка авторизации, не удалось создать подключение к БД.

4. Система сверяет введенные данные с имеющимися в БД.

Исключение №2: Ошибка авторизации, введенные данные не найдены в БД.

5. Система выдает сообщение пользователю о успешной авторизации.
6. Пользователь вводит номер телефона и текст сообщения и кликает на соответствующую кнопку для его отправки.
7. Система отправляет сообщение.

Исключение №3: При отправке сообщения произошла ошибка, которая не дает возможности совершить отправку.

8. Система выводит на экран пользователя сообщение об успешной отправке.
9. Система готова к дальнейшему использованию.

Раздел – «Исключения»:

Действия актеров:

Исключение №1: Ошибка авторизации, не удалось создать подключение к БД.

4. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 2 (с сохранением введенных данных).
5. Пользователь изменяет свои данные для авторизации (логирования) в соответствующих полях приложения.

Исключение №2: Ошибка авторизации, введенные данные не найдены в БД.

5. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 2 (с сохранением введенных данных).
6. Пользователь изменяет свои данные для авторизации (логирования) в соответствующих полях приложения.

Исключение №3: При отправке сообщения произошла ошибка, которая не дает возможности совершить отправку.

8. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 6.
9. Пользователь исправляет ошибку и снова отправляет сообщение.

2.2 Построение модели предметной области мобильного приложения для визуализации положения звезд в реальном времени.

Для построения модели предметной области отлично подходит еще одна структурная диаграмма UML–языка проектирования – диаграмма классов. Диаграмма классов (англ. class diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперaций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от концепций, которые не относятся к реальности (производительность, инкапсуляция, видимость и т. п.). Классы можно рассматривать с позиции различных уровней, выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации.

Взаимосвязи на диаграмме классов отображаются линиями, существует шесть типов связи:

– Ассоциация – тип отношения, отображающий связь классов между собой по какому-либо признаку

– Наследование (обобщение) – отношение, при котором структура одного класса является составной частью(обобщением) другого.

– Имплементация – отношение, при котором один класс реализует поведение второго класса или интерфейса.

– Зависимость – отношение, при котором изменение в основном классе влияет на работу во втором, обратный механизм при этом не работает.

– Агрегация – тип отношения, отображающий взаимосвязь между це-

лым и его частями.

– Композиция – более строгий вариант агрегации, при котором существование целого напрямую зависит от существования его частей.

На рисунке 2.2 отображено визуальное представление связей между классами в UML-диаграммах.

Рисунок 2.2 – Графическое отображение взаимосвязей между классами

Диаграмма классов является ключевым элементом при моделировании информационной системы, проектирование которой происходит посредством использования объектно-ориентированной схемы моделирования. Эта диаграмма позволяет получить примерное представление об программной составляющей проектируемой системы уже на первых этапах разработки. Однако, стоит отметить, что при непосредственной разработке программного обеспечения, программист зачастую дорабатывает и перерабатывает диаграмму классов к виду, максимально удобному для конкретной реализации в используемом скопе языков программирования.

Диаграмма классов для микросервисного приложения для организации работы контакт-центра представлена на рисунке 2.3.

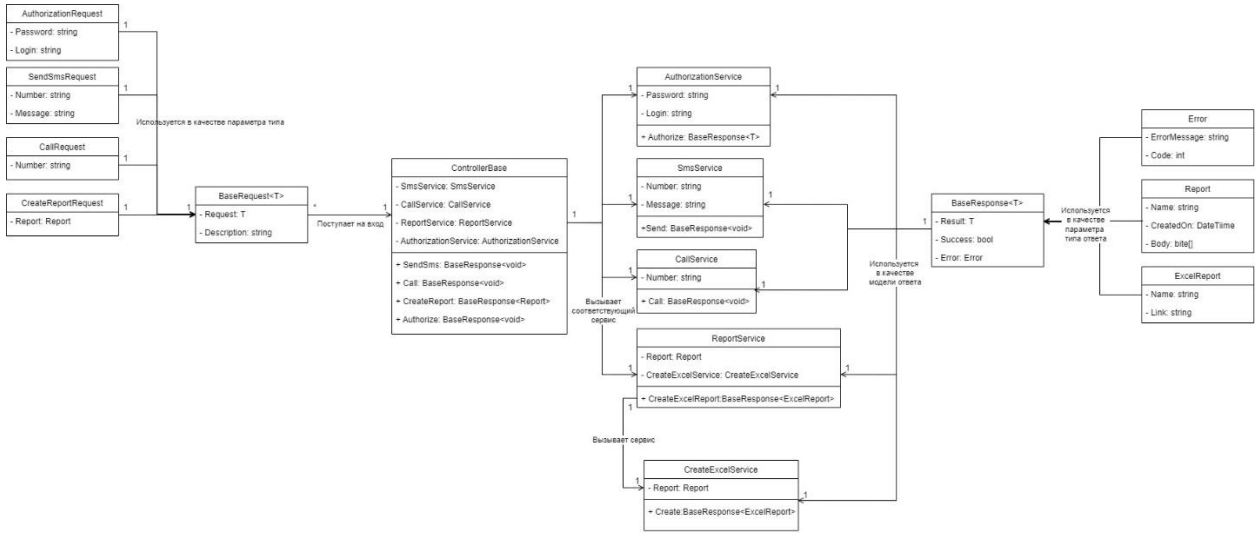

Рисунок 2.3 – Диаграмма классов разрабатываемого приложения

Диаграмма классов составляется до начала работы над программным кодом приложения, поэтому необходимо ее делать максимально абстрактной, но, при этом, необходимо максимально точно выразить функционал классов и их взаимосвязи между собой. Для упрощения понимания данной диаграммы также составляется специальная таблица, описывающая поведение конкретных классов в проектируемой системе. Описание классов приведено в таблице 2.1.

Таблица 2.1 – Описание классов

Наименование	Описание
BaseRequest<T>	Класс-модель, необходимый для передачи информации запроса с клиентской части приложения на серверную, содержит в себе описание в виде текстовой информации, а также сам запрос, имеющий один из типов, созданных для отправки запросов.
AuthorizationRequest	Класс, который является типом запроса на авторизацию пользователя. Содержит в себе информацию о данных пользователя, а именно об логине и пароле в виде текстовых данных. Данный класс используется при определении класса BaseRequest в качестве па-

	раметра типа при отправке запроса с клиентской части на серверную.
SendSmsRequest	Класс, который является типом запроса для отправки сообщения клиенту. Содержит в себе информацию о номере телефона, на который необходимо отправить сообщение и сам текст сообщения, в виде текстовых данных. Данный класс используется при определении класса BaseRequest в качестве параметра типа при отправке запроса с клиентской части на серверную.
CallRequest	Класс, который является типом запроса для звонка клиенту. Содержит в себе информацию о номере телефона, на который необходимо совершить звонок, в виде текстовых данных. Данный класс используется при определении класса BaseRequest в качестве параметра типа при отправке запроса с клиентской части на серверную.
CreateReportRequest	Класс, который является типом запроса для создания отчета. Содержит в себе информацию о создаваемом отчете виде данных класса Report. Данный класс используется при определении класса BaseRequest в качестве параметра типа при отправке запроса с клиентской части на серверную.
ControllerBase	Главный класс взаимодействия пользователя с приложением. Класс служит для приема http-запросов с клиентской части на серверную и выполняет роль маршрутизатора, то есть вызывает сервис, соответствующий полученному запросу, и передает в него полученные данные, а также принимает ответ о выполненной операции сервисом и передает его на клиентскую часть приложения, если это необходимо.

	Содержит в себе экземпляры классов, которые реализуют, необходимые для работы приложения сервисы.
AuthorizationService	Класс инкапсулирует механизм авторизации в приложении. Данный сервис вызывается из основного класса ControllerBase, на вход принимает запрос типа BaseRequest<AuthorizationRequest>. После выполнения своей задачи данный класс формирует ответ, имеющий тип BaseResponse, и передает его обратно вызывающему классу - ControllerBase.
SmsService	Класс инкапсулирует механизм отправки смс клиенту. Данный сервис вызывается из основного класса ControllerBase, на вход принимает запрос типа BaseRequest<SmsRequest>. После выполнения своей задачи данный класс формирует ответ, имеющий тип BaseResponse, и передает его обратно вызывающему классу - ControllerBase.
CallService	Класс инкапсулирует механизм совершения звонка клиенту. Данный сервис вызывается из основного класса ControllerBase, на вход принимает запрос типа BaseRequest<CallRequest>. После выполнения своей задачи данный класс формирует ответ, имеющий тип BaseResponse, и передает его обратно вызывающему классу - ControllerBase.
ReportService	Класс инкапсулирует механизм работы с отчетами. Данный сервис вызывается из основного класса ControllerBase, на вход принимает запрос типа BaseRequest<CreateReportRequest>. После выполнения своей задачи данный класс формирует ответ, имеющий тип BaseResponse<Report>, и передает его обратно вызывающему классу - ControllerBase.

CreateExcelService	Класс инкапсулирует механизм работы с Excel-данными. Данный сервис вызывается из основного сервиса ReportService, на вход принимает запрос типа Report. После выполнения своей задачи данный класс формирует ответ, имеющий тип ExcelReport, и передает его обратно вызывающему классу - ReportService.
BaseResponse<T>	Класс-модель, необходимый для передачи информации ответа с серверной части приложения на клиентскую, содержит в себе результат в виде булевского значения, ошибку типа Error, а также сам результат, имеющий один из типов, созданных для возврата ответа.
Error	Класс, который является типом для хранения информации об ошибке. Содержит в себе информацию о об ошибке в текстовом формате и код ошибки, представляющий собой числовое значение. Данный класс используется при отправке ответа в случае, если во время работы приложения произошла ошибка.
Report	Класс, который является типом для передачи информации об отчете. Содержит в себе название, создаваемого отчета в виде текстовых данных, дату его создания, а также сам отчет представляемый собой массив байт. Данный класс используется при отправке ответа в качестве результата работы сервиса по построению отчетов.
ExcelReport	Класс, который является типом для передачи Excel-данных. Содержит в себе название, создаваемого отчета в виде текстовых данных, а также ссылку на созданный отчет. Данный класс в качестве типа резуль-

	тата, возвращаемого сервисом CreateExcelReport

Таким образом, в данном разделе была построена модель предметной области будущего приложения посредством разработки диаграммы классов, кроме того, для упрощения восприятия данной диаграммы было дано краткое описание наполнения и поведения для каждого из классов.

2.3 Построение диаграммы последовательности приложения

Диаграммы последовательности (sequence diagram) являются видом диаграмм взаимодействия языка UML, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования.

Выполненная диаграмма последовательностей представлена на рис. 2.4:

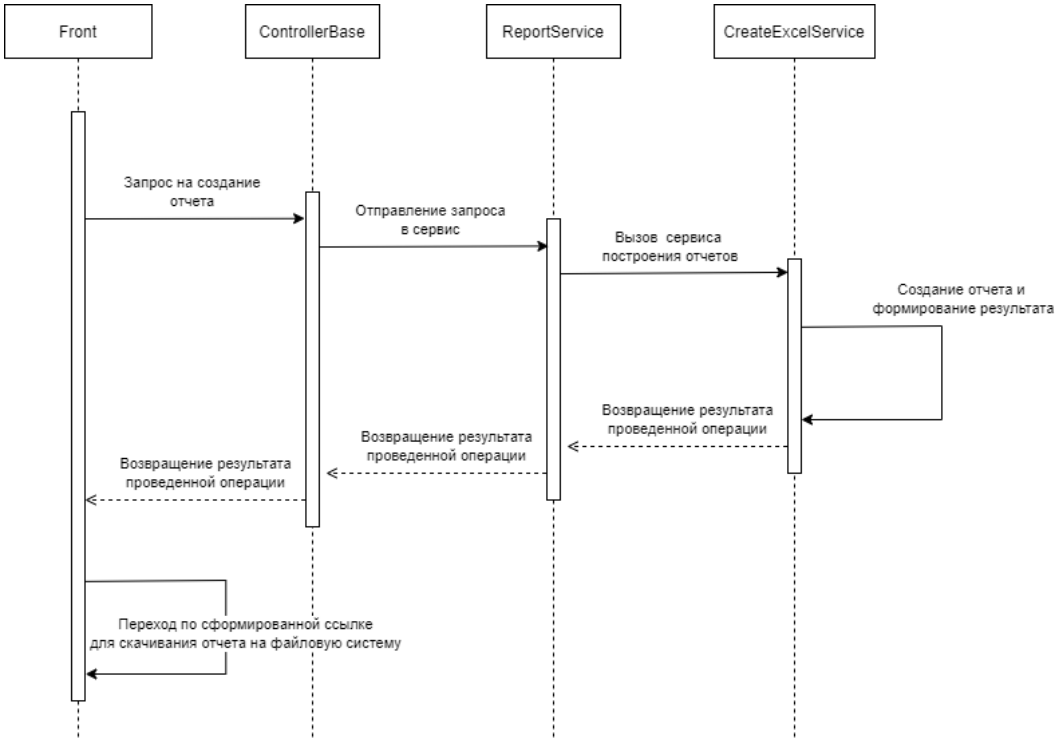

Рисунок 2.4 – Диаграмма последовательности

На рисунке 2.4 изображена диаграмма последовательности, на которой продемонстрирован процесс взаимодействия класса-контроллера с классами сервисами при осуществлении запроса на создание отчета.

Из диаграммы видно, что инициатором осуществления процессов является пользователь, использующий веб-интерфейс. Кроме того, видно, что запрос обрабатывается тремя разными классами. Это сделано для того, чтобы отделить бизнес-логику от технической реализации создания отчета. После получения результата работы программы происходит автоматическое скачивание файла.

Таким образом, в результате построения диаграммы последовательности разработаны подробные описания процессов внутри приложения, которые будут отражать функциональную ценность информационной системы.

2.4 Расчет функциональных и размерно-ориентированных метрик для разрабатываемого приложения.

Для количественной оценки разрабатываемого приложения используют LOC-оценку. LOC-оценка используется для измерения объема программного обеспечения путем подсчета количества строк исходного кода.

Цель этой деятельности - сформировать предварительные оценки, которые позволят:

- предъявить заказчику корректные требования по стоимости и затратам на разработку программного продукта;
- составить план программного проекта.

В оценке стоимости ПО используются 2-е единицы оценки: функциональная точка Function Point (FP) и строка кода Line of Code (LOC).

При расчете FP и LOC оценок необходимо выполнить анализ предварительной трудоемкости и стоимости разработки программы.

К достоинствам функционально-ориентированных метрик относят:

1 Не зависят от языка программирования;

2 Легко вычисляются на любой стадии проекта.

К недостаткам функционально-ориентированных метрик относят:

1 Результаты, основанные на субъективных данных, используются не прямые, а косвенные измерения;

2 FP-оценки легко пересчитать в LOC-оценки. Результаты пересчета зависят от языка программирования, используемого для реализации ПО.

Используется 5 информационных характеристик описание которых представлено ниже, а их оценка в таблице 1.

1 Количество внешних вводов. Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные. Вводы должны быть отделены от запросов, которые подсчитываются отдельно.

2 Количество внешних выводов. Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются.

3 Количество внешних запросов. Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы - каждый учитывается отдельно.

4 Количество внутренних логических файлов. Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом).

5 Количество внешних интерфейсных файлов. Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение.

6 Каждой из выявленных характеристик ставится в соответствие сложность. Для этого характеристике назначается низкий, средний или высокий ранг, а затем формируется числовая оценка ранга.

Данные для определения ранга и оценки сложности транзакций и файлов приведены в таблицах:

Таблица 2.2 – Ранг и оценка сложности внешних вводов

Ссылки на файлы	Элементы данных		
	1-4	5-15	>15
0-1	Низкий (3)	Низкий (3)	Средний (4)
2	Низкий (3)	Средний (4)	Высокий (6)
>2	Средний (4)	Высокий (6)	Высокий (6)

Таблица 2.3 – Ранг и оценка сложности внешних выводов

Ссылки на файлы	Элементы данных		
	1-4	5-19	>19
0-1	Низкий (4)	Низкий (4)	Средний (5)
2-3	Низкий (4)	Средний (5)	Высокий (7)
>3	Средний (5)	Высокий (7)	Высокий (7)

Таблица 2.4 – Ранг и оценка сложности внешних запросов

Ссылки на файлы	Элементы данных		
	1-4	5-19	>19
0-1	Низкий (3)	Низкий (3)	Средний (4)
2-3	Низкий (3)	Средний (4)	Высокий (6)
>3	Средний (4)	Высокий (6)	Высокий (6)

Таблица 2.5 – Ранг и оценка сложности внутренних логических файлов

Типы элементов записей	Элементы данных		
	1-19	20-50	>50
1	Низкий (7)	Низкий (7)	Средний (10)
2-5	Низкий (7)	Средний (10)	Высокий (15)
>5	Средний (10)	Высокий (15)	Высокий (15)

Таблица 2.6 – Ранг и оценка сложности внешних интерфейсных файлов

Ссылки на файлы	Элементы данных		
	1-19	20-50	>50
1	Низкий (5)	Низкий (5)	Средний (7)
2-5	Низкий (5)	Средний (7)	Высокий (10)
>5	Средний (7)	Высокий (10)	Высокий (10)

Таблица 2.7 – Исходные данные для расчета FP-оценки

Имя характеристики	Ранг, количество, сложность			
	Низкий	Средний	Высокий	Итого
Внешние вводы	0*3=0	0*4=20	10*6=0	60
Внешние выводы	0*4=0	0*5=0	10*7=70	70
Внешние запросы	1*3=3	0*4=0	0*6=0	3
Внутренние логические файлы	0*7=0	35*10=350	0*15=0	350
Внешние интерфейсные файлы	0*5=0	0*7=0	0*10=0	0
Всего				483

Кол-во функциональных указателей вычисляется по формуле 2.1:

$$FP = S \times (0{,}65 + 0{,}01 \times \sum_{i=1}^{14} F_i) \quad (2.1)$$

где F_i – коэффициенты регулировки сложности в диапазоне:

– 0 – не влияет;

– 1 – случайное влияние;

– 2 – небольшое влияние;

– 3 – среднее влияние;

– 4 – важное влияние;

– 5 – основное влияние;

Значение коэффициентов регулировки сложности представлены в таблице 2.8.

Таблица 2.8 – Значение коэффициентов регулировки сложности

№	Системный параметр	Описание	Коэффициент
1	Передача данных	Сколько средств связи требуется для передачи или обмена информацией, или с приложением или системой?	4
2	Распределенная обработка данных	Как обрабатываются распределенные данные и функции обработки?	3
3	Производительность	Нуждается пользователь в фиксации времени ответа или производительности?	4
4	Распространённость используемой конфигурации	Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение?	4
5	Скорость транзакций	Как часто выполняются транзакции? (каждый день, каждую неделю, каждый	4

		месяц)	
6	Оперативный ввода данных	Какой процент информации надо вводить в режиме онлайн?	3
7	Эффективность работы конечного пользователя	Приложение проектировалось для обеспечения эффективной работы конечного пользователя?	5
8	Оперативное обновление	Как много внутренних файлов обновляется в онлайновой транзакции?	4
9	Сложность обработки	Выполняет ли приложение интенсивную логическую или математическую обработку?	1
10	Повторная используемость	Приложение разрабатывалось для удовлетворения требований одного или многих пользователей?	1
11	Легкость инсталляции	Насколько трудны преобразование и инсталляция приложения?	1
12	Легкость эксплуатации	Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления?	3
13	Разнообразные условия размещения	Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций?	4
14	Простота изменений	Была ли спроектирована, разработана и поддержана в приложении простота изменений?	3
Всего			44

Теперь можно рассчитать количество функциональных указателей:

$$FP = 483 \times (0{,}65 + 0{,}01 \times 44) = 526{,}47$$

Так как это количество функциональных указателей, округляем его до ближайшего целого и получаем 527 функциональных точек.

Одним из преимуществ функциональной оценки является то, что при небольших манипуляциях ее можно преобразовать в размерно-ориентированную оценку. Для данного преобразования используются специальные таблицы конвертации (таблица 2.9). Для расчета LOC-оценки необходимо умножить значение из таблицы для используемого языка на количество функциональных метрик.

Line of Code (LOC) – это оценка ПО, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода. К пре-

имуществам использования LOC, как единицы размера ПО, относят простоту, а недостатками является следующее:

– размер проекта в LOC может быть определен только после его завершения;

– LOC зависит от языка программирования;

– LOC не учитывает качества кода.

Зная кол-во функциональных указателей, можем получить число строк кода. В языке C# одна функциональная точка примерно равна 53 строкам кода (Таблица 2.9).

Таблица 2.9 – Кол-во операторов на один FP

Язык программирования	**Кол-во операторов на один FP**
Assembler	320
C	128
Cobol	106
Fortran	106
Pascal	90
C++	64
LISP	64
Prolog	64
C#	53
Java	53
Kotlin	49
Visual Basic	32
Smalltalk	22
Perl	21

Формула для пересчета из FP в LOC:

LOC = FP * Количество Операторов

Получим, что число строк кода равно:

LOC = 527 * 53 = 27931 строк кода.

Таким образом в данном разделе были вычислены основные оценки проектируемой системы – размерно-ориентированные и функциональные, кроме того, были выявлены их преимущества и недостатки по сравнению с другими способами оценки разрабатываемого приложения.

2.5 Расчет модели оценки стоимости разработки мобильного приложения для визуализации положения звезд.

Очень важным показателем при разработке какого-либо программного продукта является время разработки системы. Для определения предварительного времени разработки используют ранее рассчитанную размерно-ориентированную оценку.

Constructive cost model (COCOMO – модель издержек разработки) – это алгоритмическая модель оценки стоимости разработки программного обеспечения.

Основное уравнение этой модели имеет вид (формула 2.2):

$$\text{Затраты} = A \times M_e \times \text{Размер}^B [\text{чел.} - \text{мес.}], \quad (2.2)$$

где:

– A = 2.5 – масштабный коэффициент.

– Размер – размер ПО выраженный в тыс. LOC;

– M_e – множитель поправки зависит от семи формирователей затрат, характеризирующих продукт, процесс и персонал (Таблица 2.5);

– Показатель степени B отражает нелинейную зависимость затрат от размера проекта (от длины кода LOC).

Значение показателя степени B измеряется в диапазоне от 1.01 до 1.26, зависит от 5-и масштабных факторов Wi и вычисляется по формуле 2.3:

$$B = 1{,}01 + 0{,}01 \sum_{i=1}^{5} W_i. \quad (2.3)$$

Общая характеристика масштабных факторов Wi приведена в таблице 2.10.

Таблица 2.10 – Характеристика масштабных факторов W_i

Масштабный фактор Wi	Пояснение	Wi
1 Предсказуемость, наличие прецедентов	Отражает предыдущий опыт организации в реализации проектов этого типа. Очень низкий (=5) означает отсутствие опыта. Сверхвысокий (=0) означает, что организация полностью знакома с этой прикладной областью	3-среднее
2 Гибкость разработки	Отражает степень гибкости процесса разработки. Очень низкий (=5) означает, что используется заданный процесс. Сверхвысокий (=0) означает, что клиент установил только общие цели	3-среднее
3 Разрешение рисков в архитектуре	Отражает степень выполняемого анализа риска. Очень низкий (=5) означает малый анализ. Сверхвысокий (=0) означает полный и сквозной анализ риска	4-среднее
4 Связность группы	Отражает, насколько хорошо разработчики группы знают друг друга, и насколько удачно они совместно работают. Очень низкий (=5) означает очень трудные взаимодействия. Сверхвысокий (=0) означает интегрированную группу без проблем взаимодействия	1-очень высокий
5 Зрелость процесса	Означает зрелость процесса в организации. Вычисление этого фактора может выполняться по вопроснику CMM	3-среднее
Всего		14

Из таблицы 2.10 следует что численное значение показателя B = 1,15.

Множитель поправки M_e зависит от набора формирователей затрат EM_i, перечисленных в таблице 2.11.

Таблица 2.11 – Формирователи затрат для раннего этапа проектирования

Обозначение	Название	EMi
PERS	Возможности (способности) персонала	Среднее=1
RCPX	Надежность и сложность продукта	Среднее=1
RUSE	Требуемое повторное использование	Среднее=1
PDIF	Трудность (сложность) платформы	Среднее=1
PREX	Опытность персонала	Среднее=1
FCIL	Средства поддержки	Среднее=1
SCED	Сроки	Среднее=1

Для расчета множителя поправки, зависящего от 7-и формирователей затрат, характеризующих продукт, процесс и персонал, воспользуемся формулой 2.4:

$$M_e = \prod_{i=1}^{7} EM_i. \quad (2.4)$$

Из формулы 2.4 следует что множитель поправки равен 1s.

Для расчета затрат на разрабатываемый продукт, необходимо воспользоваться формулой 2.2. В итоге получим:

$$\text{Затраты} = 2.5 \times 1 \times 2.6^{1.15} = 7.5[\text{чел.} - \text{мес.}].$$

В итоге проведённые расчетов можно сделать вывод что, один человек может выполнить поставленную задачу за 7,5 месяцев.

Таким образом в данном разделе при помощи COCOMO-модели издержек разработки и вычисленной ранее размерно-ориентированной оценки были выявлены предполагаемые сроки разработки микросервисного приложения для организации работы контакт-центра.

2.6 Вывод по второму разделу

В данном разделе был выполнен аналитический обзор требований при разработке микросервисного приложения для организации работы контакт-центра на основе диаграммы прецедентов, который заключался в определении главных ролей, актеров, прецедентов и их взаимодействии, а также в построении самой диаграммы прецедентов.

Был выполнен анализ структурных принципов проектирования приложения и модели предметной микросервисного приложения для организации работы контакт-центра, на основе которых, были построены диаграммы классов и последовательности, и определены главные функциональные задачи. Осуществлены расчеты функционально- и размерно-ориентированных метрик для микросервисного приложения контакт-центр (FP, LOC, COCOMO).