Д. А. Дробышевский
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **В. Н. Кулинченко**, ст. преподаватель

РАЗРАБОТКА ПРЕДСТАВИТЕЛЬСКОГО САЙТА ЗАО «РАСТ»

Создание представительского сайта ЗАО «РАСТ» требует не только реализации базовых функций, но и продуманной архитектуры, способной адаптироваться к растущим нагрузкам. Одной из ключевых проблем является баланс между скоростью отклика и функциональностью, особенно при интеграции динамического контента, такого как интерактивные формы, персонализированные рекомендации или взаимодействие со сторонними API [1].

В рамках проекта для фронтенда выбран фреймворк Vue.js с использованием Composition API для управления состоянием приложения и библиотеки Axios для выполнения HTTP-запросов к серверу. Это позволило организовать эффективное взаимодействие с данными и минимизировать лишние перерисовки интерфейса. Серверная часть реализована на Django с применением Django REST Framework, что обеспечило гибкость в обработке запросов и масштабируемость за счет модульной структуры.

Оптимизация производительности достигается через кэширование статического контента с помощью Django-Whitenoise и CDN-сервисов, таких как Cloudflare. Для динамических данных применён паттерн «ленивой загрузки» (lazy loading), при котором ресурсы загружаются только при необходимости, например, изображения в галерее или комментарии на странице. Дополнительно настроена предзагрузка критических ресурсов через <link rel="preload">, что сокращает время первого взаимодействия (FCP) на 15−20 %.

Важным аспектом стала безопасность. Внедрена защита от CSRF-атак с использованием токенов, а также настройка заголовков Content Security Policy (CSP) для предотвращения XSS. Для мониторинга ошибок и производительности подключены инструменты типа Sentry и Lighthouse, которые позволяют оперативно выявлять узкие места.

В ходе работы была применена архитектура, сочетающая паттерн MVC на бэкенде (реализованном на Django) и SPA (Single Page Application) на фронтенде с использованием Vue.js для управления глобальным состоянием. Для асинхронных операций, таких как загрузка данных или отправка форм, использованы стандартные HTTP-запросы через Axios, что обеспечивает обновление интерфейса без перезагрузки страницы.

Интеграция с базой данных MySQL выполнена через ORM Django, что упрощает миграции и валидацию данных. Для повышения отказоустойчивости настроена репликация базы данных между основным и резервным сервером. Авторизация реализована на основе сессий Django с интеграцей JWT-токенов, что повышает безопасность сессий.

Особое внимание уделено процессу непрерывной интеграции и доставки (CI/CD). Сборка фронтенда автоматизирована через Webpack с настройкой tree-shaking для удаления неиспользуемого кода. Процесс развертывания автоматизирован с использованием Docker, что упростило настройку окружения и деплой приложения. Тестирование включает модульные и интеграционные тесты, выполняемые с помощью pytest и встроенных средств Vue, что гарантирует стабильность работы всех компонентов системы.

Литература

1. Руководство по архитектуре веб-приложений от Microsoft [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/azure/architecture/guide/. – Дата доступа: 12.01.2025.