

**Введение:**

1.1. Цель проекта: Разработать систему для эффективного управления запросами на ипотеку через колл-центр с использованием технологии Node.js.

**Описание функциональности:**

2.1. Прием звонка от Неквалифицированного Лида (НЛ):

2.1.1. Реализовать систему идентификации номера телефона НЛ. -+

2.1.2. Вести журнал звонков с данными о дате, времени и номере телефона.-+

2.2. Заполнение формы лида на портале:

2.2.1. Создание веб-приложения на Node.js с использованием Express.js.

2.2.2. Разработка формы с полями для данных НЛ (имя, контактные данные и т.д.).

2.2.3. Интеграция с базой данных для хранения информации о лидах.

2.3. Отправка SMS с ссылкой на регистрацию:

2.3.1. Использование SMS-шлюза для отправки сообщений.

2.3.2. Генерация уникальной ссылки и отправка на указанный номер телефона.

2.3.3. Логирование отправленных SMS.

2.4. Загрузка файлов через ссылку:

2.4.1. Создание защищенного хранилища на сервере.

Добавить форму для загрузки

2.4.2. Реализация механизма загрузки файлов по уникальной ссылке.

2.4.3. Запись информации о загруженных файлах в базу данных.

2.5. Доступ сотрудника-сейла к порталу:

2.5.1. Аутентификация сотрудника с использованием системы учетных записей.

2.5.2. Разработка интерфейса для поиска лида по номеру телефона.

**Технические требования:**

3.1. Технологии:

3.1.1. Node.js для серверной части.

3.1.2. Express.js для создания веб-приложения.(molecularjs)

3.1.3. …БД…. для хранения данных о лидах.

3.1.4. Использование SMS-шлюза для отправки сообщений.( twilio)

3.2. Безопасность:

3.2.1. Шифрование данных передаваемых между сервером и клиентами.

3.2.2. Ограниченный доступ к защищенному хранилищу файлов.

3.3. Архитектура:

3.3.1. Модульная структура с фронтендом, бэкендом, базой данных и хранилищем файлов.

**Развитие проекта:**

4.1. План масштабирования системы для обработки роста запросов.

4.2. Возможность интеграции с системами других отделов.

**Требования к документации:**

5.1. Подробное описание API.

5.2. Инструкции по установке и настройке системы.

5.3. Документация по обслуживанию и обновлению.

**Тестирование:**

6.1. Разработка тестов для проверки функционала.

6.2. Тестирование безопасности.

**Вывод:**

7.1. Завершение проекта означает функциональность и удовлетворение требований заказчика.

**План работы:**

8.1. Фазы разработки (анализ, проектирование, реализация, тестирование).

8.2. Определение сроков для каждой фазы.

Порядок разработки

2. Создание базовой структуры приложения:

Создание основных директорий (например, models, routes).

Создание основного файла приложения (например, app.js).

3. Работа с базой данных:

Установка и настройка MongoDB.

Создание схемы и модели для лидов в файле models/lead.js.

module.exports = mongoose.model('Lead', leadSchema);

4. Создание роутов для обработки данных:

Создание файла routes/leads.js для обработки операций с лидами.

5. Подключение роутов к приложению:

В основном файле приложения (app.js) подключить и использовать созданный роут.

6. Тестирование базового функционала:

Запустить сервер (node app.js) и использовать инструменты, такие как Postman, для проверки работы созданного эндпоинта.

7. Добавление дополнительного функционала:

Реализовать дополнительные эндпоинты и функциональность в соответствии с вашими требованиями.

8. Интеграция с SMS-шлюзом:

Установка и настройка библиотеки для отправки SMS (например, twilio).

Добавление кода для отправки SMS с уникальной ссылкой в файле routes/leads.js.

9. Разработка фронтенда:

Создание веб-формы для сотрудника колл-центра (HTML, CSS, JavaScript).

Интеграция веб-формы с бэкендом через AJAX или другие методы.

10. Создание защищенного хранилища:

Реализация механизма создания уникальных ссылок.

Разработка эндпоинта для загрузки файлов в соответствии с уникальными ссылками.