Литература:
1. SSL VPN, 2024. – URL: https://www.cisco.com (дата обращения 17.09.2024).
2. Network Security Docs – IPSec VPN, 2024. – URL: https://docs.paloaltonetworks.com (дата обращения 15.09.2024).
3. *Соболев М.А.* Сравнительный анализ российского стандарта шифрования по ГОСТ Р 34.12–2015 и американского стандарта шифрования AES // Политехнический молодежный журнал. – 2022. – № 04(69). – DOI: 10.18698/2541-8009-2022-04-785.

Макшаков С.В.

Система поддержки принятия решений в задачах технического переоснащения в железнодорожной отрасли

Аннотация: Описание функционала и программной архитектуры системы поддержки принятия решений в задачах технического переоснащения в подразделениях ОАО «РЖД». Назначение системы – автоматизация процессов формирования, мониторинга и контроля выполнения программы технического переоснащения. Цель создания системы – повышение эффективности, точности планирования и реализации проектов по техническому переоснащению. Демонстрируется архитектурный подход и анализируются основные технологии, используемые при построении системы.

Ключевые слова: система поддержки принятия решений, техническое переоснащение, архитектура программного обеспечения, веб-приложение, АО «ВНИИЖТ», ОАО «РЖД»

В работе представлена автоматизированная информационно-аналитическая система, разработанная АО «ВНИИЖТ» для реализации технического переоснащения в холдинге ОАО «РЖД».

1. Описание проблематики и постановка вопросов

Программа технического переоснащения в железнодорожной отрасли решает проблемы обеспечения экономической и

социально-политической безопасности страны, связанной со стратегически важной инфраструктурой.

Предполагаю рассмотреть следующие вопросы:

– планирование и стратегическое управление техническим переоснащением в железнодорожной отрасли;

– методы построения средств информационной поддержки принятия решений в системе управления техническим переоснащением.

При создании системы задавались следующие вопросы: с чего начать? Что необходимо для успешного выполнения программы по техническому переоснащению в рамках всего холдинга РЖД?

2. Этапы разработки автоматизированной информационно-аналитической системы

Очевидно, что для начала нужно спланировать процесс технического переоснащения. Было решено создать программное обеспечение, способное помогать при планировании и принятии решений.

Для начала нужно было определиться со списком продукции, подлежащей замене. Требовалось создать справочник такой продукции, их аналогов, базы потенциальных поставщиков и производителей.

Затем необходимо было определиться с приоритетом переоснащения. Для этого был организован оперативный доступ к критически важной продукции с указанием прогноза даты наступления риска.

И, конечно, было важно иметь возможность добавления в систему программ и подпрограмм технического переоснащения, связанных с подразделениями ОАО «РЖД». Для этого в системе был создан раздел, позволяющий формировать подпрограммы подразделений холдинга и формирования на их основе итоговой программы, с указанием стоимости, даты начала и даты окончания, а также возможности согласования и добавления документации к программе.

Тут мы подходим к сложным вопросам по организации в системе внутренней документации. Как структурировать, хранить и искать документацию? Как искать информацию внутри документации? Как поддерживать её в актуальном состоянии и

иметь возможность обсуждать её в режиме диалога? Не предусмотреть это было бы ошибкой. Решено было создать отдельный модуль для хранения документации и метаданных о ней с возможностью поиска по содержимому и связать хранящуюся в этом модуле документацию с основной реляционной базой данных для получения быстрого доступа к информации о названии файлов, их типах, датах создания и комментариям к документации.

Были и другие задачи при построении системы. Например, для обеспечения безопасности системы и ограничения прав доступа для пользователей необходимо было создать пользовательскую ролевую модель и модуль администрирования. Приходилось обеспечивать не только такие функциональные требования, как безопасность, надёжность и доступность системы, и, конечно, требовалось реализовать удобный пользовательский интерфейс.

Это были первоочередные задачи, которые были реализованы на первом этапе развития автоматизированной информационно-аналитической системы. Актуальность программного обеспечения и его активное использование привело к решению о необходимости расширения функционала системы.

Второй этап потребовал разработки гораздо большего количества модулей и функциональных требований:

– стало понятно, что необходим модуль по работе с мероприятиями, декомпозиции мероприятий на этапы, а этапов на задачи;

– создан раздел объектов технического переоснащения, позволяющий структурировать объекты, выстраивая иерархические системы (к примеру: объекты, узлы, подвижной состав);

– выполнен автоматизированный расчёт оценки рисков с градацией по уровню критичности и связи их с продукцией и мероприятиями;

– ограничен доступ филиалов к редактированию объектов, внесённых другими филиалами;

– создан раздел инфографики;

– реализована возможность выгрузки информации в различных разрезах;

– разработан модуль работы с внешними организациями;

– разработан модуль для обратной связи для прямого общения между участниками программы технического переоснащения;

– создана возможность просмотра истории изменений;
– проведена доработка интерфейса в соответствии с пожеланиями пользователей.

На данный момент идёт разработка третьего этапа развития системы:

– добавлена оценка приоритетности мероприятий;
– разработан модуль новостей и уведомлений;
– осуществлены доработки, связанные с безопасностью системы;
– внедрён механизм упрощённой интеграции с внешними системами;
– разрабатывается нейросетевой помощник для детализации мероприятий, связанных с техническим переоснащением.

Описание этапов разработки способно продемонстрировать актуальность системы. На этом описание бизнес-логики заканчиваю и перехожу к описанию технической части.

3. Архитектура программного обеспечения

Автоматизированная информационно-аналитическая система технического переоснащения представляет из себя веб-приложение [1], иными словами, клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется преимущественно на сервере. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому система является межплатформенной.

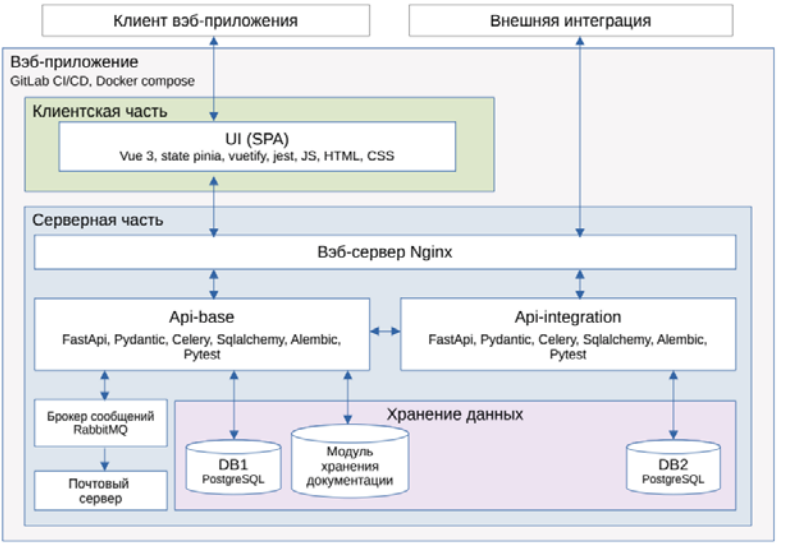

Рисунок 1 – Архитектура системы

Взаимодействие между клиентом и сервером осуществляется по сети, по средствам HTTP запросов с использованием SSL шифрования. В качестве архитектурного стиля взаимодействия используем REST.

Архитектура веб-приложения представлена на рисунке 1 в виде трехуровневой структуры: клиентская часть, серверная часть и слой хранения данных. Для лучшего понимания архитектуры разберёмся с каждым её элементом по отдельности, подробнее рассмотрим порядок взаимодействия между элементами системы и используемые в них технологии.

Клиентская часть разработана на реактивном фрэймворке Vue JS [2], в основе которого используются JavaScript, HTML, CSS. Этот фрэймворк называют реактивным не за скорость, хотя скорость является одной из его преимуществ, а за использование Virtual DOM, что позволяет обновлять на странице только те HTML-элементы, которые были изменены. Реактивность лежит в основе SPA (single-page application), которое загружает только одну HTML-страницу и динамически обновляет её содержимое без необходимости полной её перезагрузки при взаимодействии с

пользователем. Помимо этого, Vue JS позволяет использовать большое количество библиотек, упрощающих разработку. Клиентская часть взаимодействует с сервером через API, отправляя HTTP-запросы на определённые адреса.

Веб-сервер Nginx [3] выступает в качестве обратного прокси-сервера и обеспечивает асинхронность при обработке запросов в неблакирующем режиме. Он принимает входящие HTTP-запросы от клиентов и перенаправляет их на соответствующие сервисы.

API-сервисы разработаны на Python с использованием фрэймворка FastAPI [4]. Он поддерживает асинхронное программирование и предлагает автоматическую документацию API с использованием OpenAPI, обрабатывает запросы и валидирует их с помощью Pydantic. Основным преимуществом FastAPI является высокая производительность, удобство разработки и возможность использования различных библиотек. Мы используем Celery [5] в качестве асинхронной очереди задач для выполнения фоновых и отложенных работ, связанных с генерацией отчётов и отправкой уведомлений. RabbitMQ выступает в качестве брокера сообщений, он позволяет Celery выполнять задачи асинхронно и обеспечивает надёжное хранение задач, позволяя избежать их потери в случае сбоев.

В общей архитектуре два сервиса API. Api-base предназначен для работы с клиентом веб-приложения, почти вся бизнес-логика, описанная в предыдущем разделе, реализуется здесь. Api-integration предназначен для работы с запросами от внешних систем, данные для ответа он запрашивает через Api-base, а не напрямую из базы данных. Решение вынести Api-integration в отдельный сервис было связанно с обеспечением большей безопасности.

Хранение данных для каждого API сервиса организованно в отдельных базах данных. В качестве основной СУБД используется PostgreSQL [6]. PostgreSQL – это реляционная база данных, поддерживающая сложные запросы, транзакции и предоставляющая множество инструментов для работы с данными.

Для взаимодействия с FastApi используется ORM (Object-Relational Mapping) SQLAlchemy. Для автоматизации процесса миграции баз данных, управления изменениями и поддержания истории изменений, сделанных в моделях данных, используется Alembic.

Для хранения документации и поиска по содержимому используется модуль хранения документации. Для связи документов, хранящихся в этом модуле с другими данными приложения, используется ассоциативная таблица в PostgreSQL.

В конце стоит сказать, что для ускорения процесса разработки используется GitLab CI/CD, это инструмент для автоматизации непрерывной интеграция и непрерывного развёртывания нового функционала в программном обеспечении. Сборка проекта осуществляется с помощью Docker Compose.

Заключение

Автоматизированная информационно-аналитическая система позволила проследить цепочки зависимостей продукции, определить наиболее рискованные объекты и сформировать программу технического переоснащения в холдинге ОАО «РЖД».

Архитектура веб-приложения продемонстрировала свою эффективность и гибкость при развитии функционала, а также производительность, масштабируемость и удобство использования.

Литература:

1. Веб-приложение. – URL: https://ru.wikipedia.org/wiki/Веб-приложение (дата обращения 06.10.2024).

2. Документация по Vue JS. Введение. – URL: https://vuejs.org/guide/introduction.html (дата обращения 06.10.2024).

3. Nginx. – URL: https://nginx.org/ru/ (дата обращения 06.10.2024).

4. FastApi – URL: https://fastapi.tiangolo.com/ru/ (дата обращения 06.10.2024).

5. Celery – URL: https://docs.celeryq.dev/ (дата обращения 06.10.2024).

6. Документация на русском языке для PostgreSQL. – URL: https://postgrespro.ru/docs/postgresql (дата обращения 06.10.2024).