

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Систем сбора и обработки данных
(полное название кафедры)

Утверждаю

Зав. кафедрой ССОД

Бакаев М.А.
(подпись, инициалы, фамилия)

«__» _____ 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Буханцева Максима Олеговича
(фамилия, имя, отчество студента – автора работы)

Разработка сервиса телемаркетинга для финансовых организаций
(тема работы)

Автоматики и вычислительной техники
(полное название факультета)

Направление подготовки 09.03.02 Информационные системы и технологии
(код и наименование направления подготовки бакалавра)

**Руководитель
от НГТУ**

Воронов Виталий Владимирович
(фамилия, имя, отчество)

Старший преподаватель
(ученая степень, ученое звание)

(подпись, дата)

**Автор выпускной
квалификационной работы**

Буханцев Максим Олегович
(фамилия, имя, отчество)

АВТФ, АТ-04
(факультет, группа)

(подпись, дата)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра _____ *Систем сбора и обработки данных* _____
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой _____ *Бакаев М.А.* _____
(фамилия, имя, отчество)

(подпись, дата)

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту _____ *Буханцеву Максиму Олеговичу* _____
(фамилия, имя, отчество)

Направление подготовки _____ *09.03.02 Информационные системы и технологии* _____
(код и наименование направления подготовки бакалавра)

_____ *Автоматики и вычислительной техники* _____
(полное название факультета)

Тема _____ *Разработка сервиса телемаркетинга для финансовых организаций* _____
(полное название темы выпускной квалификационной работы бакалавра)

Исходные данные (или цель работы) _____ *Проектирование и разработка системы* _____
_____ *«Телемаркетинг» для финансовых организаций* _____

Структурные части работы _____

_____ *Введение* _____

_____ *1. Анализ предметной области* _____

_____ *2. Проектирование системы «Телемаркетинг»* _____

_____ *3. Описание и тестирование готовой системы* _____

_____ *Заключение* _____

_____ *Список литературы* _____

Приложение

Задание согласовано и принято к исполнению.

**Руководитель
от НГТУ**

Воронов Виталий Владимирович
(фамилия, имя, отчество)

Старший преподаватель
(ученая степень, ученое звание)

(подпись, дата)

Студент

Буханцев Максим Олегович
(фамилия, имя, отчество)

АВТФ, АТ-04
(факультет, группа)

(подпись, дата)

Тема утверждена приказом по НГТУ № _____ от « ____ » _____ 2024 г.

ВКР сдана в ГЭК № _____, тема сверена с данными приказа

(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

(фамилия, имя, отчество секретаря государственной
экзаменационной комиссии по защите ВКР)

АННОТАЦИЯ

Данная выпускная квалификационная работа состоит из 88 страниц, 52 рисунков и 23 источников.

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

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

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..... | 8 |
| ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ..... | 10 |
| 1.1. Что такое телемаркетинг: цели, принципы | 10 |
| 1.2. Телемаркетинг в финансовой сфере..... | 11 |
| 1.3. Ручной процесс телемаркетинга внутри платежной системы «Золотая Корона»..... | 12 |
| 1.4. Обзор существующих решений..... | 14 |
| 1.4.1. 1C:CRM | 14 |
| 1.4.2. MegaCRM..... | 16 |
| 1.5. Формирование требований к разрабатываемому сервису | 20 |
| ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ «ТЕЛЕМАРКЕТИНГ» | 22 |
| 2.1. Клиент-серверное взаимодействие | 22 |
| 2.2. Выбор инструментальных средств..... | 23 |
| 2.2.1. Язык программирования Java | 23 |
| 2.2.2. Фреймворк Spring..... | 24 |
| 2.2.3. Объектно-реляционный фреймворк Hibernate | 25 |
| 2.2.4. Система управления базами данных PostgreSQL | 26 |
| 2.2.5. 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 «Abram» | 30 |
| 2.3.4. Сервис телефонии «MightyCall»..... | 30 |
| 2.4. Микросервисная архитектура | 31 |
| 2.5. Предметно-ориентированное проектирование сервисов | 32 |
| 2.5.1. Домен «Займы» | 33 |
| 2.5.2. Архитектурный слой «Ядро» | 33 |
| 2.5.3. Архитектурный слой «Интеграции» | 34 |

| | | |
|---|--|----|
| 2.6. | Архитектура системы «Телемаркетинг» | 35 |
| 2.6.1. | Сервис обработки входящего потока «tml-event-generator» | 36 |
| 2.6.2. | Сервис фильтрации входящего потока «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 место оператора «tm-arm» | 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 MightyCall | 51 |
| 2.7. | Проектирование схемы хранения данных | 51 |
| 2.7.1. | Модель «сущность-связь» | 51 |
| 2.7.2. | Нормализация отношений..... | 54 |
| 2.7.3. | Таблицы и пользовательские типы данных | 57 |
| ГЛАВА 3. ОПИСАНИЕ И ТЕСТИРОВАНИЕ ГОТОВОЙ СИСТЕМЫ..... | | 63 |
| ЗАКЛЮЧЕНИЕ | | 68 |
| СПИСОК ЛИТЕРАТУРЫ..... | | 69 |
| ПРИЛОЖЕНИЕ | | 72 |
| Приложение А – Входные данные сервиса «tml-events-generator»..... | | 72 |
| Приложение Б – Входные и выходные данные сервиса «tm-security-service», метод аутентификации «/login» | | 74 |
| Приложение В – Входные и выходные данные сервиса «tm-security-service», метод выхода из системы «/logout»..... | | 75 |
| Приложение Г – Входные и выходные данные сервиса «tm-security-service», метод авторизации «/checkAccess»..... | | 76 |

| | |
|---|----|
| Приложение Д – Входные и выходные данные сервиса «tm-arm», метод аутентификации «/login» | 77 |
| Приложение Е – Входные и выходные данные сервиса «tm-arm», метод получения списка незавершенных коммуникаций текущего оператора «/listCommunications»..... | 78 |
| Приложение Ж – Входные и выходные данные сервиса «tm-arm», метод получения информации о наличии изменения для оператора «/hasChanges» | 79 |
| Приложение З – Входные и выходные данные сервиса «tm-arm», метод получения карточки клиента для текущей коммуникации «/getClientCard».. | 80 |
| Приложение И – Входные и выходные данные сервиса «tm-arm», метод сохранения промежуточного результата коммуникации «/saveCommunicationDraft» | 83 |
| Приложение К – Входные и выходные данные сервиса «tm-arm», метод сохранения результата коммуникации «/saveCommunicationResult» | 84 |
| Приложение Л – Входные и выходные данные сервиса «tm-arm», метод получения карточки клиента текущего оператора по id коммуникации «/getClientCardByCommunicationId» | 85 |
| Приложение М – Входные и выходные данные сервиса «tm-arm», метод выхода из системы «/logout» | 88 |

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 языка программирования, фреймворков и базы данных.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. 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.

1.2. 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 потенциальных [3];

Проводить маркетинговые исследования: операторы могут осуществлять быстрые опросы, чтобы выявить потребности рынка или собрать отзывы продуктах и услугах [3].

1.3. Ручной процесс 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.

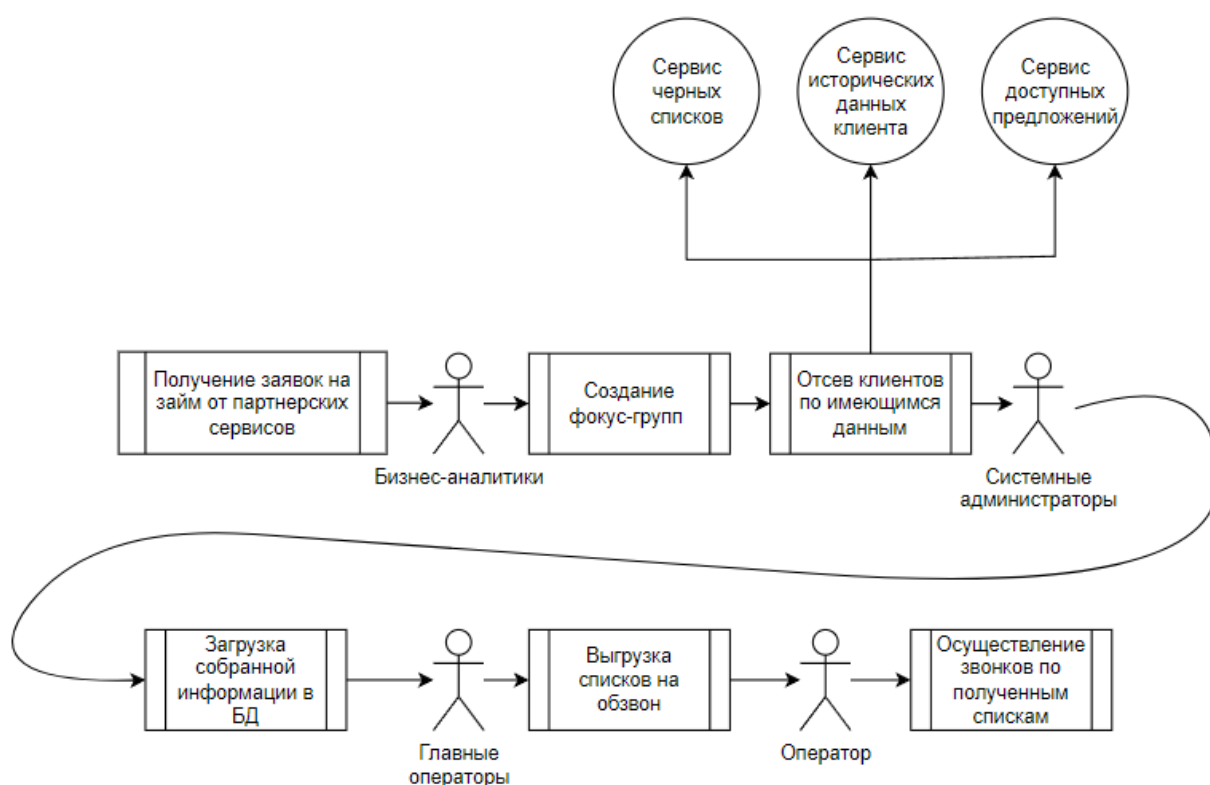


Рисунок 1 - Ручной бизнес-процесс телемаркетинга

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 главные 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 упростили бы сопровождение этого бизнес-процесса.

1.4. 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. 1.3.

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

1.4.1. 1C:CRM

«1C:CRM» — 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]

The screenshot shows a web form titled "Регистрация нового клиента" (Registration of a new client). At the top, there are two tabs: "Юридическое лицо" (Legal entity) and "Физическое лицо" (Physical person). The form is divided into several sections. The first section contains fields for "Наименование:" (Name) and "Подчинить:" (Subordinate to), with a dropdown arrow next to the latter. The second section contains fields for "Фамилия, имя, отчество:" (Surname, name, patronymic) and "Бизнес-регион:" (Business region). Below these are several checkboxes: "Клиент" (checked), "Поставщик", "Конкурент", "Прочие отношения", "Обслуживается торговым представителем", and "Предъявляет НДС по ставкам 4% и 2%". The third section, titled "Контактная информация" (Contact information), contains fields for "Вид телефона:" (Phone type), "Вид e-mail:" (Email type), and "Вид адреса:" (Address type). There is also a button "Найти похожих клиентов" (Find similar clients). The fourth section contains a checkbox "Создать контрагента при регистрации партнера" (Create counterparty when registering a partner) and a field for "ИНН:" (Tax ID). The fifth section, titled "Контактное лицо" (Contact person), contains fields for "Фамилия:", "Имя:", "Отчество:", and "Должность:", with a dropdown arrow next to the latter. The final section contains fields for "Вид телефона:", "Вид e-mail:", and "Адрес:".

Рисунок 2 - Форма регистрации нового клиента

Планирование звонков: «1С:CRM» позволяет настраивать расписание звонков, создавать задачи для сотрудников и отслеживать их статус выполнения [4].

Рисунок 3 - Форма планирования звонков

Система «1С:CRM» не предусматривает связи с телефонией — для этого существует продукт под названием «СофтФон» [5]. Хотя «СофтФон» и позволяет получать информацию о звонящем, проводить звонки, но это отдельный продукт, который имеет свои минусы.

Например, при поступлении звонка он не сразу попадает в «1С:CRM», а только при команде пользователя «Передать в 1С». С одной стороны, это хорошо, так как неизвестные звонки не попадут в систему, но есть опасность того, что не все звонки клиентов попадут в базу [5].

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

Карточка телефонного звонка позволяет учитывать затраты времени, участников разговора и основные запросы клиента.

Однако «1С:CRM» 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.

1.4.2. 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].

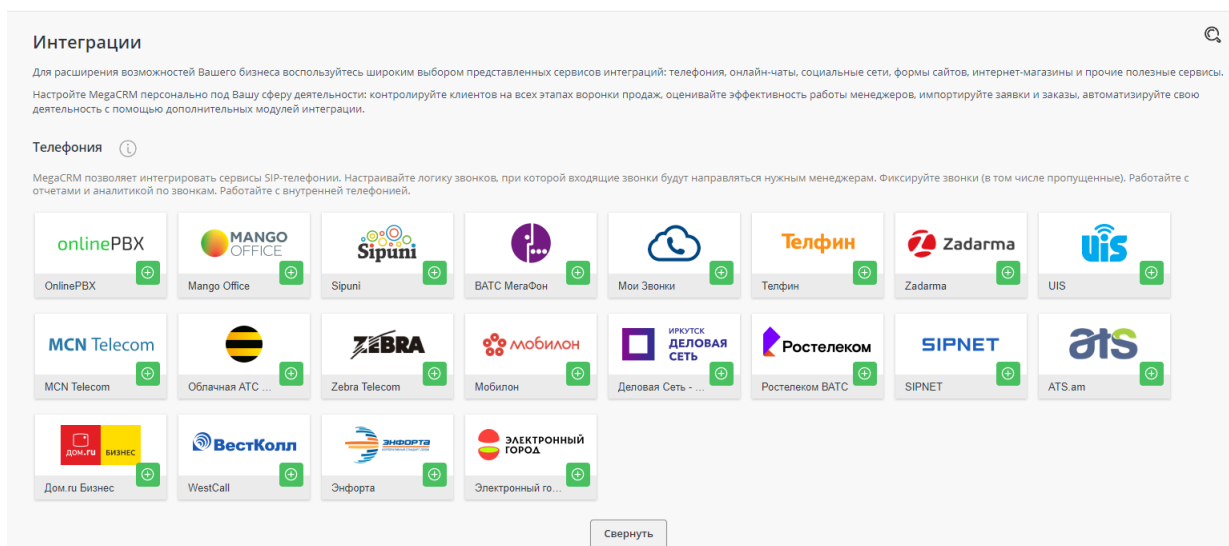


Рисунок 5 - Интеграция «MegaCRM» с телефоний

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

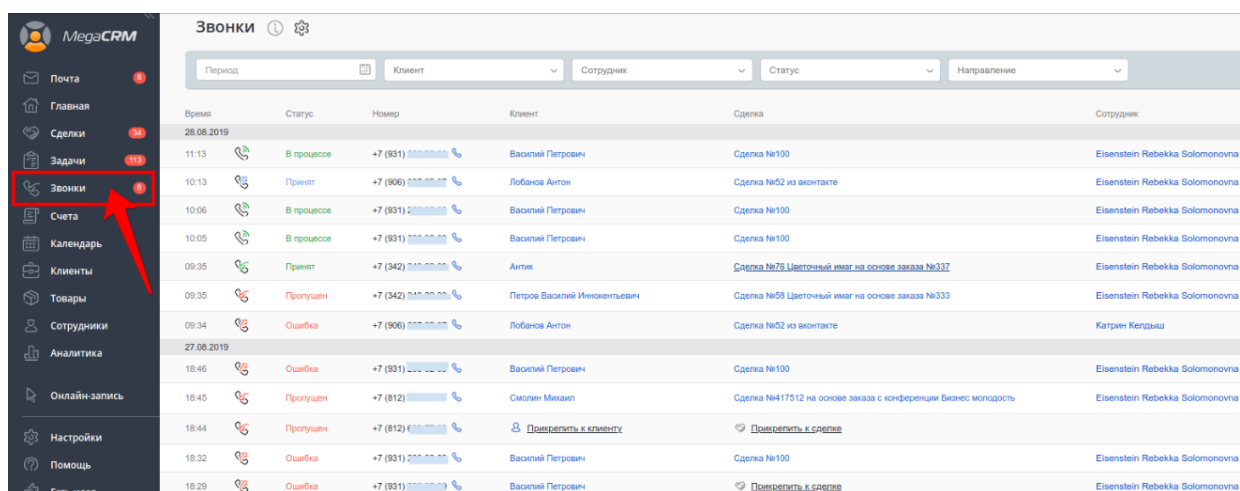


Рисунок 6 - Раздел обзвона клиентов

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]. Благодаря централизованному хранению

информации о клиенте, компания может значительно повысить эффективность работы с клиентской базой.

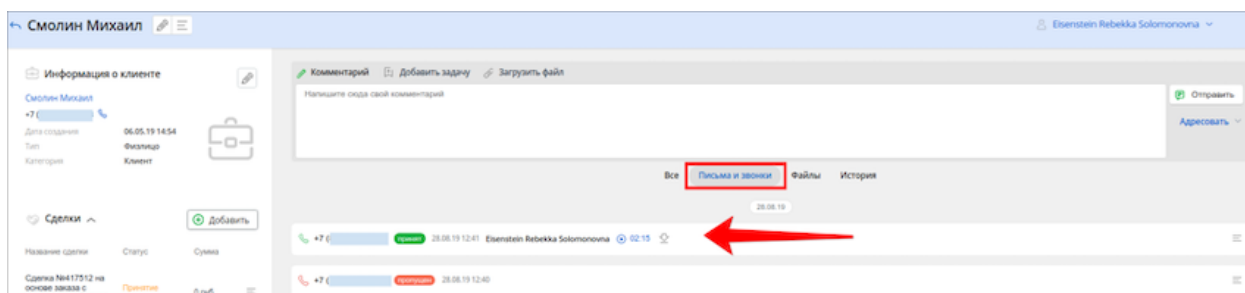


Рисунок 7 - Карточка клиента

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 взаимодействия 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.

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].

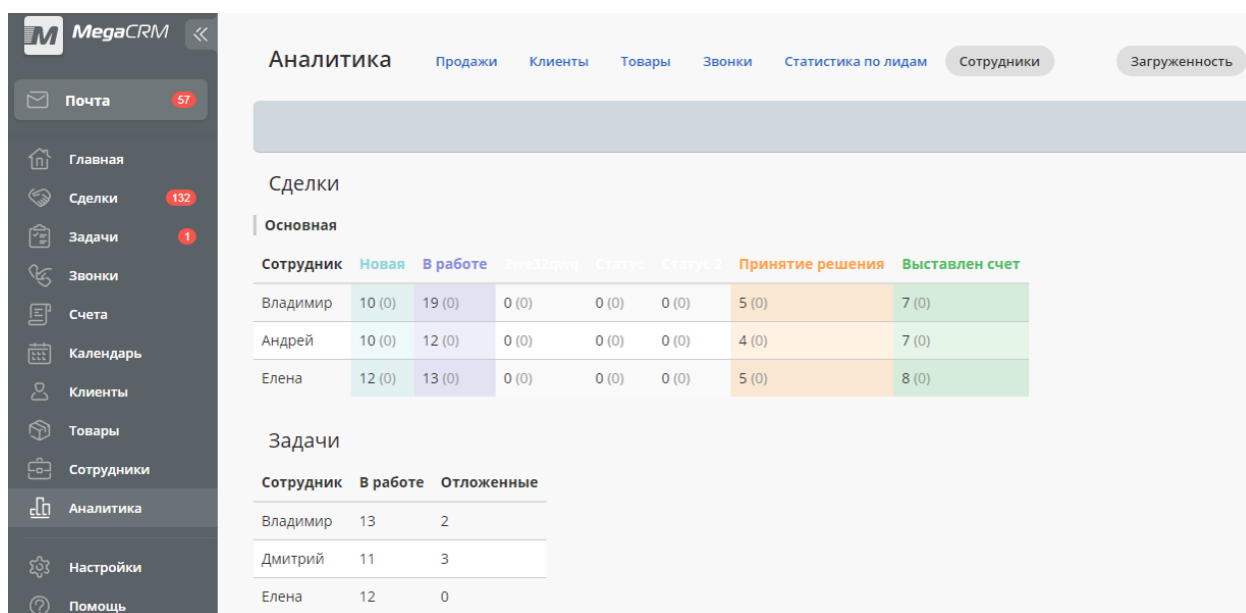


Рисунок 8 - Раздел «Аналитика», вкладка «Сотрудники»

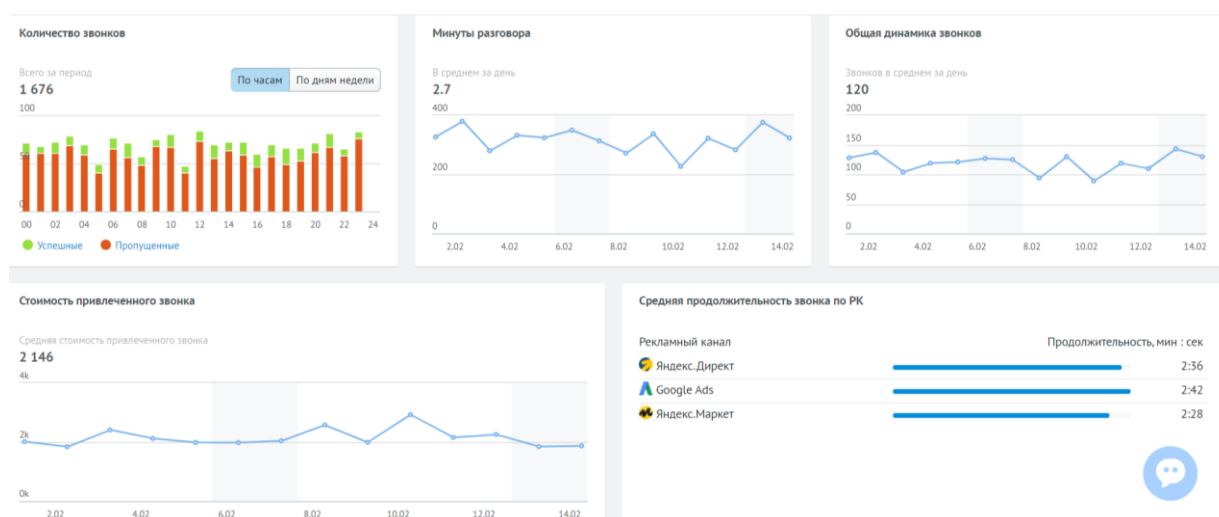


Рисунок 9 - Раздел «Аналитика», вкладка «Звонки»

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 могла бы полностью автоматизировать бизнес-процесс, описанный в п. 1.3, 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».

1.5. Формирование требований к разрабатываемому сервису

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

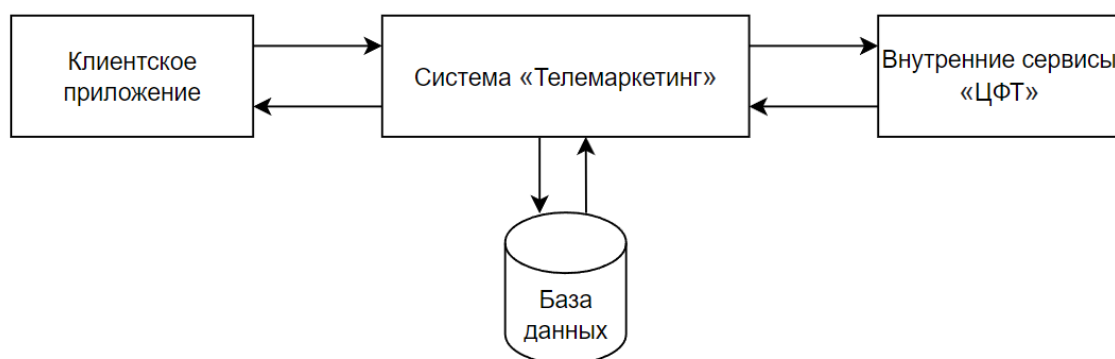


Рисунок 10 - Схема взаимодействия систем

Система «Телемаркетинг» будет 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:

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

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ «ТЕЛЕМАРКЕТИНГ»

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

2.1. Клиент-серверное взаимодействие

Клиентское приложение будет взаимодействовать с программным интерфейсом серверного приложения посредством архитектурного стиля Representational State Transfer (REST). Фактически REST API представляет собой просто набор конечных точек, к которым клиент обращается с помощью HTTP-запросов для получения информации 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 форматы данных [10].

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

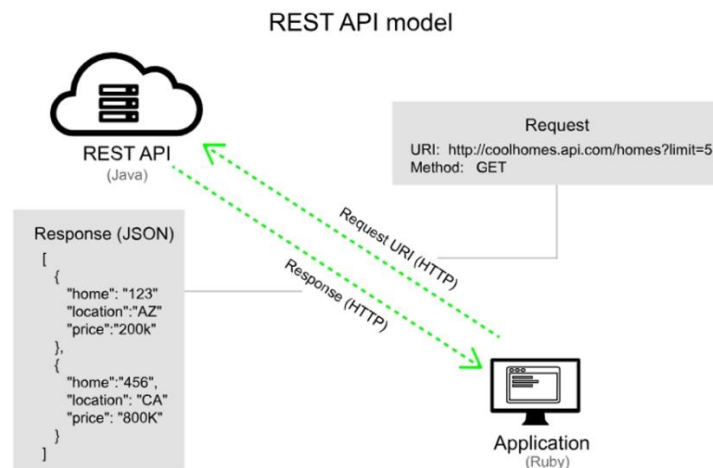


Рисунок 11 - Схема клиент-серверного взаимодействия

2.2. Выбор инструментальных средств

Для решения поставленных задач необходимо выбрать технологии, средства разработки и фреймворки, позволяющие реализовать систему «Телемаркетинг».

2.2.1. 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 C, 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.

2.2.2. Фреймворк Spring

Spring – 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 состоит из набора 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.

2.2.3. Объектно-реляционный фреймворк Hibernate

Hibernate — 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 данных [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].

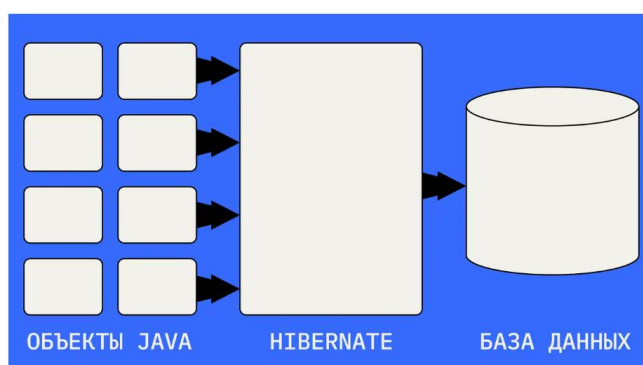


Рисунок 12 - Схема отображения Java – объектов в записи базы данных с помощью фреймворка Hibernate

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].

2.2.4. Система управления базами данных PostgreSQL

PostgreSQL — это универсальная система управления базами данных (СУБД), которая сочетает в себе объектно-реляционные возможности. Она предоставляет возможность создания надёжных баз данных, широко используемых в финансовой сфере, разработке веб-приложений и других областях [15].

Этот продукт бесплатен и доступен под лицензией с открытым исходным кодом, что означает отсутствие необходимости платить за его использование, независимо от целей, коммерческих или нет. Существуют также платные редакции от сторонних разработчиков, предлагающие дополнительные функции, однако их использование является опциональным [15].

PostgreSQL совместим с различными операционными системами, такими как Windows, Linux и macOS. Пользователи могут легко установить и использовать СУБД без необходимости дополнительных инструментов.

PostgreSQL поддерживает множество типов данных и структур, включая сетевые адреса, JSON, геометрические данные для геопозиций, XML и другие. Пользователи могут создавать собственные пользовательские типы данных, что помогает упростить работу с базой данных и установить необходимые ограничения [15].

ACID — это набор требований, обеспечивающих надежность и целостность данных. Аббревиатура расшифровывается как atomicity, consistency, isolation, durability, то есть атомарность, согласованность, изолированность, устойчивость. Эти требования обеспечивают надёжную работу системы в реальном времени. PostgreSQL соответствует всем четырем принципам ACID, обеспечивая сохранность данных при выполнении транзакций и других операций [15].

2.2.5. 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 связывает 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

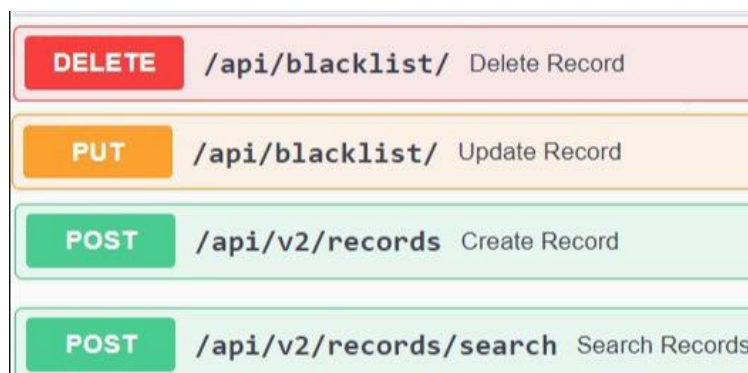
2.3. Сторонние сервисы

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

2.3.1. 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 клиенту.

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



| | | |
|--------|------------------------|----------------|
| DELETE | /api/blacklist/ | Delete Record |
| PUT | /api/blacklist/ | Update Record |
| POST | /api/v2/records | Create Record |
| POST | /api/v2/records/search | Search Records |

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.

```

{
  "query": {
    "$or": [
      {
        "phone": "79131234567"
      },
      {
        "device_id": "4fsa86198as461"
      }
    ]
  }
}

```

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

2.3.2. 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.

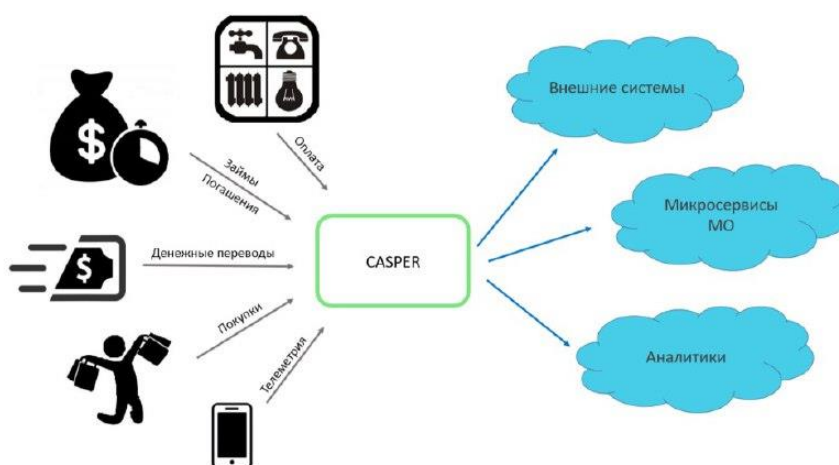


Рисунок 16 - Взаимодействие системы «Casper»

Назначение сервиса «Casper» 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 будем 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.

2.3.3. Сервис получения набора доступных продуктов «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.

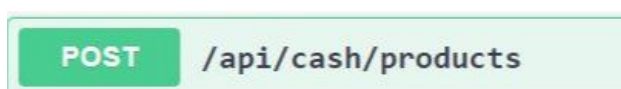


Рисунок 17 - Метод определения типа заёмщика и набор доступных для него продуктов

2.3.4. 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.

2.4. Микросервисная архитектура

Микросервисная архитектура — это подход проектирования, при котором приложение состоит из набора небольших сервисов, каждый из которых отвечает за определенный процесс и взаимодействует с другими модулями [18].

Одним из главных плюсов микросервисов является повышение отказоустойчивости, доступности и масштабируемости системы. В микросервисной архитектуре, как правило, перестанет работать только несколько модулей, а остальная система продолжит функционировать. Так как сами по себе модули автономны, то их относительно легко внедрять и тестировать производительность системы, что упрощает масштабируемость по горизонтальному пути [18].

Несмотря на множество плюсов, стоит отметить и некоторые минусы по сравнению с монолитной архитектурой. Разделение приложения на микросервисы, хотя и кажется простым, на самом деле является сложной технической задачей. Некоторые преимущества микросервисной архитектуры могут также стать минусами:

- Мониторинг: в случае с монолитом нужно отслеживать только один экземпляр, в то время как с микросервисами их могут быть десятки,

сотни или тысячи, что усложняет процесс мониторинга и управления системой [18].

- Технологическая независимость каждого модуля: каждый микросервис может быть написан на разных языках программирования и использовать различные технологии, что может создавать сложности [18].

2.5. 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 реализуется 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.

2.5.1. Домен «Займы»

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

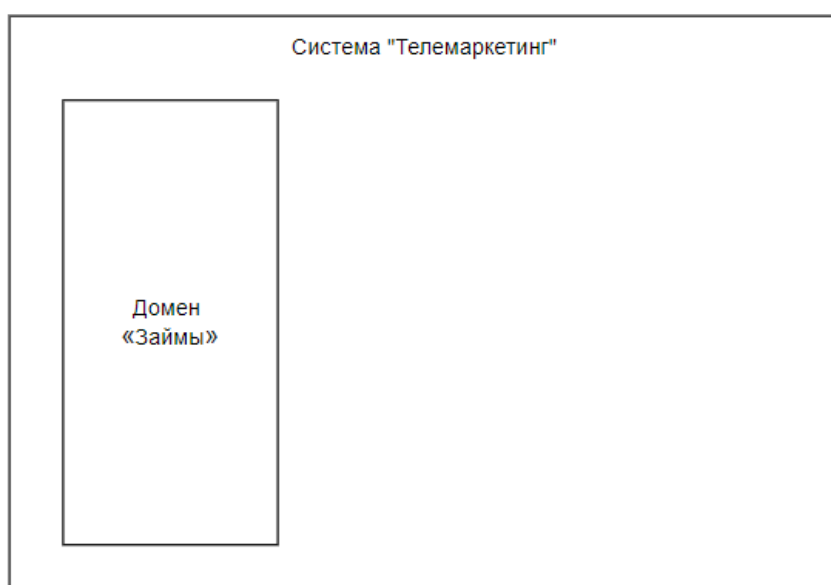


Рисунок 18 - Выделение доменной области «Займы»

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.

2.5.2. Архитектурный слой «Ядро»

Архитектурный слой «Ядро» представляет собой центральную архитектуру 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»

2.5.3. 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 [20].



Рисунок 20 - Выделение архитектурного слоя «Интеграции»

2.6. 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.

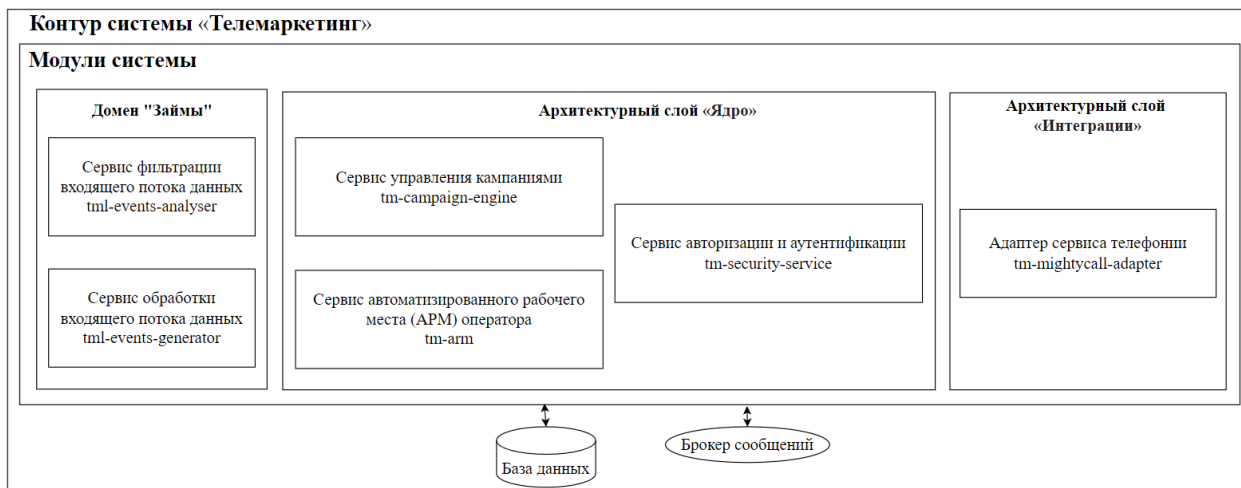


Рисунок 21 - Микросервисная архитектура системы «Телемаркетинг»

2.6.1. Сервис обработки входящего потока «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 получение займа в мобильном приложении «KoronaPay», но по каким-то причинам не довели её до конца;
- «pos_reject» – Клиенты, которые оформляли заявку на получение займа непосредственно в магазине-партнере, но получили автоматический отказ по каким-то причинам. Такие клиенты уходят на ручной разбор в нашу систему, чтобы оператор мог перенаправить их 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 себя проверки по столбцам на наличие обязательности и формата строки, описанных в *приложении А*.

Если форматный контроль выполнен неуспешно, 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».

На *рисунке 22* приведен пример конфигурации папок для хранения результатов обработки входных потоков данных.

```
20 # базовая папка с входящими csv-файлами
21 csv-events-base-folder: "/u/var/tml-events-generator/csv/"
22
23 events:
24   # папки для хранения результатов обработки
25   success-folder-name: "success"
26   error-folder-name: "error"
27   # описание входных потоков данных
28   data-streams:
29     - name: MOBILE_LOANS
30       folder: ${csv-events-base-folder}/mobile_loans
31       topic: "tml-incoming-events"
32     - name: POS_REJECT
33       folder: ${csv-events-base-folder}/pos_reject
34       topic: "tml-incoming-events"
```

Рисунок 22 - Конфигурация потоков данных

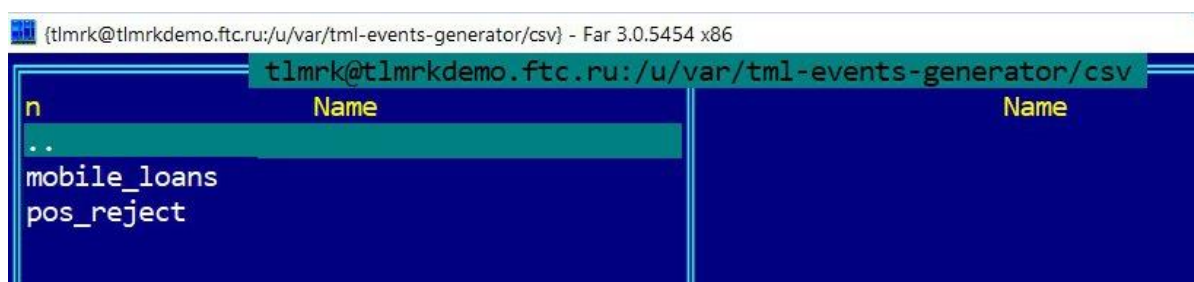


Рисунок 23 - Папки потоков данных



Рисунок 24 - Папки для успешной и неуспешной обработки файлов

2.6.2. Сервис фильтрации входящего потока «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;
- Гражданство;
- Возраст;
- Минимальное и максимальное значение грейс-периода займа;
- Наличие клиента в черном списке;
- Наличие доступных продуктов для клиента;
- И т.д.

На *рисунке 25* приведен пример конфигурации фильтров для потока клиентов из мобильного приложения «KoronaPay».

```

42 events-analyser:
43   campaigns:
44     - campaign-id: 1
45       data-streams: MOBILE_LOANS #Поток из мобильного приложения "Корона Pay"
46       request-statuses: registered, scored #Статус клиента: зарегистрирован, прошел первичную оценку в приложении
47       filters-order: #Порядок фильтров
48         - RESIDENCE #Гражданство
49         - AGE #Возраст
50         - PHONE_BLACK_LIST #Не находится в черном списке ГК "ЦФТ"
51         - ML_DEVICE #Данная проверка от сервиса ML должна быть пройдена
52         - ML_PHONE #Данная проверка от сервиса ML должна быть пройдена
53         - ML_FACE #Данная проверка от сервиса ML должна быть пройдена
54         - ML_DOCUMENT #Данная проверка от сервиса ML должна быть пройдена
55
56       ml-skip-error: false #При ошибке от ML сервисов не пропускаем дальше клиента
57       citizenship-unavailable-countries: #Недоступные страны
58         - BLR
59         - UKR
60       min-age: 18 #Минимальный возраст клиента
61       phone-templates: #Маски для номеров телефона
62         - 7\d{5}4\d{4}
63         - 7\d{5}3\d{4}

```

Рисунок 25 - Пример конфигурации фильтров

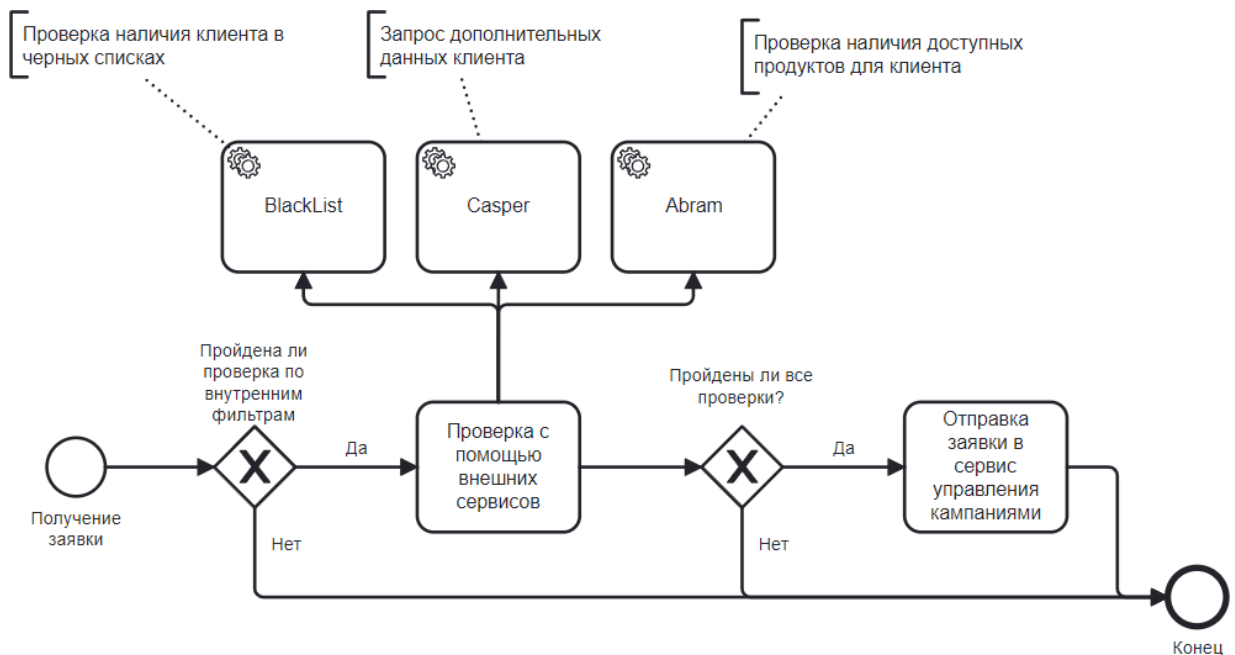


Рисунок 26 - Диаграмма моделирования бизнес-процесса «Фильтрации заявок через каскад фильтров»

Данный модуль кроме внутренних проверок применяет 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» с помощью брокера сообщений.

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

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

На вход поступают сообщения от модуля «tml-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 расписанию, 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 27*.

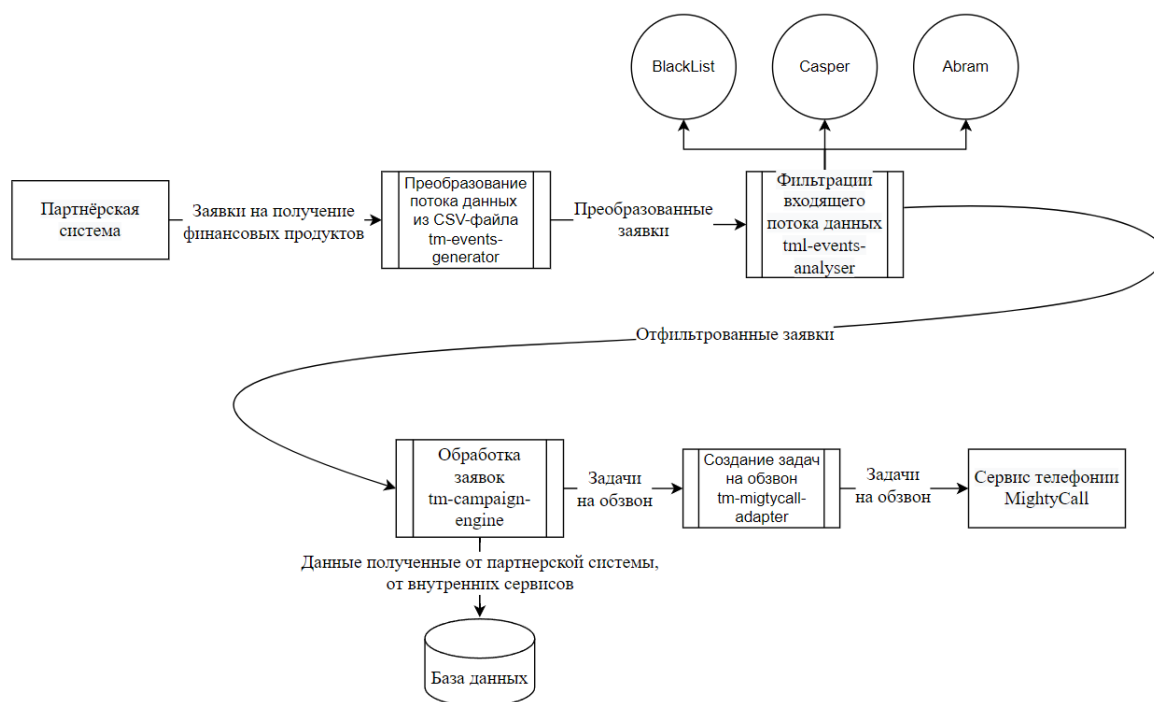


Рисунок 27 - Диаграмма потока данных «Создание задачи на обзвон»

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».

2.6.4. 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 авторизованным пользователям.

Модуль имеет три HTTP POST-метода:

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

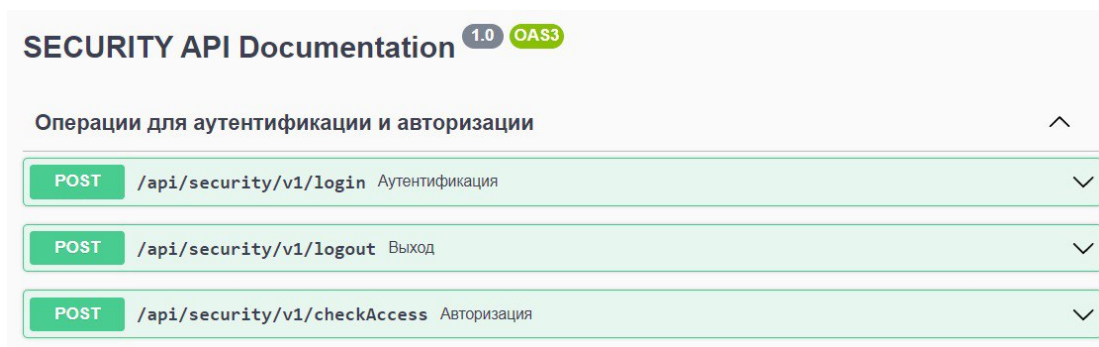


Рисунок 28 - Методы модуля «tm-security-service»

Входные и выходные данные методов, отраженных на *рисунке 28*, находятся в *приложении Б*, *приложении В*, *приложении Г* соответственно.

2.6.4.1. Аутентификация

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

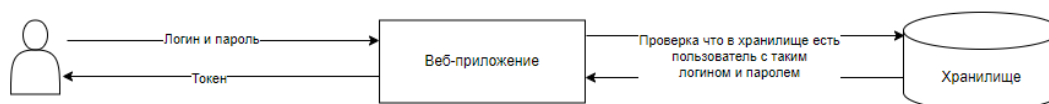


Рисунок 29 - Процесс аутентификации

На вход метода поступают данные, необходимые для аутентификации пользователя, — логин и пароль. Затем осуществляется поиск оператора в соответствующей таблице по логину и хеш-сумме от пароля.

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

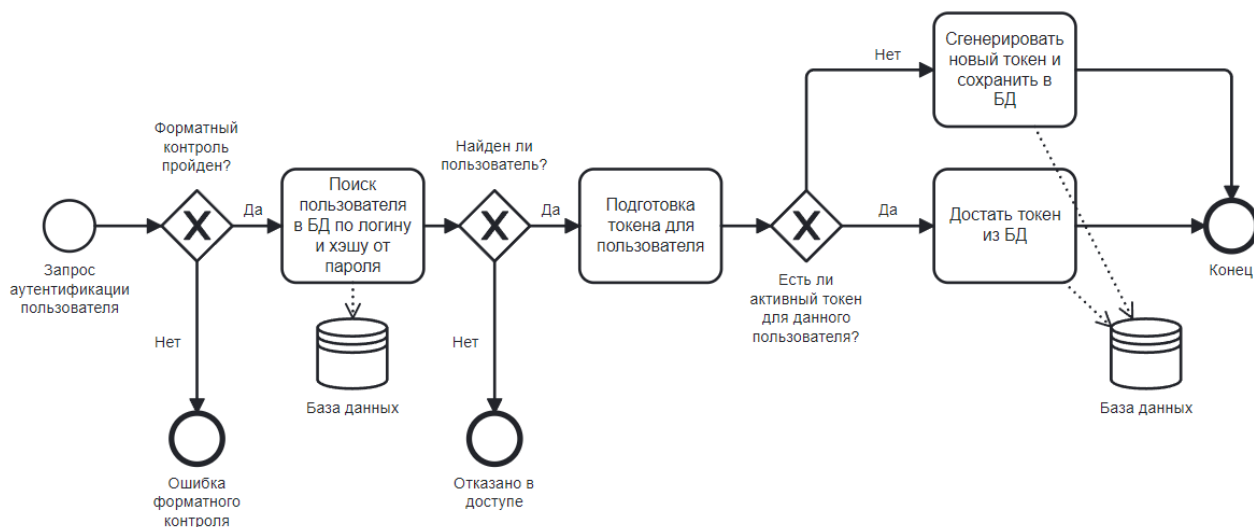


Рисунок 30 - Диаграмма моделирования бизнес-процесса «Запрос аутентификации пользователя»

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

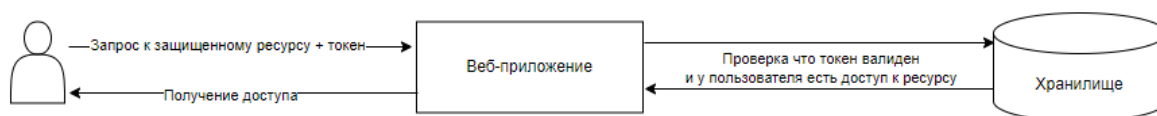


Рисунок 31 - Процесс авторизации

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

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

```
{
  "result": {
    "code": "OK",
  }
}
```

Рисунок 32 - Успешное прохождение авторизации

Иначе будет сообщение о том, что доступ запрещен.

```
{
  "result": {
    "code": "ACCESS_DENIED",
    "description": "Доступ запрещен"
  }
}
```

Рисунок 33 - Ошибка «ACCESS_DENIED»

Данный процесс подробно отражен ниже, на *рисунке 34*.

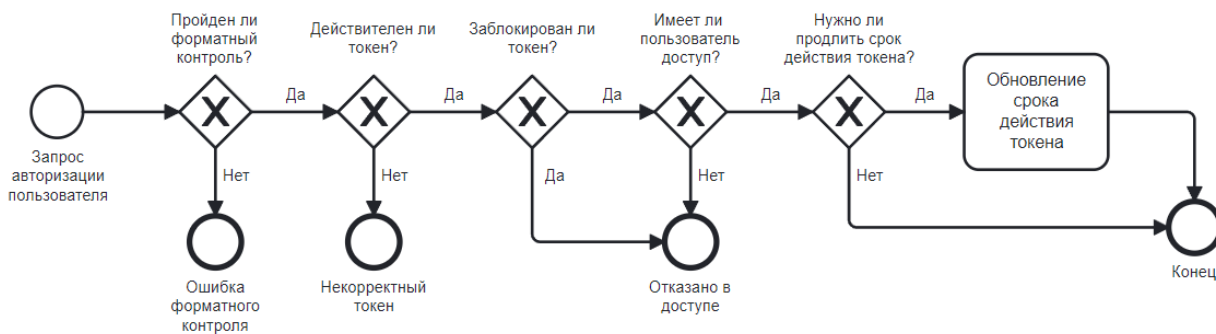


Рисунок 34 - Диаграмма моделирования бизнес-процесса «Запрос авторизации пользователя»

2.6.4.3. 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.

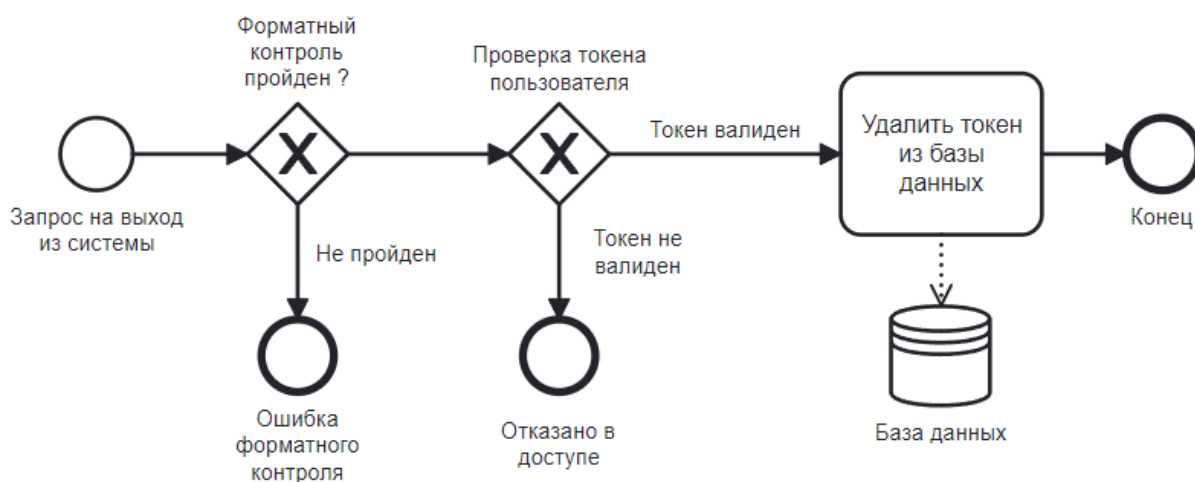


Рисунок 35 - Диаграмма моделирования бизнес-процесса «Запрос выхода из системы»

2.6.5. Автоматизированное рабочее место оператора «tm-arm»

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

| | | | |
|------|--|--|-----|
| POST | /api/arm/v1/login | Аутентификация | ✓ |
| GET | /api/arm/v1/listCommunications | Получение списка незавершенных коммуникаций текущего оператора | ✓ 🔒 |
| GET | /api/arm/v1/hasChanges | Получение информации о наличии изменений (для отображения) для оператора | ✓ 🔒 |
| GET | /api/arm/v1/getClientCard | Получение карточки клиента для текущей коммуникации | ✓ 🔒 |
| POST | /api/arm/v1/saveCommunicationDraft | Сохранение промежуточного результата коммуникации | ✓ 🔒 |
| POST | /api/arm/v1/saveCommunicationResult | Сохранение результата коммуникации | ✓ 🔒 |
| POST | /api/arm/v1/getClientCardByCommunicationId | Получение карточки клиента текущего оператора по ID коммуникации | ✓ 🔒 |
| GET | /api/arm/v1/logout | Выход из системы | ✓ 🔒 |

Рисунок 36 - Интерфейс АРМ оператора

На *рисунке 36* изображен API сервиса. Входные и выходные данные всех запросов приведены в *приложении Д, приложении Е, приложении Ж, приложении З, приложении И, приложении К, приложении Л и приложении М* соответственно.

Как видно, АРМ 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».

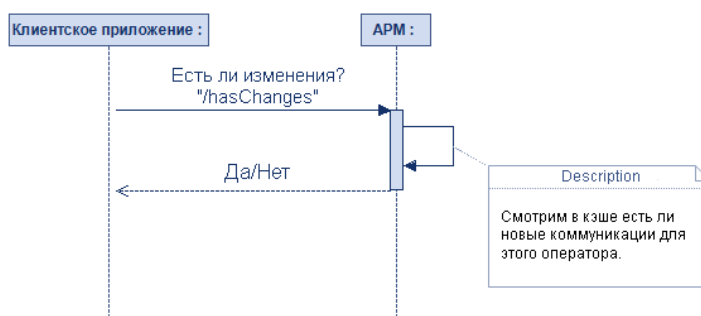


Рисунок 37 - Диаграмма последовательности «Процесс опроса АРМа клиентским приложением»

Данные о новых звонках 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».

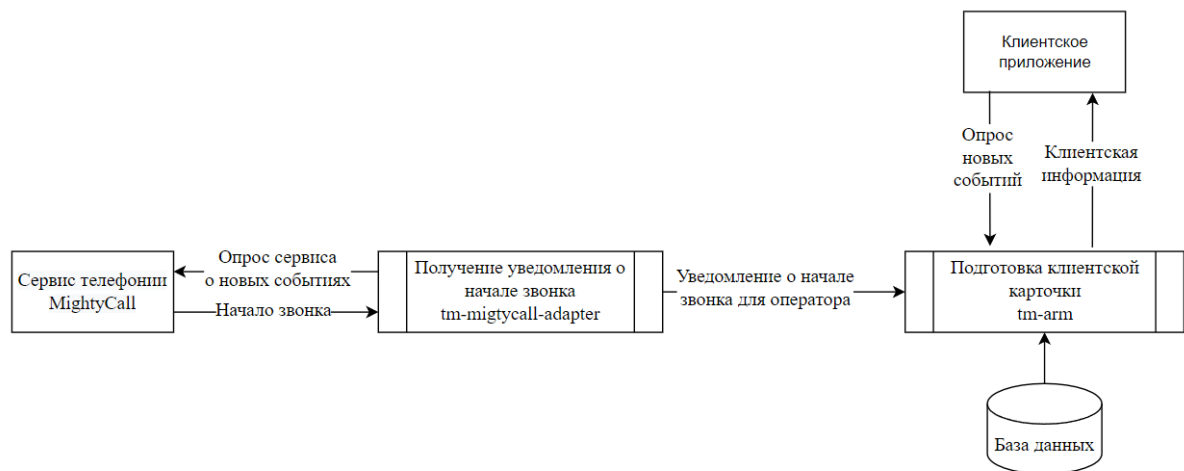


Рисунок 38 - Диаграмма потока данных «Обработка начала звонка»

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

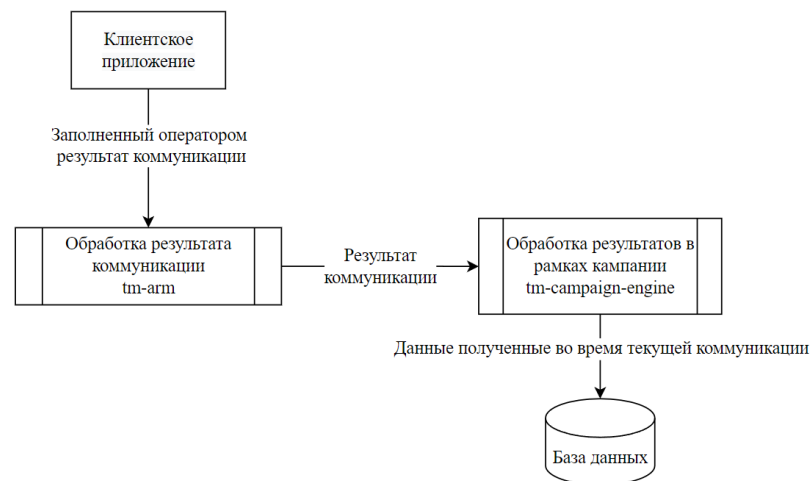


Рисунок 39 - Диаграмма потока данных «Обработка результатов звонка»

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 60 секунд после того, как оператор закончил звонок с клиентом.

2.6.6. 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.

2.6.6.1. 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.

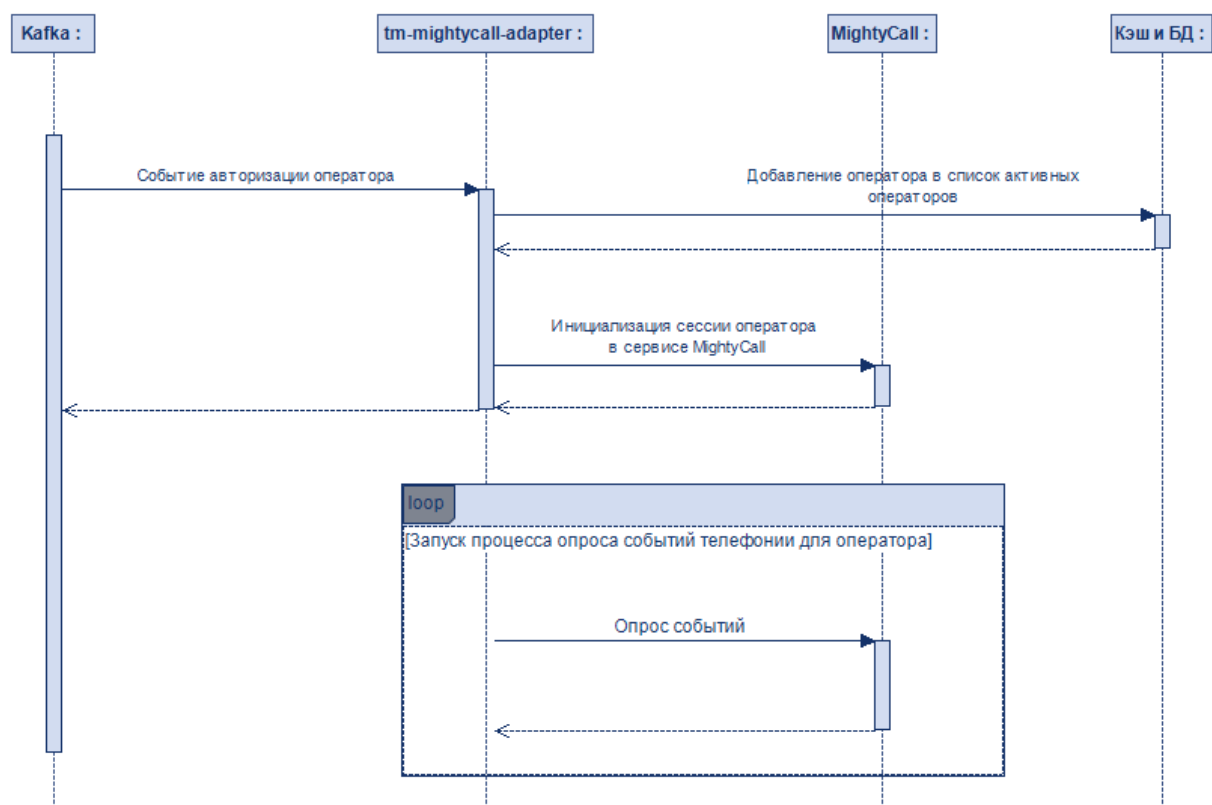


Рисунок 40 - Диаграмма последовательности «Открытие сессии оператора»

2.6.6.2. Заккрытие сессии оператора

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



Рисунок 41 - Диаграмма последовательности «Закрытие сессии оператора»

2.6.6.3. Процесс опроса событий для оператора

По каждому активному оператору адаптер опрашивает MightyCall на наличие новых событий, для того чтобы передать эту информацию в АРМ с помощью брокера сообщений. Данный процесс отражен на *рисунке 42*.

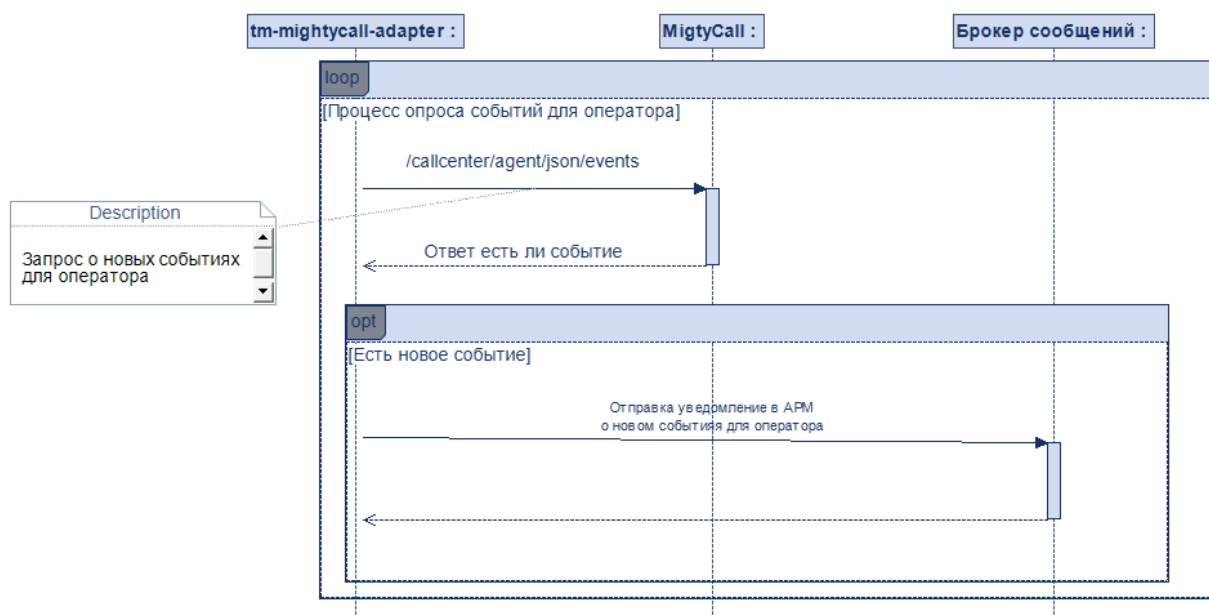


Рисунок 42 - Диаграмма последовательности «Процесс опроса событий для оператора»

2.6.6.4. 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 существующей 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 43*.

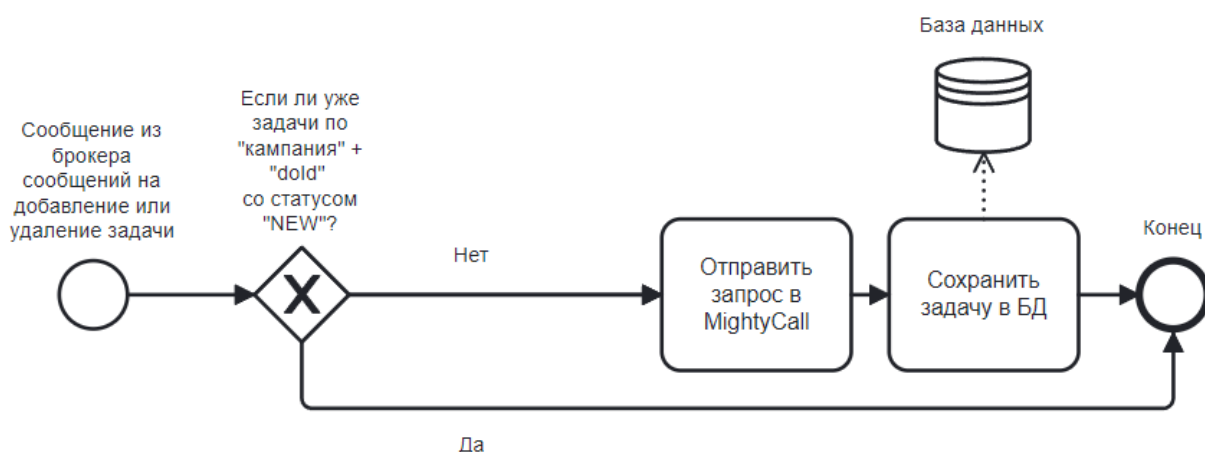


Рисунок 43 - Диаграмма моделирования бизнес-процесса «Работа с очередью задач MightyCall»

2.7. Проектирование схемы хранения данных

2.7.1. Модель «сущность-связь»

Основной задачей концептуального проектирования базы данных является определение сущностей проектируемой системы и связей между

сущностями. Один из методов такого проектирования является Entity Relation (ER) модель, или модель «сущность-связь».

Базовыми понятиями ER-модели являются: сущность, связь, атрибуты.

Сущность – объект реального мира, информация о котором должна храниться в базе данных. Сущность подразделяется на такие понятия, как класс и экземпляр. Класс – некоторый объект реального мира. Экземпляр класса – конкретный уникальный объект класса [22]. Сущности подразделяются на **сильные** и **слабые**. Существование сильной сущности не зависит от какого-либо другого типа сущности. Существование слабой сущности зависит от какого-либо другого типа сущности, поэтому слабая сущность обязательно должна быть связана с сильной [23].

Атрибут – свойство сущности или связи, поэтому сущность можно рассматривать как набор атрибутов. Атрибут, состоящий из одного элемента данных, – простой, а многоэлементные атрибуты – составные [24].

Ключ сущности – потенциальный ключ, который выбран в качестве рабочего [24].

Связь выражает взаимодействия между сущностями [24]. **Степень связи** – количество классов (типов) сущностей, охваченных данной связью. Связи могут быть различной степени. Наиболее распространена бинарная связь (то есть между двумя сущностями) [24].

Кратность связи – количество возможных элементов одной сущности, связанных с одним экземпляром другой сущности [24].

У связей могут быть следующие разновидности:

- **один-к-одному** – один объект первого типа связан с одним объектом второго типа [23];
- **один-ко-многим** – один объект первого типа связан с несколькими объектами второго типа [23];
- **многие-ко-многим** – каждый элемент первого типа может быть связан с несколькими элементами второго типа, и каждый элемент второго типа может быть связан с несколькими элементами первого типа [23];

Для данной модели необходимо определить состав сущностей, состав атрибутов и ключи всех сущностей, связи между сущностями и их кратности.

Условные обозначения: первичный ключ, внешний ключи.

Сущность – **Кампании**. Атрибуты: Уникальный идентификатор, Название, Описание, Домен. Все атрибуты, кроме описания, обязательны для полного определения каждой кампании.

Сущность – **Операторы**. Атрибуты: Уникальный идентификатор, Логин, Пароль, Блокировка, Имя, Фамилия, Токен, Срок действия токена, Логин MightyCall, Пароль MightyCall, Сессия MightyCall. Все атрибуты обязательны для полного определения каждого оператора.

Сущность – **Задачи кампаний**. Атрибуты: Уникальный идентификатор, Номер телефона, Информация о клиенте, Время отложенного запуска, Тип, Статус, Тип MightyCall, Статус MightyCall, Уникальный идентификатор кампании. Все атрибуты обязательны для полного определения каждой задачи кампании.

Сущность – **Коммуникации**. Атрибуты: Уникальный идентификатор, Статус, Результат, Уникальный идентификатор задачи кампании, Уникальный идентификатор оператора. Все атрибуты обязательны для полного определения каждой коммуникации.

К сильным сущностям можно отнести «Кампании» и «Операторы». Сущность «Задачи кампаний» является слабой, поскольку без кампании не будет существовать. Сущность «Коммуникации» является слабой, потому что не может существовать без задачи кампании и оператора.

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В данной работе используется метод «Crow's foot notation». Сущности будут показаны как прямоугольники. Внутри прямоугольников будут описаны атрибуты сущности. РК возле атрибута означает Primary Key (первичный ключ) сущности. FK возле атрибута означает Foreign Key (внешний ключ).

Связи между сущностями будут изображены с помощью линий, показывая при этом кардинальности связи. [21]

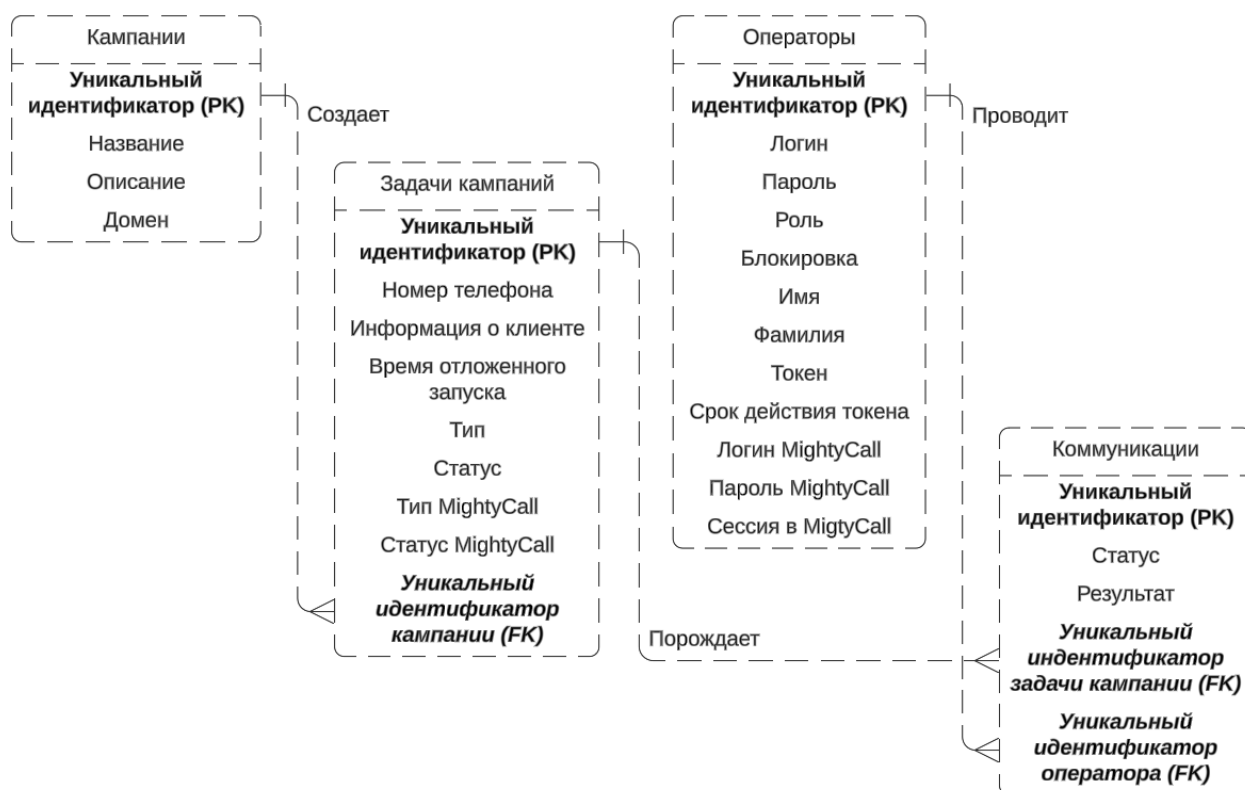


Рисунок 44 - Диаграмма «сущность-связь»

2.7.2. Нормализация отношений

Нормализация – процедура преобразования исходного набора отношений базы данных к виду нормальных форм на основе их ключей и функциональных связей между атрибутами [24].

Всего выделяют 6 нормальных форм (НФ). Каждая следующая нормальная форма должна удовлетворять критериям предыдущей и включать свои требования, тем самым устраняя определенный набор аномалий. На практике добиваются второй нормальной формы (2НФ), но, чтобы не было аномалий при обновлении данных, добьемся нормальной формы Бойса-Кодда (НФБК).

Ниже, в *таблице 1*, приведено описание нормальных форм до НФБК.

Таблица 1 - Нормальные формы

| Нормальная форма | Условие |
|--|---|
| Первая нормальная форма (1НФ) | Отношение находится в первой нормальной форме, если все его атрибуты являются атомарными и нет повторяющихся атрибутов в таблице [24]. |
| Вторая нормальная форма (2НФ) | Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый его неключевой атрибут неприводимо зависит от первичного ключа. Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость [24]. |
| Третья нормальная форма (3НФ) | Отношение находится в третьей нормальной форме, когда находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа [24]. |
| Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы) | Отношение находится в нормальной форме Бойса-Кодда, если оно находится в третьей нормальной форме и в нем отсутствует зависимость атрибутов составного первичного ключа от неключевых атрибутов. Особое отношение этой форме уделяется, когда в отношении есть несколько составных потенциальных ключей с общими атрибутами [24]. |

В разрабатываемой БД нет повторяющихся кортежей, составные атрибуты разбиваются на атомарные, поэтому отношения находятся в 1НФ. 2НФ соблюдена, поскольку все неключевые атрибуты зависят от первичного ключа. Чтобы добиться 3НФ нужно:

- Вынести данные об операторе MightyCall из сущности «Операторы» в отдельную сущность «Операторы MightyCall»;
- Информацию об активных сессиях операторов необходимо вынести в отдельную сущность «Активные операторы MightyCall»;
- Данные токена оператора из сущности «Операторы» вынести в отдельную сущность «Аутентификации операторов»;

- Из сущности «Задачи кампаний» вынести данные о задачах MightyCall в отдельную сущность «Задачи MightyCall».

У сущностей нет составных ключей, следовательно, все отношения находятся в НФБК.

В результате нормализации отношений приведем окончательный список сущностей с их атрибутами. Условные обозначения: первичный ключ, внешний ключ.

Сущность – **Кампании**. Атрибуты: Уникальный идентификатор, Название, Домен, Описание.

Сущность – **Операторы**. Атрибуты: Уникальный идентификатор, Логин, Пароль, Роль, Блокировка, Имя, Фамилия.

Сущность – **Аутентификации операторов**. Атрибуты: Уникальный идентификатор, Срок действия, Токен, Уникальный идентификатор оператора.

Сущность – **Операторы MightyCall**. Атрибуты: Уникальный идентификатор, Логин MightyCall, Пароль MightyCall, Уникальный идентификатор оператора.

Сущность – **Активные операторы MightyCall**. Атрибуты: Уникальный идентификатор, Уникальный идентификатор оператора.

Сущность – **Задачи кампаний**. Атрибуты: Уникальный идентификатор, Номер телефона, Информация о клиенте, Время отложенного запуска, Тип, Статус, Уникальный идентификатор кампании.

Сущность – **Задачи MightyCall**. Атрибуты: Уникальный идентификатор, Тип MightyCall, Статус MightyCall, Уникальный идентификатор задачи кампании.

Сущность – **Коммуникации**. Атрибуты: Уникальный идентификатор, Статус, Результат, Уникальный идентификатор задачи кампании, Уникальный идентификатор оператора.

2.7.3. Таблицы и пользовательские типы данных

Таблица 2 – Кампании (TM\$CAMPAIGNS)

| Название | Тип | Описание |
|-------------|--|--------------------------------------|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| NAME | VARCHAR(64), NOT NULL | Название |
| DOMAIN | DOMAINS, NOT NULL | Домен, к которому относится кампания |
| DESCRIPTION | VARCHAR(256) | Описание |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |

Был добавлен пользовательский тип данных «Домены» (DOMAINS), который может принимать следующие значения:

- LOANS.

Таблица 3 - Задачи кампаний (TM\$CAMPAIGN_TASKS)

| Название | Тип | Описание |
|-------------|--|-----------------------------------|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| PHONE | VARCHAR(15), NOT NULL | Номер клиента |
| CLIENT_INFO | JSONB, NOT NULL | Информация о клиенте |
| TASK_TIME | TIMESTAMP, NOT NULL | Время отложенного запуска задачи |
| TASK_TYPE | CAMPAIGN_TASK_TYPES, NOT NULL | Тип задачи |
| TASK_STATE | CAMPAIGN_TASK_STATES, NOT NULL | Статус задачи |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| CAMPAIGN_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор кампании |

Был добавлен пользовательский тип данных «Типы задач кампании» (CAMPAIGN_TASK_TYPES), который может принимать следующие значения:

- PHONE_MIGHTYCALL.

Также был введен тип данных «Статусы задач кампании» (CAMPAIGN_TASK_STATES), он принимает следующие значения:

- NEW;
- IN_PROGRESS;
- DONE.

Таблица 4 - Коммуникации (TM\$COMMUNICATIONS)

| Название | Тип | Описание |
|------------------|--|--|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| STATE | COMMUNICATION_STATES, NOT NULL | Статусы коммуникации |
| RESULT_DETAILS | JSONB, NOT NULL | Результат коммуникации |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| CAMPAIGN_TASK_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор задачи кампании |
| OPERATOR_ID | BIGINT, NOT NULL | Уникальный идентификатор оператора |

Был добавлен пользовательский тип данных «Статус коммуникации» (TM\$COMMUNICATION_STATES), который может принимать следующие значения:

- NEW;
- DRAFT;
- FIXED.

Таблица 5 - Операторы (TM\$OPERATORS)

| Название | Тип | Описание |
|------------|--|--------------------------------|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| LOGIN | VARCHAR(64), NOT NULL | Логин оператора |
| PASSWORD | VARCHAR(256), NOT NULL | Зашифрованный пароль оператора |
| LOCKED | BOOLEAN, NOT NULL | Флаг заблокирован ли оператор |
| ROLE | ROLES, NOT NULL | Роль оператора |
| FIRST_NAME | VARCHAR(32), NOT NULL | Имя оператора |
| LAST_NAME | VARCHAR(32), NOT NULL | Фамилия оператора |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |

Был добавлен пользовательский тип данных «Роли пользователя» (ROLES), который может принимать следующие значения:

- OPERATOR.

Таблица 6 - 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(512), NOT NULL | Токен |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| OPERATOR_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 7 - Оператор в MightyCall (TM\$MIGHTYCALL_OPERATORS)

| Название | Тип | Описание |
|-------------|--|---|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| LOGIN | VARCHAR(64), NOT NULL | Логин оператора |
| PASSWORD | VARCHAR(256), NOT NULL | Зашифрованный пароль оператора для системы MightyCall |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| OPERATOR_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 8 – Открытые сессии операторов в MighyCall
(TM\$MIGHTYCALL_ACTIVE_OPERATORS)

| Название | Тип | Описание |
|-------------|--|---|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| OPERATOR_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор оператора системы «Телемаркетинг» |

Таблица 9 - Задачи для MightyCall (TM\$MIGHTYCALL_TASKS)

| Название | Тип | Описание |
|------------------|--|--|
| ID | PRIMARY KEY, BIGSERIAL, NOT NULL | Уникальный идентификатор |
| TASK_TYPE | MIGHTYCALL_TASK_TYPES, NOT NULL | Тип задачи |
| TASK_STATE | MIGHTYCALL_TASK_STATES, NOT NULL | Состояние задачи |
| WTIME | TIMESTAMP, NOT NULL | Время создания записи |
| UTIME | TIMESTAMP, NOT NULL | Время обновления записи |
| CAMPAIGN_TASK_ID | FOREIGN KEY, BIGINT, NOT NULL | Уникальный идентификатор задачи кампании |

Был добавлен пользовательский тип данных «Типы задачи в MightyCall» (MIGHTYCALL_TASK_TYPES), который может принимать следующие значения:

- ADD;
- DELETE.

Также был введен тип данных «Статусы задачи в MightyCall» (MIGHTYCALL_TASK_STATES), он принимает следующие значения:

- NEW;
- IN_PROGRESS;
- DONE.

Ниже, на *рисунке 45* изображена итоговая схема спроектированных таблиц и пользовательских типов данных.

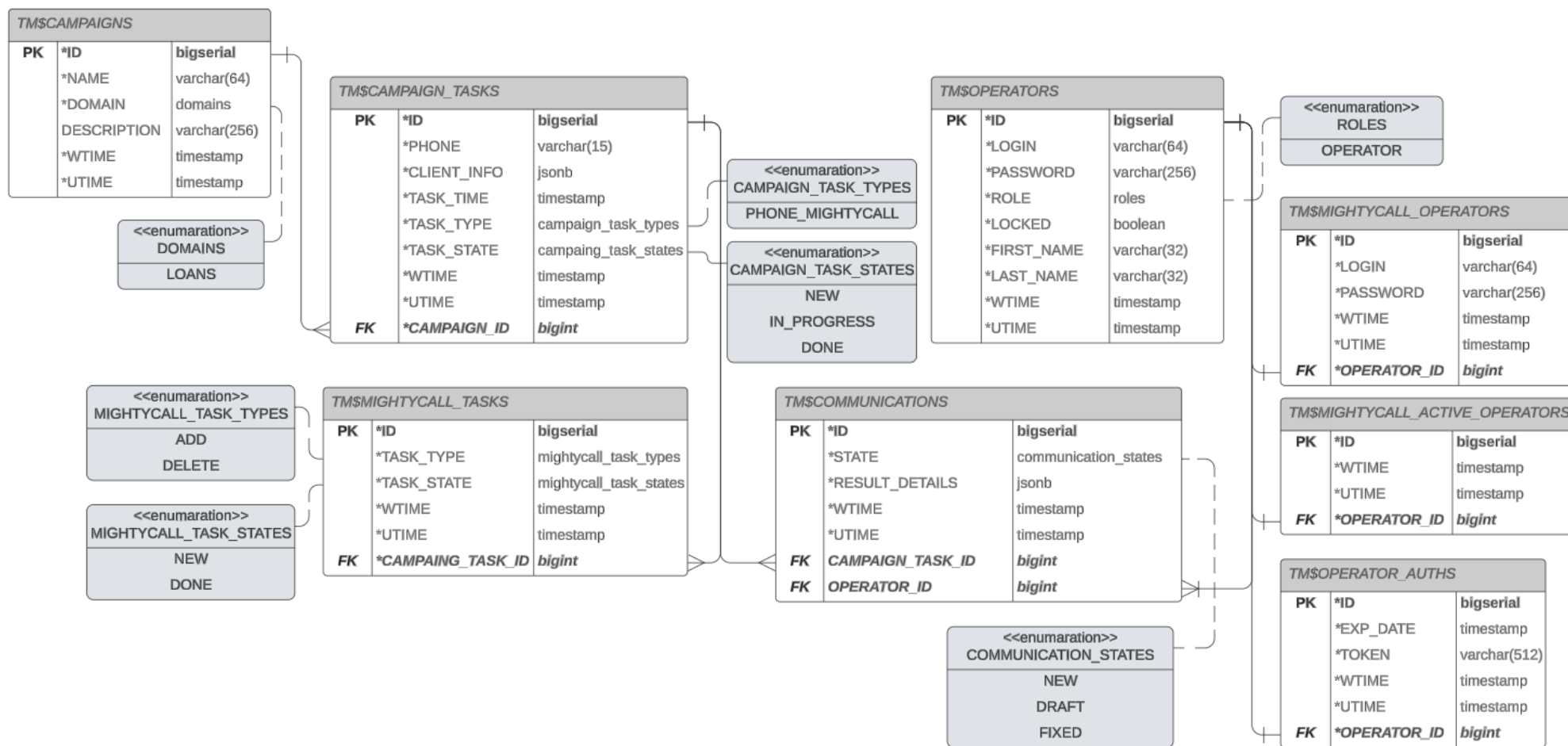
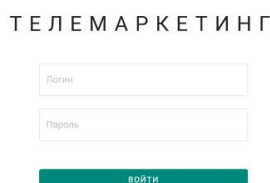


Рисунок 45- Схема базы данных

ГЛАВА 3. ОПИСАНИЕ И ТЕСТИРОВАНИЕ ГОТОВОЙ СИСТЕМЫ

Как говорилось ранее, в ГК «ЦФТ» уже есть готовое клиентское приложение, с помощью которого можно протестировать разработанный функционал системы «Телемаркетинг», поэтому в данной главе всё взаимодействие будет происходить с помощью него.

Для взаимодействия с системой необходимо пройти процесс аутентификации. Форма клиентского приложения на *рисунке 46* нужна для входа оператора в систему:



ТЕЛЕМАРКЕТИНГ

Логин

Пароль

Войти

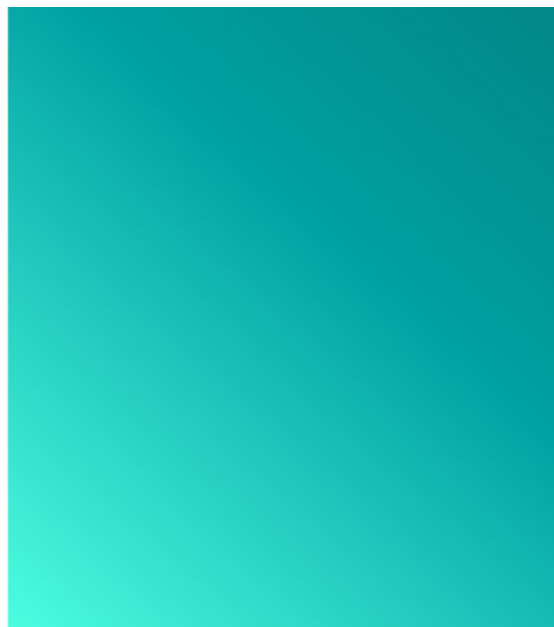


Рисунок 46 - Форма входа в систему

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

Ниже, на *рисунке 47*, у оператора для всех звонков результаты разговоров завершены.

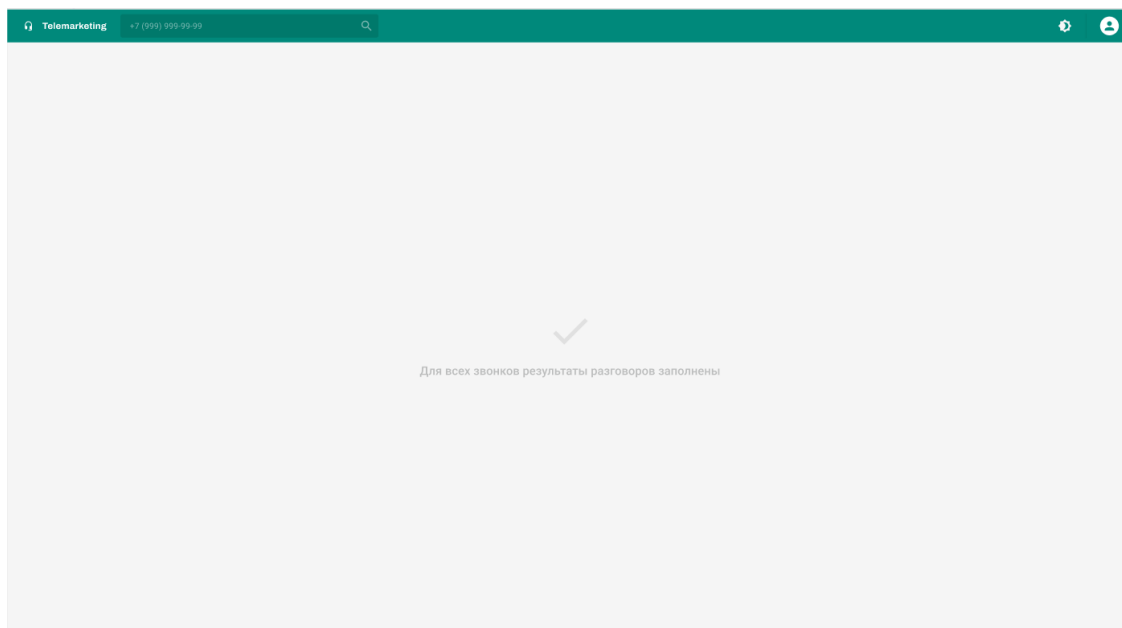


Рисунок 47 - Главный экран. Нет незаполненных результатов коммуникаций

Если у оператора есть незаполненные результаты, то они будут отображены на главном окне в разделе «Карточки на доработку». На *рисунке 48* у оператора есть 4 коммуникации, результат которых он должен заполнить.

 The screenshot shows the main interface of the Telemarketing application with the section 'Карточки на доработку' (Cards for rework) active. It displays a table with 4 rows of communication data. Each row has a green phone icon on the left. The 'Результат' (Result) column for all rows shows 'Не сохранен' (Not saved) in red text.

| Дата | Номер заявки | Тип | Кампания | Телефон | Результат |
|------------------|--------------|--------|-----------------|--------------------|-------------|
| 01.12.2019 12:45 | 123456789 | Заявка | Доведи до займа | +7 (***) ***-67-89 | Не сохранен |
| 01.12.2019 12:45 | 123456789 | Заявка | Доведи до займа | +7 (***) ***-67-89 | Не сохранен |
| 01.12.2019 12:45 | 123456789 | Заявка | Доведи до займа | +7 (***) ***-67-89 | Не сохранен |
| 01.12.2019 12:45 | 123456789 | Заявка | Доведи до займа | +7 (***) ***-67-89 | Не сохранен |

Рисунок 48 - Главный экран. Есть незаполненные результаты коммуникаций

Кроме этого, как только оператор успешно авторизировался в системе, в клиентском приложении запускается фоновый процесс опроса АРМ, который отправляет запрос `/hasChanges` для получения уведомления о начале звонка.

Как только клиентское приложение получает положительный результат в ответ на запрос `/hasChanges`, оно делает запрос `/getClientCard`, чтобы

получить карточку клиента, в которой содержится собранная системой информация о клиенте:

- Гражданство;
- Возраст;
- Телефон;
- ФИО;
- История заявок;
- История займов;
- Кампания.

Telemarketing +7 (999) 999-99-99

ХОЖАНАЗАРОВ ЯСУРБЕК МИРЗАНАЗАРОВИЧ
KHOZHANAZAROV YASURBEK MIRZANAZAROVICH

Уснувший клиент

Гражданство: Таджикистан, Возраст: 27 лет, Телефон: +7 (***) ***-67-89

| Дата (МСК) | Номер | Тип | Статус | Продукт | Сумма | Срок | Грейс | ОС | Флоу | Заполнение анкеты | Зачисление |
|------------------|-----------|--------|-----------------|---------|----------|--------|-------|----|------|----------------------|------------|
| 01.12.2019 12:45 | 123456789 | ЗАЯВКА | Зарегистрирован | pdf | 10 000 P | 21 дн | 25 дн | PI | PI | Селфи - ДУЛ - ДолДок | all |
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Оформлен | pdf | 10 000 P | 12 мес | 2 мес | | | | last |
| 01.12.2019 12:45 | 123456789 | ЗАЯВКА | Зарегистрирован | pdf | 10 000 P | 21 дн | 25 дн | PI | PI | Селфи - ДУЛ - ДолДок | |

| Дата | Номер | Тип | Кампания | Результат разговора | Комментарий | Оператор | Телефон |
|------------------|-----------|--------|-----------------|--|-----------------------------|------------------|--------------------|
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Доведи до займа | Результат: Сброс звонка | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Доведи до займа | Результат: Отказ от взаимодействия | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЯВКА | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЯВКА | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЯВКА | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |
| 01.12.2019 12:45 | 123456789 | ЗАЙМ | Доведи до займа | Причина: Не было с собой документа; Результат: Все понятно, заполнит позже | Все понятно, заполнит позже | Абросимова Елена | +7 (***) ***-67-89 |

Результат разговора

Причина

Результат

Комментарий

СОХРАНИТЬ

Рисунок 49 - Форма карточки клиента

В самом конце, когда звонок подходит к завершению, оператор должен заполнить результат проведенной коммуникации. На *рисунке 50* представлен блок заполнения результата коммуникации.

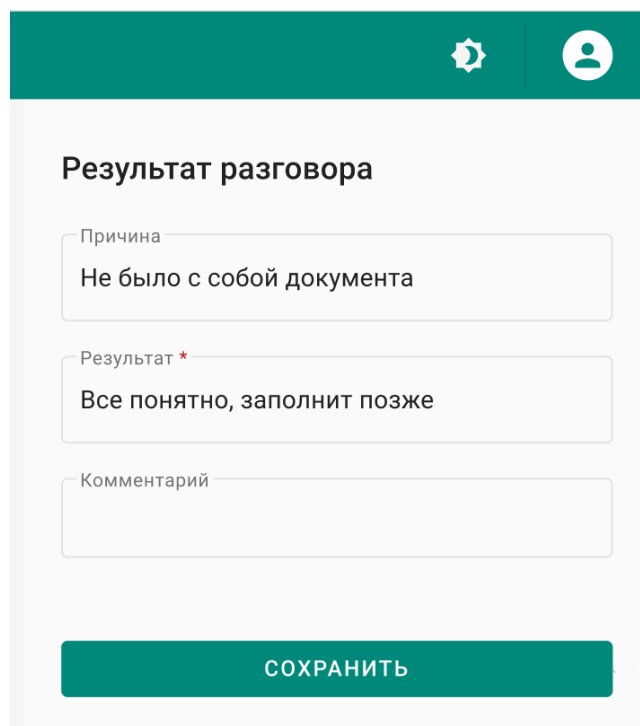


Рисунок 50 - Блок заполнения результата разговора

Так как работа колл-центра поставлена на поток, то у оператора ограниченное количество времени на заполнение результата коммуникации – 1 минута.

Если оператор успел заполнить результат, то он успешно сохраняется запросом ``/saveCommunicationResult`` в АРМ. В клиентском приложении об этом сообщает статус, который показан на *рисунке 51*.

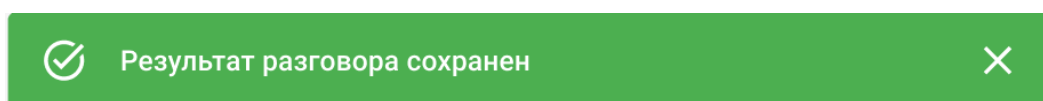


Рисунок 51 - Результат успешно сохранен

Если оператор не успел заполнить результат, то клиентское приложение отправляет запрос ``/saveCommunicationDraft`` в АРМ, который сохраняет черновик, чтобы оператор заполнил результат позже. В клиентском приложении об этом сообщает статус, который изображен на *рисунке 52*.



Результат разговора по предыдущему клиенту не сохранен.
Карточка вернется на доработку.



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

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

ЗАКЛЮЧЕНИЕ

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

- Интеграция с закрытыми сервисами ГЦ «ЦФТ»;
- Интеграция с телефонией;
- Автоматическая обработка входящего потока заявок;
- Обработка и подготовка данных о клиенте;
- Обработка результатов коммуникаций;
- Удобное автоматизированное рабочее место оператора.

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

Система, безусловно, требует доработок, в перспективе – добавление функционала для главных операторов: возможность загрузки 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? [Электронный ресурс]. - URL: <https://www.unisender.com/ru/glossary/chto-takoe-telemarketing> (Дата обращения: 30.11.2023)
2. Telemarketing 101: Definition, History, Strategy, And Benefits. [Электронный ресурс]. - URL: <https://www.freedomtoascend.com/marketing/telemarketing/> (Дата обращения: 30.11.2023)
3. Telemarketing For Financial Services: Navigating The Path To Success. [Электронный ресурс]. - URL: <https://www.appointmentsetting.co.uk/telemarketing-for-financial-services-navigating-the-path-to-success/> (Дата обращения: 14.12.2023)
4. 1С:CRM. [Электронный ресурс]. - URL: <https://1crm.ru/help/telemarketing/> (Дата обращения: 12.01.2024)
5. 1С:CRM. Обзор. [Электронный ресурс]. - URL: <https://habr.com/ru/companies/trinion/articles/296406/> (Дата обращения 12.01.2024)
6. CRM для телемаркетинга. [Электронный ресурс]. - URL: <https://megacrm.ru/telemarketing> (Дата обращения 13.01.2024)
7. Работа интеграции MegaCRM и Облачной АТС Дом.ру Бизнес. [Электронный ресурс]. - URL: <https://wiki.domru.biz/megacrm> (Дата обращения 13.01.2024)
8. Как работать со звонками в MegaCRM. [Электронный ресурс]. - URL: <https://help.megagroup.ru/kak-rabotat-so-zvonkami-v-megacrm> (Дата обращения 13.01.2024)
9. Дашборд в Коллтрекинге Roistat. [Электронный ресурс]. - URL: https://help-ru.roistat.com/features/Calltracking/Svodnyi_otchet/ (Дата обращения 14.01.2024)

10. What is REST? [Электронный ресурс]. - URL: <https://restfulapi.net/> (Дата обращения 16.01.2024)
11. Что такое Java? [Электронный ресурс]. - URL: <https://aws.amazon.com/ru/what-is/java/> (Дата обращения 17.02.2024)
12. Introduction to Spring Framework. [Электронный ресурс]. - URL: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html> (Дата обращения 24.02.2024)
13. Hibernate ORM. [Электронный ресурс]. - URL: <https://hibernate.org/orm/> (Дата обращения 26.02.2024)
14. About PostgreSQL. [Электронный ресурс]. - URL: <https://www.postgresql.org/about/> (Дата обращения 1.03.2024)
15. APACHE KAFKA QUICKSTART. [Электронный ресурс]. - URL: <https://kafka.apache.org/quickstart> (Дата обращения 2.03.2024)
16. Возможности MightyCall. [Электронный ресурс]. - URL: <https://mightycall.ru/features/> (Дата обращения 5.03.2024)
17. Что такое микросервисная архитектура. [Электронный ресурс]. - URL: <https://selectel.ru/blog/what-is-microservice-architecture/> (Дата обращения 7.03.2024)
18. Domain driven design. [Электронный ресурс]. - URL: <https://simpleone.ru/glossary/domain-driven-design/> (Дата обращения 10.03.2024)
19. Архитектурный слой (в корпоративной разработке). Понятие, определение, представление. [Электронный ресурс]. - URL: <https://habr.com/ru/articles/511252/> (Дата обращения 14.03.2024)
20. Джонсон Р., Гамма Э. Паттерны объектно-ориентированного проектирования // -2022, - с. 171. ISBN 978-5-4461-1595-2ю (Дата обращения 18.03.2024)
21. ER-модель, Crow's Foot [Электронный ресурс]. - URL: <https://ru.wikipedia.org/wiki/ER-модель> (Дата обращения 12.04.2024)
22. Курс лекций «Управление данными» [Электронный ресурс] : конспект лекций /Ю. И. Щетинин ; Новосиб. гос. техн. ун-т (Дата обращения 13.04.2024)

23. Нормализация отношений. Шесть нормальных форм [Электронный ресурс]. URL: - <https://habr.com/ru/post/254773/> (Дата обращения 20.04.2024)

ПРИЛОЖЕНИЕ

Приложение А – Входные данные сервиса «tml-events-generator»

| # | Описание | Обязательность | Формат строки | Пример | Комментарий | Обоснование необходимости |
|---------------------|-----------------------------------|----------------|--|----------------------|--|---------------------------|
| 1. Заголовок заявки | | | | | | |
| 1.1 | Номер заявки | Да | Число | 34697463 | | ИД |
| 1.2 | Дата создания заявки / черновика | Да | DD.MM.YYYY | 07.10.2022 | таймзона MCK | фильтр |
| 1.3 | Время создания заявки / черновика | Да | hh:mm:ss | 00:00:00 | таймзона MCK | фильтр |
| 1.4 | Дата значимого изменения заявки | Да | DD.MM.YYYY | 07.10.2022 | Дата последнего изменения заявки таймзона MCK | фильтр |
| 1.5 | Время значимого изменения заявки | Да | hh:mm:ss | 00:00:00 | Время последнего изменения заявки таймзона MCK | фильтр |
| 1.6 | Бизнес статус заявки | Да | Строка | created | Возможные значения: <ul style="list-style-type: none"> created - Создана registered - Зарегистрирована scored - Скоринг пройден canceled - Отменена | фильтр |
| 1.7 | Дата значимого изменения займа | Нет | DD.MM.YYYY | 07.10.2022 | Дата последнего изменения займа таймзона MCK | фильтр |
| 1.8 | Время значимого изменения займа | Нет | hh:mm:ss | 00:00:00 | Время последнего изменения займа таймзона MCK | фильтр |
| 1.9 | Бизнес статус займа | Нет | Строка | recieved | Возможные значения: <ul style="list-style-type: none"> Пустое значение transferred - Перевод отправлен received - Выдан canceled - Отменен repaid - Погашен / закрыт | фильтр |
| 1.10 | Результат скоринга | Нет | Строка | approved | Заполняется, только если Бизнес статус заявки = scored - Скоринг пройден Возможные значения: <ul style="list-style-type: none"> Пустое значение approved - Одобрено rejected - Отказано | фильтр |
| 2. Клиент | | | | | | |
| 2.1 | Тип клиента | Да | Строка | pioneer | Возможные значения: <ul style="list-style-type: none"> pioneer - первичник repeater - повторник | фильтр |
| 2.2 | Номер телефона | Да | 11 цифр без "+" в начале, первая цифра 7 | 79057896050 | 11 цифр, без "+" в начале первая цифра 7 (т.к. звоним только по РФ) | ИД, фильтр, ML |
| 2.3 | faceID | Нет | Строка | 2Zz484MBY7almrgo4-B0 | | ML |
| 2.4 | Гражданство | Нет | Строка | KGZ | если полная идентификация, Примечание: не работаем с Беларусь, Грузия, Украина | фильтр |
| 2.5 | Серия | Нет | Строка | 1234 | | данные, ML |
| 2.6 | Номер | Нет | Строка | AM123456 | | данные, ML |
| 2.7 | Фамилия | Нет | Строка | БЕКИШОВ | | фильтр |
| 2.8 | Имя | Нет | Строка | АЙДАРБЕК | | фильтр |
| 2.9 | Отчество | Нет | Строка | КУБАНЫЧБЕКОВИЧ | | фильтр |
| 2.10 | Дата рождения | Нет | DD.MM.YYYY | 01.02.1995 | | фильтр |
| 2.11 | Пол | Нет | Строка | M | если полная идентификация, Возможные значения: <ul style="list-style-type: none"> M F | фильтр |
| 3. Девайс | | | | | | |
| 3.1 | ОС | Нет | Строка | ANDROID | | фильтр |
| 3.2 | Версия МП | Нет | Строка | 3.102.0 | | фильтр |
| 3.3 | DEVICE ID | Нет | Строка | be86e2a103acf33a | | фильтр, ML |

| 4. Продукт | | | | | | |
|------------|---------------------------------------|-----|-----------------|---------|---|--------|
| 4.1 | Желаемая сумма | Нет | Число, в рублях | 25000 | | фильтр |
| 4.2 | Одобренная сумма | Нет | Число, в рублях | 20000 | | фильтр |
| 4.3 | Тип продукта | Нет | Строка | express | Возможные значения: <ul style="list-style-type: none"> • express - микрозайм (заявка на займ) с оформлением через мобильное приложение • consumer - потребительский кредит с аннуитетными платежами • overdraft - кредитный лимит • securedByCar - займ под залог авто • onlineCreditTransfer - переводы в долг • webExpress - веб займы • и др. | фильтр |
| 4.4 | Срок займа | Нет | Число | 63 | | фильтр |
| 4.5 | Размерность срока займа | Нет | Строка | day | Возможные значения: <ul style="list-style-type: none"> • day • month | фильтр |
| 4.6 | Длительность грейса | Нет | Число | 7 | | фильтр |
| 4.7 | Размерность срока длительности грейса | Нет | Строка | day | Возможные значения: <ul style="list-style-type: none"> • day • month | фильтр |

Приложение Б – Входные и выходные данные сервиса «tm-security-service», метод аутентификации «/login»

POST

/api/security/v1/login

Аутентификация

^

Parameters

Try it out

Request body required

application/json

Example Value | Schema

LoginRequestDto

login*

string

maxLength: 2048

example: somelogin

Логин

password*

string

maxLength: 2048

example: somepassword

Пароль

}

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example Value | Schema

LoginResponseDto {
 result*
 ResultResponseDto {
 code*
 {
 description: Код ответа
 }
 example: OK
 description
 {
 description: Описание результата
 }
 }
 }
 data
 TokenResponseDto {
 token*
 Token string
 maxLength: 2048
 example: eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJhODMxZjY2Ni1lMDhLLTRlNmItY
 Токен
 operatorId*
 OperatorId number
 example: 683
 Идентификатор оператора
 }
 }
 }
}

Приложение В – Входные и выходные данные сервиса «tm-security-service»,
метод выхода из системы «/logout»

POST /api/security/v1/logout Выход

Parameters

Try it out

Request body required

application/json

Example Value | Schema

LogoutRequestDto

token*

Token string

maxLength: 2048

example: eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJhODMxZjY2Ni1lNDhLLTRlNmItY

Токен

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example Value | Schema

LogoutResponseDto

result*

ResultResponseDto

code*

description: Код ответа

Приложение Г – Входные и выходные данные сервиса «tm-security-service», метод авторизации «/checkAccess»

POST

/api/security/v1/checkAccess

Авторизация

^

Parameters

Try it out

Request body required

application/json

Example Value | Schema

CheckAccessRequestDto

token*

Token string

maxLength: 2048

example: eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJhODMxZjY2Ni1lMDhLLTRlNmItY

Токен

updateExpire

boolean

Признак обновления срока действия токена

}

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example Value | Schema

CheckAccessResponseDto

result*

ResultResponseDto

code*

description:

Код ответа

example: OK

Приложение Д – Входные и выходные данные сервиса «tm-arm», метод аутентификации «/login»

POST

/api/arm/v1/login

Аутентификация

Try it out

Parameters

Request body required

application/json

Example Value

Schema

LoginRequestDto

login*

string

maxLength: 2048

example: somelogin

Логин

password*

string

maxLength: 2048

example: somepassword

Пароль

| Responses | | |
|---|-------------|----------|
| Code | Description | Links |
| 200 | OK | No links |
| <div>Media type</div> <div>application/json</div> <div>Controls Accept header.</div> <div><div>Example Value</div><div>Schema</div><div><div>LoginResponseDto</div><div><div>status*</div><div>ResponseStatusName</div><div>string</div><div>example: OK</div><div>Статус выполнения запроса</div></div><div><div>data</div><div><div>TokenResponseDto</div><div><div>token*</div><div>Token</div><div>string</div><div>maxLength: 2048</div><div>example: eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJhODMxZjY2Ni1lMDhLLTRlNmItY</div><div>Token</div></div></div></div></div></div> | | |

Приложение Е – Входные и выходные данные сервиса «tm-arm», метод получения списка незавершенных коммуникаций текущего оператора «/listCommunications»

GET

/api/arm/v1/listCommunications

Получение списка незавершенных коммуникаций текущего оператора

⌵ 🔒

Parameters

Try it out

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example Value

Schema

CommunicationListResponseDto

status

ResponseStatusName

string

example: OK

Статус выполнения запроса

data

CommunicationListDto

communications

time

string(\$datetime)

pattern: dd.MM.yyyy HH:mm

example: 01.12.2019 12:45

Дата

id

CommunicationId

integer

minimum: 0

example: 123456789

Номер карточки

doId

CommunicationNumber

string

maxLength: 32

example: 123456678

Номер заявки/займа

campaign

Campaign

displayName

string

maxLength: 32

example: Доведи до займа

Название кампании

description

string

maxLength: 32

example: Короткая рекомендация как продать займ

Описании кампании

callPhone

Phone

string

maxLength: 11

minLength: 11

pattern: [0-9]{11}

example: 79963802789

result

CommunicationResult

description

key-value значения результата коммуникации

< * >

string

example: OrderedMap { "COMMENT": "Дополним позже", "RESULT": "Дополним позже" }

operator

description

Имя оператора

name

string

example: Елена

Имя оператора

lastName

string

example: Авросиова

Приложение Ж – Входные и выходные данные сервиса «tm-arm», метод получения информации о наличии изменения для оператора «/hasChanges»

GET

/api/arm/v1/hasChanges

Получение информации о наличии изменений (для отображения) для оператора

^

🔒

Parameters

Try it out

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example ValueSchema

HasChangesResponseDto

status*

ResponseStatusName string

example: OK

Статус выполнения запроса

data

ChangesDto

hasChanges*

boolean

example: true

Флаг наличия изменений

changes

[ChangeType string

maxLength: 32

example: CLIENT_CARD_READY

Тип изменения

Приложение 3 – Входные и выходные данные сервиса «tm-arm», метод получения карточки клиента для текущей коммуникации «/getClientCard»

GET

/api/arm/v1/getClientCard

Получение карточки клиента для текущей коммуникации

^

🔒

Parameters

Try it out

Responses

| Code | Description | Links |
|------|-------------|----------|
| 200 | OK | No links |

Media type

application/json

Controls Accept header

Example Value

Schema

GetClientCardResponseDto

status*

ResponseStatusName string

example: OK

Статус выполнения запроса

data

ClientCardResponseDto

clientCard*

ClientCardDto

clientInfo*

ClientInfoDto

phone*

Phone string

maxLength: 11

minLength: 11

pattern: [0-9]{11}

example: 79963802789

Номер телефона клиента

fullNameRu

ClientFullName

name*

string

maxLength: 32

example: Шахноза

Имя клиента

lastName*

string

maxLength: 32

example: Ибрадова

Фамилия клиента

middleName

string

maxLength: 32

example: Шахноза

Отчество клиента

fullNameEn

ClientFullName

name*

string

maxLength: 32

example: Шахноза

Имя клиента

lastName*

string

maxLength: 32

example: Ибрадова

Фамилия клиента

middleName

string

maxLength: 32

example: Шахноза

Отчество клиента

citizenship

string

maxLength: 92

example: Россия

birthDate

string(\$date)

maxLength: 10

pattern: dd.MM.yyyy

example: 12.05.1995

Дата рождения

age

integer

maximum: 128

minimum: 18

example: 27

Возраст

| | | |
|-----------------------|---------------------|--|
| currentCommunication* | CommunicationDto | <div> <div>time*</div> <div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> <div>Дата</div> </div> </div> |
| id* | CommunicationId | <div> <div>integer</div> <div>minimum: 0</div> <div>example: 123456789</div> <div>Номер карточки</div> </div> |
| campaign | Campaign | <div> <div> <div>displayName*</div> <div>string</div> <div>maxLength: 32</div> <div>example: Доведу до займа</div> <div>Название кампании</div> </div> <div> <div>description</div> <div>string</div> <div>maxLength: 32</div> <div>example: Короткая рекомендация как продать займ</div> <div>Описании кампании</div> </div> </div> |
| callPhone* | Phone | <div> <div>string</div> <div>maxLength: 11</div> <div>minLength: 11</div> <div>pattern: [0-9]{11}</div> <div>example: 79963802789</div> <div>Номер телефона клиента</div> </div> |
| result | CommunicationResult | <div> <div> <div>description:</div> <div>Значения результата коммуникации</div> </div> <div> <div>reason</div> <div>string</div> <div>example: Не было с собой документа</div> </div> <div> <div>result</div> <div>string</div> <div>example: Заполнит позже</div> </div> <div> <div>comment</div> <div>string</div> </div> </div> |
| operator* | | <div> <div> <div>description:</div> <div>Имя оператора</div> </div> <div> <div>name*</div> <div>string</div> <div>example: Елена</div> <div>Имя оператора</div> </div> <div> <div>lastName*</div> <div>string</div> <div>example: Абросимова</div> <div>Фамилия оператора</div> </div> </div> |

| | | |
|---------------------|----------------|---|
| productList | [ProductDto | <div> <div> <div> <div>startTime*</div> <div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> <div>Дата начала</div> </div> </div> <div> <div>endTime*</div> <div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> <div>Дата окончания</div> </div> </div> </div> </div> |
| statusDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| productDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| requestedAmount | integer | <div> <div>minimum: 0</div> <div>example: 12000000</div> <div>Запрошенная сумма в копейках</div> </div> |
| approvedAmount | integer | <div> <div>minimum: 0</div> <div>example: 10000000</div> <div>Одобренная сумма в копейках</div> </div> |
| term* | integer | <div> <div>minimum: 0</div> <div>example: 21</div> <div>Срок (в заданных единицах)</div> </div> |
| termUnitDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| grace* | integer | <div> <div>minimum: 0</div> <div>example: 25</div> <div>Грейс (в заданных единицах)</div> </div> |
| graceUnitDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| filling | [AppDocument | <div> <div> <div> <div>name*</div> <div>string</div> <div>maxLength: 32</div> <div>example: Селфи/ДУЛ</div> <div>Название документа на фото</div> </div> <div> <div>documentDict</div> <div>DictionaryItem</div> <div>> {...}</div> </div> <div> <div>status*</div> <div>string</div> <div>maxLength: 32</div> <div>example: ERROR</div> <div>Статус документа</div> </div> <div> <div>errors</div> <div>[FillingError</div> <div> <div> <div>errorDescription</div> <div>string</div> <div>example: Плохое качество</div> </div> <div> <div>advice</div> <div>string</div> <div>example: Доводите фокусировку камеры и сделайте фото документа</div> <div>Рекомендация</div> </div> </div> </div> </div> </div> |
| current* | boolean | <div> <div>example: true</div> <div>Признак текущего продукта</div> </div> |
| retailChainNameDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| deviceOsDict | DictionaryItem | <div> <div>> {...}</div> </div> |
| interestRate | integer | <div> <div>minimum: 0</div> <div>example: 80</div> <div>Процентная ставка в день. Передается в формате: % * 100</div> </div> |
| | | <div> <div>}}</div> </div> |

Приложение И – Входные и выходные данные сервиса «tm-arm», метод сохранения промежуточного результата коммуникации «/saveCommunicationDraft»

Request body application/json

Example Value | Schema

```
SaveCommunicationDraftRequestDto {
  communicationId* CommunicationId integer
    minimum: 0
    example: 123456789
    Номер карточки

  clientPhone* Phone string
    maxLength: 11
    minLength: 11
    pattern: [0-9]{11}
    example: 79963802789
    Номер телефона клиента

  communicationResult CommunicationResult {
    description: Значения результата коммуникации

    reason string
    example: Не было с собой документа

    result string
    example: Заполнит позже

    comment string
  }
}
```

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type application/json

Controls Accept header.

Example Value | Schema

```
StatusResponseDto {
  status* ResponseStatusName string
    example: OK
    Статус выполнения запроса
}
```

Приложение К – Входные и выходные данные сервиса «tm-arm», метод сохранения результата коммуникации «/saveCommunicationResult»

Request body application/json

Example Value | Schema

```
SaveCommunicationRequestDto {
  communicationId* CommunicationId integer
    minimum: 0
    example: 123456789
    Номер карточки

  clientPhone* Phone string
    maxLength: 11
    minLength: 11
    pattern: [0-9]{11}
    example: 79963802789
    Номер телефона клиента

  communicationResult CommunicationResult {
    description: Значения результата коммуникации

    reason string
      example: Не было с собой документа

    result string
      example: Заполнит позже

    comment string
  }
}
```

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type application/json

Controls Accept header.

Example Value | Schema

```
StatusResponseDto {
  status* ResponseStatusName string
    example: OK
    Статус выполнения запроса
}
```

Приложение Л – Входные и выходные данные сервиса «tm-arm», метод получения карточки клиента текущего оператора по id коммуникации «/getClientCardByCommunicationId»

POST

/api/arm/v1/getClientCardByCommunicationId

Получение карточки клиента текущего оператора по ID коммуникации

Try it out

Parameters

Request body

application/json

Example Value

Schema

GetClientCardByCommunicationIdRequestDto

communicationId*

CommunicationId integer

minimum: 0

example: 123456789

Номер карточки

Responses

Code

Description

200

OK

Media type

application/json

Controls Accept header

Example Value

Schema

GetClientCardResponseDto

status*

ResponseStatusName string

example: OK

Статус выполнения запроса

data

ClientCardResponseDto

clientCard*

ClientCardDto

clientInfo*

ClientInfoDto

phone*

Phone string

maxLength: 11

minLength: 11

pattern: [0-9]{11}

example: 79963802789

Номер телефона клиента

fullNameRu

ClientFullName

name*

string

maxLength: 32

example: Шахноза

Имя клиента

lastName*

string

maxLength: 32

example: Ибрахова

Фамилия клиента

middleName

string

maxLength: 32

example: Шахноза

Отчество клиента

fullNameEn

ClientFullName

name*

string

maxLength: 32

example: Шахноза

Имя клиента

lastName*

string

maxLength: 32

example: Ибрахова

Фамилия клиента

middleName

string

maxLength: 32

example: Шахноза

Отчество клиента

citizenship

string

maxLength: 92

example: Россия

birthDate

string(\$date)

maxLength: 10

pattern: dd.MM.yyyy

example: 12.05.1991

Дата рождения

age

integer

maximum: 128

minimum: 18

example: 27

Возраст

| | | |
|-----------------------|------------------|--|
| currentCommunication* | CommunicationDto | <div> <div>time*</div> <div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> </div> </div> <div> <div>id*</div> <div> <div>Дата</div> <div>CommunicationId integer</div> <div>minimum: 0</div> <div>example: 123456789</div> <div>Номер карточки</div> </div> </div> <div> <div>campaign</div> <div> <div>Campaign</div> <div> <div>displayName*</div> <div>string</div> <div>maxLength: 32</div> <div>example: Доведеи до займа</div> <div>Название кампании</div> </div> <div>description</div> <div>string</div> <div>maxLength: 32</div> <div>example: Короткая рекомендация как продать займ</div> <div>Описании кампании</div> </div> </div> |
|-----------------------|------------------|--|

| | | |
|-------------|---------------------|---|
| productlist | [ProductDto | <div> <div> <div>startTime*</div> <div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> </div> </div> <div> <div>endTime*</div> <div> <div>Дата начала</div> <div>string(\$datetime)</div> <div>pattern: dd.MM.yyyy HH:mm</div> <div>example: 01.12.2019 12:45</div> <div>Дата окончания</div> </div> </div> <div> <div>statusDict</div> <div>DictionaryItem > {...}</div> </div> <div> <div>productDict</div> <div>DictionaryItem > {...}</div> </div> <div> <div>requestedAmount</div> <div> <div>integer</div> <div>minimum: 0</div> <div>example: 12000000</div> <div>Запрошенная сумма в копейках</div> </div> </div> <div> <div>approvedAmount</div> <div> <div>integer</div> <div>minimum: 0</div> <div>example: 10000000</div> <div>Одобренная сумма в копейках</div> </div> </div> <div> <div>term*</div> <div> <div>integer</div> <div>minimum: 0</div> <div>example: 21</div> <div>Срок (в заданных единицах)</div> </div> </div> <div> <div>termUnitDict</div> <div>DictionaryItem > {...}</div> </div> <div> <div>grace*</div> <div> <div>integer</div> <div>minimum: 0</div> <div>example: 25</div> <div>Грейс (в заданных единицах)</div> </div> </div> <div> <div>graceUnitDict</div> <div>DictionaryItem > {...}</div> </div> <div> <div>filling</div> <div> <div>AppDocument</div> <div> <div>name*</div> <div>string</div> <div>maxLength: 32</div> <div>example: Селфи/ДПЛ</div> <div>Название документа на фото</div> <div>documentDict</div> <div>DictionaryItem > {...}</div> <div>status*</div> <div>string</div> <div>maxLength: 32</div> <div>example: ERROR</div> <div>Статус документа</div> <div>errors</div> <div> <div>FillingError</div> <div> <div>errorDescription</div> <div>string</div> <div>example: Плохое качество</div> <div>Текст ошибки</div> <div>advice</div> <div>string</div> <div>example: Дожидесь фокусировки камеры и сделайте фото документа</div> <div>Рекомендация</div> </div> </div> </div> </div> </div> </div> |
| | current* | <div> <div>boolean</div> <div>example: true</div> <div>Признак текущего продукта</div> </div> |
| | retailChainNameDict | <div> <div>DictionaryItem > {...}</div> </div> |
| | deviceOsDict | <div> <div>DictionaryItem > {...}</div> </div> |
| | interestRate | <div> <div>integer</div> <div>minimum: 0</div> <div>example: 80</div> <div>Процентная ставка в день. Передается в формате: % * 100</div> </div> |
| | | <div> <div>}}</div> </div> |

```

filling
  AppDocument {
    name* string
      maxLength: 32
      example: Селфи/ДП/
      Название документа на фото

    documentDict DictionaryItem > {...}

    status* string
      maxLength: 32
      example: ERROR
      Статус документа

    errors [FillingError {
      errorDescription string
        example: Плохое
        качество
      Текст ошибки
      advice string
        example: Дожидитесь
        фокусировки камеры и
        сделайте фото
        документа
      Рекомендация

    }]

    current* boolean
      example: true
      Признак текущего продукта

    retailChainNameDict DictionaryItem > {...}

    deviceOsDict DictionaryItem > {...}

    creditMethodDict DictionaryItem > {...}

    interestRate integer
      minimum: 0
      example: 80
      Процентная ставка в день. Передается в формате: % * 100

  }
}

```

```

history
  CommunicationDto {
    time* string($datetime)
      pattern: dd.MM.yyyy HH:mm
      example: 01.12.2019 12:45
      Дата

    id* CommunicationId integer
      minimum: 0
      example: 123456789
      Номер карточки

    campaign Campaign {
      displayName* string
        maxLength: 32
        example: Доведи до займа
        Название кампании

      description string
        maxLength: 32
        example: Короткая рекомендация как продать займ
        Описание кампании

    }

    callPhone* Phone string
      maxLength: 11
      minLength: 11
      pattern: [0-9]{11}
      example: 79963802789
      Номер телефона клиента

    result CommunicationResult {
      description: Значения результата коммуникации
      reason string
        example: Не было с собой документа
      result string
        example: Заполнил позже
      comment string

    }

    operator* {
      description: Имя оператора
      name* string
        example: Елена
        Имя оператора
      lastName* string
        example: Абросимова
        Фамилия оператора

    }

  }
}
}

```

```

DictionaryItem {
  dictionaryKey* string
    example: Ключ
    Ключ

  dictionaryValue* string
    example: Значение
    Значение

}

```

Приложение М – Входные и выходные данные сервиса «tm-arm», метод выхода из системы «/logout»

GET

/api/arm/v1/logout

Выход из системы

^

🔒

Parameters

Try it out

Responses

| Code | Description |
|------|-------------|
| 200 | OK |

Media type

application/json

Controls Accept header.

Example Value | Schema

StatusResponseDto

{
 status*
 ResponseStatusName string
 example: OK
 Статус выполнения запроса
}