Задача.

Разработать ПО для взаимодействия сервисов для ипотечного проекта, использование Moleculer.js в качестве фреймворка для создания микросервисной архитектуры.

Основные шаги:

**Колл-центр:**

Когда Неквалифицированный лид (НЛ) звонит в колл-центр, сотрудник колл-центра взаимодействует с Moleculer-сервисом "Колл-центр" для обработки запроса.

Сервис "Колл-центр" может запустить действие, для обработки данных, предоставленных НЛ.

**SMS-шлюз:**

Сервис "Колл-центр" может взаимодействовать с сервисом "SMS-шлюз" для отправки SMS с уникальной ссылкой на страницу регистрации хранилища.

Сервис "SMS-шлюз" взаимодействует с Twilio API для отправки SMS.

**Хранилище:**

После отправки SMS НЛ переходит по уникальной ссылке и взаимодействует с сервисом "Защищенное хранилище".

Сервис " хранилище" может предоставлять методы для загрузки документов.

**Система управления:**

Сейл, имеющий доступ к порталу системы управления, может использовать сервис "Система управления " для поиска лида по номеру телефона или другим критериям.

Сервис "Система управления" может предоставлять методы для просмотра данных о лиде и его загруженных документах.

*Техническое задание*

**1.Введение:**

Описание проекта:

Проект направлен на создание системы управления ипотечными заявками для банка. Система будет обеспечивать цикл от подачи заявки до проверки документов

Цели проекта:

Упрощение процесса обработки ипотечных заявок.

Повышение эффективности работы колл-центра и сотрудников по продажам.

Обеспечение безопасного хранения данных клиентов.

**2. Обзор предметной области:**

2.1 Процесс получения ипотеки:

Клиент связывается с колл-центром, проходит процедуру анкетирования, после чего информация передается в базу данных. Сейл получает доступ к системе для дальнейшей работы с клиентами.

**3. Функциональные требования:**

3.1 Колл-центр:

Система обработки входящих звонков.

Заполнение формы лида с учетом информации, полученной от клиента (номер телефона и ФИО?).

Отправка SMS-уведомления с уникальной ссылкой для регистрации (uuid или crypto)

3.2 База данных:

Хранение информации о лидах, документов, статус заявки

**4. Нефункциональные требования:**

4.1 Производительность:

Система должна поддерживать обработку не менее N заявок в минуту в периоды пиковой нагрузки.

4.2 Безопасность:

Шифрование данных при передаче и хранении. (PostgreSQL- ПШД)

Аутентификация с использованием механизмов безопасности(passport.js)

**5. Требования к архитектуре:**

5.1 Moleculer.js:

Использование Moleculer.js в качестве фреймворка для построения микросервисной архитектуры.

Микросервисы для колл-центра, базы данных и система управления

5.2 PostgreSQL:

Использование PostgreSQL в качестве базы данных для хранения информации о клиентах и заявках.

5.3 Взаимодействие:

Коммуникация между микросервисами через Moleculer Брокер.

RESTful API для взаимодействия с внешними системам

1

2 5

С

Система управления

Портал КЦ

Портал для продажников

Админ - Панель

ЛК для загрузки документов

4

3

БД/Хранилище

1. Лид звонит в КЦ
2. КЦ заполняет форму Лида с номером телефона и добавляет в базу
3. Система управления отправляет СМС лиду (ссылка)
4. Лид переходит по ссылке и загружает документы
5. Сейл подтверждает (или возвращает ошибки в документах) и т.д

Примерная архитектурная диаграмма взаимодействия:

КЦ-сервис

Внешний сервис

(напр. Twilio API)

SMS-шлюз

Хранилище

Система управления для Сейла

База данных

Moleculer Брокер