# АННОТАЦИЯ

Данная выпуская квалификационная работа состоит из 69 страниц, 51 рисунка и 8 источников.

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

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

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

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

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

СОДЕРЖАНИЕ

[1 ВВЕДЕНИЕ 3](#_Toc166617125)

[2 ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc166617126)

[2.1 Что такое телемаркетинг: цели, принципы 6](#_Toc166617127)

[2.2 Телемаркетинг в финансовой сфере. 7](#_Toc166617128)

[2.3 Специфика бизнес – процесса внутри рассматриваемой платежной системы «Золотая Корона» 8](#_Toc166617129)

[2.4 Обзор существующих решений 9](#_Toc166617130)

[2.4.1 1С:CRM 9](#_Toc166617131)

[2.4.2 MegaCRM 13](#_Toc166617132)

[2.5 Формирование требования к разрабатываемой системе 18](#_Toc166617133)

[3 ГЛАВА 2. Проектирование системы 19](#_Toc166617134)

[3.1 Выбор инструментальных средств 19](#_Toc166617135)

[3.1.1 Язык программирования Java 19](#_Toc166617136)

[3.1.2 Автоматизированная сборка проектов с помощью Apache Maven 20](#_Toc166617137)

[3.1.3 Фреймворк Spring 21](#_Toc166617138)

[3.1.4 Объектно-реляционный фреймворк Hibernate 26](#_Toc166617139)

[3.1.5 Система управления базами данных PostgreSQL 28](#_Toc166617140)

[3.1.6 Apache Kafka 30](#_Toc166617141)

[3.2 Взаимодействие клиент – серверной архитектуры 31](#_Toc166617142)

[3.3 Проектирование сервисной архитектуры разрабатываемой системы 32](#_Toc166617143)

[3.3.1 Микросервисная архитектура 32](#_Toc166617144)

[3.3.2 Предметно-ориентированное проектирование сервисов. Проектирование домена «Займы». 33](#_Toc166617145)

[3.3.3 Интеграции со сторонними сервисами 34](#_Toc166617146)

[3.3.4 Проектирование основного контура «Телемаркетинг» 42](#_Toc166617147)

[3.4 Проектирование структуры хранения данных 65](#_Toc166617148)

[4 ГЛАВА 3 66](#_Toc166617149)

[5 ЗАКЛЮЧЕНИЕ 67](#_Toc166617150)

[6 СПИСОК ЛИТЕРАТУРЫ 68](#_Toc166617151)

[7 ПРИЛОЖЕНИЯ 69](#_Toc166617152)

# **ВВЕДЕНИЕ**

1 – 2 страницы. Описываю проблему, как люди живут без неё, почему её нужно решить, актуальность, и т.д.

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.

Система телемаркетинга в финансовой организации решает определенны проблемы:

Во-первых, это увеличение продаж. Благодаря прямому контакту с потенциальными клиентами, предоставлению информации о продуктах и услугах, а также предложению индивидуальных условий и акций, система телемаркетинга позволяет эффективно привлекать новых клиентов и увеличивать продажи финансовых продуктов и услуг.

Во-вторых, система телемаркетинга помогает улучшить качество обслуживания клиентов. Оперативное реагирование на вопросы и запросы клиентов, решение проблем и предоставление необходимой информации способствуют повышению удовлетворенности клиентов.

Кроме того, система телемаркетинга позволяет собирать и анализировать данные о клиентах, их потребностях и предпочтениях. Это помогает лучше понимать свою целевую аудиторию и разрабатывать более эффективные маркетинговые стратегии.

Также система телемаркетинга позволяет управлять клиентской базой, отслеживать историю взаимодействия с клиентами, проводить сегментацию и персонализацию маркетинговых кампаний.

Наконец, система телемаркетинга автоматизирует процесс обзвона клиентов, что позволяет сократить время и ресурсы, затрачиваемые на контакт с каждым клиентом.

В целом, разработка системы телемаркетинга в финансовой организации может быть обоснована необходимостью повышения эффективности продаж, улучшения обслуживания клиентов, сбора и анализа данных, управления клиентской базой и экономии времени и ресурсов. 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».

Сложность заключается в том, что готовые системы от сторонних разработчиков не подходят для интеграции с существующим процессом, поэтому будет трудно внедрить стороннее решение, не перестраивая существующие бизнес-процессы, что может иметь непредвиденные расходы на переобучение персонала.

Разработка собственного сервиса телемаркетинга с учетом привычных бизнес-процессов внутри компании позволяет решить это. Не придется перестраивать процессы, но все этапы станут автоматизированными, что позволит легко внедрить систему и увеличить эффективность работы.

В результате выполнения данной работы будет разработан и реализован сервис телемаркетинга для платежной системы «Золотая Корона» группы компаний «Центр Финансовых Технологий».

Также выполнение проекта подразумевает выбор инструментов для разработки: языка программирования, фреймворков, базы данных. Будут изучены существующие инструменты, а по результатам их сравнения произведён окончательный выбор.

15 – 20 страниц. Глава будет посвящена более полному погружению в проблему. Нужно сделать плавный переход из введения. Взять проблему, описать варианты готовых решений/как еще можно решить эту проблему. В конце сделаю вывод: как буду решать проблему, почему есть мотивация её решать вообще.

Ставим задачу перед собой: «Будем делать …, вот так …, потому что …, опираясь на обзор существующей системы …».

Зарисовка как было до системы телемаркетинга:

Была уже выстроенная цепочка действий начиная от того, как бизнес собирает как-то данные и т.д. После этого появляется какая-то группа клиентов, которым мы хотим позвонить. Т.е. бизнес составляет Excel-файлы с группами пользователей, далее эти файлики вручную выгружают админы в БД, а в конце супероператоры выгружают опять вручную списки в MightyCall (сервис-звонилка). В итоге, операторы совершают звонки. В этой цепочке очень много ручного труда, который хотелось автоматизировать, кроме этого, на каждом этапе находятся разные люди из разных команд -> сложнее контролировать процесс (что/где/как сломалось)

Беглый анализ:

Как видно из описания, процесс достаточно уникальный, поэтому я не уверен, что есть какое-то коробочное решение, которое можно было бы внедрить сюда, не ломая существующий процесс. Т.е., наверное, есть какие-то системы для звонков, но они вряд ли дадут себя настроить так как уже построен процесс в золотой короне. -> у нас кроме затрат на внедрения появляются затраты на перестройку самого процесса совершения звонков, переобучение персонала и т.д.

В любом случае, нужно поискать какие-то системы со схожим функционалом, чтобы точно дать оценку.

# **ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## Что такое телемаркетинг: цели, принципы

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 (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.

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.

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.

Кроме того, широкое использование баз данных и автоматизированной рассылки звонков привело к улучшению обслуживания клиентов. Это улучшило 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;

• Исходящий - обзвон пользователей операторами.

В данной выпускной квалификационной работе будет рассматриваться исходящий вид телемаркетинга.

## Телемаркетинг в финансовой сфере.

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

Телемаркетинг предоставляет возможность прямого контакта с клиентом, что приносит 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. 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.
* **Повышать спрос:** финансовые консультанты могут использовать телемаркетинг для привлечения "холодных" потенциальных клиентов или отслеживания "теплых" потенциальных клиентов.
* **Проводить маркетинговые исследования:** можно проводить быстрые опросы или интервью, чтобы понять потребности рынка или собрать отзывы о новых продуктах или услугах.

## Специфика бизнес – процесса внутри рассматриваемой платежной системы «Золотая Корона»

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

В самом начале платежная система «Золотая Корона» получает заявки на займы от партеров. В качестве сервисов, из которых идет поток заявок может быть мобильное приложение «KoronaPay» или другие сервис – партнеры.

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

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

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

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

Результаты каждой коммуникации точно также 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 локализовывать проблемы и разбирать инциденты. Поэтому разработка сервиса, или использование готового решения значительно упростило бы сопровождение этого бизнес – процесса.

## Обзор существующих решений

Перед тем как разрабатывать своё решение необходимо проанализировать уже существующие решения, чтобы понять можно ли внедрить их с учетом процесса, описанного в п. 1.3.

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

### 1С:CRM

<https://1crm.ru/help/telemarketing/>

<https://habr.com/ru/companies/trinion/articles/296406/>

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

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

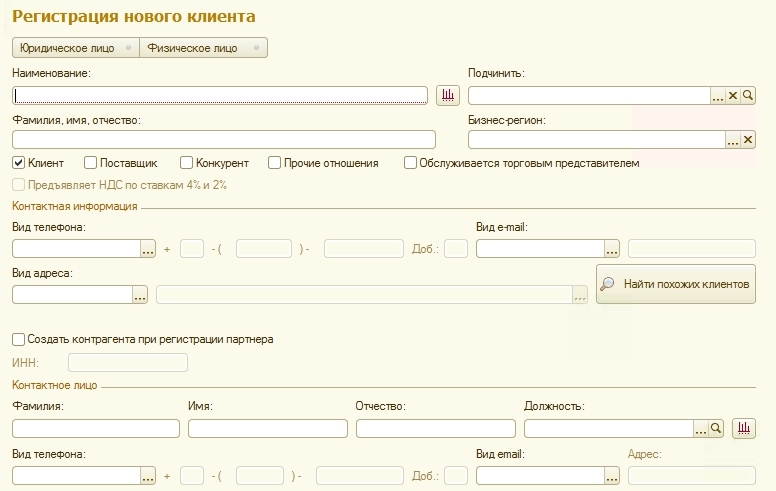


Рисунок 1 – Форма регистрации нового клиента

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

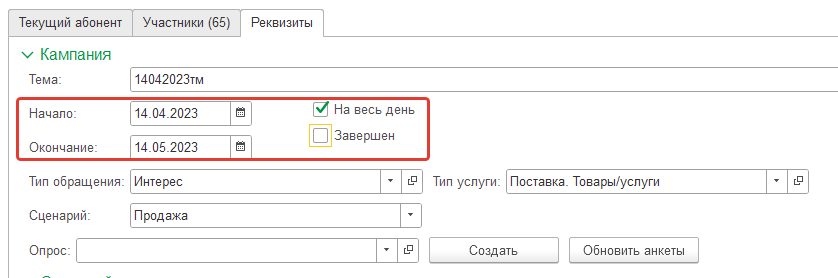


Рисунок 2 – Форма планирования звонков

Сама система 1С:CRM не предусматривает связи с телефонией — для этого существует продукт под названием 1С-Рарус: СофтФон. 1С-Рарус: СофтФон обеспечивает интеграцию с офисной АТС (телефонной станцией) и необходим для того, чтобы вы могли фиксировать исходящие и входящие звонки в системе 1С:CRM. Звонить с использованием СофтФона можно, но сама по себе его настройка — дело не тривиальное. 1С-Рарус: СофтФон — это встроенный продукт. СофтФон нельзя дорабатывать, его код закрыт и написан не на языке 1С.

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

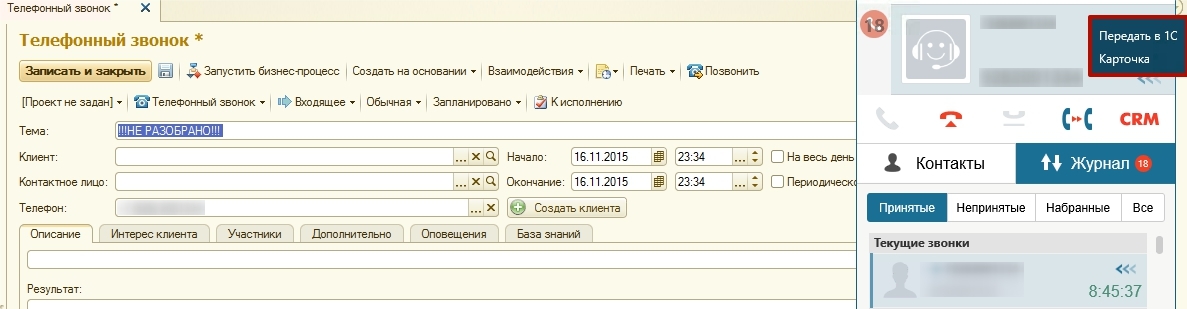


Рисунок 3 – Карточка телефонного звонка. Совместная работа «1С Телемаркетинг» и «1С-Рарус: СофтФон»

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

Телефонные обзвоны: в системе 1С:CRM предусмотрена также возможность регистрации телефонных обзвонов. Эта функция называется «Телемаркетинг». «Телемаркетинг» предоставляет возможность создавать список обзвона вручную, либо по сегментам, либо на основании результатов какого-либо отчета. В документе телемаркетинга можно написать сценарий телефонного звонка в свободной форме. Телемаркетинг дает простейшую аналитику: количество обработанных звонков, количество неуспешных звонков.

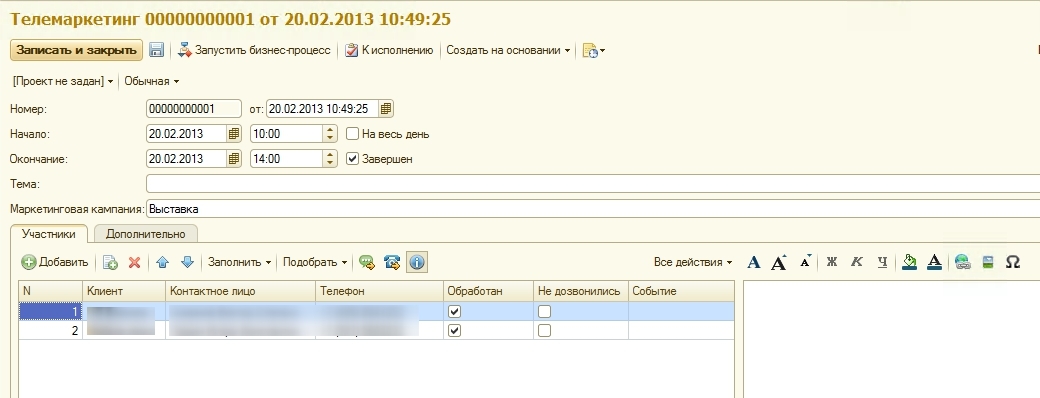


Рисунок 4 – Форма создания телефонных обзвонов

Однако, «1C Телемаркетинг» не имеет интеграций с системами, 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.

Таким образом, телемаркетинг представляет из себя, скорее, инструмент контроля сотрудников, осуществляющих звонки. Здесь фиксируется факт того, позвонили клиенту или нет, нежели результат звонка.

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

### MegaCRM

<https://megacrm.ru/telemarketing>

<https://help.megagroup.ru/kak-rabotat-so-zvonkami-v-megacrm#step--1>

https://wiki.domru.biz/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.

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.

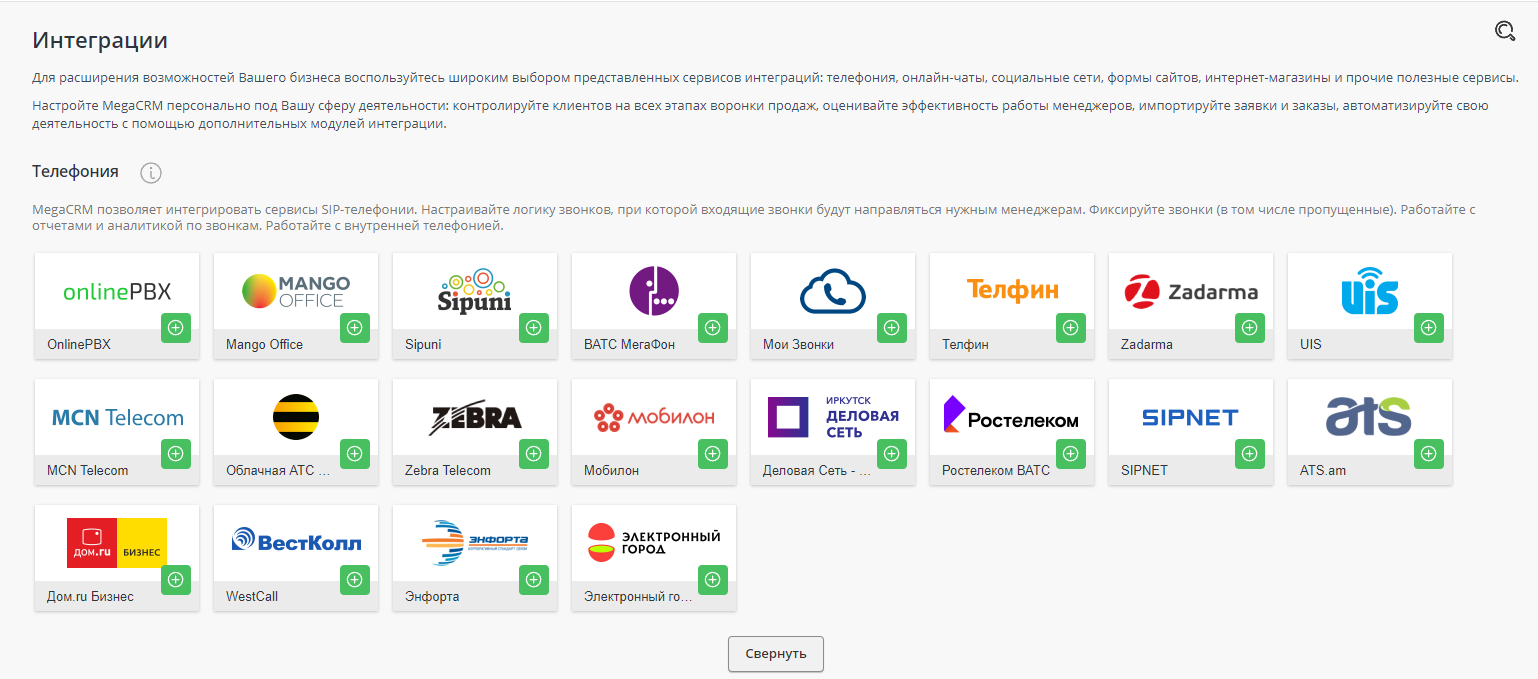
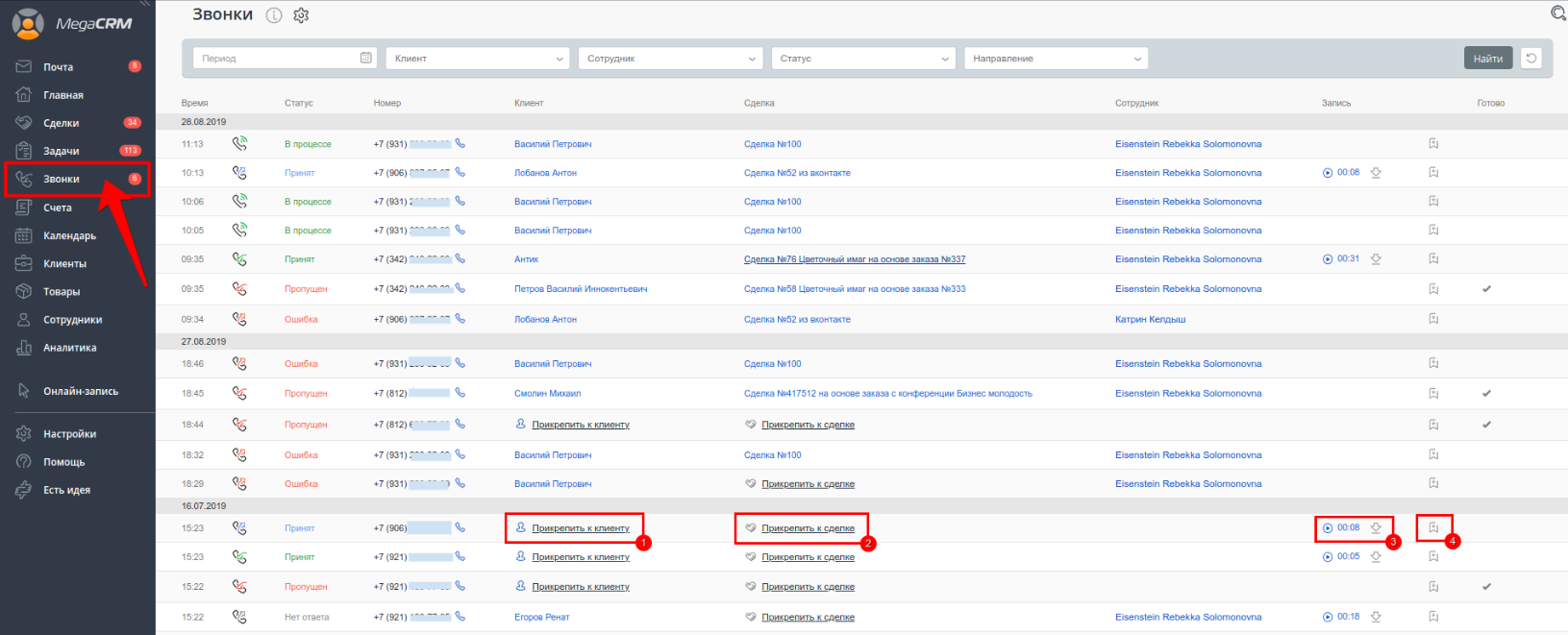


Рисунок 5 – Интеграция MegaCRM с телефоний

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

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.

Рисунок 7 – Карточка клиента (ВСТАВИТЬ)

Основные элементы, которые есть в карточке клиента:

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

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

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

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

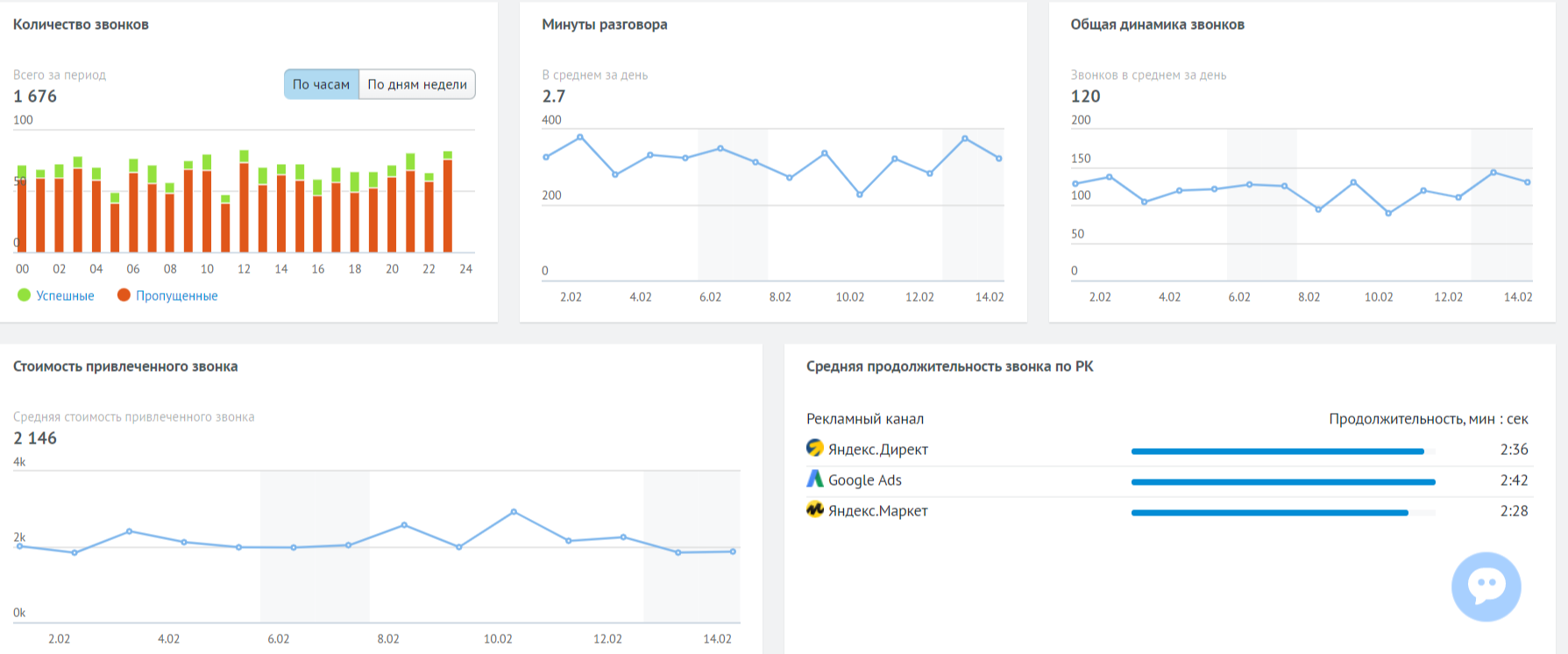


Рисунок 9 – Раздел «Аналитика», вкладка «Звонки»

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

Однако, MegaCRM не имеет интеграций с системами, 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, при использовании платформы MegaCRM, пользователи могут столкнуться с несколькими потенциальными недостатками. В частности, высокая стоимость лицензий и дополнительных функций может быть проблемой для малых компаний с ограниченным бюджетом. Кроме того, сложность настройки и необходимость обучения сотрудников при внедрении новой CRM-системы также могут вызвать определенные трудности. Ограничения в конфигурировании, безопасности, зависимость от стабильного интернет-соединения и возможные проблемы с интеграцией с другими инструментами также могут создать препятствия для эффективного использования MegaCRM в повседневной работе компании.

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

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

## Формирование требования к разрабатываемой системе

Основными требованиями к системе является открытость, доступность платформы, расширяемость и удобство использования.

Система предназначена для обработки поступающего потока клиентских заявок, фильтрации этого потока, подготовки и сбора данных о клиентах для дальнейшего звонка и фиксации результата звонка.

В ходе обзора существующих решений были выявлены как достоинства, так и недостатки. Исходя из этого можно выявить ряд требований к проектируемой системе:

* Автоматическая обработка входящего потока заявок;
* Конфигурируемая фильтрация потока заявок;
* Разделение клиентов по кампаниями, в рамках которых будет осуществляться звонок;
* Интеграция с внешними сервисами;
* Обработка и подготовка данных о клиенте;
* Механизм перезвона клиенту;
* Сохранение истории коммуникаций
* Удобное автоматизированное рабочее место оператора (АРМ Оператора).

СХЕМА?

Рассказ: немного телемаркетинга, как он строится, завязка на внутренние сервисы и т.д.

Тут общие слова, нужны картинки  
Нужно поставить задачу т.д.

# **ГЛАВА 2. Проектирование системы**

Тут описать что есть готовый фронт (арм), есть БД, сервисы ЦФТ

Чтобы связать, нужна прослойка – система моя, для того чтобы связать и т.д.

Слева фронт, справа сервисы цфт

## Выбор инструментальных средств

### Язык программирования Java

Java - популярный язык программирования, используемый уже более 20 лет. Миллионы приложений написаны на Java и используются и 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.

Java выделяется среди других языков программирования по нескольким причинам:

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

### Автоматизированная сборка проектов с помощью Apache Maven

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

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

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

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

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

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

### Фреймворк Spring

Spring — это набор взаимосвязанных мини-фреймворков, созданных для работы над разными частями приложения. Подключать их можно по отдельности, в зависимости от задач.

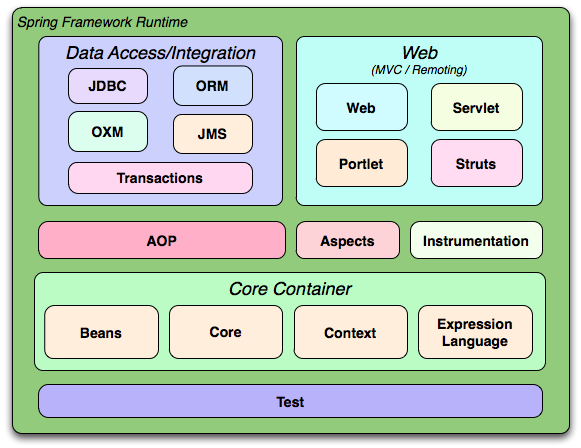


Рисунок 10 – Экосистема фреймворков Spring

* Основной контейнер (Core Container):

Контейнер Core состоит из модулей Core, Beans, Context и Expression Language.

Модули Core и Beans обеспечивают основные компоненты платформы, включая инверсию контроля (Inversion of Control) и внедрения зависимостей (Dependency Injection). BeanFactory представляет собой сложную реализацию шаблона factory. Это устраняет необходимость в программируемых синглтонах и позволяет отделить конфигурацию и спецификацию зависимостей от реальной логики программы.

Модуль Context основан на надежной базе, предоставляемой модулями Core и Beans: это средство доступа к объектам в стиле framework, аналогичное реестру JNDI. Модуль Context наследует свои функции от модуля Beans и добавляет поддержку интернационализации (используя, например, пакеты ресурсов), распространения событий, загрузки ресурсов и прозрачного создания контекстов, например, с помощью контейнера сервлетов. Модуль Context также поддерживает функции Java EE, такие как EJB, JMX и базовое удаленное взаимодействие. Интерфейс ApplicationContext является центральным элементом модуля Context.

Модуль Expression Language предоставляет мощный язык выражений для запросов к объектному графу и управления им во время выполнения. Он является расширением унифицированного языка выражений (unified EL), указанного в спецификации JSP 2.1. Язык поддерживает установку и получение значений свойств, присвоение свойств, вызов методов, доступ к контексту массивов, коллекций и индексаторов, логических и арифметических операторов, именованных переменных и поиск объектов по имени из контейнера IoC Spring. Он также поддерживает проекцию и выбор списка, а также общие агрегации списков.

* Доступ к данным/интеграция (Data Access/Integration)

Уровень доступа к данным/интеграции состоит из модулей JDBC, ORM, OXM, JMS и транзакций.

Модуль JDBC предоставляет уровень абстракции JDBC, который устраняет необходимость в утомительном кодировании на JDBC и анализе кодов ошибок, зависящих от поставщика базы данных.

Модуль ORM предоставляет уровни интеграции для популярных API объектно-реляционного отображения, включая JPA, JDO, Hibernate и iBatis. Используя пакет ORM, мы можем использовать все эти платформы O/R-сопоставления в сочетании со всеми другими функциями, предлагаемыми Spring, такими как простая функция декларативного управления транзакциями, упомянутая ранее.

Модуль OXM предоставляет уровень абстракции, который поддерживает реализации сопоставления объектов и XML для JAXB, Castor, XMLBeans, JiBX и XStream.

Модуль Java Messaging Service (JMS) содержит функции для создания и использования сообщений.

Модуль транзакций поддерживает программное и декларативное управление транзакциями для классов, реализующих специальные интерфейсы, и для всех ваших POJO (обычных старых объектов Java).

* Web

Веб-уровень состоит из модулей Web, Web-Servlet, Web-Struts и Web-Portlet.

Веб-модуль Spring предоставляет базовые функции интеграции, ориентированные на Веб, такие как функция загрузки файлов из нескольких частей и инициализация контейнера IoC с использованием прослушивателей сервлетов и контекста веб-приложения. Он также содержит связанные с Веб части поддержки удаленного взаимодействия Spring.

Модуль Web-Servlet содержит реализацию Spring model-view-controller (MVC) для веб-приложений. Платформа Spring MVC обеспечивает четкое разделение между кодом модели предметной области и веб-формами и интегрируется со всеми другими функциями платформы Spring Framework.

Модуль Web-Struts содержит классы поддержки для интеграции классического веб-уровня Struts в приложение Spring. Обратите внимание, что начиная с версии Spring 3.0 эта поддержка устарела. Рассмотрите возможность переноса вашего приложения на Struts 2.0 и его интеграции с Spring или на решение Spring MVC.

Модуль Web-Portlet предоставляет реализацию MVC для использования в среде портлетов и отражает функциональность модуля Web-Servlet.

* AOP и инструментарий

Модуль AOP от Spring предоставляет реализацию аспектно-ориентированного программирования, совместимую с AOP Alliance, позволяющую вам определять, например, перехватчики методов и точечные разрезы для четкого разделения кода, реализующего функциональность, которая должна быть разделена. Используя функциональность метаданных исходного кода, вы также можете включить информацию о поведении в свой код, аналогично тому, как это делается с атрибутами .NET. Модуль separate Aspects обеспечивает интеграцию с AspectJ. Модуль Instrumentation обеспечивает поддержку class instrumentation и реализацию classloader для использования на определенных серверах приложений.

* Тестирование

Тестовый модуль поддерживает тестирование компонентов Spring с помощью JUnit или TestNG. Он обеспечивает согласованную загрузку Spring Applicationcontext и кэширование этих контекстов. Он также предоставляет макетные объекты, которые вы можете использовать для изолированного тестирования вашего кода.

Особенности и преимущества Spring

* Простота, гибкость, универсальность. Фреймворк разработан для поддержки всех типов развертывания, от небольших приложений до крупных корпоративных решений. А для начинающих это означает, что кривая обучения будет крута, но потенциал для применения — огромен.
* Возможность внедрения зависимостей (DI). DI — это шаблон проектирования, который позволяет реализовывать слабую связь, что упрощает тестирование и поддержку кода. Его можно сравнить с личным помощником, который приносит вам то, что вам нужно, именно когда вам это нужно. Это делает процесс написания кода более гладким и эффективным, что особенно полезно для новичков.
* Контейнер инверсии управления (IoC) Spring предоставляет мощный способ управления компонентами приложения. Это похоже на главный контроллер, который обрабатывает создание и связывание объектов, что помогает избежать большого объёма ручной работы.
* Поддержка аспектно-ориентированного программирования (АОР). Это парадигма программирования, которая стремится увеличить модульность, позволяя разделять сквозные аспекты разработки. Она как волшебная палочка, которая может добавлять дополнительную функциональность там, где вам это нужно, не затрагивая существующий код. Это позволяет изменять подход к организации кода, что делает его более понятным для начинающих и упрощает навигацию по структуре сложных проектов.
* Интеграция с другими Java-фреймворками и технологиями. Spring хорошо совместим с другими популярными фреймворками, такими как Hibernate, Struts и JUnit, JPA и даже облачными платформами. Вы можете объединить преимущества различных моделей для создания надёжного и эффективного приложения. Возможность работать в гармонии с другими инструментами может значительно улучшить опыт обучения для новичков, предоставив им более широкое понимание Java-экосистемы.
* Spring Boot, проект в составе экосистемы Spring, предоставляет более быстрый и эффективный способ настройки и запуска приложений Spring. Он устраняет много шаблонного кода конфигурации, позволяя вам сконцентрироваться на написании бизнес-логики.

### Объектно-реляционный фреймворк Hibernate

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

Hibernate Framework - это фреймворк, созданный для работы с базами данных. Он реализует объектно-реляционную модель - технологию, которая связывает программные сущности и соответствующие записи в базе данных.

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

Если описывать упрощённо, ORM предлагает присвоить столбцу имя переменной объекта кода, а в строку записать значение. Причём типы данных должны определяться автоматически. Всё это требует настройки, и без специальных инструментов приходилось бы проводить все операции вручную. К счастью, готовые инструменты для автоматизации этого процесса существуют.

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

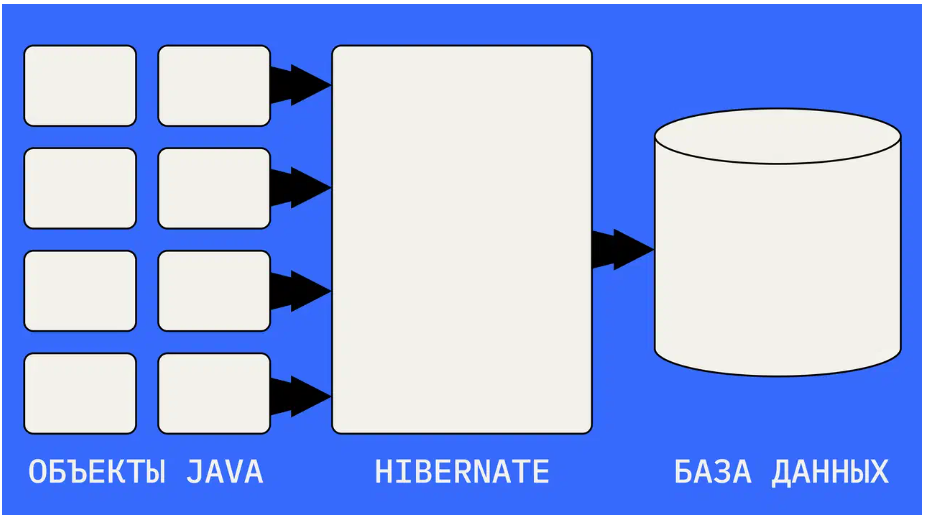


Рисунок 11 – Схема отображения Java – объектов в записи базы данных с помощью фреймворка Hibernate

Для работы с БД необязательно использовать возможности Hibernate — можно писать весь код стандартными средствами Java. Однако фреймворк значительно облегчает работу:

* ускоряет написание кода;
* создаёт удобную модель отображения уже существующей базы внутри кода;
* позволяет задать стандарт переноса информации из кода в БД.

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.

Внутри фреймворка используется Hibernate Query Language (HQL), который напоминает SQL, но является проще и более компактным, что делает его удобным для работы с базами данных из Java

Hibernate поддерживает "ленивую загрузку", которая позволяет эффективно работать с базой данных, загружая данные только по необходимости, что повышает производительность приложения

### Система управления базами данных PostgreSQL

PostgreSQL – это объектно-реляционная система управления базами данных (СУБД), наиболее развитая СУБД с открытым исходным кодом. Является альтернативой коммерческим СУБД.

Преимуществами СУБД PostgreSQL являются:

* ***Свободный доступ*** – любой специалист может бесплатно скачать, установить и начать работу с базами данных (БД);
* ***Поддержка различных платформ*** – PostgreSQL подходит для работы на всех популярных операционных системах (ОС): Windows, Linux, macOS. Пользователь получается готовую систему «из коробки» - чтобы установить и использовать программу, не нужны дополнительные инструменты;
* ***Поддержка множества типов данных*** – PostgreSQL поддерживает много разных типов и структур данных, в том числе сетевые адреса, данные в текстовом формате JSON, геометрические данные для координат геопозиций, XML и д.р. Все эти форматы можно хранить и обрабатывать в СУБД. Также, можно создавать собственные типы данных, их называют пользовательскими. Пользовательские типы данных нужны, чтобы упростить работу с БД или установить ограничения;
* ***Работа с большими размерами данных*** *–* размер базы в PostgreSQL не ограничен и зависит от того, сколько свободной памяти есть в месте хранения: на сервере, локальном компьютере или облаке. Максимальный размер таблицы – 32 терабайта. Этого более чем достаточно для хранения данных компании типа Amazon. Одна строка в БД не может превышать 1,6 терабайта, а максимальный размер одной ячейки – 1 гигабайт. В такую ячейку можно записать даже видео;
* ***Соответствие требованиям ACID***

*Атомарность* (от англ. *Atomicity*) - возможность совершать транзакции. Транзакции — это операции или действия, которые происходят полностью или не происходят вообще,

*Согласованность* (от англ. *Consistency*) - В отличие от атомарности, согласованность означает, что транзакция будет выполнена только в том случае, если она не нарушает согласованность данных в базе.

*Изолированность* (от англ. *Isolation*) - означает, что параллельные действия не влияют друг на друга.

*Устойчивость* (от англ. *Durability*) - означает, что уже выполненная транзакция не отменится из-за технических неполадок, например, если отключат свет в серверной.

Это четыре требования для надёжной работы систем, которые обрабатывают данные в режиме реального времени. Если все требования выполняются, данные не будут теряться из-за технических ошибок или сбоев при работе оборудования.

PostgreSQL соответствует всем четырём требованиям ACID и обеспечивает сохранность данных при выполнении транзакций и других работ.

* ***Поддержка сложных запросов*** - В PostgreSQL можно писать собственные функции — пользовательские блоки кода, которые выполняют те или иные действия. Эта возможность есть практически в любых СУБД, но PostgreSQL поддерживает больше языков, чем аналоги. Кроме стандартного SQL, в PostgreSQL можно писать на C и C++, Java, Python, PHP, Lua и Ruby. Он поддерживает V8 — один из движков JavaScript, поэтому JS тоже можно использовать совместно с PgSQL. Реализована поддержка Delphi, Lisp и прочих редких языков. При необходимости можно расширить систему под другие ЯП.

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

### Apache Kafka

## Взаимодействие клиент – серверной архитектуры

REST API — это архитектурный подход, который устанавливает ограничения для API: как они должны быть устроены и какие функции поддерживать. Это позволяет стандартизировать работу программных интерфейсов, сделать их более удобными и производительными.

Слово REST — акроним от Representational State Transfer, что переводится на русский как «передача состояния представления», «передача репрезентативного состояния» или «передача „самоописываемого“ состояния».

В отличие от, например, SOAP API, REST API — не протокол, а простой список рекомендаций, которым можно следовать или не следовать. Поэтому у него нет собственных методов. С другой стороны, его автор Рой Филдинг создал ещё и протокол HTTP, так что они очень хорошо сочетаются, и REST обычно используют в связке с HTTP.

## Проектирование сервисной архитектуры разрабатываемой системы

В этом разделе описаны основные этапе проектирования системы «Телемаркетинг».

До инструментов!!!

### Микросервисная архитектура

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

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

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

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

### Предметно-ориентированное проектирование сервисов. Проектирование домена «Займы».

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

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

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

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.

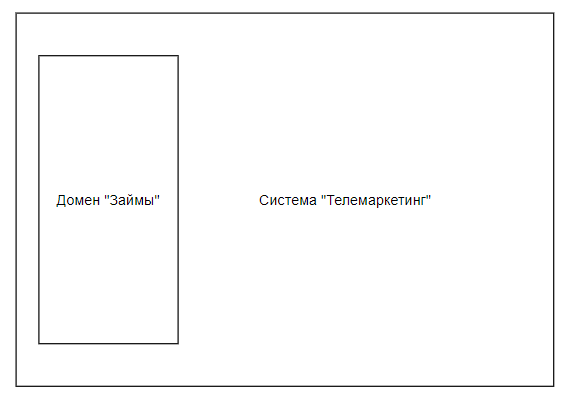


Рисунок 12 – Выделение доменной области

Доменной областью в проектируемой системе являются «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.

### Интеграции со сторонними сервисами

#### Сервис черных списков «BlackList»

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

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

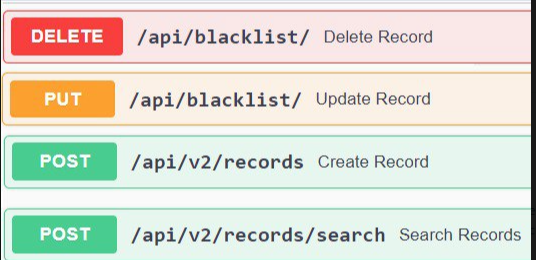


Рисунок 13 – API сервиса «BlackList»

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

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

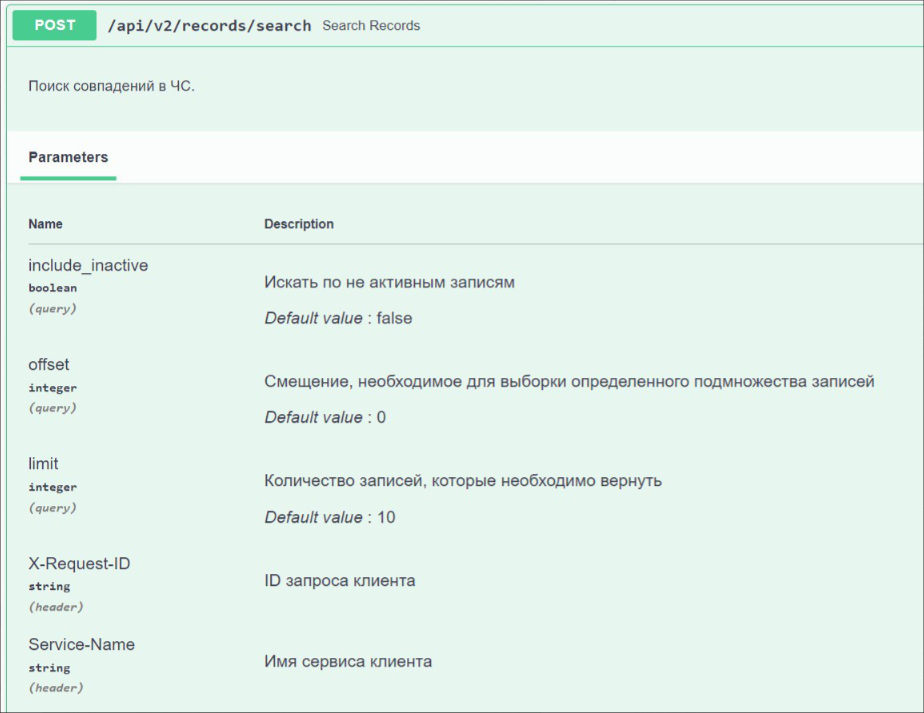


Рисунок 14 – Параметры запроса для поиска записей в черных списках

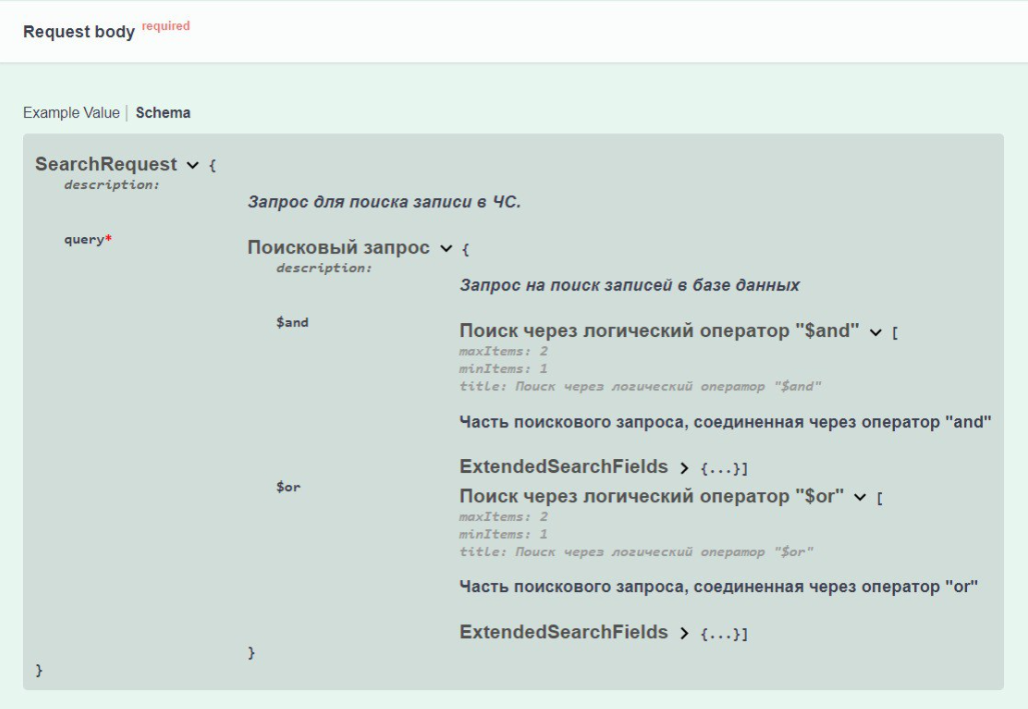


Рисунок 15 – Тело запроса для поиска записей в черных списках

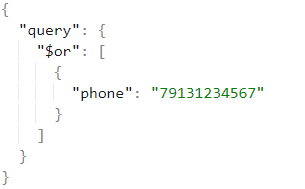


Рисунок 16 – Пример запроса на поиск клиента по номеру телефона



Рисунок 17 – Пример запроса на поиск клиента по номеру телефона или по уникальному идентификатору устройства

#### Сервис предварительных отказов «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.

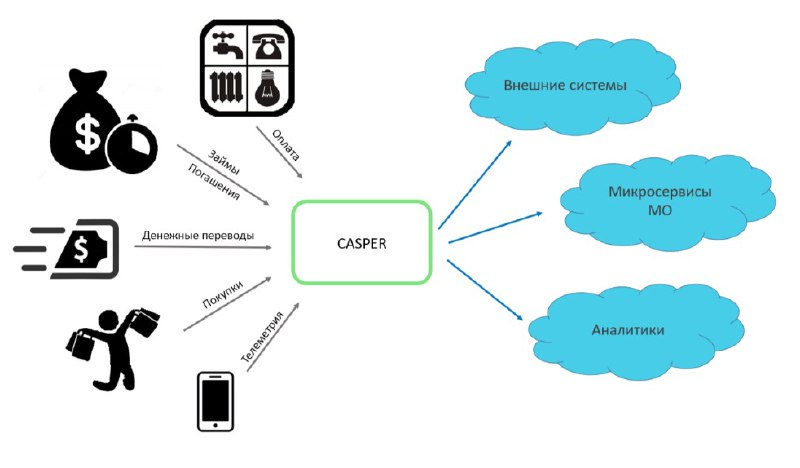


Рисунок 18 – Взаимодействие системы «Casper»

Назначение сервиса «Сasper» в системе «Телемаркетинг» - фильтра при обработке входящих заявок. Проверка по данным из этого сервиса позволяет принять решение – имеет ли смысл звонить по заявке или нет.

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

В ответ Casper будет отправлять JSON, в котором содержится большое количество клиентских данных вместе с проверками, с помощью которых мы будем определять нужно ли звонить клиенту.

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

При оценке клиента используются ML-модели, обученные дата-аналитиками на основе признаков клиента. Модели выделены в отдельный подсервис - ядро, разработкой которого занимаются аналитики. В ядре реализована логика расчета одобряемых сумм и подбор продуктов. Признаки - в нашей терминологии это фичи - насчитываются из данных о клиенте, например, количество переводов за последний год или сумма предыдущего займа.

В зависимости от значения фич на скоринге продуктов клиента определяют в некоторый грейд и одобряют некоторый набор продуктов. На скоринге от значения фич зависит одобренная сумма займа, которую клиент сможет полностью и вовремя вернуть.

Далее в зависимости от типа клиента, его продуктовой группы, флоу и насчитанных фич модель в ядре рассчитывает минимальную и максимальную суммы займа, который подойдет клиенту, и возвращает набор продуктов в этой ценовой категории.

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



Рисунок 19 – Метод определения типа заёмщика и набор доступных для него продуктов.



Рисунок 20 – Тело запроса для получения типа заемщика и набора продуктов

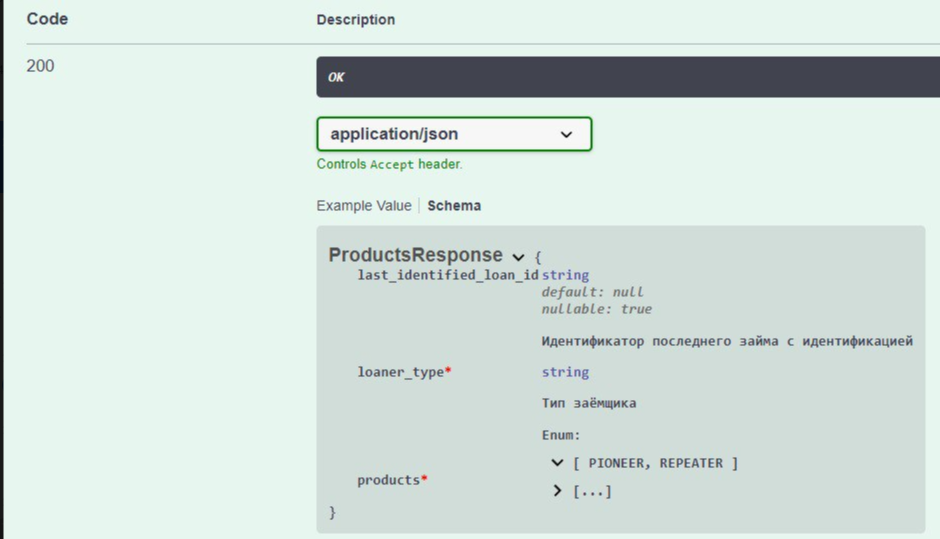


Рисунок 21 – Ответ запроса на получения типа заемщика и набора продуктов



Рисунок 22 – Набор продуктов доступный для клиента

#### Сервис телефонии MightyCall

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

### Проектирование основного контура «Телемаркетинг»

#### Общая архитектура системы «Телемаркетинг»

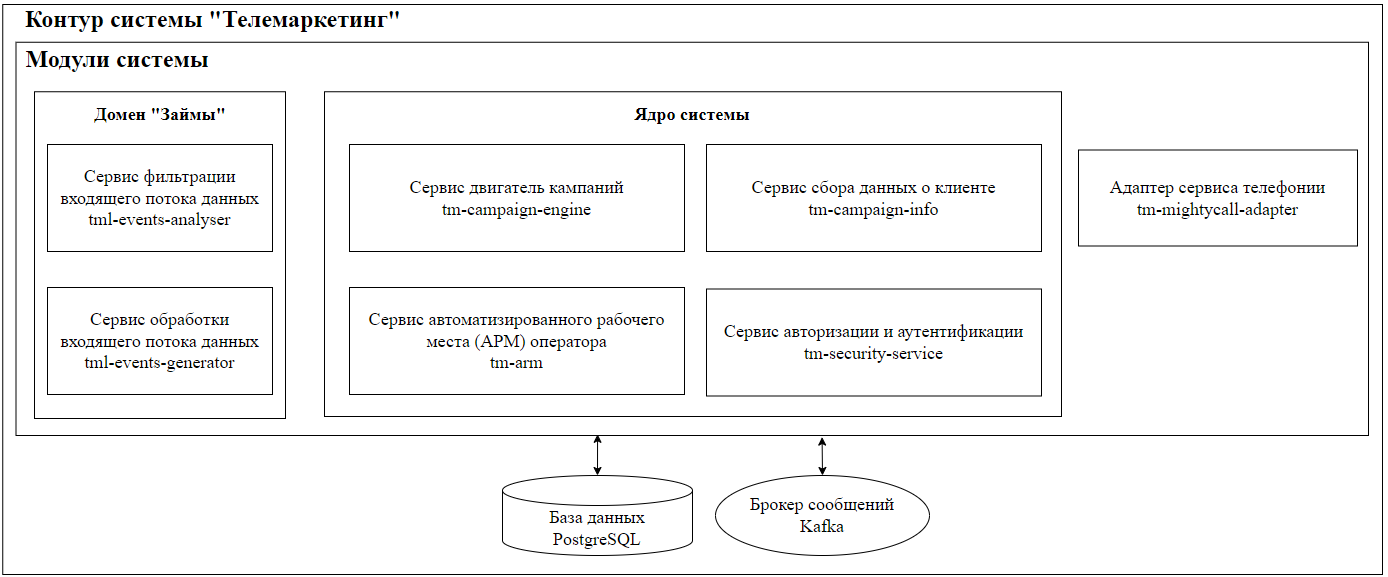


Рисунок 23 – Общая архитектура системы «Телемаркетинг»

#### Проектирование домена «Займы»

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

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

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

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

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

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

Если форматный контроль пройден, то каждую строку как слепок заявки сервис отправляет дальше на обработку в сервис фильтрации входящего потока данных «tml-events-analyser» через Kafka.

После успешной обработки файл перекладывается в папку «success».

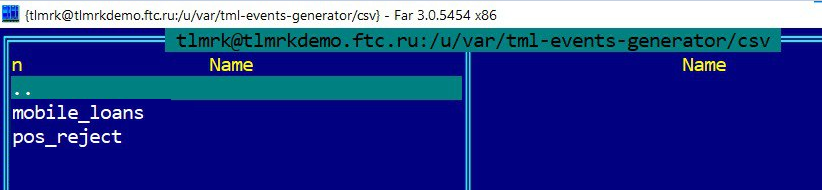


Рисунок 24 – Папки потоков данных

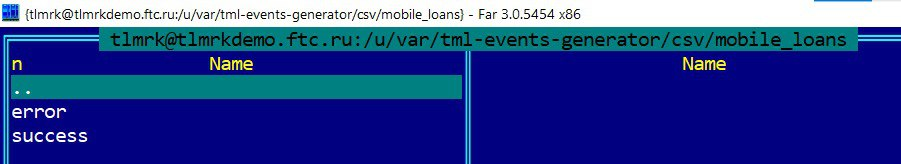


Рисунок 25 – Папки для успешной и не успешной обработки файлов

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

В этом модуле заявка проходит через каскад фильтров. Прохождение каскада означает что заявка становится кандидатом на участника кампании. Если один из фильтров не пройден, то обработка заявки прекращается – заявка точно не является кандидатом.

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

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

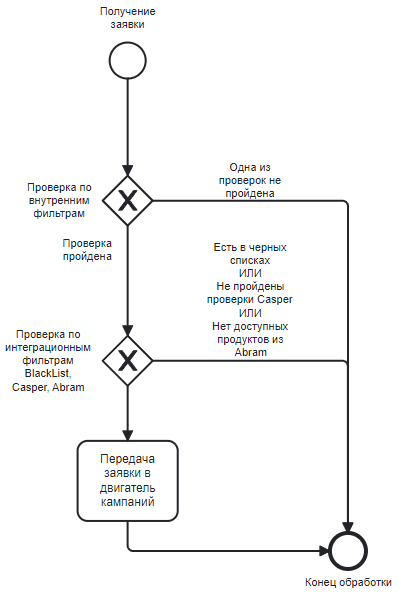


Рисунок 26 – Алгоритм фильтрации заявок

ПРИМЕР КОНФИГА В ПРИЛОЖЕНИИ

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

При успешном прохождении всех фильтров заявка отправляется в сервис двигатель кампаний «tm-campaign-engine» c помощью Kafka.

#### Проектирование ядра системы

##### Сервис двигатель кампании «tm-campaign-engine»

Модуль кампаний отвечает за обработку клиента в рамках его кампании.

Он получает сообщения от модуля tm-event-analyser и обрабатывает их. 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 по расписанную, которое задается в конфигурационном файле. Обработчик достает все заявки со статусом «NEW» - запускает действие на обзвон. Теперь задача переходит в статус «IN\_PROGRESS» и отправляется в адаптер сервиса телефонии – tm-mightycall-adapter, который отвечает за звонки.

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

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

##### Сервис сбора данных о клиенте tm-campaign-info

Объединить вместе с сервисом фильтрации. Обосновать можно так: получили все необходимые для процессинга данные в тот момент и можем их использовать в рамках кампании. 😊 Там же данные «оседают» в бд и т.д.

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

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

Аутентификация — это процесс проверки подлинности учетных данных пользователя с целью установления его личности. В основном контексте онлайн безопасности, аутентификация используется для проверки того, что пользователь, который утверждает, что он есть, действительно таковым является. Это могут быть различные учетные данные, такие как логин/пароль, биометрические данные (например, отпечаток пальца, распознавание лица), RFID-карты, одноразовые пароли и т.д.

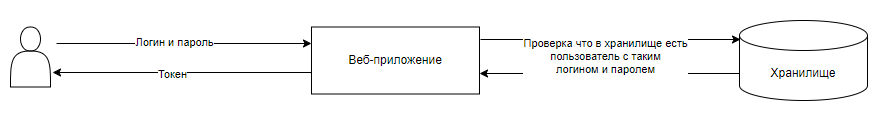
Когда аутентификация происходит успешно, пользователь получает доступ к системе или ресурсам.

Рисунок 27 – Процесс аутентификации

Основная цель аутентификации - защита от несанкционированного доступа и обеспечение безопасности данных и ресурсов путем проверки подлинности пользователей.

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.

Простыми словами, аутентификация ответственна за проверку того, кто пользователь, а авторизация - за то, что он может делать.

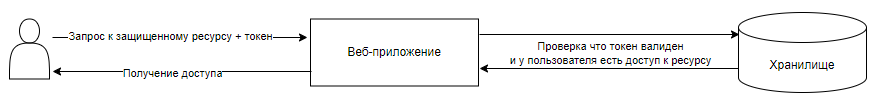
Механизмы авторизации могут включать в себя роли, разрешения, политики доступа и другие атрибуты, которые определяют контроль над функциональностью и данными. Например, администратор системы имеет больше прав, чем обычный пользователь, и поэтому он может видеть и изменять больше данных.

Рисунок 28 – Процесс авторизации

Авторизация играет важную роль в обеспечении безопасности информационных систем, поскольку она позволяет эффективно управлять доступом и предотвращать несанкционированные действия пользователей в системе.

Модуль имеет три HTTP POST-метода:

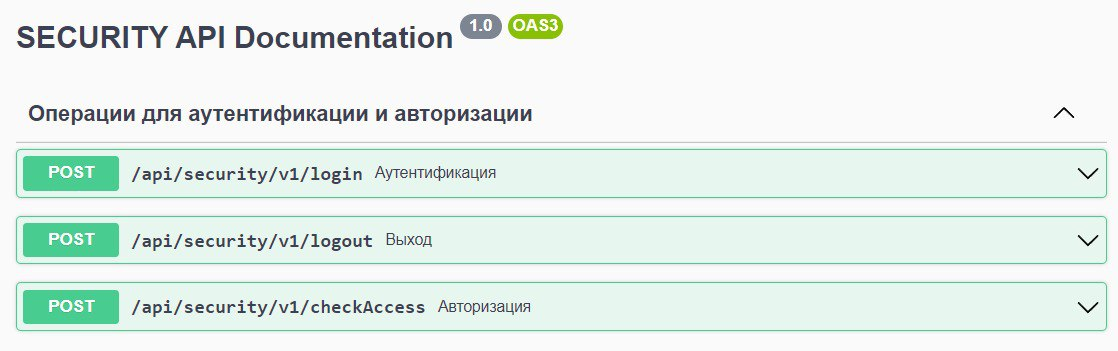
* «/api/security/v1/login» - аутентификация
* «/api/security/v1/logout» - выход
* «/api/security/v1/ checkAccess» - авторизация

Рисунок 29 – Документация модуля «tm-security-service»

###### Аутентификация

Метод используется для аутентификации пользователя по логину и паролю и генерации токена.

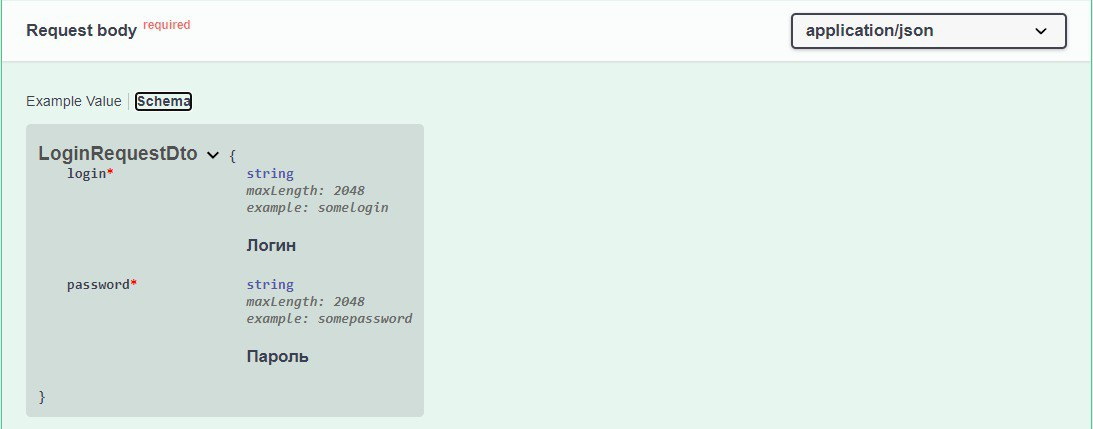
На вход метода поступают данные в формате JSON, необходимые для аутентификации пользователя. После получения запроса модуль осуществляет форматный контроль – проверяет наличие обязательных полей, ограничения, наложенные на поле, и так далее.

Рисунок 30 – Входные данные запроса «/api/security/v1/login»

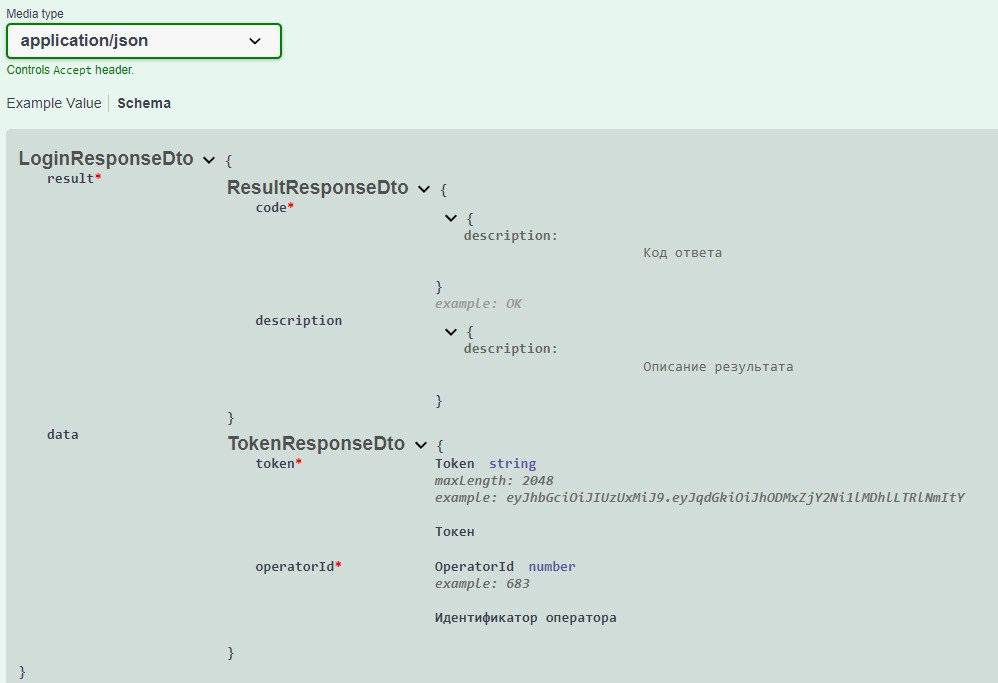


Рисунок 31 – Выходные данные запроса «/api/security/v1/login»

Если форматный контроль выполнен не успешно, то возвращается код и описание ошибки – «DATA\_FORMAT\_INCORRECT – Неверный формат данных запроса». Конец обработки.

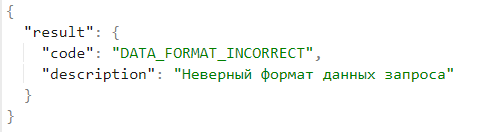


Рисунок 32 – Ответ при не успешном прохождении форматного контроля

Если форматный контроль пройден, то осуществляется поиск оператора в соответствующей таблице по следующим условиям:

* Логин
* Хэш от пароля

Если пользователь не найден или заблокирован, то возвращается соответствующий код и описание ошибки. Конец обработки.

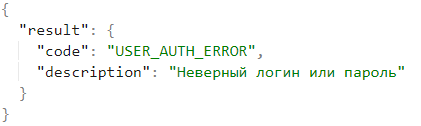


Рисунок 33 – Ответ при не успешном поиске пользователя

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

Если токен не найден или имеет истекший срок действия, то создается новый токен. Иначе срок действий токена продлевается и пользователь получает токен.

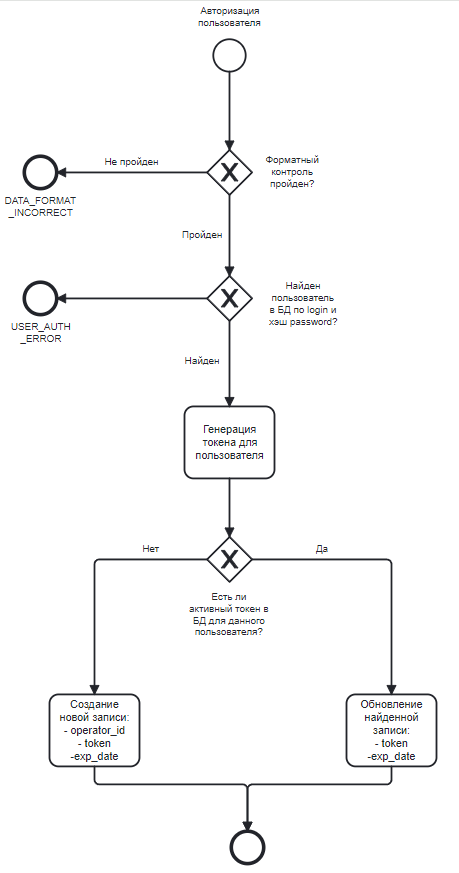
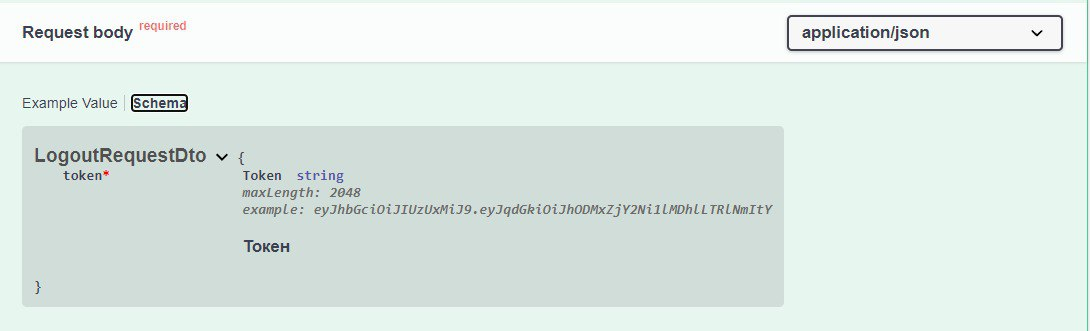


Рисунок 34 – Схема обработки запроса аутентификации ПЕРЕРИСОВАТЬ

###### Выход

Метод используется для выхода пользователя из системы.

На вход метода поступают данные в формате JSON, необходимые для аутентификации пользователя. После получения запроса модуль осуществляет форматный контроль – проверяет наличие обязательных полей, ограничения, наложенные на поле, и так далее.

Рисунок 35 – Входные данные запроса «/api/security/v1/logout»

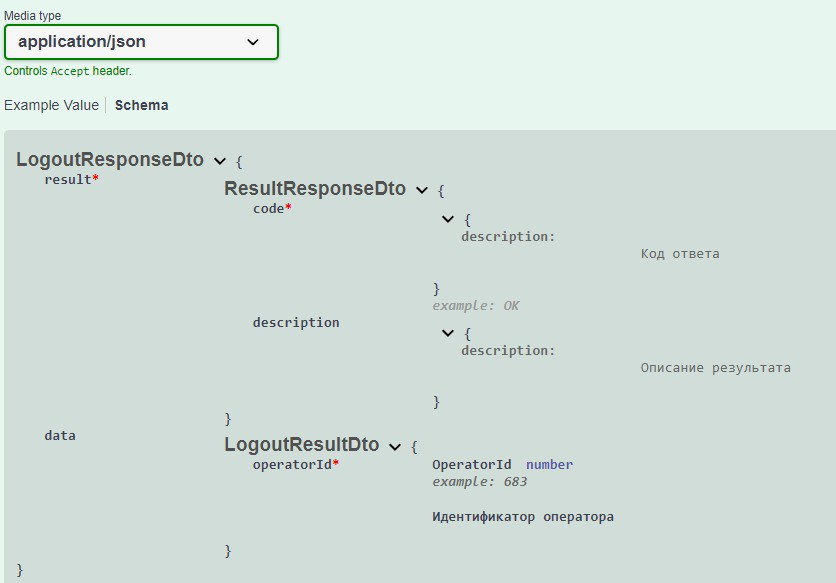


Рисунок 36 – Выходные данные запроса «/api/security/v1/logout»

Если форматный контроль выполнен не успешно, то возвращается код и описание ошибки – «DATA\_FORMAT\_INCORRECT – Неверный формат данных запроса». Конец обработки.

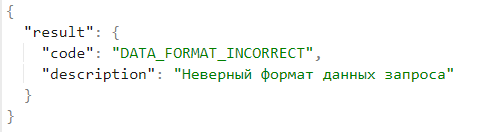


Рисунок 37 – Ответ при не успешном прохождении форматного контроля

Если форматный контроль пройден, то выполняется поиск токена в базе данных. Если токен найден, то удалить найденную запись. Если токен не найден, то конец обработки.

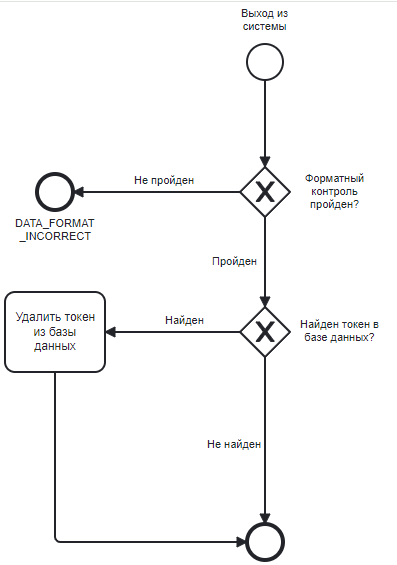


Рисунок 38 – Схема обработки запроса выхода из системы

###### Авторизация

Метод используется для проверки авторотационного токена и доступа к ресурсам.

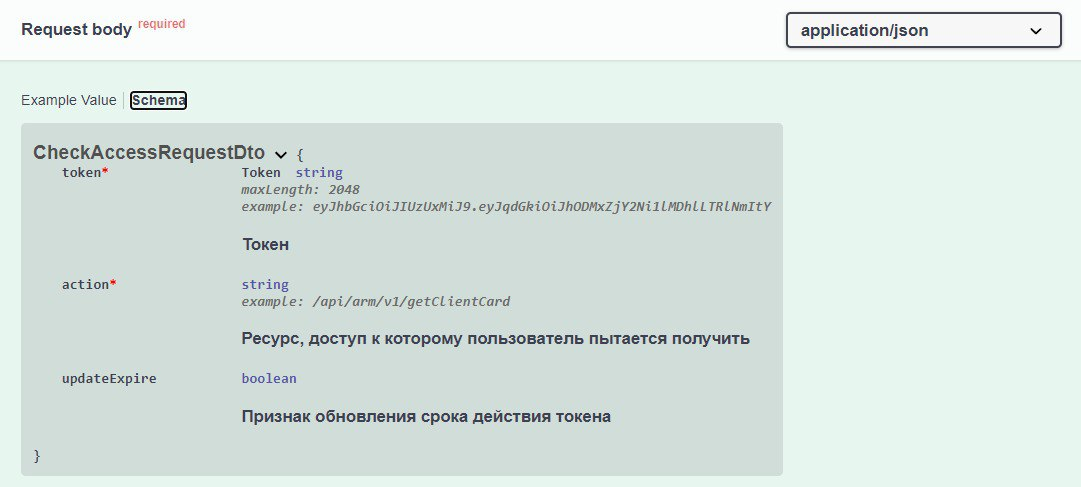


Рисунок 39 – Входные данные запроса «/api/security/v1/checkAccess»



Рисунок 40 – Выходные данные запроса «/api/security/v1/checkAccess»

Если форматный контроль выполнен не успешно, то возвращается код и описание ошибки «DATA\_FORMAT\_INCORRECT»:

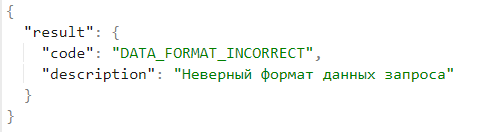


Рисунок 41 – Ответ при не успешном прохождении форматного контроля

Если форматный контроль выполнен успешно, то выполняется логическая обработка.

При недействительно токене сервис в ответ отправляет ошибку:

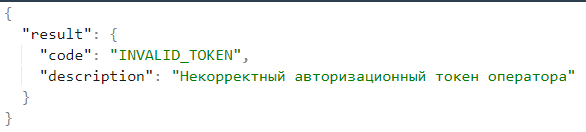


Рисунок 42 – Ошибка «INVALID\_TOKEN»

Если пользователь заблокирован или ему на хватает прав для доступа к запрашиваемому ресурсу:

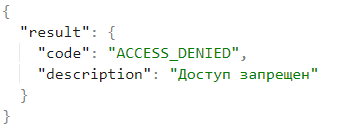


Рисунок 43 – Ошибка «ACCESS\_DENIED»

Если запрос имел на вход указание о том, что нужно продлить срок действия токена, то срок действия токена продлевается на время, которое задается из конфигурационного файла модуля.

В случае успешного прохождения авторизации пользователя сервис отправляет код «OK»

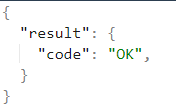


Рисунок 44 – Успешное прохождение авторизации

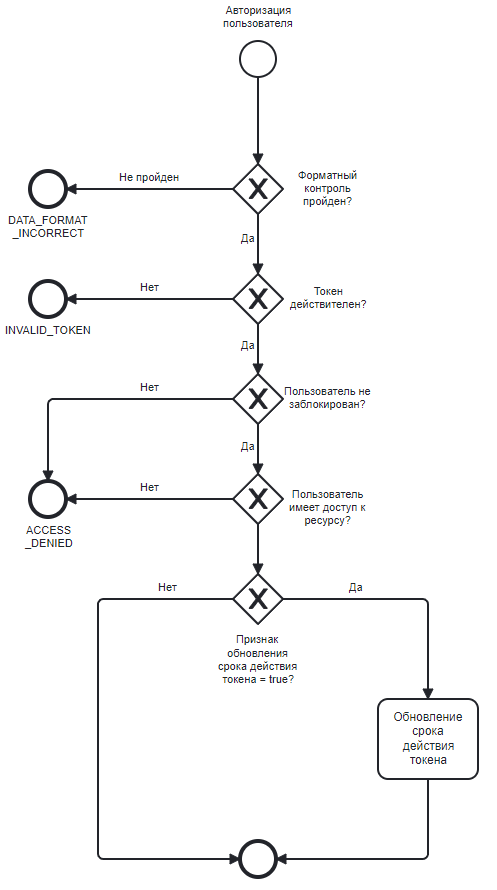


Рисунок 45 – Схема обработки процесса авторизации

##### Автоматизированное рабочее места оператора

#### Проектирование адаптера сервиса телефонии tm-mightycall-adapter

Модуль отвечает за работу системы «Телемаркетинг» с сервисом телефонии MightyCall. Основной задачей модуля является создание задач на обзвон и оброс сервиса MightyCall о начале новых звонков.

Перед работой с сервисом необходимо инициализировать оператора. Как только оператор заходит в АРМ, адаптер получает сообщение из Kafka для инициализации операторской сессии. После этого система должна обращаться в MightyCall не реже, чем раз в 15 минут для поддержания сессии и получения событий. В противном случае потребуется повторная инициализация оператора.

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

##### Открытие и закрытие сессии оператора

Открытие сессии

Адаптер принимает сообщение из Kafka, сформированное в результате аутентификации оператора в АРМ оператора. Далее сервис сохраняет оператора в список активных в БД и в кэш. После этого происходит инициализация оператора в сервисе MightyCall. После этого запускается процесс опроса событий для оператора.

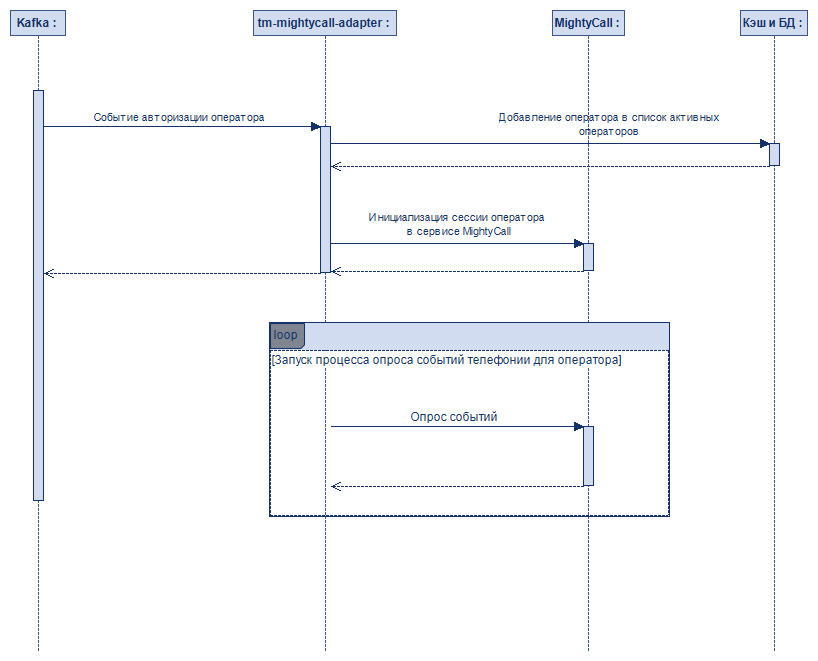


Рисунок 46 – Открытие сессии оператора

Закрытие сессии

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

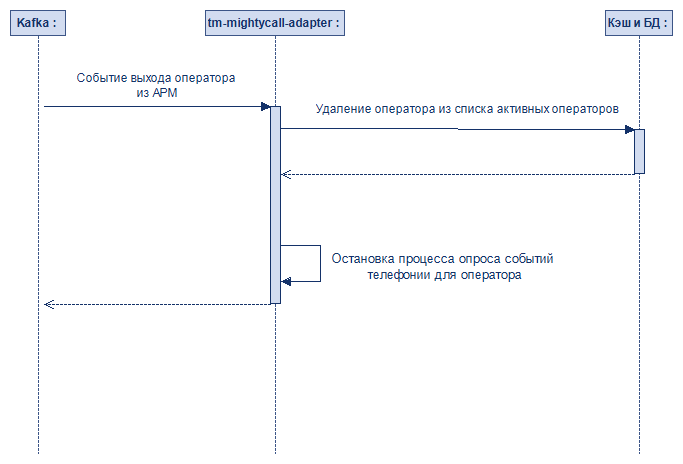


Рисунок 47 – Закрытие сессии оператора

##### Добавление и удаление задачи в очередь MightyCall

Адаптер принимает из Kafka сообщение, сформированное в результате работы сервиса обработки входящего потока – tm-campaign-engine, и добавляет или удаляет запись в очередь MightyCall.

Обработки сообщений Kafka на добавление задачи

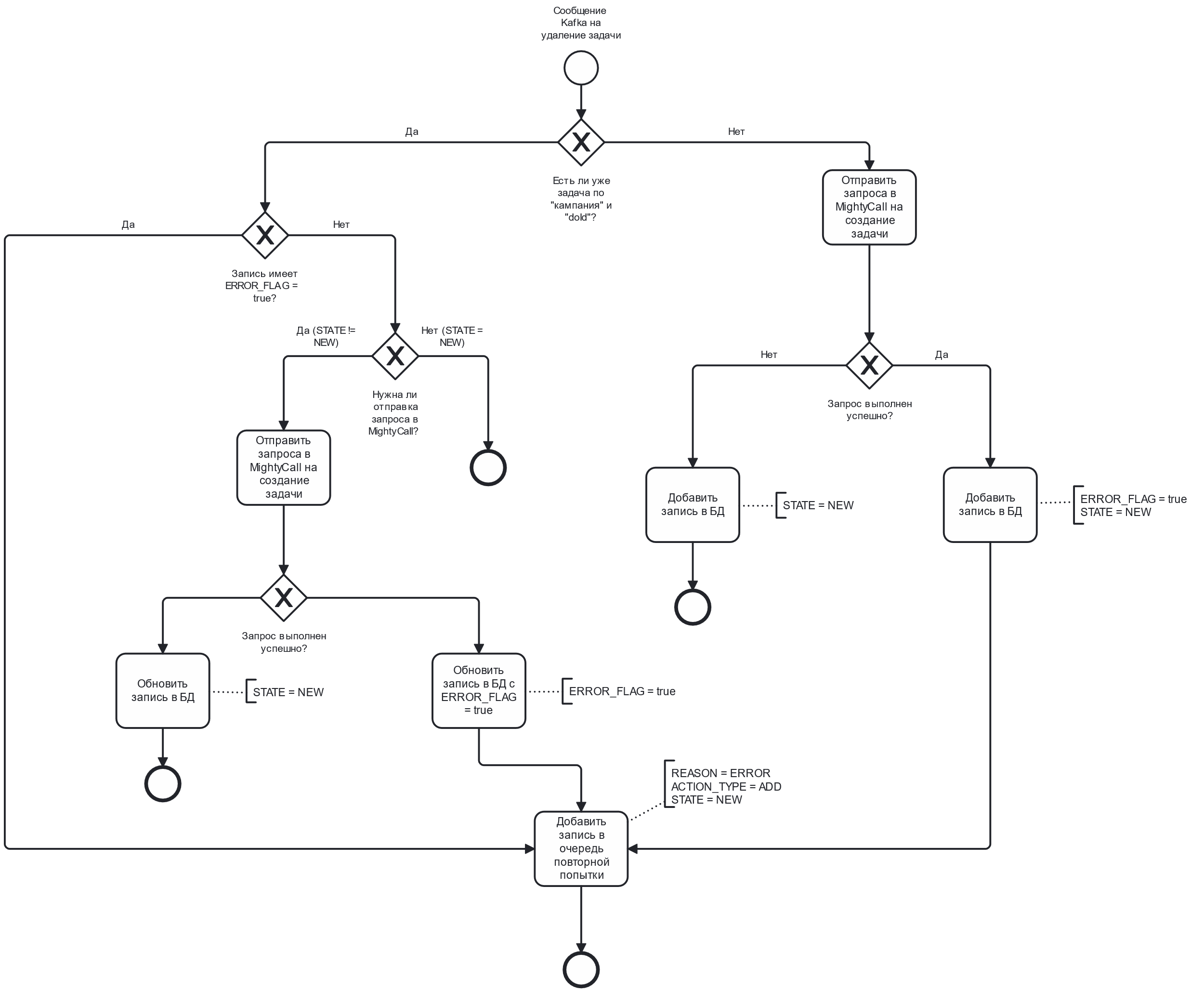


Рисунок 48 – Процесс добавления задачи

Обработка сообщений Kafka на удаление задачи

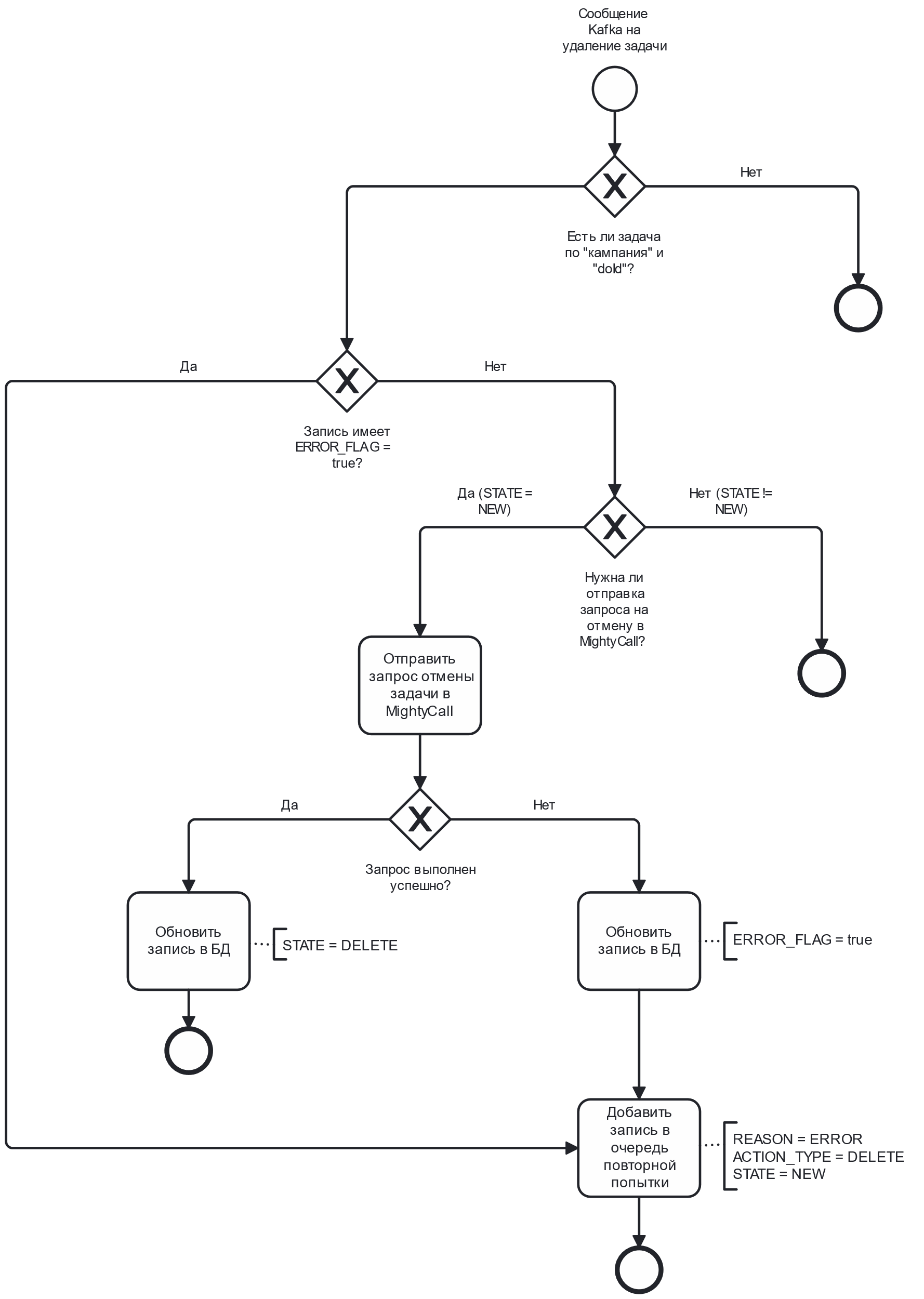


Рисунок 49 – Процесс удаления задачи

##### Процесс опроса событий телефонии для оператора

По кажд

##### Повторная попытка отправки задачи в MightyCall

Повторная попытка — это процесс, который запускается по расписанию, которое задание в конфигурационном файле модуля. Процесс достает из БД все записи с ошибками и в хронологическом порядке отправляет запросы в MightyCall.

Если запрос отправлен успешно, или его не нужно отправлять, то запись считается успешно отработанной. В противном случае все неуспешные запросы отправляется на повторную попытку снова.

*Можно нарисовать схемку, как в прошлых пунктах*

## Проектирование структуры хранения данных



Рисунок 50 – Таблицы в базе данных

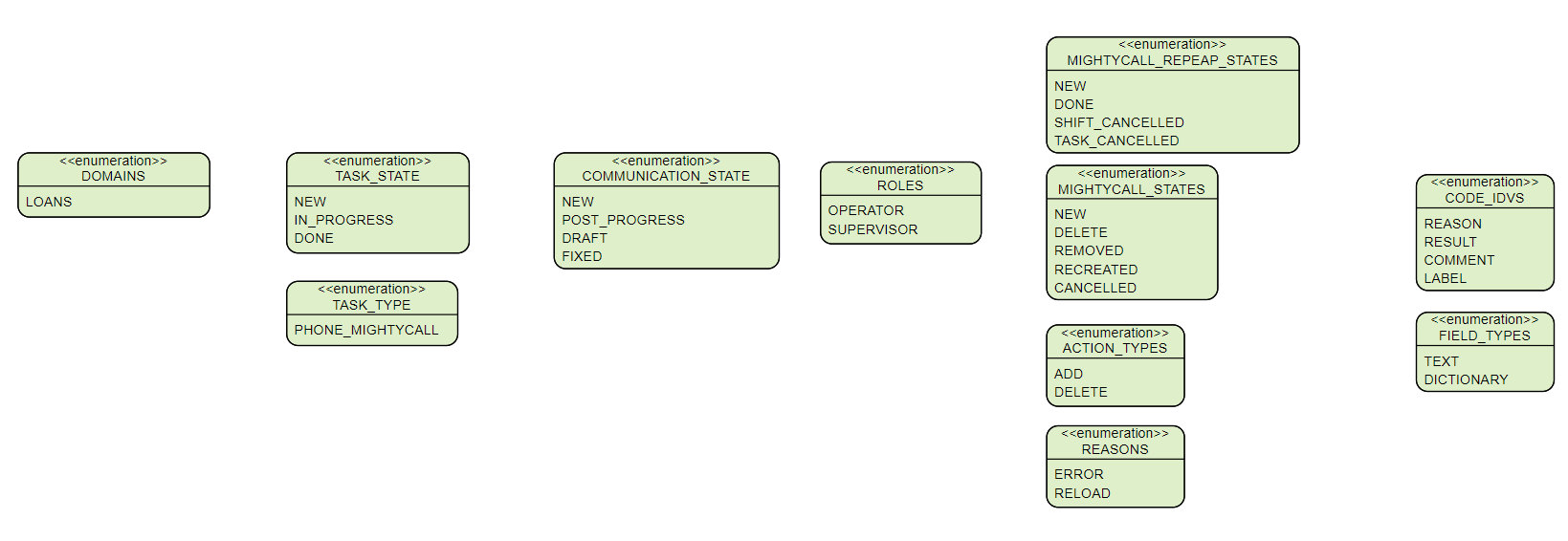


Рисунок 51 – Пользовательские типы в базе данных

# ГЛАВА 3

+- 15 страниц.

Описываю то что получилось. Т.к. разрабатываю программу, то в конце нужно написать некое руководство пользователя: картинки экранов, некоторые пользовательские сценарии.

«Делаем …, видим …»

# ЗАКЛЮЧЕНИЕ

Возврат к введению, передо мной стояла задача …, я её выполнил успешно …

Чего добились, куда двигаться дальше…

# СПИСОК ЛИТЕРАТУРЫ

1. https://www.mango-office.ru/products/calltracking/for-marketing/osnovy/chto-takoe-telemarketing-i-kak-ego-primenyat/
2. https://sendpulse.com/ru/support/glossary/telemarketing
3. https://www.unisender.com/ru/glossary/chto-takoe-telemarketing/#anchor-6
4. <https://1crm.ru/help/telemarketing/>
5. <https://megacrm.ru/telemarketing>
6. <https://martinfowler.com/bliki/DomainDrivenDesign.html>
7. <https://spring.io/projects/spring-framework/>
8. https://kafka.apache.org/intro

Лучше дописывать по ходу работы, т.к. так проще будет аннотировать. 15-20 источников. Если нет ссылки на источник, то он тут не нужен.

«Доп. инф. по теме можно прочитать …»

# ПРИЛОЖЕНИЯ

Буду использовать, если не хватает объема. В сумме работа должна быть +- 70 страниц

Обычно сюда выносятся кусочки кода, если они больше страницы. Можно вынести теоретические информацию, но лучше внешний ресурс.