ООО «Диджитал Маркетс»

Система проектного управления «Дистанционный офис. Облако»

Описание процессов, обеспечивающих поддержание жизненного цикла

Язык программирования: Typescript - 85%.

Информация о персонале, необходимом для обеспечения поддержки ПО:

- Руководитель разработки
- Frontend-разработчик
- Backend-разработчик
- Тестировщик

Используемые технологии:

В процессе разработки мы используем современные технологии, которые обеспечивают высокую производительность, масштабируемость и надежность наших решений.

Backend. Основной язык разработки: NestJS, который предоставляет модульную архитектуру и высокую гибкость для разработки серверной части.

Frontend:

Веб-приложения: разрабатываются на Vue.js с использованием фреймворка Nuxt, который позволяет создавать серверные рендеринг-приложения (SSR) и статические сайты. Для управления состоянием клиента применяется Pinia, обеспечивающая реактивное хранилище данных.

Мобильные приложения:

Кроссплатформенная разработка: если технически возможно, нативные мобильные приложения разрабатываются на Kotlin Multiplatform, что позволяет использовать один и тот же код на Android и iOS.

Нативные технологии: в случаях, когда необходима более высокая производительность или сложность приложения, используются Jetpack Compose для Android и SwiftUI для iOS.

Структура приложения

Frontend:

Публичная часть: Основной интерфейс, доступный для пользователей, разработан с использованием Vue.js и Nuxt. Включает в себя все основные функции и возможности, доступные конечным пользователям.

Административная панель: Специальный интерфейс для администраторов и модераторов системы, который позволяет управлять данными, следить за состоянием системы и выполнять административные задачи.

Backend:

API: Вся серверная логика организована в виде API, разделенного по модулям в рамках NestJS. Каждый модуль отвечает за определенную функциональность, что позволяет легко поддерживать и масштабировать приложение.

Эта архитектура позволяет легко добавлять новые функции и модули, обеспечивая при этом высокую производительность и надежность системы.

Запуск и мониторинг приложения:

Веб-приложение запускается в контейнере Docker. При его работе приложение генерирует логи — записи всех событий и операций, которые происходят в процессе выполнения программы. Эти логи автоматически отправляются в Loki — специальный инструмент для хранения и обработки логов. В Grafana настроены дашборды и алерты, которые следят за логами в Loki. Если в логах появляется ошибка, связанная с 500 кодом (критическая ошибка на стороне сервера), Grafana отправляет уведомление об этом.

Уведомления об ошибках:

Когда Grafana фиксирует появление 500 ошибки, она автоматически отправляет алерт (уведомление) в Telegram. В этом уведомлении указано, что произошла критическая ошибка, и в каком компоненте системы она возникла. Команда, получившая алерт в Telegram, знает, что необходимо срочно разобраться с проблемой, так как 500 ошибка может привести к тому, что пользователи не смогут воспользоваться веб-приложением.

Анализ и устранение неисправностей:

После получения алерта команда начинает разбирать проблему. Сначала проверяют логи в Loki, чтобы найти конкретное место в коде или событие, вызвавшее 500 ошибку. Затем анализируют эти данные, чтобы понять причину ошибки — возможно, это неправильный запрос, сбой в работе базы данных или

ошибка в коде приложения. Как только причина найдена, команда приступает к её устранению, после чего обновляет приложение и повторно проверяет его работу, чтобы убедиться, что ошибка больше не возникает.