Мурашов А.А.,
Лозинская А.М.
ФГБОУ ВО "Уральский государственный педагогический университет"
г. Екатеринбург, Россия

WEB-СЕРВИС УЧЕТА ТЕХНОЛОГИЧЕСКИХ ПОКАЗАТЕЛЕЙ ПРОИЗВОДСТВА

Аннотация

В статье рассматриваются вопросы разработки web-приложения промышленного предприятия, предназначенного для накопления и обработки технологических показателей различных групп производственных процессов. Особое внимание уделяется методам и средствам программирования, с помощью которых решаются многоплановые задачи исследования.

Ключевые слова: ReactJS, JavaScript, web-программирование, REST API, C#, СУБД, Sql Server, Oracle.

Murashov A.A.,
Lozinskaya A.M.
Ural State Pedagogical University
Yekaterinburg, Russia

WEB-SERVICE FOR ACCOUNTING TECHNOLOGICAL PRODUCTION INDICATORS

Annotation

The article deals with the development of a web-application of an industrial enterprise, intended for the accumulation and processing of technological indicators of various groups of production processes. Particular attention is paid to the methods and tools of programming, with the help of which multifaceted research tasks are solved.

Keywords: ReactJS, JavaScript, web programming, REST API, C #, DBMS, Sql Server, Oracle.

Развитие компьютерных наук и технологий стимулировало процессы цифровизации всех социльно-экономических сфер деятельности человека,

разработку программного обеспечения и сетевых ресурсов, поддерживающих процессы управления экономикой и производством.

Создание информационных баз позволяет оптимизировать накопление, обработку и использование данных, связанных с производственными процессами и показателями.

Цель нашего исследования заключалась в разработке web-приложения базы данных технологических показателей предприятия, работающего на операционных системах Windows и Linux, и обеспечивающего высокое быстродействие и масштабируемость. Для достижения цели были сформулированы следующие основные задачи:

- определение архитектуры web-приложения,
- выбор технологии и средств разработки,
- разработка Frontend-части (клиентской части сайта),
- разработка Backend-части (серверной части сайта),
- развертывание и настройка IIS-сервера,
- апробация web-приложения, анализ и коррекция работы модулей.

Предприятие тяжелой промышленности имеет несколько цехов и участков, работает с большими объемами данных, связанных с технологическими и экономическими параметрами производства. Анализ системного программного обеспечения показал необходимость разработки web-ресурса, работающего под операционными системами Windows (XP, 7, 8, 10) и Linux.

Обзор современных решений архитектур web-приложений позволил выделить несколько возможных вариантов разработки [5]:

1. Клиент-Сервер – совокупность взаимодействующих двух компонент (клиентов и серверов): клиенты обращаются к серверам с запросами, сервера их обрабатывают и возвращают результат. В качестве сервера может выступать система управления базами данных (СУБД), обеспечивающая выполнение запросов клиента, который в свою очередь реализует интерфейс пользователя.

2. Сервисная модель – услугу реализуют несколько серверов, представляемых клиенту как единое целое. Этот вариант отлично подходит для сервисов, приостановление обслуживания которых недопустимо.

3. Тонкий клиент – клиент выполняет очень ограниченную по функционалу задачу (часто – только прием ввода с клавиатуры и других устройств и обработка команд рисования). Клиентская программа передает весь ввод пользователя (нажатия клавиш, движение мыши и т.д.) по сети серверу; сервер разбирает и обрабатывает этот ввод и передает клиенту готовые экраны, которые тот просто отображает.

4. Трёхуровневая архитектура – архитектура клиент-сервер, в которой между клиентом и сервером появляется сервер бизнес-логики, отвечающий за выполнение запросов к СУБД и дальнейшую передачу ответа клиенту.

Для того, чтобы реализовать высокую производительность приложения и его дальнейшую масштабируемость была выбрана трёхуровневая архитектура **клиент→сервер бизнес-логики→сервер баз данных**.

Для написания клиентской части приложения могут быть использованы следующие *frameworks (фреймворки – библиотеки готовых программных продуктов)*:

VueJS – реализация интерактивных сценариев JavaScript на отдельных страницах, в качестве фундамента для полноценных промышленных приложений *[2]*;

Angular (компания Google) – создание одностраничных клиентских приложений, SPA-решений (Single Page Application) [2];

ReactJS (компании Facebook) – создание пользовательского интерфейса с помощью отдельных компонентов приложения, которые в дальнейшем можно использовать в других проектах [1].

Для написания клиентской части был выбран язык JavaScript с использованием фреймворка ReactJS, поскольку он (1) позволяет создавать и повторно использовать компоненты (UI блоки), благодаря чему можно более гибко создавать приложения и более точечно управлять его частями; (2) увеличивает производительность отрисовки приложений с помощью Virtual DOM (виртуальное DOM-дерево страницы); (3) его легко интегрировать с другими фреймворками (Angular, Vue); (4) для вёрстки разметки используется синтаксис JSX (JavaScript XML), позволяющий писать HTML-код внутри JS-файла, что значительно упрощает работу. Для разработки интерфейса была взята готовая библиотека компонентов **React Ant Design 2.8.0**, для сборки использовался Webpack, который собирает все модули, написанные на JavaScript.

Для написания Backend-части (сервера приложений) могут использоваться:

NodeJS – среда выполнения кода на JavaScript, построенная на основе движка JavaScript Chrome V8, с помощью которого вызовы на языке JavaScript транслируются в машинный код [2].

ASP.NET Core MVC – фреймворк для создания сайтов и web-приложений с помощью реализации паттерна MVC (Model – View – Controller).

ASP.NET Core WebAPI – web-служба, которая может взаимодействовать с различными приложениями (ASP.NET, мобильными, десктопными) [4].

В связи с тем, что проектируемая СУБД – Microsoft SQL Server, а для разработки Frontend-части выбран ReactJS, сервер приложения было решено реализовывать с использованием web-службы **ASP.NET Core WebAPI**. Данная технология поддерживает кроссплатформенность и интегрированные средства для работы с базами данных Microsoft SQL Server и Oracle. Благодаря кроссплатформенности, приложение может быть развернуто на операционных системах семейства Windows и Linux. Данные технологических показателей работы некоторых цехов ведутся в разных базах данных (Microsoft SQL Server, Oracle). С помощью ASP.NET Core и технологии Entity Framework можно писать различные запросы к базам данных, соединять данные по ключевым полям и отправлять клиенту без написания запросов и хранимых процедур на языке Transact-SQL. Entity Framework транслирует синтаксис на языке C# в язык запросов Transact-SQL, благодаря чему и происходит обращение к СУБД [4].

На сервере СУБД используются такие СУБД как Oracle и Microsoft SQL Server. Для выполнения запросов от клиента к службе WebAPI могут быть использованы следующие архитектуры:

1. SOAP API – использует формат обмена данными SOAP XML для запросов и ответов.

2. REST API – использует протокол HTTP в качестве транспортного протокола, предоставляющего возможности вести запросы и заголовки запросов, ответы и заголовки ответов и др.

Поскольку в REST-API в основном используется формат JSON, который легче анализировать и обрабатывать, было решено использовать именно эту архитектуру. Данная архитектура предполагает применение следующих методов или типов запросов HTTP для взаимодействия с сервером:

- GET – получение данных с сервера БД;
- POST – отправка данных на сервер (используется для создания записей в БД);
- PUT – обновление данных в БД;
- DELETE – удаление данных из БД.

Концепция выбранной архитектуры web-приложения представлена на рисунке 1.

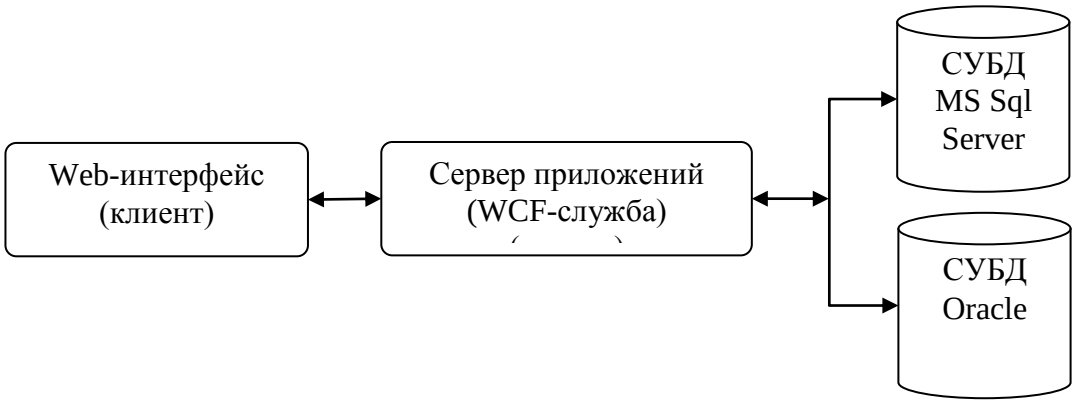

Рис. 1. Трехуровневая архитектура web-приложения

Анализ информационной структуры производства позволил определить компонентный состав модели web-приложения и требования к функциональным возможностям модулей. Также были определены основные требования к дизайну интерфейса приложения (язык, адаптивность, положение модулей, цветовая гамма, фирменный стиль).

Разработанное приложение имеет следующую структуру:

1. Диспетчер цеха № 1.
   - Временная часть (график и текущие отчеты).
   - Материальная часть (учет единиц материальной части).
   - Производственная часть (показатели соблюдения технологий).
2. Диспетчер цеха № 2.
   - Производственная часть (показатели работы участков и оборудования).
   - Складская часть (поиск и оборот единиц материальной части).
   - Временная часть (график и текущие отчеты по производству).
3. Диспетчер цеха № 3.
   - Производственная часть (учет заявок).
4. Диспетчер цеха № 4.
   - Временная часть (график и текущие отчеты по производству).
5. Диспетчер цеха № 5.
   - Временная часть (график и текущие отчеты по производству).
   - Справочники

С помощью web-приложения можно решить следующие задачи по вводу, обработке и выводу данных:

1. Авторизация пользователя на портале.
2. Ввод информации о показателях производства.
3. Вывод информации о показателях производства.

4. Изменение данных о показателях производства.
5. Удаление данных.

Тестирование web-приложения экспертами производственных подразделений позволило установить достаточно хороший уровень разработанности и удобства использования (*usability*), высокое быстродействие, потенциал масштабирования в процессе эксплуатации. Замечания и рекомендации экспертов, а также результаты апробации продукта систематизированы для анализа, доработки и развития web-приложения.

Реализованная архитектура позволила вынести бизнес-логику приложения в отдельную WCF-службу, функционирующую на сервере приложения. Пользователи, работающие с приложением, получили доступ к модулям соответствующих подразделений и возможность вводить информацию в редактируемые таблицы, выводить на печать отчеты по технологическим показателям, выгружать информацию в таблицы Microsoft Excel. Результаты проведенного исследования имеют высокую практическую значимость и отличаются новизной выработанных теоретико-методологических подходов.

ЛИТЕРАТУРА

1. Вирух, Р. Путь к изучению React. 2018. 231с.
2. Закас, Н. JavaScript для профессиональных веб-разработчиков. СПб: Питер, 2015. 960 с.
3. Бен-Ган, И. Microsoft SQL Server 2012. Основы T-SQL. М.: Эксмо, 2015. 400 с.
4. Вагнер, Б. C#: Эффективное программирование. М.: Альфа-Книга, 2017. 224 с.
5. Типы архитектур программного обеспечения [Электронный ресурс]. URL: https://habr.com/ru/company/1cloud/blog/424911/ (дата обращения 26.05.2020)