**АННОТАЦИЯ**

Данная выпуская квалификационная работа состоит из 66 страниц, 52 рисунка и 21 источник.

Ключевые слова: телемаркетинг, автоматизация, разработка, серверное приложение.

Цель выпускной квалификационной работы - разработка системы для автоматизации процесса телемаркетинга.

В современном мире, информационные технологии развиваются стремительно, их применение в финансовой сфере имеет первостепенное значение. Финансовым организациям необходимо продавать свои продукты и услуги в условиях растущей конкуренции, поэтому необходимо использовать содержательные и заманчивые предложения, которые могли бы убедить потребителей воспользоваться данными товарами и услугами.

Одним из основных способов продвижения является телемаркетинг. Телемаркетинг — это способ продвижения товаров и услуг, при котором осуществляется коммуникация с потенциальными клиентами через телефонные звонки с целью продвижения продукции, получение обратной связи и проведения опросов.

В результате выполнения данной работы было разработано серверное приложение, которое реализует поставленную в данной работе цель.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7

1.1 Что такое телемаркетинг: цели, принципы 7

1.2 Телемаркетинг в финансовой сфере. 8

1.3 Специфика ручного телемаркетинга внутри платежной системы «Золотая Корона» 9

1.4 Обзор существующих решений 11

1.4.1 1С:CRM 11

1.4.2 MegaCRM 14

1.5 Формирование требований к разрабатываемой системе 19

ГЛАВА 2. Проектирование системы «Телемаркетинг» 21

2.1 Клиент-серверное взаимодействие 21

2.2 Выбор инструментальных средств 22

2.2.1 Язык программирования Java 22

2.2.2 Автоматизированная сборка проектов с помощью Apache Maven 23

2.2.3 Фреймворк Spring 23

2.2.4 Объектно-реляционный фреймворк Hibernate 25

2.2.5 Система управления базами данных PostgreSQL 26

2.2.6 Брокер сообщений Apache Kafka 27

2.3 Сторонние сервисы 28

2.3.1 Сервис черных списков «BlackList» 28

2.3.2 Сервис получения клиентских данных «Casper» 29

2.3.3 Сервис получения набора доступных продуктов для клиента «Abram» 31

2.3.4 Сервис телефонии MightyCall 31

2.4 Микросервисная архитектура 32

2.5 Предметно-ориентированное проектирование сервисов. 33

2.5.1 Домен «Займы» 34

2.5.2 Архитектурный слой «Ядро» 34

2.5.3 Архитектурный слой «Интеграции» 35

2.6 Архитектура системы «Телемаркетинг» 36

2.6.1 Сервис обработки входящего потока данных «tml-event-generator» 37

2.6.2 Сервис фильтрации входящего потока данных «tml-events-analyser» 38

2.6.3 Сервис управления кампаниями «tm-campaign-engine» 40

2.6.4 Сервис аутентификации и авторизации tm-security-service 41

2.6.4.1 Аутентификация 42

2.6.4.2 Авторизация 43

2.6.4.3 Выход из системы 45

2.6.5 Автоматизированное рабочее места оператора 45

2.6.6 Адаптер сервиса телефонии tm-mightycall-adapter 48

2.6.6.1 Открытие сессии оператора 48

2.6.6.2 Закрытие сессии оператора 49

2.6.6.3 Процесс опроса событий для оператора 50

2.6.6.4 Работа с очередью задач в MightyCall 51

2.7 Проектирование структуры хранения данных 52

ГЛАВА 3 Описание и тестирование готовой системы 58

ЗАКЛЮЧЕНИЕ 63

СПИСОК ЛИТЕРАТУРЫ 64

ПРИЛОЖЕНИЯ 66

# **ВВЕДЕНИЕ**

Финансовым организациям необходимо продавать свои продукты и услуги в условиях растущей конкуренции. Стремительное развитие информационных технологий в финансовой сфере повысило уровень сбора информации о клиентах. В настоящее время для общения с потребителем все чаще используется телемаркетинг.

Телемаркетинг — это способ продвижения товаров и услуг, при котором осуществляется коммуникация с потенциальными клиентами через телефонные звонки с целью продвижения продукции, получения обратной связи и проведения опросов.

Система телемаркетинга в финансовой организации, благодаря прямому контакту с потенциальными клиентами, помогает увеличивать продажи, улучшать качество обслуживания клиентов, собирать и анализировать данные о клиентах. Система автоматизирует процесс обзвона клиентов, что позволяет сократить время и ресурсы, затрачиваемые на контакт с каждым клиентом. Поэтому разработка сервиса телемаркетинга является актуальной задачей.

В данной выпускной квалификационной работе будет рассматриваться разработка сервиса телемаркетинга на базе платежной системы «Золотая Корона» группы компаний «Центр Финансовых Технологий».

В компании уже есть ручной процесс телемаркетинга, который необходимо автоматизировать. Сложность заключается в том, что готовые системы от сторонних разработчиков не подходят для интеграции с существующим процессом, так как в нем используются закрытые сервисы внутри ГК «ЦФТ». Поэтому при внедрении существующих решений для построения телемаркетинга могут возникнуть трудности, из-за которых не получится в полной мере автоматизировать процесс не перестраивая его.

В результате выполнения данной работы будет спроектировано и разработано серверное приложение «Телемаркетинг» для платежной системы «Золотая Корона» группы компаний «Центр Финансовых Технологий».

Также выполнение проекта подразумевает выбор инструментов для разработки, языка программирования, фреймворков, базы данных.

# **ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## Что такое телемаркетинг: цели, принципы

Телемаркетинг (или telemarketing) представляет собой использование телефона и современных средств связи для ведения коммуникаций и продаж без посредников. Этот метод хоть и один из старшейших, но при этом остается самым эффективным для увеличения объема продаж, изучения целевой аудитории, привлечения новых клиентов и повышения узнаваемости бренда на рынке [1].

Предпосылки возникновения телемаркетинга появились во второй половине XIX века с изобретением телефона. Первое зарегистрированное использование телефона для продвижения товаров относится к 1909 году. В то время телемаркетинг в основном использовался для продажи подписки на газеты и журналы. Только в 1960-х годах телемаркетинг превратился в полноценную возможность для бизнеса [2].

В 1965 году AT&T создала первую систему телемаркетинга как способ охвата клиентов. Со временем достижения в области коммуникационных технологий и деловой практики позволили телемаркетингу стать мощным инструментом в мире продаж и маркетинга. Сегодня это один из самых популярных методов работы с клиентами, позволяющий предприятиям устанавливать контакты с большой клиентской базой экономически эффективным способом [2].

Кроме того, широкое использование баз данных и автоматизированной рассылки звонков привело к улучшению обслуживания клиентов [2]. Это увеличило охват клиентов и дало компаниям преимущество в привлечении и удержании клиентов. Телемаркетинг продолжает оставаться важной частью успешной маркетинговой стратегии для предприятий любого размера.

Телемаркетинг делится на два вида:

• Входящий - прием звонков от клиентов [2];

• Исходящий - обзвон пользователей операторами [2].

В данной выпускной квалификационной работе будет рассматриваться исходящий вид телемаркетинга.

## Телемаркетинг в финансовой сфере.

Когда дело доходит до финансовых услуг — будь то страхование, управление капиталом или кредитование, — ничто не может заменить индивидуальный подход.

Телемаркетинг предоставляет возможность прямого контакта с клиентом, что дает компании целый ряд преимуществ в ее продвижении.

Одним из основных плюсов является мгновенная обратная связь и общение с клиентом в режиме реального времени. Никакой другой инструмент маркетинга не обеспечивает такую быструю возможность понимания настроений клиента и оказания влияния на него в пользу бизнеса.

При использовании этой тактики компания может взаимодействовать с потенциальными клиентами и создавать списки новых потенциальных контактов.

Главный плюс использования телемаркетинга для продвижения финансовых продуктов заключается в том, что он позволяет Вам мгновенно оценить, насколько интересен клиенту Ваш сервис или продукт [3]. Более того, он позволяет:

* **удерживать клиентов**: телемаркетинг может быть эффективным инструментом для связи с текущими клиентами и информирования их о новых услугах или продуктах, которые могут их заинтересовать, тем самым повышая уровень удержания [3];
* **повышать спрос:** финансовые консультанты могут использовать телемаркетинг для привлечения "холодных" потенциальных клиентов или отслеживания "теплых" потенциальных клиентов [3];
* **проводить маркетинговые исследования:** можно проводить быстрые опросы или интервью, чтобы понять потребности рынка или собрать отзывы о новых продуктах или услугах [3].

## Специфика ручного телемаркетинга внутри платежной системы «Золотая Корона»

В рамках данной работы разработка системы телемаркетинга осуществляется на базе платежной системы «Золотая Корона», поэтому необходимо учитывать специфику и потребности этого бизнеса. Предлагаю рассмотреть, как сейчас вручную работает данный процесс.

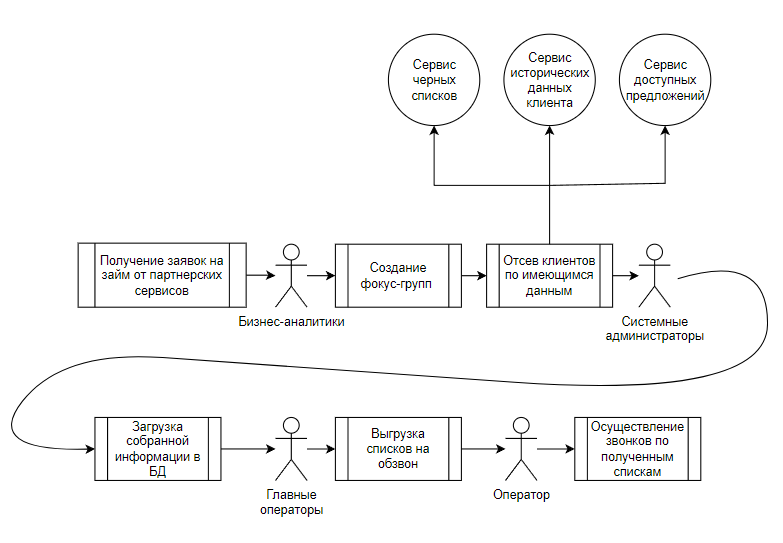


Рисунок 1 - Ручной бизнес-процесс телемаркетинга

Как видно на *рисунке 1*, в самом начале платежная система «Золотая Корона» получает заявки на займы от внешних партнерских систем. В качестве сервисов, из которых идет поток заявок, может быть мобильное приложение «KoronaPay» или другие сервисы-партнеры.

Далее бизнес-аналитики на основе этих заявок создают фокус-группы клиентов, которые обычно называют кампаниями, собирают данные из внутренних сервисов, чтобы определить, каким клиентам и на каких условиях «Золотая Корона» могла бы выдать финансовые продукты.

Потом идет отсев потенциальных клиентов с помощью полученных данных и уже имеющихся черных списков. Под данными в этом случае подразумевается: история коммуникаций, информация о клиенте, которая была получена во время оформления заявки на получение финансовых продуктов, и т.д.

Вся собранная информация хранится в виде Excel-документов, которые передаются «из рук в руки». Далее системные администраторы сгружают их в базы данных для дальнейшей обработки.

В самом конце супер-операторы снова вручную выгружают списки на обзвон в сервис телефонии. И только теперь обычные операторы могут совершать звонки по выбранным фокус-группам.

Результаты каждой коммуникации точно так же формируются оператором вручную в Excel-документах, которые отправляются в конце каждой смены и сгружаются системными администраторами в базу данных.

Как видно, в этой цепочке очень много человеческого труда, который можно было бы автоматизировать. Кроме этого, на каждом этапе находятся люди из разных команд, что усложняет процесс контроля и коммуникации между ними. С ростом нагрузки становится сложнее обрабатывать клиентские заявки, локализовывать проблемы и разбирать инциденты. Поэтому разработка сервиса или использование готового решения значительно упростило бы сопровождение этого бизнес-процесса.

## Обзор существующих решений

Перед тем как разрабатывать своё решение, необходимо проанализировать уже существующие системы, чтобы понять, можно ли использовать их с учетом процесса, описанного в п. 2.3.

Сейчас на рынке есть несколько основных решений для построения системы телемаркетинга. Предлагаю рассмотреть их.

### 1С:CRM

1С CRM Телемаркетинг — это специализированная программа, разработанная для автоматизации процессов телемаркетинга и управления взаимоотношениями с клиентами. Она предназначена для использования в компаниях, которые активно занимаются продажами по телефону [4]. Основные функции программы включают:

Управление базой данных клиентов: программа позволяет хранить информацию о клиентах, включая контактные данные, историю покупок, предпочтения и т.д. [5].

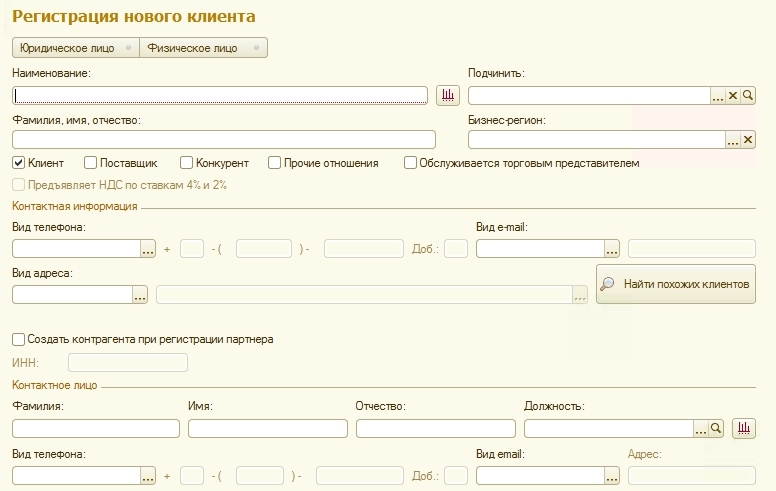


Рисунок 2 - Форма регистрации нового клиента

Планирование звонков: с помощью 1С CRM Телемаркетинг можно создавать расписание звонков, назначать ответственных за них сотрудников и отслеживать выполнение задач [4].

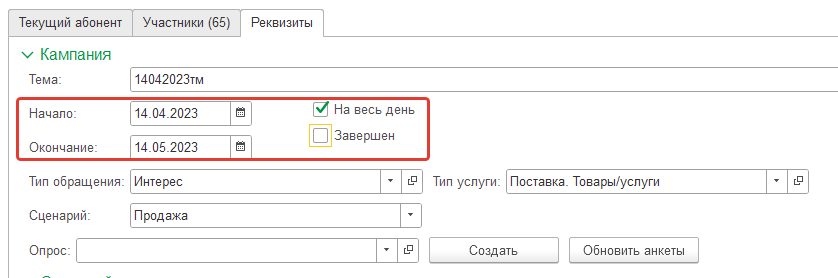


Рисунок 3 - Форма планирования звонков

Сама система 1С:CRM не предусматривает связи с телефонией — для этого существует продукт под названием 1С-Рарус: СофтФон [5].

1С-Рарус: СофтФон обеспечивает интеграцию с офисной АТС (телефонной станцией) и необходим для того, чтобы вы могли фиксировать исходящие и входящие звонки в системе 1С:CRM. Звонить с использованием СофтФона можно, но сама по себе его настройка — дело не тривиальное. СофтФон нельзя дорабатывать, его код закрыт и написан не на языке 1С [5].

СофтФон позволяет получать информацию о звонящем, делать исходящие звонки — то есть со своей минимальной задачей он справляется. Но все-таки то, что СофтФон — это отдельный продукт, ощущается в процессе работы. При поступлении звонка он не сразу попадает в 1С:CRM, а только при команде пользователя «Передать в 1С». Конечно, с одной стороны, это хорошо: сторонние звонки не попадут в систему, но есть опасность того, что и не все звонки клиентов попадут в базу [5].

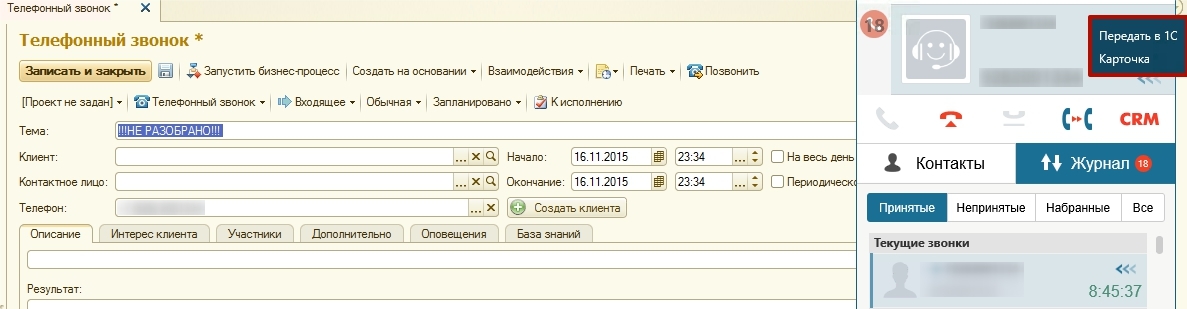


Рисунок 4 - Карточка телефонного звонка. Совместная работа «1С Телемаркетинг» и «1С-Рарус: СофтФон»

Карточка телефонного звонка предоставляет широкие возможности, например, такие, как учет трудозатрат, участников события, первичного спроса клиента. Пользоваться этими возможностями или ограничиться полем «Описание», где в свободной форме фиксируется результат звонка, — решать вам.

Однако, «1C Телемаркетинг» не имеет интеграций с внутренними системами ГК «ЦФТ», которые используются для получения кредитной истории, кредитного потенциала и т.д. Также приложение не подразумевает автоматическую обработку входящих заявок на обзвон, что не дает возможности полностью автоматизировать процесс телемаркетинга.

Резюмируя, хочется сказать, что система имеет богатый функционал. Но невозможность доработки решения для платежной системы «Золотая Корона» не дает полной автоматизации бизнес-процесса. Поэтому это решение не освобождает нас от ручного труда, который так хотелось бы автоматизировать.

### MegaCRM

MegaCRM — это программа, которая помогает компаниям управлять своими контактами и продажами. Она позволяет создавать и управлять списками контактов, отслеживать историю взаимодействия с клиентами, а также автоматизировать процесс продаж [6].

Одной из главных особенностей MegaCRM является возможность интеграции с другими программами и сервисами, такими как телефония, электронная почта и социальные сети. Это позволяет пользователям получать доступ к информации о клиентах из разных источников в одном месте [7].

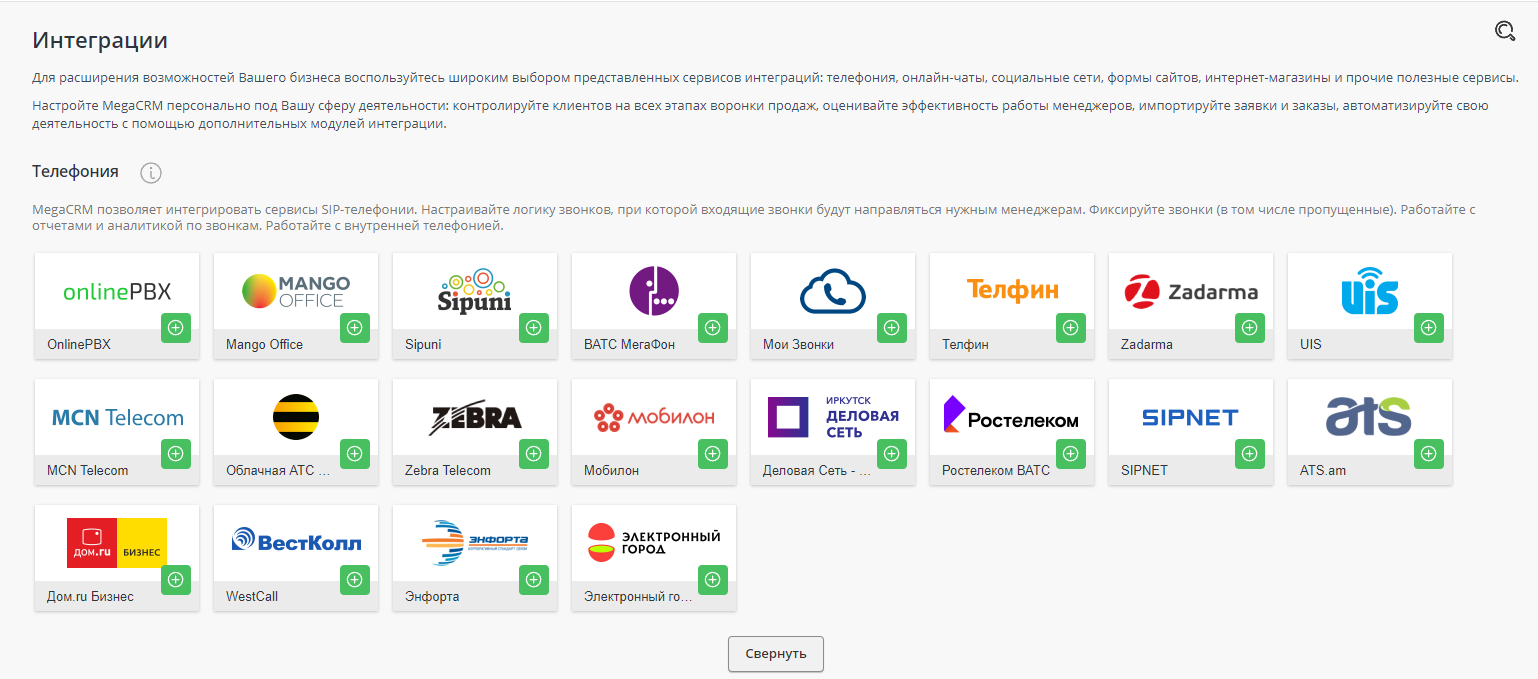


Рисунок 5 - Интеграция MegaCRM с телефоний

Программа также имеет функцию автоматического обзвона клиентов, что значительно упрощает процесс продаж. Пользователи могут создавать сценарии звонков и настраивать правила для автоматического обзвона [8].

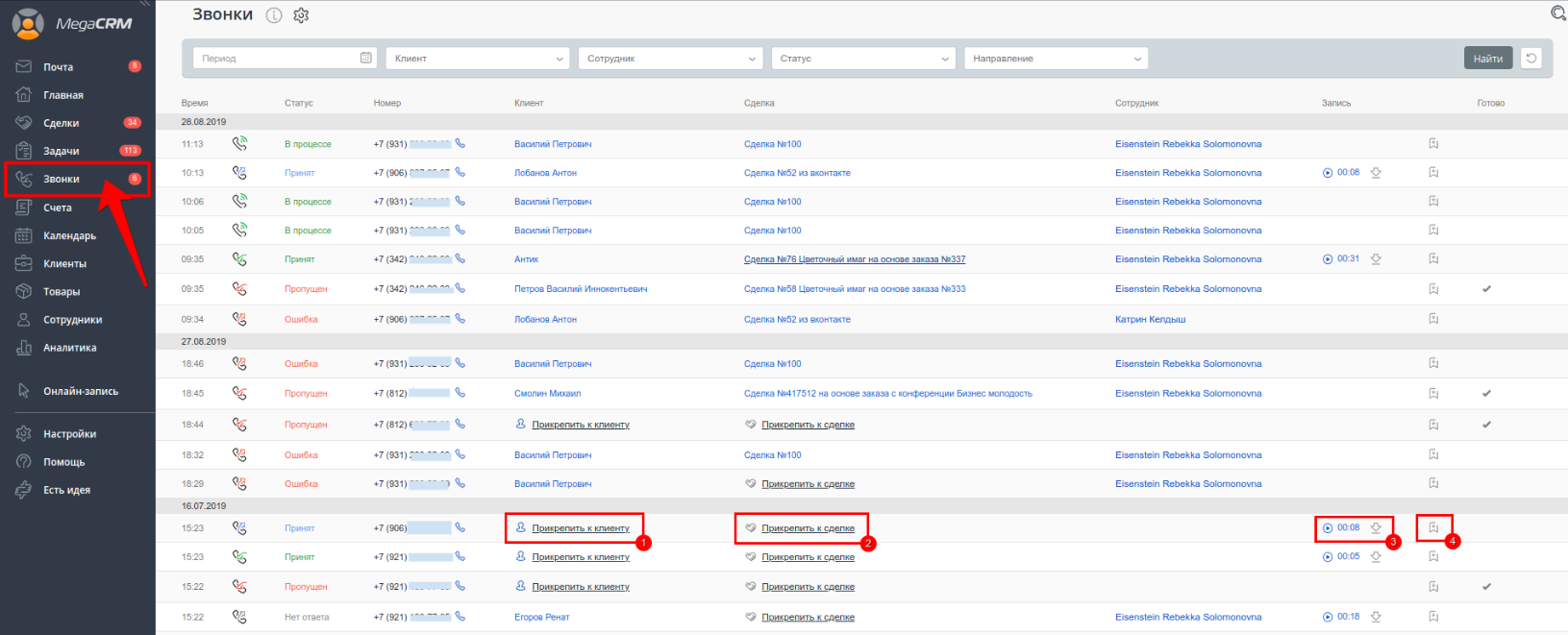


Рисунок 6 - Раздел обзвона клиентов

Карточка клиента представляет собой место, где хранится вся информация о клиенте и история коммуникаций с ним. Карточка клиента является ключевым инструментом для эффективного управления отношениями с клиентами и обеспечивает персонализированный подход в работе с каждым клиентом [8].

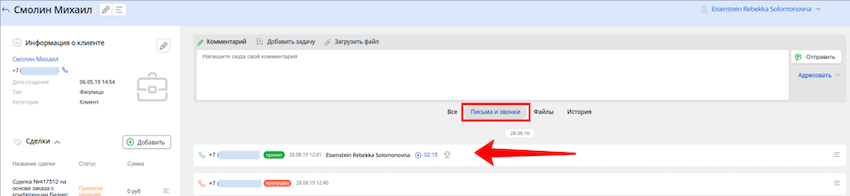


Рисунок 7 - Карточка клиента

Основные элементы, которые есть в карточке клиента:

* Контактная информация: основные данные о клиенте, такие как имя, контактный номер телефона, электронная почта и адрес.
* История взаимодействия: записи о предыдущих контактах и взаимодействии с клиентом, такие как звонки, встречи, электронные письма, чаты и т.д.
* Сделки и заказы: информация о текущих и прошлых сделках, заказах, счетах и договорах с клиентом.
* Интересы и предпочтения: информация о предпочтениях клиента, его интересах, покупках, предыдущих обращениях и любой другой информации, которая может помочь улучшить работу с клиентом.
* Аналитика и отчетность: статистика по взаимодействию с клиентом, конверсии, динамике продаж и другие ключевые показатели для анализа эффективности работы с клиентом.

Карточка клиента помогает операторам лучше понять потребности и предпочтения клиента, улучшить взаимодействие и обслуживание, а также эффективно вести процессы продаж. Благодаря централизованному хранению информации о клиенте, компания может значительно повысить эффективность работы с клиентской базой.

Кроме того, MegaCRM предоставляет возможность анализировать данные о продажах и клиентах. Пользователи могут создавать отчеты по различным параметрам, таким как количество звонков, время разговора и конверсия [9].

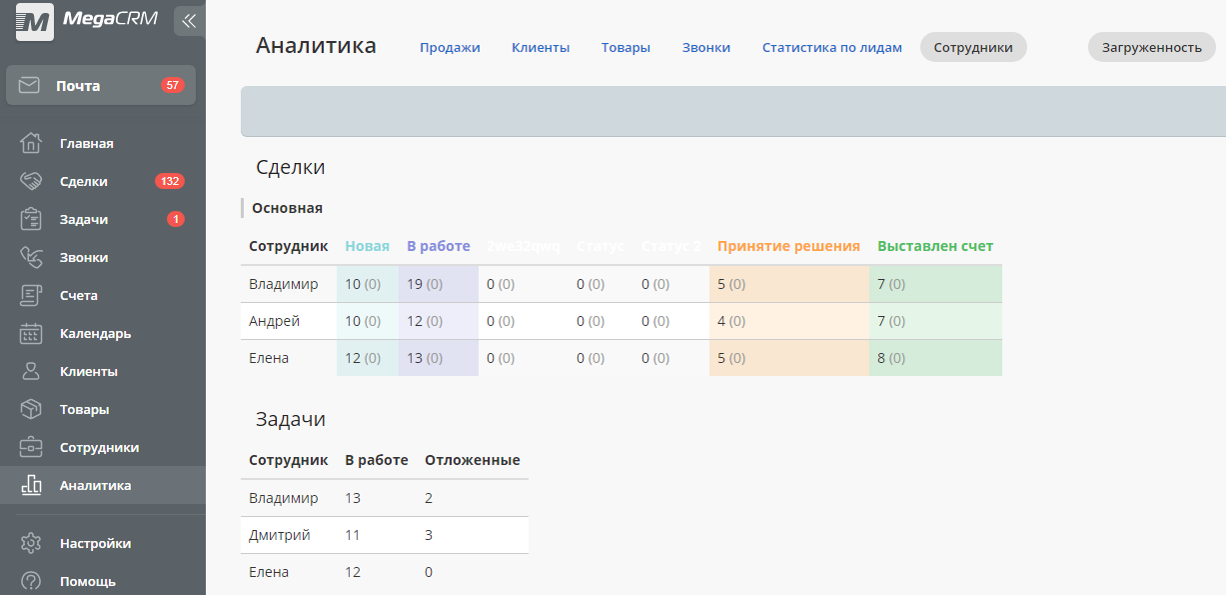


Рисунок 8 - Раздел «Аналитика», вкладка «Сотрудники»

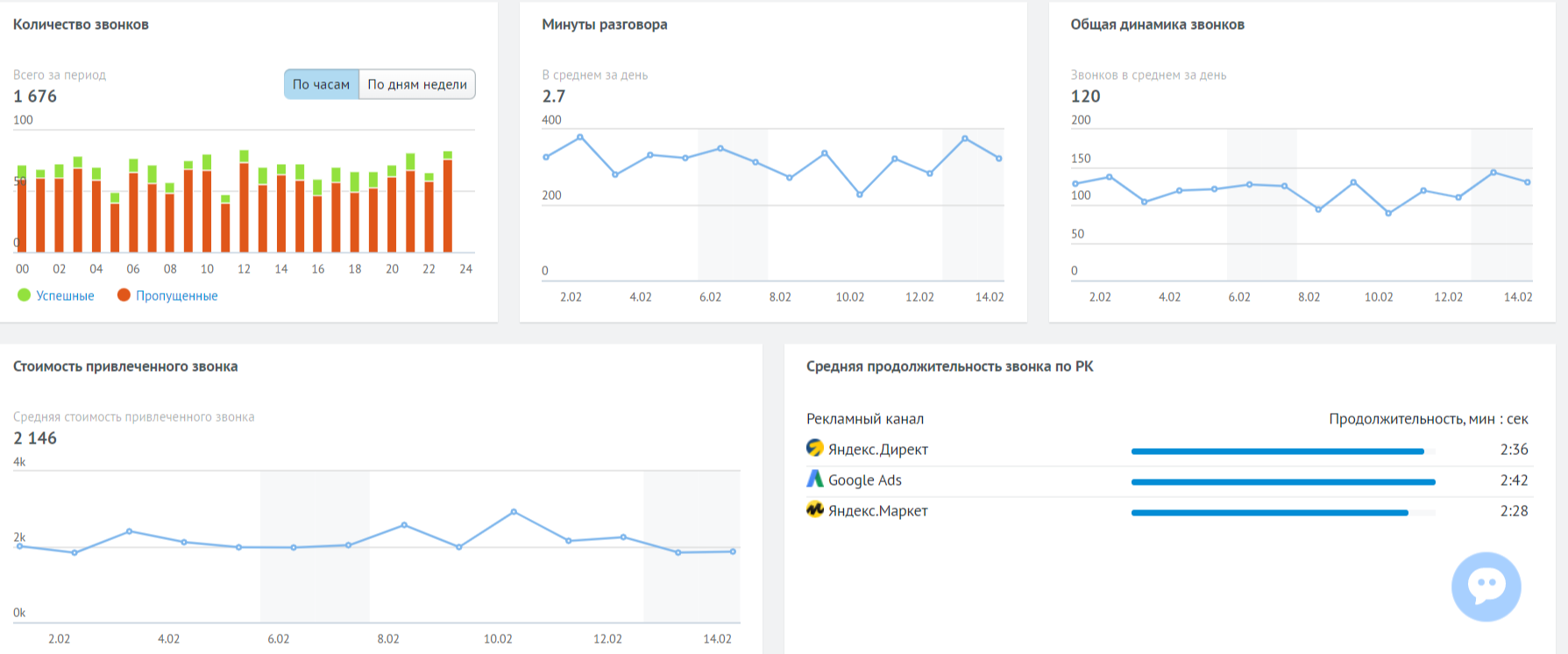


Рисунок 9 - Раздел «Аналитика», вкладка «Звонки»

В целом MegaCRM - это мощный инструмент для управления продажами и контактами. Он позволяет автоматизировать процесс продаж и упростить работу с клиентами.

Однако MegaCRM не имеет интеграций с внутренними системами ГК «ЦФТ», которые используются для получения кредитной истории, кредитного потенциала и т.д. Также приложение не подразумевает автоматическую обработку входящих заявок на обзвон, что не дает возможности полностью автоматизировать процесс телемаркетинга.

Кроме того, сложность настройки и необходимость обучения сотрудников при внедрении новой CRM-системы также могут вызвать определенные трудности. Ограничения в конфигурировании, безопасности, зависимость от стабильного интернет-соединения и возможные проблемы с интеграцией с другими инструментами также могут создать препятствия для эффективного использования MegaCRM в повседневной работе компании.

В результате проведенного анализа существующих систем я выявил проблему: среди имеющихся решений нет системы, которая бы могла бы полностью автоматизировать бизнес-процесс, описанный в п. 2.3., так как рассматриваемые решения не предусматривают интеграцию для получения данных от внешних сервисов, которыми пользуется компания.

Они также не являются решениями с открытым исходным кодом, который можно было бы доработать для полной автоматизации процесса внутри платежной системы «Золотая Корона».

## Формирование требований к разрабатываемой системе

В компании уже есть готовые сервисы, с помощью которых можно получить и отфильтровать данных клиента, а также разработанное клиентское приложение, поэтому нам необходимо сосредоточиться на разработке серверного приложения, которое бы могло взаимодействовать с ними.

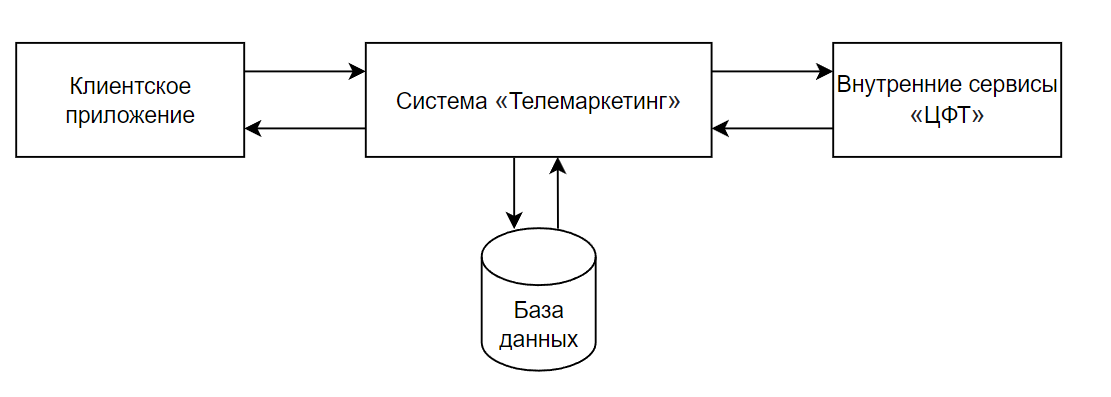


Рисунок 10 - Схема взаимодействия систем

Система «Телемаркетинг» предназначена для обработки поступающего потока клиентских заявок, фильтрации поступающего потока с помощью сторонних систем, сбора и подготовки клиентских данных, которые используются для осуществления коммуникации.

В ходе обзора существующих решений были выявлены их достоинства и недостатки, которые необходимо учесть в разрабатываемой системе, а именно:

1. интеграция с закрытыми сервисами ГЦ «ЦФТ»;
2. интеграция с телефонией;
3. автоматическая обработка входящего потока заявок;
4. обработка и подготовка данных о клиенте;
5. обработка результатов коммуникаций
6. удобное автоматизированное рабочее место оператора

# **ГЛАВА 2. Проектирование системы «Телемаркетинг»**

В данной главе будет описан процесс проектирования серверного приложения и выбор инструментов.



## Клиент-серверное взаимодействие

Взаимодействие клиента с программным интерфейсом приложения будет основано на использовании архитектурного стиля Representational State Transfer (сокращённо REST) [10].

Фактически REST API — это всего лишь набор конечных точек, обращаясь к которым с помощью HTTP-запросов, клиент получает в ответ информацию с сервера

REST API не принуждает использовать какой-то конкретный язык программирования. Клиентская и серверная части нашего приложения могут быть написаны на разных языках программирования. Для передачи информации можно использовать JSON, но поддерживаются и другие формат данных [10].

Ниже, на рисунке 11, представлено изображение взаимодействия компонентов в клиент-серверной архитектуре.

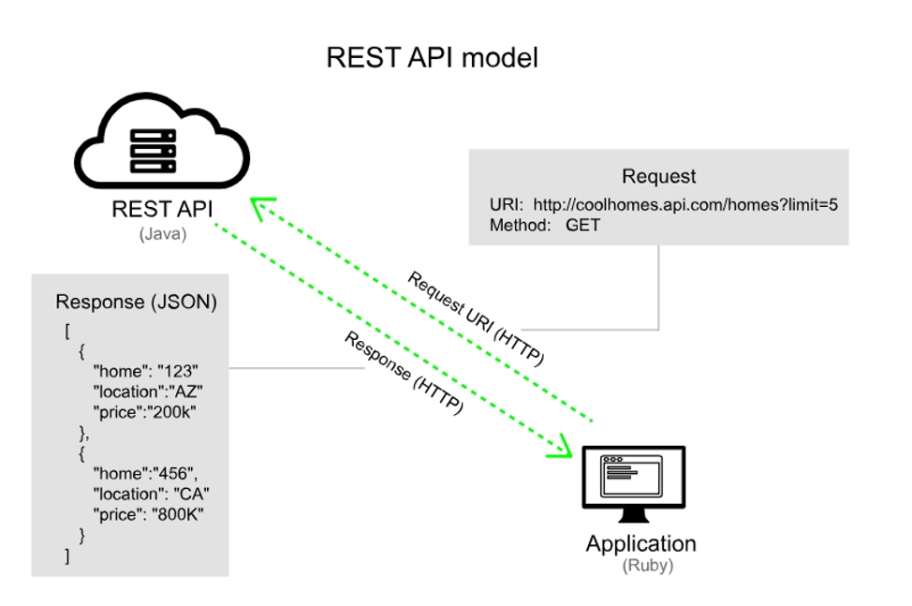


Рисунок 11 - Схема клиент-серверного взаимодействия

## Выбор инструментальных средств

Для решения поставленных задач необходимо выбрать технологии,

средства разработки и библиотеки, позволяющие реализовать систему «Телемаркетинг».

### Язык программирования Java

Java популярный язык программирования, используемый уже более 20 лет. Миллионы приложений написаны на Java и используются по сей день. Он может выступать как платформа сам по себе. Java быстрый, безопасный и надежный язык, подходящий для широкого спектра приложений, от мобильных до корпоративных, от обработки больших данных до серверных технологий [11].

Java выделяется среди других языков программирования по нескольким причинам:

* Синтаксис, похожий на С, что делает его понятным для многих программистов, а также облегчает изучение языка новичками.
* Возможности объектно-ориентированного программирования, позволяющие создавать гибкие системы, способные легко взаимодействовать с новыми компонентами без изменений в коде.
* Статическая типизация, которая помогает избежать ошибок на ранних этапах разработки путем явного определения типов переменных.
* Java ориентирована на Java Virtual Machine (JVM), что позволяет запускать Java-код на различных устройствах и операционных системах. Это также обеспечивает обратную совместимость, позволяя разработчикам использовать старый код с новыми версиями JVM без изменений.

### Автоматизированная сборка проектов с помощью Apache Maven

Apache Maven это инструмент для автоматизации сборки проектов, который широко используется Java-разработчиками, но также имеет плагины для интеграции с другими языками, такими как C/C++, Ruby, Scala, PHP и другие [12].

Для простых проектов на Java можно воспользоваться командной строкой для сборки. Однако при разработке более сложного ПО, требующего сторонних библиотек и ресурсов, процесс сборки становится сложнее. Здесь на помощь приходит Maven, созданный для упрощения этой задачи.

Одной из ключевых особенностей Maven является декларативное описание проекта. Это означает, что разработчику не нужно детально настраивать каждый аспект сборки, так как все необходимые параметры уже установлены по умолчанию. Если нужно отклониться от стандартных настроек, изменения требуется внести в нужные параметры конфигурации.

Еще одно преимущество Maven гибкое управление зависимостями. Он может автоматически загружать сторонние библиотеки в локальный репозиторий, выбирать нужные версии пакетов и обрабатывать транзитивные зависимости.

Также стоит отметить, что Maven независим от операционной системы. В то время как при работе из командной строки параметры могут зависеть от платформы, Maven позволяет не обращать внимания на этот аспект.

В случае необходимости систему сборки можно настраивать под индивидуальные потребности, используя готовые плагины или создавая собственные, если подходящего решения не нашлось.

### Фреймворк Spring

Spring Framework – универсальный фреймворк с открытым исходным кодом для JVM-платформы [13]. Фреймворк — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Spring Framework — это большой набор разных мини-фреймворков. Каждый из них нужен для работы над определёнными приложениями или их частями [13].

Основными компонентами являются:

* Inversion of Control (Ioc) – этот модуль лежит в основе Spring и контролирует взаимодействие разных компонентов между собой. С помощью специальных аннотаций можно описать зависимости между компонентами, соединяя их в единую архитектуру. В итоге компоненты работают сами по себе, а связями между ними управляет специальный контейнер. Это позволяет даже в случае изменения компонентов сохранять работоспособность всей системы [13];
* Aspect Oriented Programming (AOP)-модуль - позволяет реализовать не совсем стандартную парадигму — аспектно-ориентированное программирование. С его помощью программу можно сделать более выразительной и чёткой за счёт разделения на отдельно функционирующие части с разными зонами ответственности. АОП позволяет неявно работать с разными функциями системы так, что они не могут помешать работе друг друга. Это улучшает архитектуру приложения и позволяет программисту не отвлекаться от бизнес-задач на рутину [13];
* Модуль Model View Controller (MVC) - этот модуль реализует популярную схему веб-приложений — разделение её на три части. Первая часть отвечает за данные, вторая — за отображение интерфейсов, третья — за изменение данных в ответ на действия пользователя [13].

В итоге, основной задачей Spring Framework является упрощение и сокращение работы программиста при создании Java-приложения, благодаря большому набору готовых решений, которые можно использовать.

### Объектно-реляционный фреймворк Hibernate

Hibernate Framework — это фреймворк для языка Java, предназначенный для работы с базами данных. Он реализует объектно-реляционную модель — технологию, которая «соединяет» программные сущности и соответствующие записи в базе [14].

Объектно-реляционная модель, или ORM (Object-Relational Mapping), позволяет создать программную «виртуальную» базу данных из объектов, описанных с применением принципов ООП.

Hibernate — это инструмент, который работает по принципу ORM. Он основан на особом наборе правил — спецификации JPA 2.1. Она описывает, как именно можно сохранять данные из Java-кода в базу данных, однако это только теоретические правила. Hibernate превращает теорию в практику и позволяет загружать объекты из кода Java в базы данных напрямую [14].

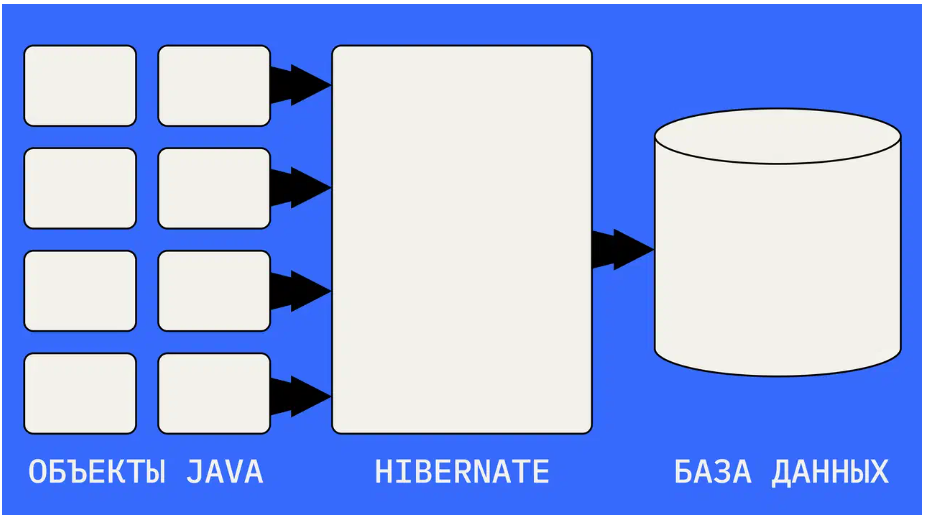


Рисунок 12 - Схема отображения Java – объектов в записи базы данных с помощью фреймворка Hibernate

Для работы с БД необязательно использовать Hibernate, можно писать весь код стандартными средствами Java. Однако фреймворк значительно упрощает работу.

Hibernate совместим с различными СУБД, такими как MySQL, PostgreSQL, Oracle, Microsoft SQL Server и другими. Благодаря общим принципам работы с базами, он подходит для большинства реляционных баз данных [14].

### Система управления базами данных PostgreSQL

PostgreSQL - это объектно-реляционная система управления базами данных (СУБД) с широким спектром применения. С помощью неё можно создавать устойчивые транзакционные базы данных. БД этого класса используются в финансах, веб-разработке и т.д. [15].

Это бесплатный продукт с лицензией свободного распространения. Вне зависимости от того, используете вы ее в коммерческих целях или нет, СУБД не потребует денег. Есть различные платные редакции от сторонних производителей, которые вносят дополнительные изменения в исходный код, но они опциональны.

PostgreSQL подходит для работы на всех популярных операционных системах (ОС): Windows, Linux, macOS. Пользователь получается готовую систему «из коробки» - чтобы установить и использовать программу, не нужны дополнительные инструменты [15].

PostgreSQL поддерживает много разных типов и структур данных, в том числе сетевые адреса, данные в текстовом формате JSON, геометрические данные для координат геопозиций, XML и др. Все эти форматы можно хранить и обрабатывать в СУБД. Также можно создавать собственные типы данных, их называют пользовательскими. Пользовательские типы данных нужны, чтобы упростить работу с БД или установить ограничения [15].

ACID — это набор требований, который обеспечивает сохранность данных. Аббревиатура расшифровывается как atomicity, consistency, isolation, durability, то есть атомарность, согласованность, изолированность, устойчивость. Это четыре требования для надёжной работы систем, которые обрабатывают данные в режиме реального времени. Если все требования выполняются, данные не будут теряться из-за технических ошибок или сбоев при работе оборудования. PostgreSQL соответствует всем четырём требованиям ACID и обеспечивает сохранность данных при выполнении транзакций и других работ [15].

Исходя из всех вышеперечисленных преимуществ, было решено в качестве системы управления базами данных выбрать PostgreSQL.

### Брокер сообщений Apache Kafka

Современные серверные приложения сложны и включают множество модулей, которые должны взаимодействовать друг с другом. Для этой цели служат системы обмена сообщениями (брокеры сообщений). Это разветвленная система, которая связываем модули друг с другом. Хорошо построенная система сообщений позволяет сервисам ставить друг другу задачи, сообщать об изменениях в системе и уведомлять заинтересованные части логики приложения о своих состояниях.

Apache Kafka – это система обмена сообщениями с открытым исходным кодом. Хранение и пересылка сообщений идет параллельно, что обеспечивает скорость и надежность. Записи в Kafka хранятся в виде журнала сообщений, который выглядит как очередь, в которую можно добавлять сообщения, но нельзя удалять или модифицировать. Данный подход дает большую надежность и простоту отслеживания изменений [16].



Рисунок 13 - Схема обмена сообщениями между модулями с помощью брокера сообщений

## Сторонние сервисы

Система «Телемаркетинг» должна уметь взаимодействовать с внешними сервисами, с помощью которых можно осуществлять фильтрацию клиентов и проводить звонки.

### Сервис черных списков «BlackList»

Черные списки – сервис для блокировки нежелательных клиентов на определенный период или бессрочно. Под блокировкой понимается внесение данных клиента с причиной в базу сервиса. При последующем обращении к сервису с запросом на поиск клиента возвращается запись (или список записей), на основании чего сторонние сервисы принимают решение по клиентку.

В функционал сервиса входит запись данных клиента, обновление, удаление, а также поиск записей по одному или нескольким параметрам.

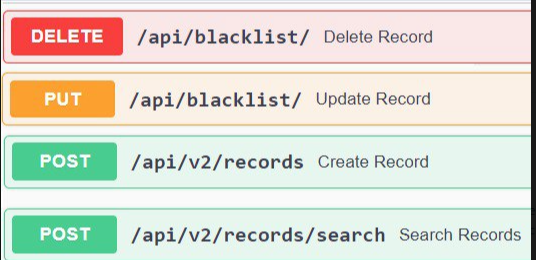


Рисунок 14 - Интерфейс сервиса «BlackList»

Назначение данного сервиса в системе «Телемаркетинг» – не пропускать клиента, если он находится в общем черном списке.

Поиск представляет из себя сложную структуру, но такой подход дает гибкие возможности для поиска. Чем-то запрос напоминает язык SQL, где с помощью логических операций осуществляется выборка данных.



Рисунок 15 - Пример запроса на поиск клиента по номеру телефона или по уникальному идентификатору устройства

### Сервис получения клиентских данных «Casper»

Casper – это платформа для учета большого объема структурированных или неструктурированных данных в реальном времени. Обработанные данные используются для статистики, анализа, прогнозов или принятия решений.

В Casper нескончаемым потоком поступают данные о разного рода совершенных финансовых событиях – платежи, переводы, погашения, займы и прочие операции.

Получаемые данные Casper сортирует, обрабатывает, обогащает, систематизирует.

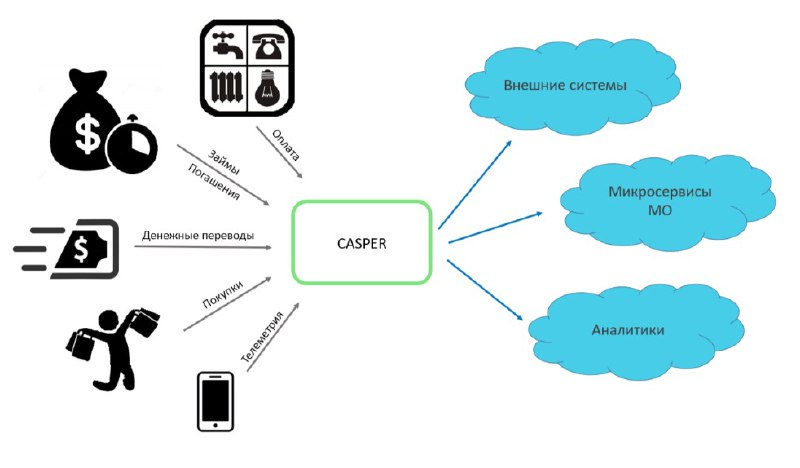


Рисунок 16 - Взаимодействие системы «Casper»

Назначение сервиса «Сasper» в системе «Телемаркетинг» - получение дополнительных данных о клиенте. Если клиент оставлял где-то внутри сервисов ГК «ЦФТ» свой цифровой след, то Casper хранит эту информацию.

На вход мы будет отправлять номер телефона клиента, уникальный идентификатор устройства, серию и номер паспорта клиента.

В ответ Casper будет отправлять JSON, в котором содержится большое количество клиентских данных.

### Сервис получения набора доступных продуктов для клиента «Abram»

Одним из факторов принятия решения о том нужно ли звонить клиенту является наличие доступных продуктов для него. Для этой цели используются ML-модели, обученные дата-аналитиками на основе признаков клиента.

В зависимости от информации о клиенте происходит подбор продуктов доступных для клиента, рассчитывается минимальная и максимальная суммы займа процент и период займа, которые подойдут клиенту.

В системе «Телемаркетинг» сервис используется для получения набора продуктов, которые в дальнейшем оператор будет предлагать клиенту.



Рисунок 17 - Метод определения типа заёмщика и набор доступных для него продуктов

### Сервис телефонии MightyCall

MightyCall – решения для организации профессиональных контактных центров и отделов продаж [17].

Этот сервис позволяет компаниям создать виртуальную телефонную линию. С помощью MightyCall, пользователи могут управлять входящими и исходящими звонками, записывать разговоры, установить автоответчик, создать голосовое меню, распределить звонки между сотрудниками, а также использовать другие функции, улучшающие процесс общения с клиентами [17].

В целом, MightyCall — это мощный инструмент для бизнесов, которые хотят улучшить свою телефонную систему, повысить эффективность работы и улучшить обслуживание клиентов.

В системе «Телемаркетинг» сервис используется для осуществления звонков по клиентским заявкам.

## Микросервисная архитектура

Микросервисная архитектура – один из подходов проектирования, при котором приложение строится из набора небольших сервисов. Каждый модуль отвечает за определенный процесс и взаимодействует с другими модулями [18].

Одним из главным плюсов микросервисов является повышение отказоустойчивости, доступности и масштабируемости системы. В микросервисной архитектуре, как правило, перестает работать только несколько модулей, а остальная система продолжит функционировать. Также важно, что в приложении локализуются сложности и риски отказов, то есть производительность системы может легко масштабироваться по горизонтальному пути. Модули автономны, поэтому их легче внедрять и тестировать. [18]

Несмотря на многие плюсы, стоит отметить и некоторые минусы в сравнении с монолитной архитектурой. При кажущейся простоте деление приложения на микросервисы – сложная техническая задача. Некоторые плюсы микросервисной архитектуры могут обернуться и минусами:

* Мониторинг – монолит один, поэтому отслеживать как он работает проще. Микросервисов могут быть десятки, сотни и тысячи, поэтому следить за каждым физически сложно. Из-за этого приходится обращать много внимания системам управления и мониторинга [18].
* Технологическая независимость каждого модуля – каждый микросервис может быть написан на разных язык программирования и на тех технология, которые были удобны команде, которая его разрабатывала. С одной стороны, это хорошо, так как какие-то языки и фреймворки лучше подходят под определенные задачи. С другой стороны, это минус – не все разработчики смогут быстро разбираться в большом количестве технологий, из-за этого будет сложнее поддерживать и вносить изменения в систему [18].
* Сложность развертывания – для выполнения требований отказоустойчивости микросервисы должны развёртываться на отдельных серверах. Поэтому необходимо использовать системы оркестрации и деплоймента, что дополнительно усложняет процесс сопровождения [18].

## Предметно-ориентированное проектирование сервисов.

В качестве построения системы на основе микросервисов был выбран предметно-ориентированный подход.

Предметно-ориентированный подход – метод разработки программного обеспечения, основанный на программировании предметной области [19]. Этот подход особенно подходит для сложных областей, где требуется упорядочить множество часто запутанных логических операций. Основной принцип – разделение приложения на домены и архитектурные слои.

Домен – предметная область, которая является совокупностью проблем и целей бизнеса [19].

Архитектурный слой — это определённый набор ресурсов, с помощью которых реализуются множество прикладных задач, характерных для данного слоя [20].

Процесс предметно-ориентированного проектирования предусматривает сотрудничество разработчиков и не разработчиков. В идеале должна быть общая модель с общими языками, поэтому, когда люди из разных областей с разными точками зрения обсуждают решение, у них будет общая база знаний с общими концепциями.

### Домен «Займы»

Доменной областью в проектируемой системе являются «Займы». Данные домен отвечает за получение и обработку специфических данных о клиенте, необходимых для телемаркетинга внутри финансовой организации.

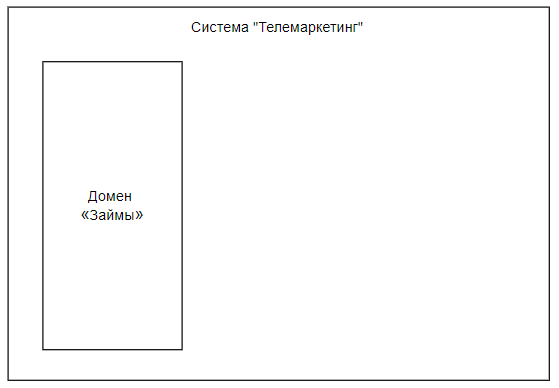


Рисунок 18 - Выделение доменной области «Займы»

Система «Телемаркетинг» в будущем может применяться не только для финансовой сферы, но и для любой другой, в которой может пригодиться эта система. При таком подходе нужно всего лишь написать специфичный под требования бизнеса домен.

### Архитектурный слой «Ядро»

Архитектурные слой «Ядро» представляет собой центральную архитектуры приложения, которая содержит и реализует бизнес-логику, правила и функциональные модели. Ядро является фундаментом и управляет основными аспектами автоматизации ручного процесса телемаркетинг. В данном слое находится логика обработки клиентской заявки в рамках кампании.

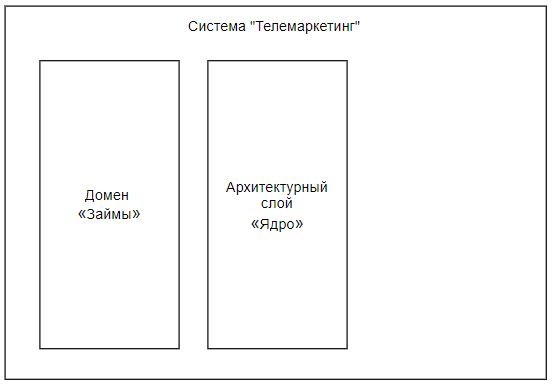


Рисунок 19 - Выделение архитектурного слоя «Ядро»

### Архитектурный слой «Интеграции»

Архитектурный слой "Интеграции" ответствен за связь и взаимодействие между различными системами или сервисами. В вашем случае, этот слой будет обеспечивать интеграцию между системой телемаркетинга и системой телефонии MightyCall для осуществления коммуникации с клиентами.

В этом слое будут содержаться компоненты, обеспечивающие передачу данных между системой телемаркетинга и системой телефонии. Как правило, в архитектуре для реализации слоя интеграции используются адаптеры.

Адаптер – паттерн, преобразующий интерфейс одной системы в другой. Адаптер обеспечивает совместную работу систем с несовместимыми интерфейсами, которая без него была бы невозможна [8].

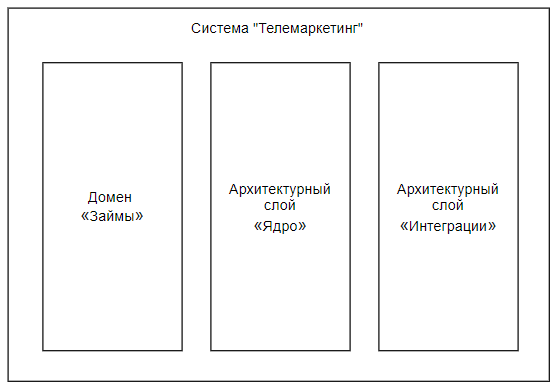


Рисунок 20 - Выделение архитектурного слоя «Интеграции»

## Архитектура системы «Телемаркетинг»

На *рисунке 21* представлена архитектура системы «Телемаркетинг». Как видно, здесь есть 6 модулей, которые распределены с помощью предметно-ориентированного подхода. В домене «Займы» находятся модули, специфичные для финансовой организации – сервис обработки входящего потока данных и сервис фильтрации этого потока. В архитектурном слое «Интеграции» находится сервис-адаптер для телефонии MightyCall. В архитектурном слое «Ядро» находится центральная часть приложения, которая независимо от домена и сервиса телефонии занимается обработкой клиентов – это сервис управления кампаниями и автоматизированное рабочее место (АРМ) оператора.



Рисунок 21 - Микросервисная архитектура системы «Телемаркетинг»

### Сервис обработки входящего потока данных «tml-event-generator»

Данный модуль занимается приемом и первичной обработкой входящего потока данных. Данные поступают в виде CSV-файлов, в которых построчно находятся клиентские заявки. CSV-файлы раскладываются по потокам в папки. Есть несколько потоков:

* «mobile\_loans» – Клиенты, которые начали оформлять заявку на займ в мобильном приложении «KoronaPay», но по каким-то причинам не довели её до конца;
* «pos\_reject» – Клиенты, которые оформляли заявку на займ прямо в магазине – партнере, но получили отказ автоматический отказ по каким-то причинам. Такие клиенты уходят на ручной разбор в нашу систему, чтобы оператор перенаправить его в мобильное приложение «Korona Pay».

Если в папке потока находится более одного файла для обработки, то импортируются все файлы, обнаруженные в папке – в порядке создания файлов от старых к новым.

Далее выполняется форматный контроль CSV-файла, который включает в себе проверки по столбцам на наличие обязательности и формата строки, описанных в ПРИЛОЖЕНИИ X.

Если форматный контроль выполнен не успешно, то файлы перекладываются в папку «error» на ручной разбор.

После успешного прохождения форматного контроля каждая строка как слепок заявки отправляется сервисом дальше – на обработку в сервис фильтрации входящего потока данных «tml-events-analyser» через брокер сообщений, а файл перекладывается в папку «success».

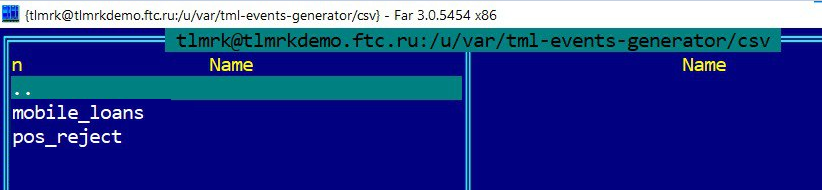


Рисунок 22 - Папки потоков данных

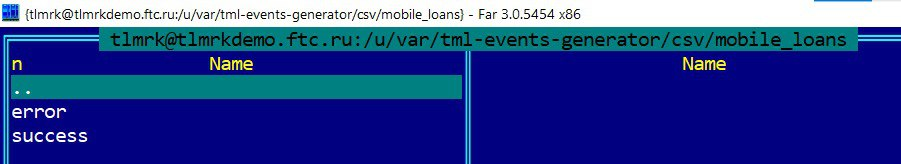


Рисунок 23 - Папки для успешной и не успешной обработки файлов

### Сервис фильтрации входящего потока данных «tml-events-analyser»

В этом модуле заявка проходит через каскад фильтров. Прохождение каскада означает что заявка становится кандидатом на участника кампании. Кампания – это причина, по которой система пытается осуществить коммуникацию с клиентом. Если один из фильтров не пройден, то обработка заявки прекращается – заявка точно не является кандидатом в кампанию.

Для каждой кампании существуют внутренние фильтры, которые настраиваются в конфигурационном файле. Доступные настройки:

* Набор масок для номера телефона
* Результат скоринга
* Тип флоу
* Гражданство
* Возраст
* Минимальное и максимальное значение грейс-периода займа
* Наличие клиента в черном списке
* Наличие доступных продуктов для клиента
* И т.д.

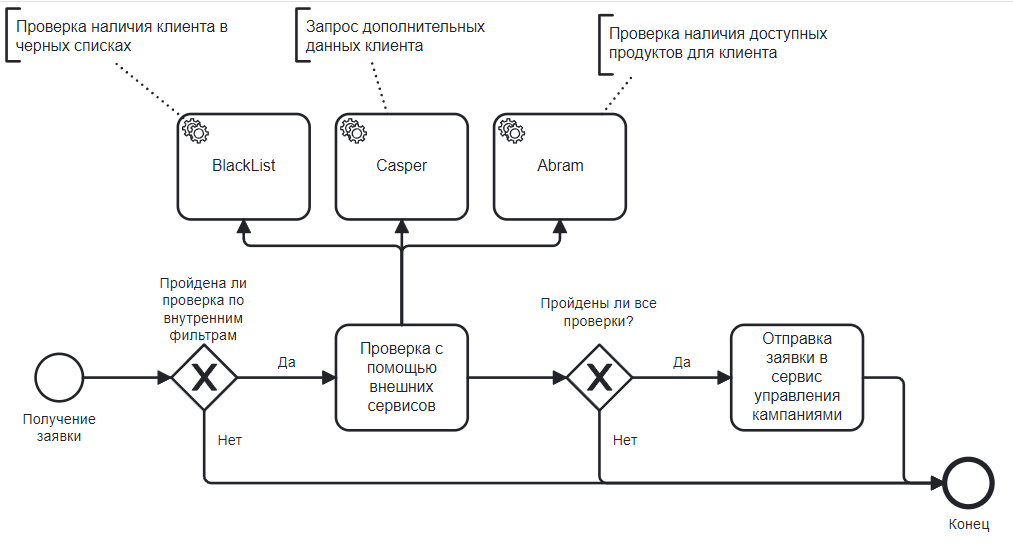


Рисунок 24 - Диаграмма моделирования бизнес-процесса «Фильтрации заявок через каскад фильтров»

ПРИМЕР КОНФИГА В ПРИЛОЖЕНИИ X.

Данный модуль кроме внутренних фильтров использует интеграционные фильтры, которые используют результаты из сервисов «BlackList», «Casper» и «Abram».

При успешном прохождении всех фильтров заявка отправляется в сервис управления кампаниями «tm-campaign-engine» c помощью брокера сообщений.

### Сервис управления кампаниями «tm-campaign-engine»

Модуль управления кампаниями отвечает за обработку клиентов в рамках их кампаний.

На вход поступают сообщения от модуля «tm-event-analyser». Если клиент уже является участником кампании и соответствующее задание находится в процессе выполнения, то предыдущее задание отменяется и обработка начинается в рамках нового задания. Новое задание сохраняется в базу данных с статусом «NEW».

Далее обработка продолжается обработчиком по расписанную, которое задается в конфигурационном файле. Обработчик достает все заявки со статусом «NEW» - запускает действие на обзвон. Теперь задача переходит в статус «IN\_PROGRESS» и отправляется в адаптер сервиса телефонии «tm-mightycall-adapter», который отвечает за интеграцию с MightyCall. Данный процесс отражен на *рисунке 25.*

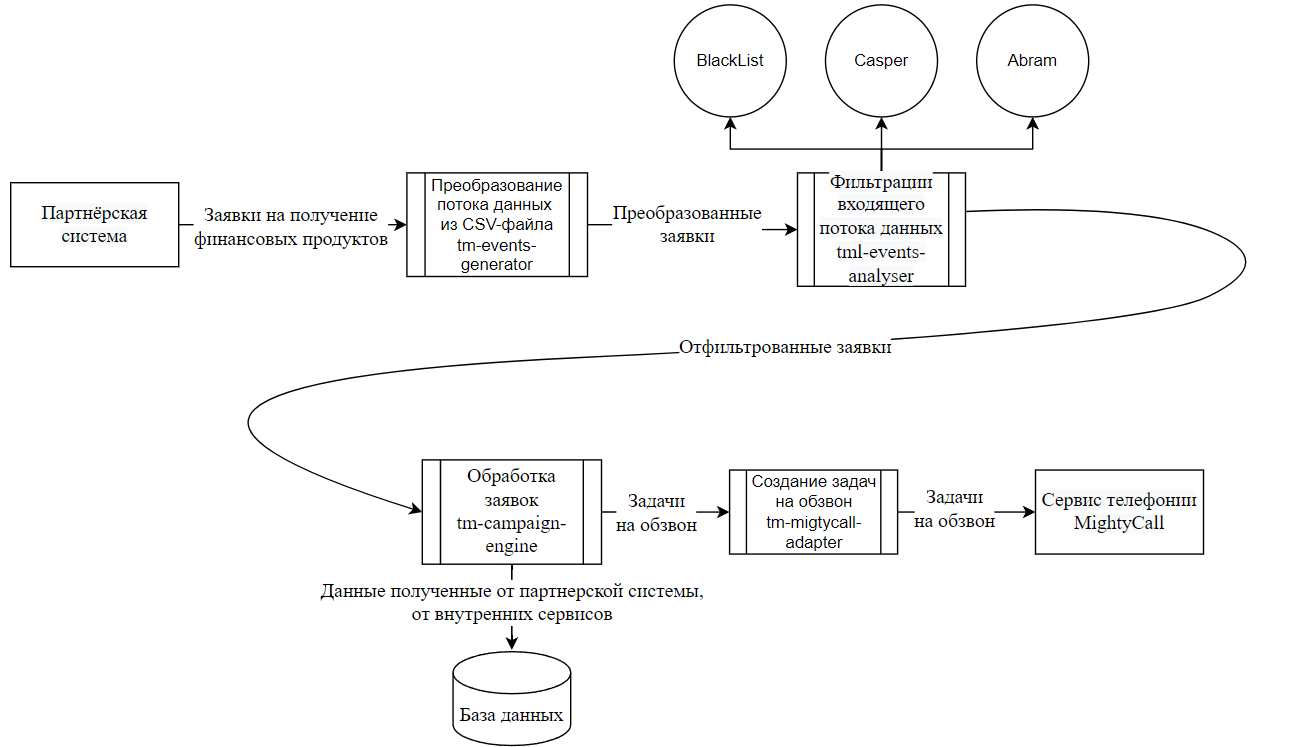


Рисунок 25 - Диаграмма потока данных «Создание задачи на обзвон»

Как только состоится звонок, модуль кампаний получит об этом уведомление, найдет участника в БД и переведет его задачу в статус «DONE».

Последняя обработка, которую выполнит модуль – сохранение результата коммуникации. В соответствующую таблицу будет занесена информация, которую заполнил оператор по результатам коммуникации с клиентом. Если клиента не удалось заинтересовать, то он помечается как кандидат к повторному обзвону, выставляется время, на которое он становится заблокированным, как только время блокировки подойдет к концу, он снова вернется в обработку со статусом «NEW».

### Сервис аутентификации и авторизации tm-security-service

Модуль аутентификации и авторизации отвечает за обеспечение безопасности системы путем аутентификации и авторизации пользователей. Его основная цель – защитить ресурсы и данные, предоставляя доступ только авторизированным пользователям.

Модуль имеет три HTTP POST-метода:

* «/api/security/v1/login» - аутентификация
* «/api/security/v1/logout» - выход
* «/api/security/v1/ checkAccess» - авторизация

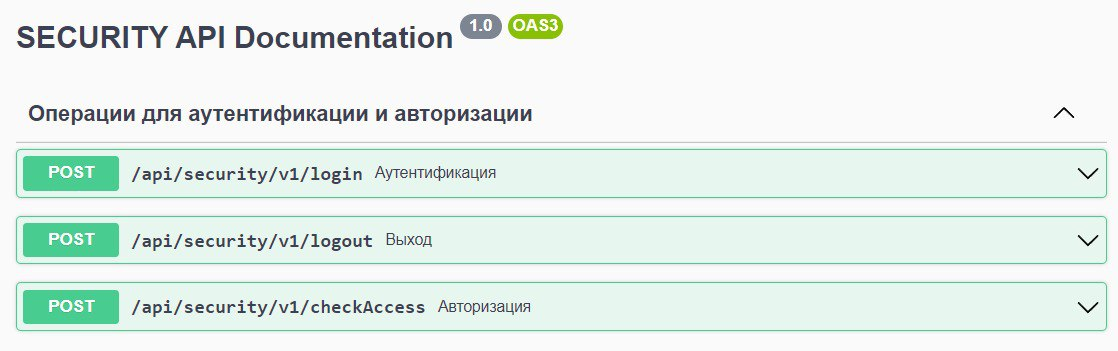


Рисунок 26 - Методы модуля «tm-security-service»

Входные и выходные данные методов, отраженных на *рисунке 26*, находятся в приложении X.

#### Аутентификация

Аутентификация — это процесс проверки подлинности учетных данных пользователя с целью установления его личности.

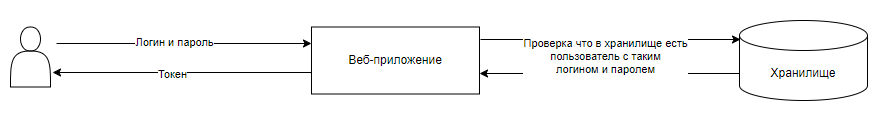


Рисунок 27 - Процесс аутентификации

На вход метода поступают данные, необходимые для аутентификации пользователя – логин и пароль. Затем существляется поиск оператора в соответствующей таблице по следующим условиям:

* Логин
* Хэш от пароля

Если пользователь найден и не заблокирован, то выполняется выдача токена для пользователя.

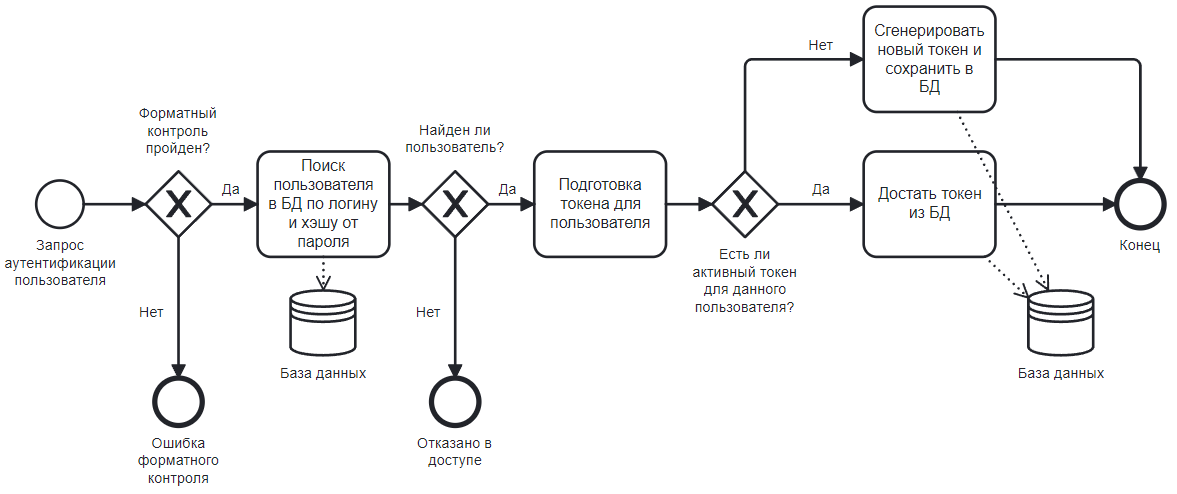


Рисунок 28 - Диаграмма моделирования бизнес-процесса «Запрос аутентификации пользователя»

#### Авторизация

Авторизация — это процесс определения разрешений и прав доступа, которые пользователь имеет в системе или к определенным ресурсам после успешной аутентификации. После того, как пользователь успешно прошел аутентификацию, и система удостоверилась в его личности, авторизация определяет, какие действия пользователь имеет право выполнять и к каким данным или функциональности он имеет доступ.

На данные момент в системе есть только одна роль «Оператор», с помощью которой можно получить доступ к любому ресурсу АРМ оператора.

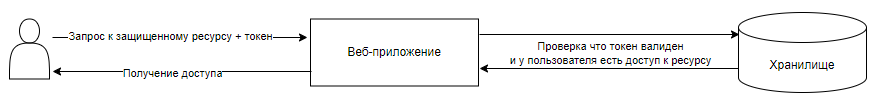


Рисунок 29 - Процесс авторизации

Авторизация осуществляется с помощью клиентского токена, который он получил во время аутентификации.

Если пользователь не заблокирован и ему на хватает прав для доступа к запрашиваемому ресурсу, то модуль сообщает об этом с помощью код «OK»

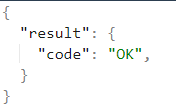


Рисунок 30 - Успешное прохождение авторизации

Иначе будет сообщение о том что доступ запрещен

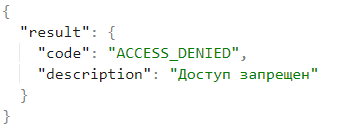


Рисунок 31 - Ошибка «ACCESS\_DENIED»

Данный процесс подробно отражен на рисунке 32

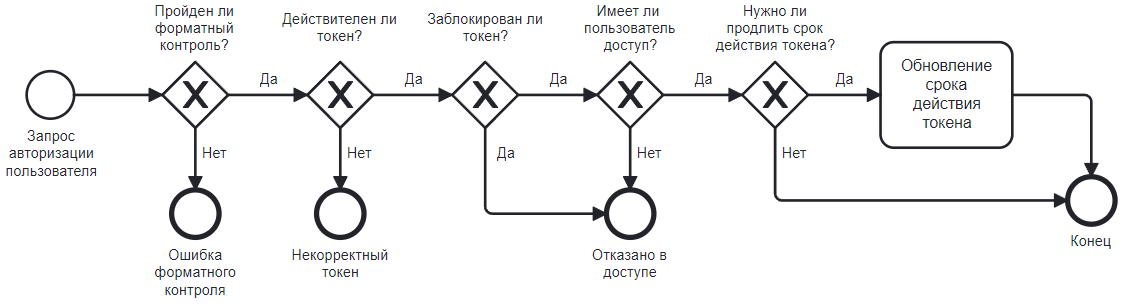


Рисунок 32 - Диаграмма моделирования бизнес-процесса «Запрос авторизации пользователя»

#### Выход из системы

Данный метод используется для выхода пользователя из системы. На вход поступают данные, необходимые для авторизации пользователя - токен.

Выполняется поиск токена в базе данных. Если токен найден, то он удаляется, что означает что пользователь больше не сможет авторизоваться с помощью него.

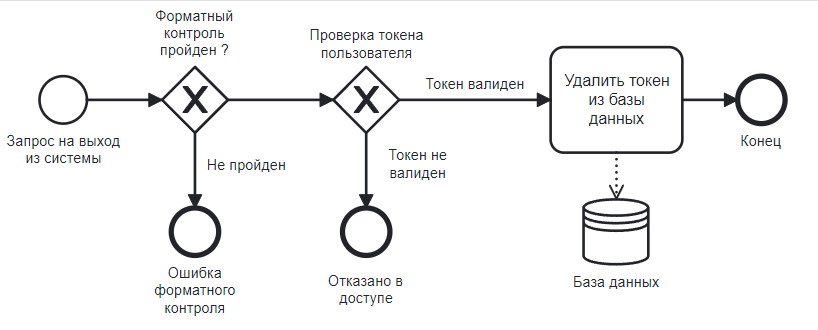


Рисунок 33 - Диаграмма моделирования бизнес-процесса «Запрос выхода из системы»

### Автоматизированное рабочее места оператора

Автоматизированное рабочее место является точкой взаимодействия клиентского приложения и системы «Телемаркетинг», предоставляет программный интерфейс для работы оператора с системой.

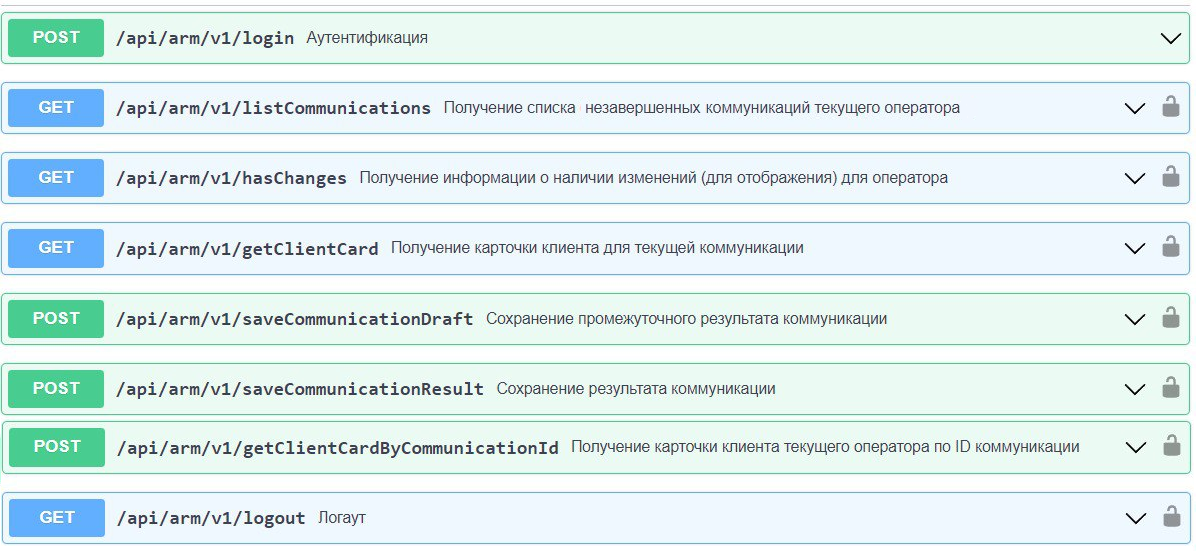


Рисунок 34 - Интерфейс АРМ оператора

На *рисунке 34* предоставлен API сервиса. Как видно, он имеет все необходимые методы для работы оператора – аутентификация в систему, выход из системы, запрос незавершенных коммуникаций, получение карточки клиента и сохранение результата коммуникации.

Как только оператор успешно проходит аутентификацию в АРМ, клиентское приложение запускает процесс опроса о начале нового звонка для этого оператора с помощью метода «/hasChanges».

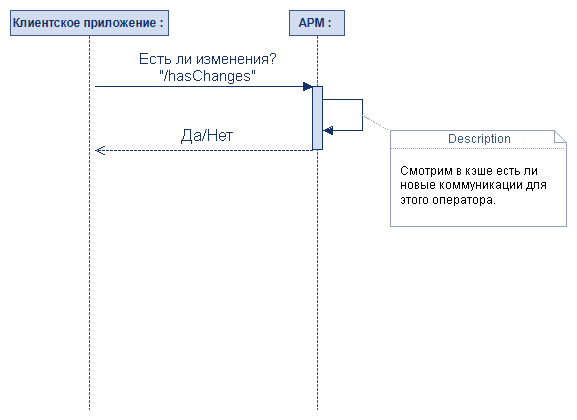


Рисунок 35 - Диаграмма последовательности «Процесс опроса АРМа клиентским приложением»

Данные о новых звонка хранятся в кэше сервиса. Как только сервис получает уведомление из брокера сообщений от адаптера телефонии о начале звонка, он начинает подготавливать карточку клиента. Под карточкой клиента подразумевается история прошлых коммуникаций с клиентом, список доступных продуктов, личные данные клиента. Собранная карточка отдается клиентскому приложению методом «/getClientCard».

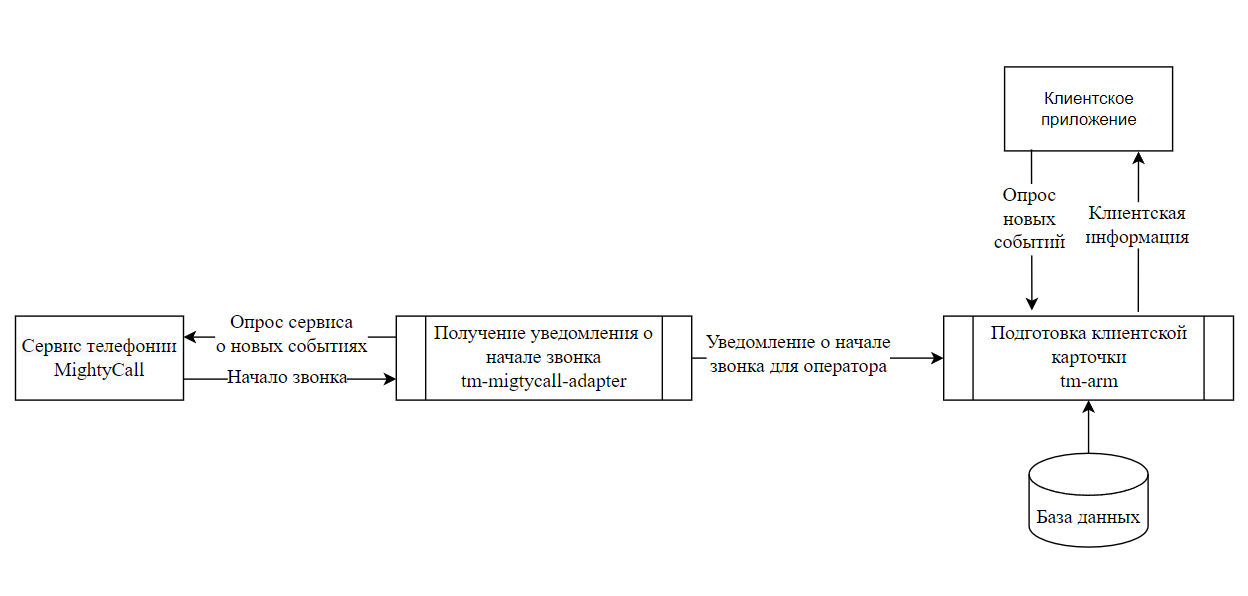


Рисунок 36 - Диаграмма потока данных «Обработка начала звонка»

Финальной обработкой коммуникации является сохранение результата, заполненного оператором в клиентском приложении.

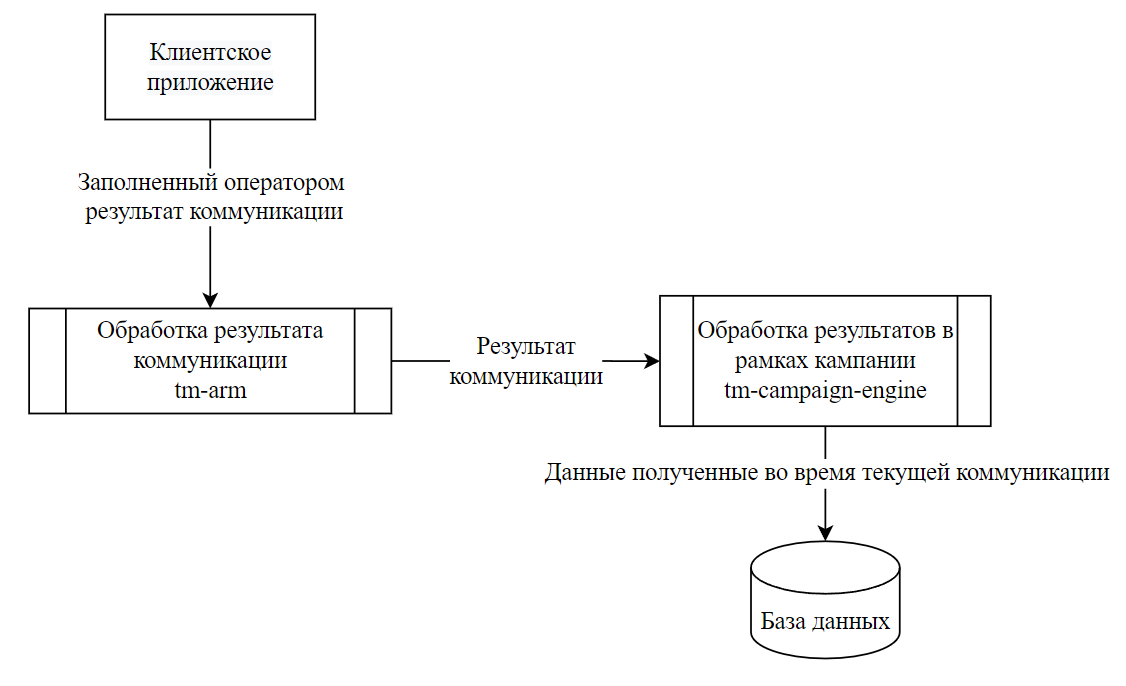


Рисунок 37 - Диаграмма потока данных «Обработка результатов звонка»

Если оператор успешно сохранил коммуникацию методом «/saveCommunicationResult», то она является завершенной. В противном случае коммуникация сохраняется методом «/saveCommunicationDraft» и уходит в доработку, которую оператор должен совершить позже с помощью метода «/getClientCardByCommunicationId». Этот метод нужен, потому что на заполнение результата отводится 30 секунд после того, как оператор закончил звонок с клиентом.

### Адаптер сервиса телефонии tm-mightycall-adapter

Модуль отвечает за взаимодействие системы «Телемаркетинг» с сервисом телефонии MightyCall. Основной задачей модуля является открытие и закрытие сессии оператора, создание задач на обзвон и опрос сервиса о начале новых звонков.

#### Открытие сессии оператора

Перед работой с сервисом необходимо инициализировать оператора. Как только оператор заходит в АРМ, адаптер получает сообщение из брокера сообщений для инициализации операторской сессии. После этого система начинает процесс опроса о начале новых событий для этого оператора.

Данные по активным операторам сохраняются в кэш адаптера и в соответствующую таблицу БД. Данные из кэша используются, чтобы знать по каким операторам необходимо опрашивать события из MightyCall, двойное хранение в БД необходимо, чтобы не потерять данные при перезагрузке модуля. При запуске адаптер копирует список активных операторов из таблицы в БД в пустой кэш.

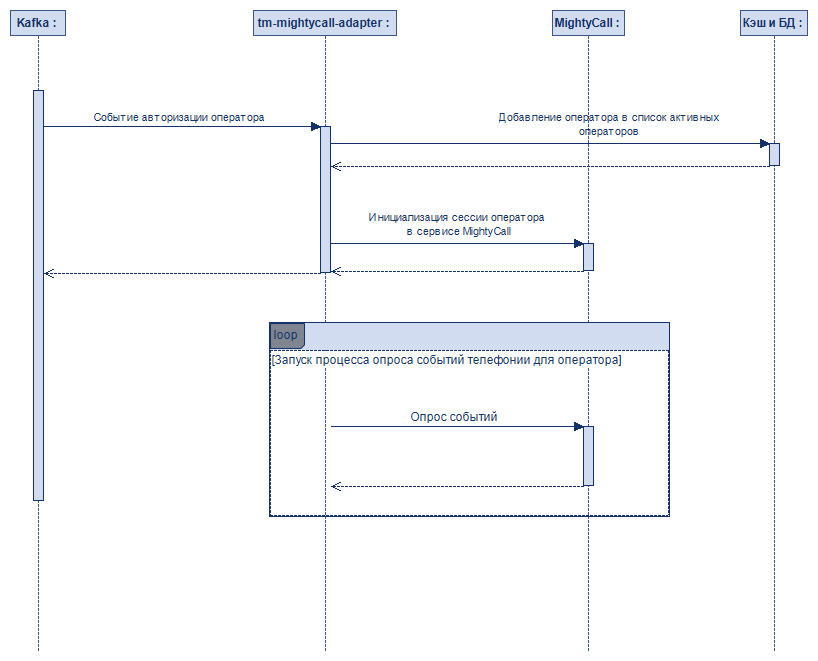


Рисунок 38 - Диаграмма последовательности «Открытие сессии оператора»

#### Закрытие сессии оператора

Адаптер принимает сообщение о выходе оператора из системы «Телемаркетинг» из АРМ оператора. Далее сервис удаляет оператора из списока активных в БД и в кэше. После этого процесс опроса событий для оператора прекращается. Закрытие сессии в MightyCall просходит автоматически, если за последние 15 минут не было запросов от оператора.

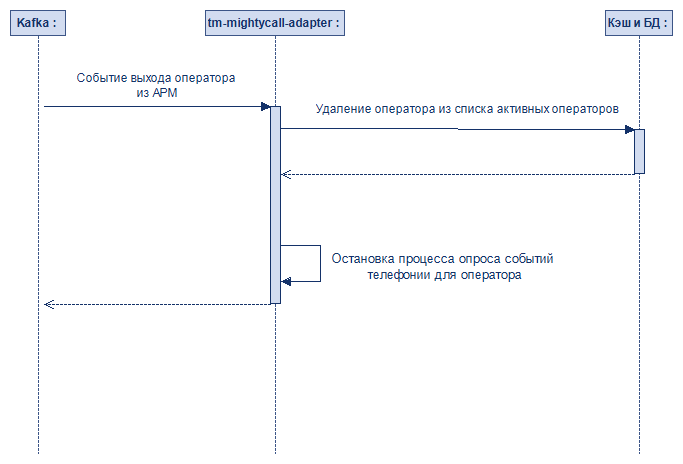


Рисунок 39 - Диаграмма последовательности «Закрытие сессии оператора»

#### Процесс опроса событий для оператора

По каждому активному оператору адаптер опрашивает MightyCall на наличие новых событий, для того чтобы передать эту информацию в АРМ c помощью брокера сообщений. Данный процесс отражен на *рисунке 40.*

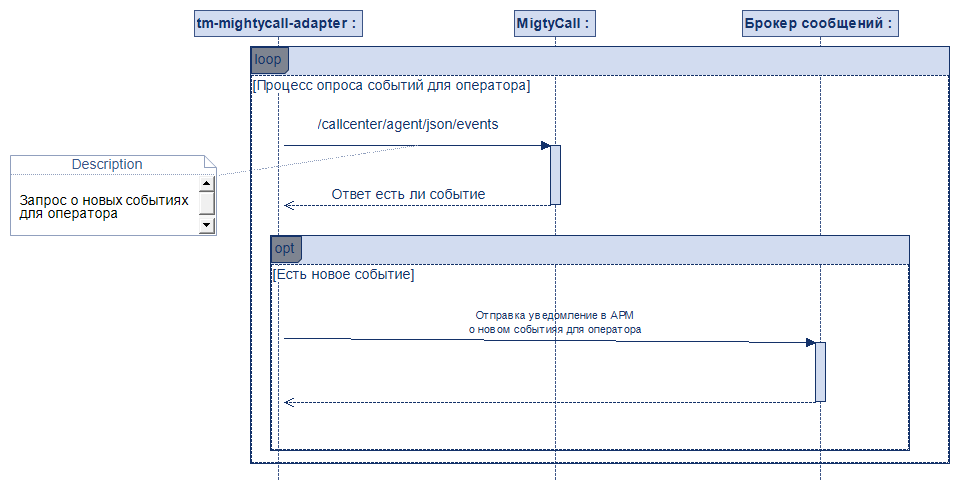


Рисунок 40 - Диаграмма последовательности «Процесс опроса событий для оператора»

#### Работа с очередью задач MightyCall

Адаптер принимает сообщение из брокера сообщений, сформированное в результате работы сервиса обработки входящего потока – tm-campaign-engine.

В зависимости от полученной задачи, адаптер отправляет задачу на создание новой задачи или отмену существущей в MightyCall. Если ранее адаптер уже отправлял эту задачу, и она не выполнена, то есть находится в статусе «NEW», то такая задача считается дублируемой и её не нужно снова отправлять в MightyCall.

Данный процесс отражен ниже, на *рисунке 41.*

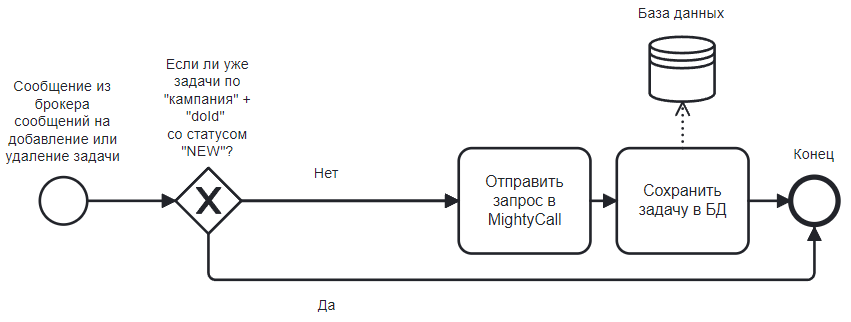


Рисунок 41 - Диаграмма моделирования бизнес-процесса «Работа с очередью задач MightyCall»

## Проектирование структуры хранения данных

Ниже представлены сущности предметной области, их атрибуты, описание таблиц сущностей и их связи

Таблица 1 – Кампании (TM$CAMPAIGNS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| NAME | VARCHAR(64),  NOT NULL | Название кампании |
| DOMAIN | DOMAINS, NOT NULL | Домен, к которому относится компания |
| DESCRIPTION | VARCHAR(255) | Описание |
| WTIME | TIMESTAMP, NOT NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |

Таблица 2 - Участники кампании (TM$CAMPAIGNERS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| PHONE | VARCHAR(15), NOT NULL | Номер клиента |
| DO\_ID | VARCHAR(255), NOT NULL | Уникальный идентификатор процесса в рамках всей системы |
| CLIENT\_INFO | JSONB, NOT NULL | Информация о клиенте |
| TASK\_TIME | TIMESTAMP, NOT\_NUL | Время отложенной задачи клиента |
| TASK\_TYPE | TASK\_TYPES NOT\_NULL | Тип коммуникации с клиентом |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| CAMPAIGN\_ID | FOREIGN\_KEY BIGINT, NOTNULL | Уникальный идентификатор кампании, к которой принадлежит клиент |

Таблица 3 - Коммуникации (TM$COMMUNICATIONS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| DO\_ID | VARCHAR(255), NOT NULL | Уникальный идентификатор процесса в рамках всей системы |
| PHONE | VARCHAR(15), NOT NULL | Номер телефона |
| STATE | COMMUNICATION\_STATES,  NOT NULL | Состояние коммуникации |
| CAMPAIGN\_DETAILS | JSONB,  NOT NULL |  |
| FIELD\_VALUES\_DETAILS |  |  |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| CAMPAIGN\_ID | FOREIGN\_KEY BIGINT, NOTNULL | Уникальный идентификатор кампании, к которой принадлежит коммуникация |
| OPERATOR\_ID | BIGINT, NOT NULL | Уникальный идентификатор оператора, который проводил коммуникацию |

Таблица 4 - Оператор (TM$OPERATORS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| LOGIN | VARCHAR(128), NOT NULL | Логин оператора |
| PASSWORD | VARCHAR(255), NOT NULL | Зашифрованный пароль оператора |
| LOCKED | BOOLEAN, NOT NULL | Флаг заблокирован ли оператор |
| ROLE | ROLES, NOT NULL | Роль оператора |
| FIRST\_NAME | VARCHAR(128), NOT NULL | Имя оператора |
| LAST\_NAME | VARCHAR(128), NOT NULL | Фамилия оператора |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |

Таблица 5 - Токен оператора (TM$OPERATOR\_AUTHS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| EXP\_DATE | TIMESTAMP, NOT NULL | Срок действия токена |
| TOKEN | VARCHAR(255) | Токен |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| OPERATOR\_ID | FOREIGN\_KEY, BIGINT, NOT NULL, | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 6 - Оператор в MightyCall (TM$MIGHTYCALL\_OPERATORS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| LOGIN | VARCHAR(128), NOT NULL | Логин оператора |
| PASSWORD | VARCHAR(255), NOT NULL | Зашифрованный пароль оператора для системы MightyCall |
| DOMAIN | DOMAINS, NOT NULL | Домен, к которому принадлежит оператор |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| OPERATOR\_ID | FOREIGN\_KEY, BIGINT, NOT NULL, | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 7 - Активные операторы (TM$OPERATORS\_STATES)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| OPERATOR\_ID | FOREIGN\_KEY, BIGINT, NOT NULL, | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 8 - Задачи для MightyCall (TM$MIGHTYCALL\_REQUESTS)

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| DO\_ID | VARCHAR(255), NOT NULL | Уникальный идентификатор процесса в рамках всей системы |
| STATE | MIGHTYCALL\_STATES, NOT NULL | Состояние задачи |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| CAMPAIGN\_ID | FOREIGN\_KEY BIGINT, NOTNULL | Уникальный идентификатор кампании, к которой принадлежит коммуникация |

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| NAME |  |  |
| FRONT\_CODE\_IDV |  |  |
| TYPE |  |  |
| REQUIRED |  |  |
| ORDER |  |  |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| CAMPAIGN\_ID | FOREIGN\_KEY BIGINT, NOTNULL | Уникальный идентификатор кампании, к которой принадлежит коммуникация |
| FIELD\_DICTIONARY\_ID | FOREIGN\_KEY BIGINT, NOTNULL |  |

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| NAME |  |  |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| ID | PRIMARY KEY, BIGSERIAL,  NOT NULL | Уникальный идентификатор |
| NAME |  |  |
| VALUE |  |  |
| ORDER |  |  |
| WTIME | TIMESTAMP NOT\_NULL | Время создание записи |
| UTIME | TIMESTAMP, NOT\_NULL | Время обновления записи |
| FIELD\_DICTIONARY\_ID |  |  |

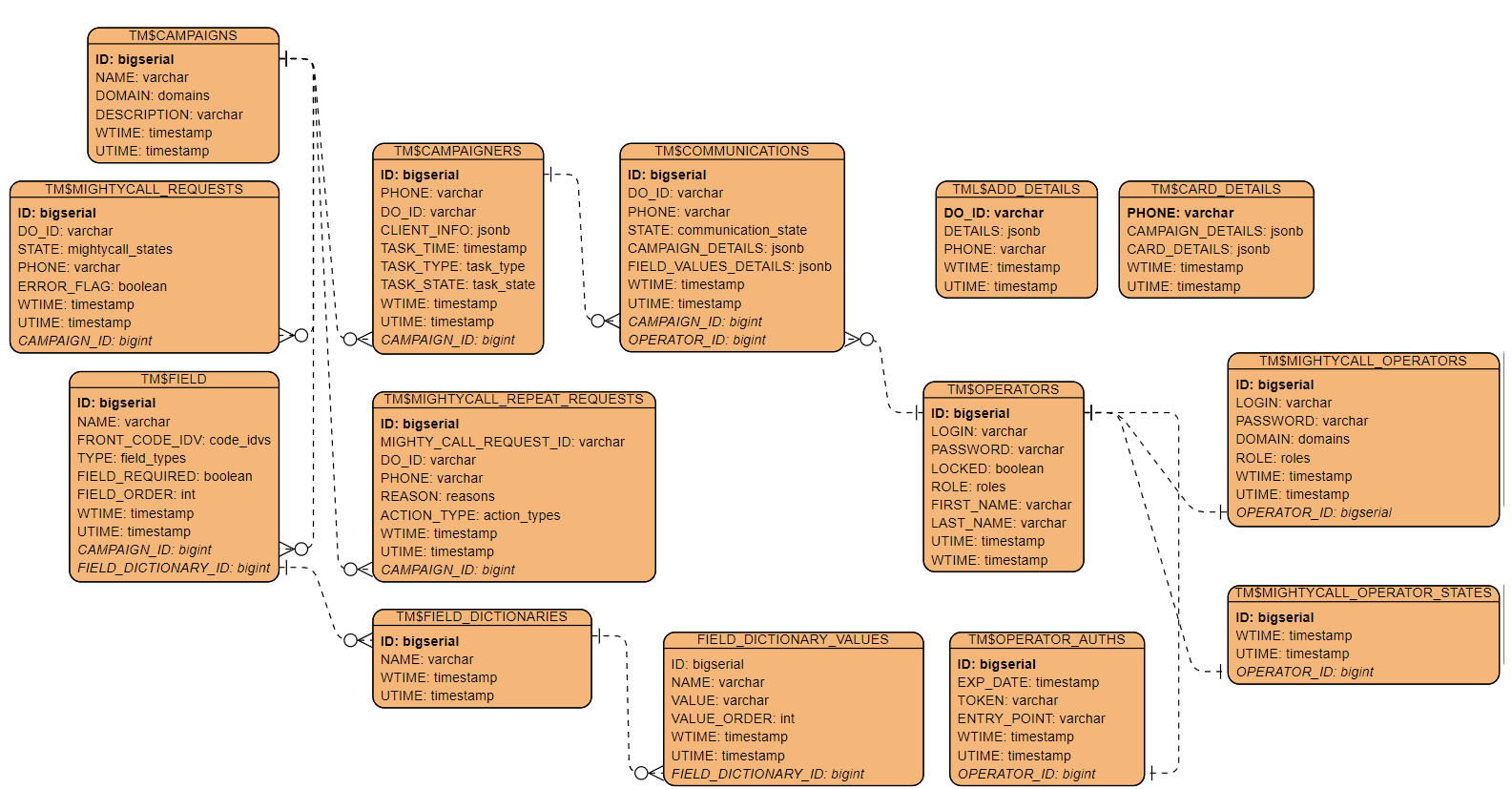


Рисунок 44 - Таблицы в базе данных

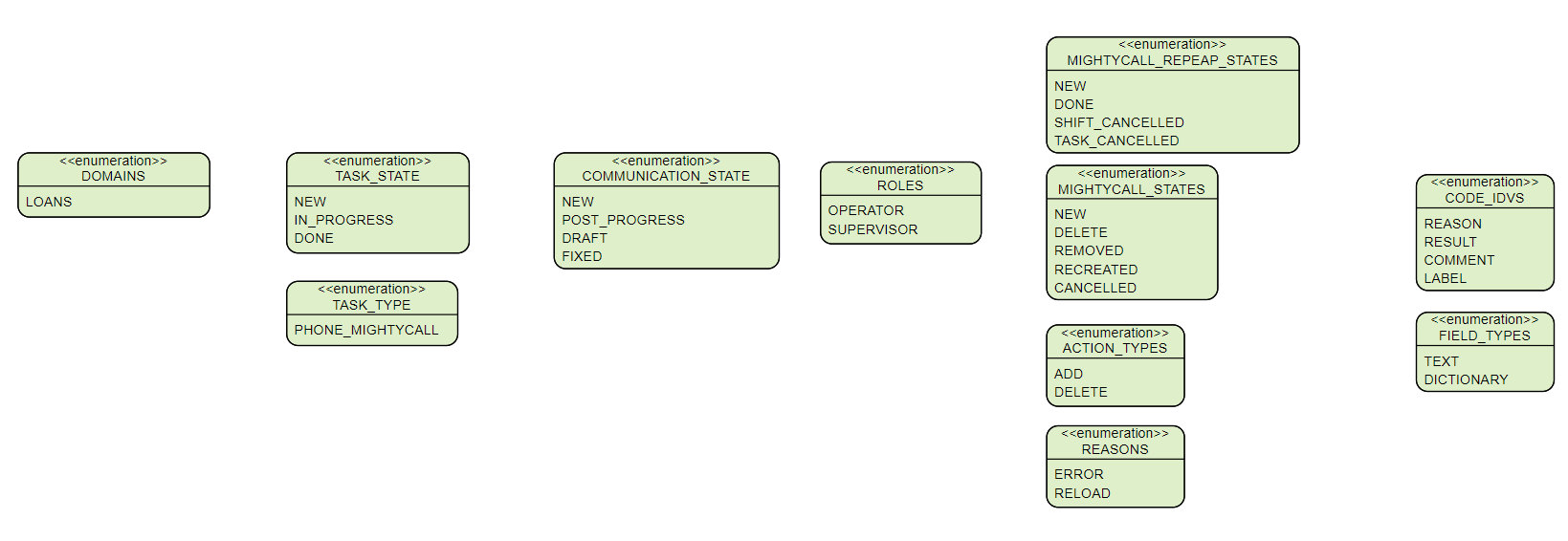


Рисунок 45 - Пользовательские типы в базе данных

# **ГЛАВА 3 Описание и тестирование готовой системы**

Форма на *рисунке 46* появляется при входе систему для входа оператора в систему:

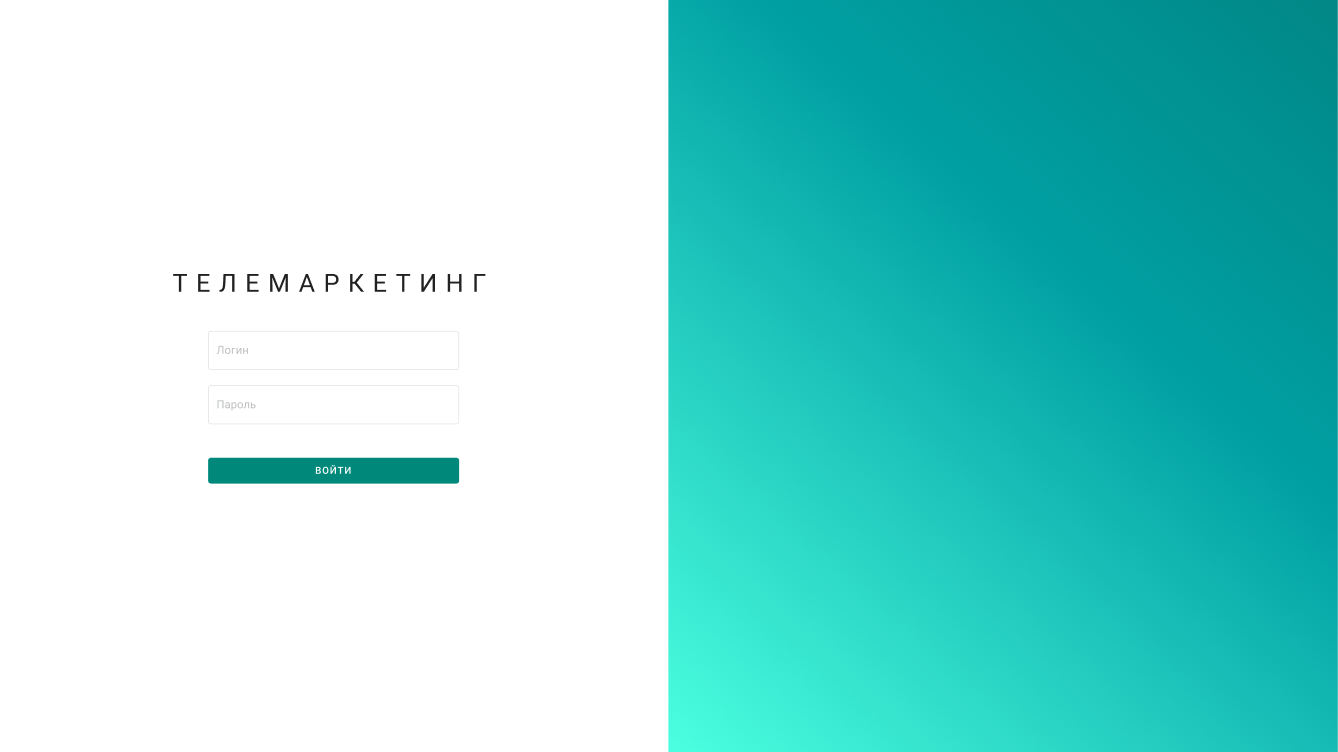


Рисунок 46 - Форма входа в систему

Как только пользователь успешно авторизовался в систему, перед ним открывается список незавершённых результатов коммуникаций, которые были запрошены из АРМ оператора по запросу `/listCommunications`.

На рисунке 47 у оператора для всех звонков результаты разговоров завершены.

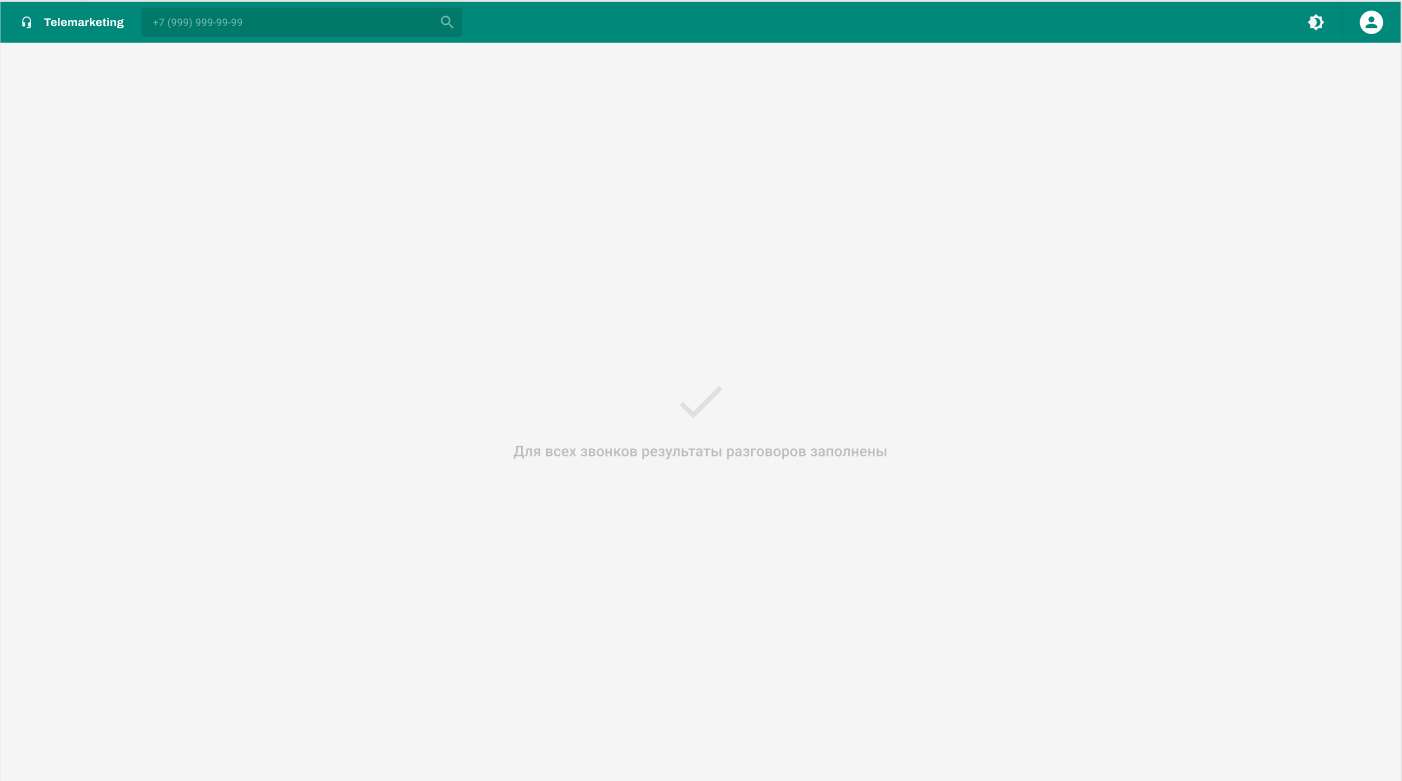


Рисунок 47 - Главный экран. Нет незаполненных результатов коммуникаций

На рисунке 48 у оператора есть 4 незавершенных результаты, которые он должен заполнить

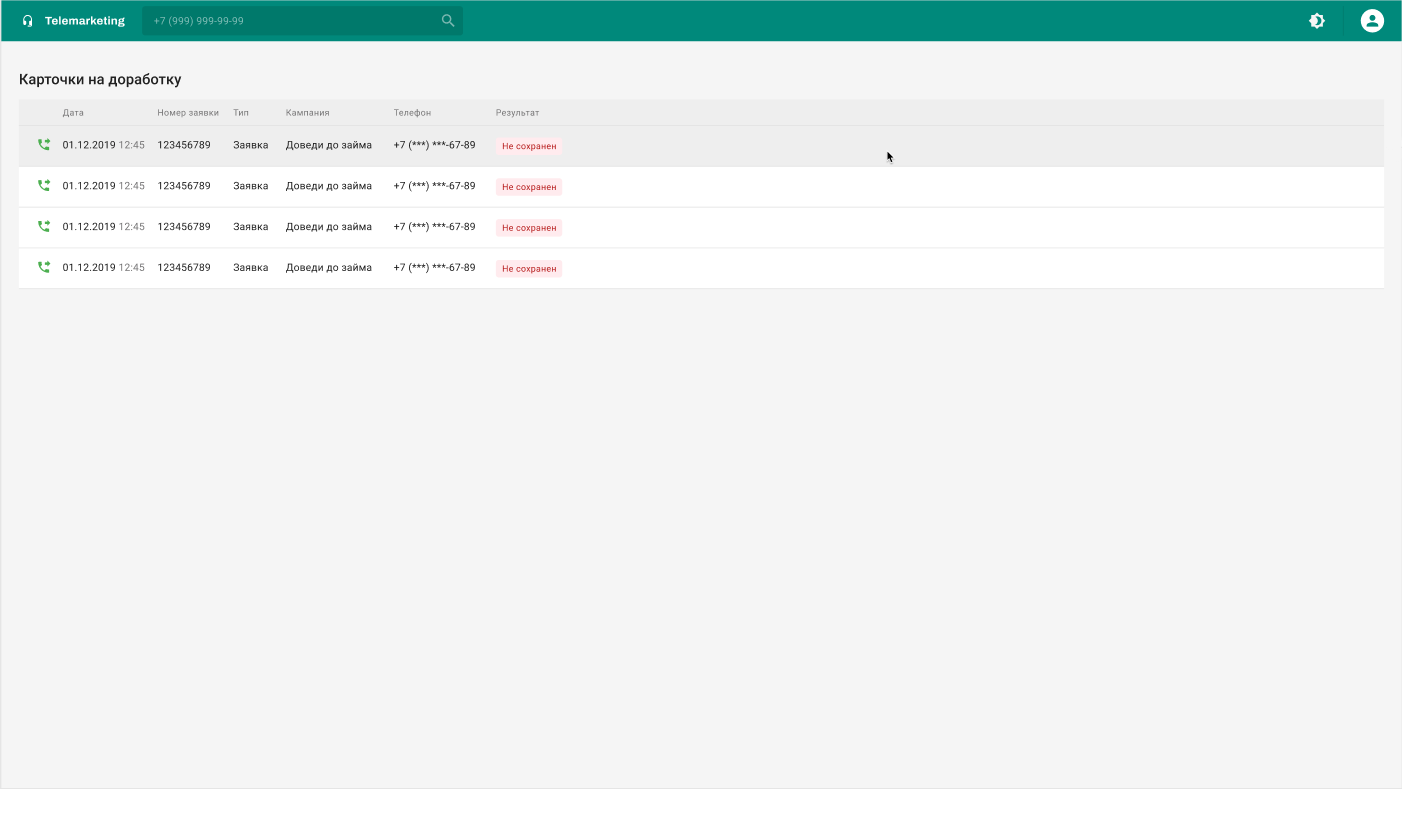


Рисунок 48 - Главный экран. Есть незаполненные результаты коммуникаций

Все это время идет фоновый процесс, который отправляет запрос `/hasChanges` в АРМ для получения уведомления о начале звонка. Как клиентское приложение получает положительный результат, оно делает запрос `/getClientCard`, чтобы получить карточку клиента, в которой содержится клиентская информация: история коммуникаций, история заявкой и займов, компания клиента.

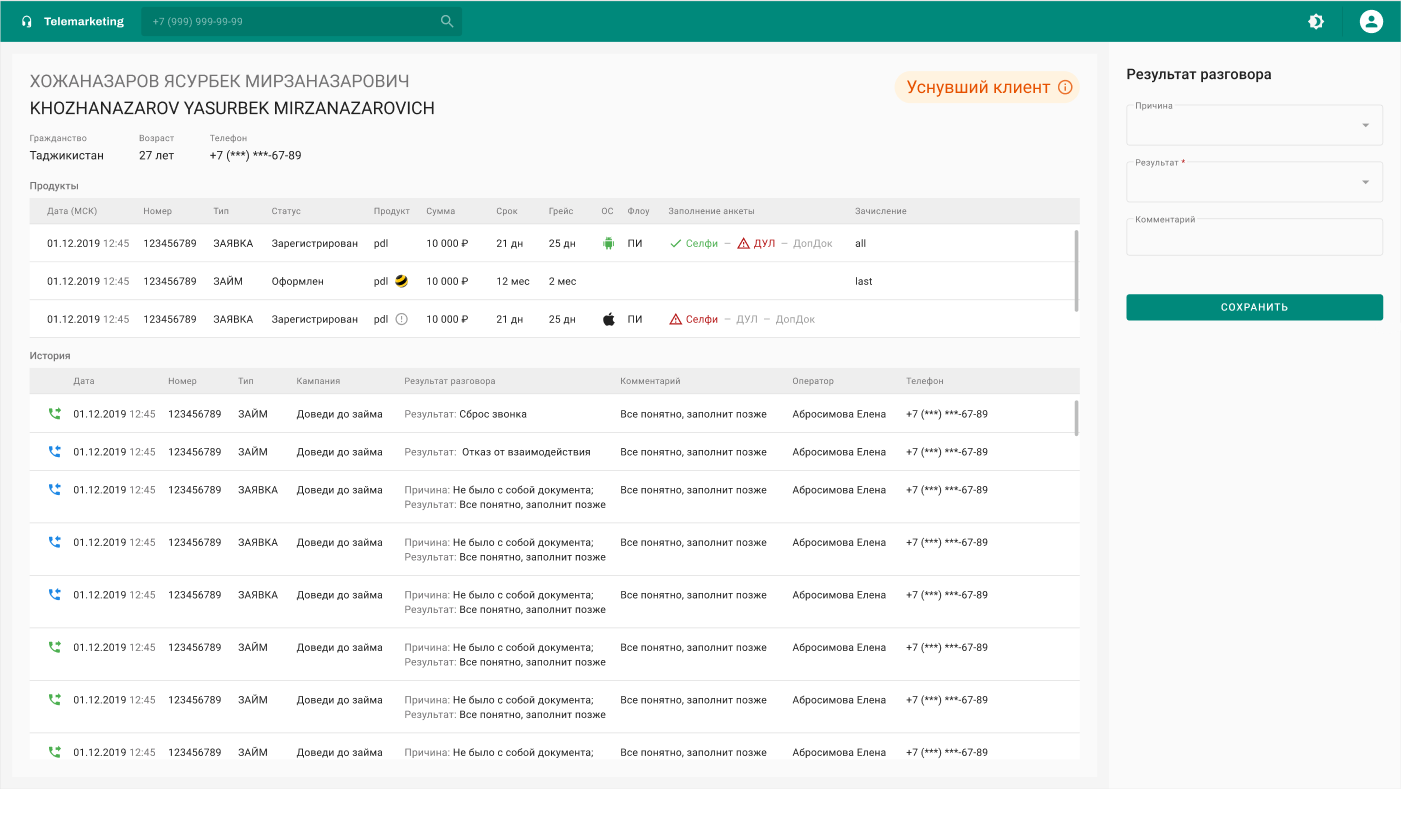


Рисунок 49 - Форма карточки клиента

Когда звонок подходит к концу оператор должен заполнить результат звонка. Нужно где-то описать что в зависимости от кампании меняется содержимое этой формы. Получаем вместе с getClientCard.

В зависимости от кампании клиента оператор заполняет результат… Криво

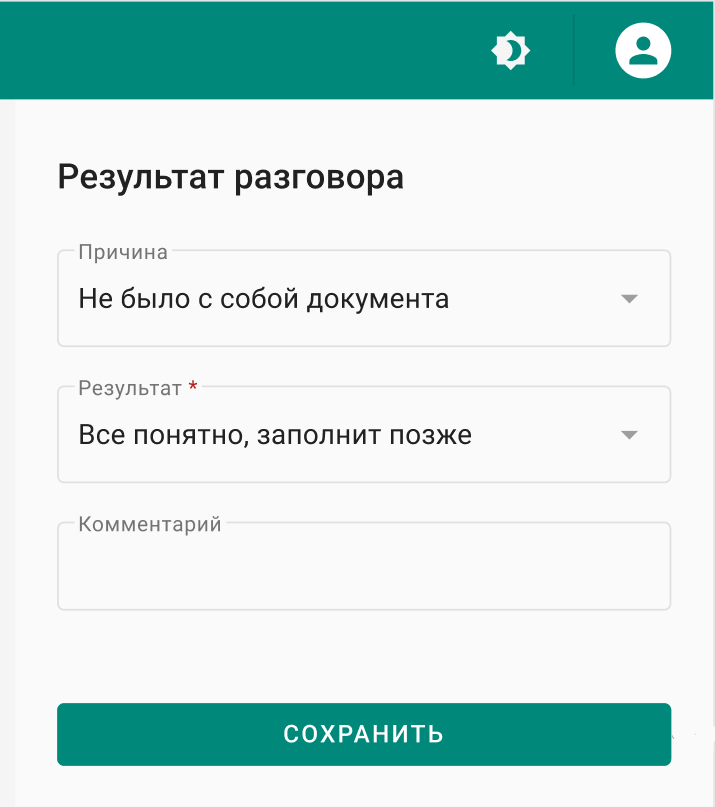


Рисунок 50 - Блок заполнения результата разговора

Если оператор успел заполнить результат, то он успешно сохраняется, о чем сообщается статус на рисунке 51

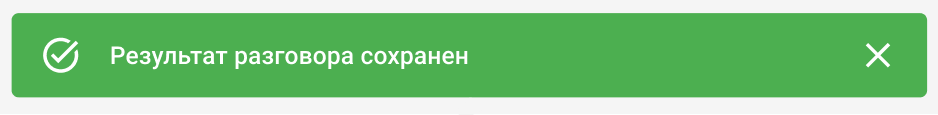


Рисунок 51 - Результат успешно сохранен

Если оператор не успел заполнить, то сохраняется черновик, который оператор должен заполнить позже, о чем сообщает статус на рисунке 52

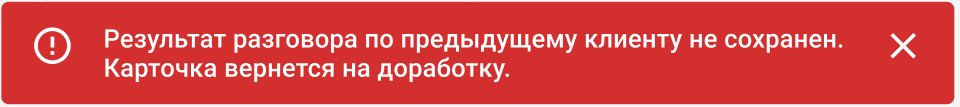


Рисунок 52 - Результат коммуникации не был сохранен. Отправлен на доработку

В результе, мы видим что реализован весь необходимый функционал для работы системы «Телемаркетинг». Клиентское приложение успешно работает с разработанной системой.

# **ЗАКЛЮЧЕНИЕ**

В результате проделанной работы была спроектирована и разработана система «Телемаркетинг». Были выполнены следующие задачи:

1. интеграция с закрытыми сервисами ГЦ «ЦФТ»;
2. интеграция с телефонией;
3. автоматическая обработка входящего потока заявок;
4. обработка и подготовка данных о клиенте;
5. обработка результатов коммуникаций
6. удобное автоматизированное рабочее место оператора

Разработанная система телемаркетинга позволяет в полной мере автоматизировать ручной бизнес-процесс.

Приложение безусловно требует доработок, в планах добавить функционал для супер-операторов: редактирование выпадающих списков результата коммуникации, возможность загрузка списков на обзвон в формате Excel-документов, возможность блокировки аккаунтов операторов.

В рамках учебного задания можно считать задачу выполненной, приложение соответствует предоставленным требованиям и удовлетворяет потребности компании.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Что такое телемаркетинг? - URL: <https://www.unisender.com/ru/glossary/chto-takoe-telemarketing>
2. Telemarketing 101: Definition, History, Strategy, And Benefits. – URL: <https://www.freedomtoascend.com/marketing/telemarketing/>
3. Telemarketing For Financial Services: Navigating The Path To Success. - URL: <https://www.appointmentsetting.co.uk/telemarketing-for-financial-services-navigating-the-path-to-success/>
4. 1С:CRM Телемаркетинг. – URL: <https://1crm.ru/help/telemarketing/>
5. 1C:CRM. Обзор. – URL: <https://habr.com/ru/companies/trinion/articles/296406/>
6. CRM для телемаркетинга. – URL: https://megacrm.ru/telemarketing
7. Работа интеграции MegaCRM и Облачной АТС Дом.ру Бизнес. - URL: <https://wiki.domru.biz/megacrm>
8. Как работать со звонками в MegaCRM. – URL: <https://help.megagroup.ru/kak-rabotat-so-zvonkami-v-megacrm>
9. Дашборд в Коллтрекинге Roistat. – URL: <https://help-ru.roistat.com/features/Calltracking/Svodnyi_otchet/>
10. What is REST? – URL: <https://restfulapi.net/>
11. Что такое Java? – URL: https://aws.amazon.com/ru/what-is/java/
12. Maven Getting Started Guide. – URL: https://maven.apache.org/guides/getting-started/
13. Introduction to Spring Framework. – URL: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html
14. Hibernate ORM. – URL: https://hibernate.org/orm/
15. About PostgreSQL. – URL: https://www.postgresql.org/about/
16. APACHE KAFKA QUICKSTART. – URL: <https://kafka.apache.org/quickstart>
17. Возможности MightyCall. – URL: <https://mightycall.ru/features/>
18. Что такое микросервисная архитектура. – URL: https://selectel.ru/blog/what-is-microservice-architecture/
19. Domain driven design. – URL: <https://simpleone.ru/glossary/domain-driven-design/>
20. Архитектурный слой (в корпоративной разработке). Понятие, определение, представление. – URL: <https://habr.com/ru/articles/511252/>
21. Паттерны объектно-ориентированного проектирования | Джонсон Ральф, Гамма Эрих, 2022, ISBN 978-5-4461-1595-2, c. 171

# ПРИЛОЖЕНИЯ