МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА RESTFUL API ДЛЯ МОНИТОРИНГА И ОТЧЕТНОСТИ В НЕФТЕГАЗОВОЙ ОТРАСЛИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Копьева Максима Игоревича

Научный руководитель
доцент, к. ф.-м. н. _____ Г. Г. Наркайтис

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2019

ВВЕДЕНИЕ

Нефтегазовая промышленность является самым главным элементом российской экономики. На территории Российской Федерации сосредоточена примерно третья часть мировых запасов природного газа. По всей стране открыто множество месторождений, процесс добычи, обработки полезных ископаемых идет непрерывно. Для обеспечения транспортировки в государстве была создана система газоснабжения, объединяющая компрессорные станции, сеть газопроводов, хранилища ресурса и месторождения. На территории России в данный момент действует около 12 крупных нефтегазовых компаний. Эти компании нуждаются в адекватном программном обеспечении, которое не будет зависеть от зарубежного. Сейчас большая часть российских компаний пользуются импортными сервисами. Эта ситуация порождает зависимость российской промышленности от зарубежных компаний. Необходимость импортозамещения программного обеспечения в Российской Федерации подтверждается важностью нефтегазовой отрасли для поддержания стабильного финансового состояния России, а также отсутствием российских разработок в этой сфере. Природные энергоносители такие как уголь, нефть и газ, являются одними из самых важных полезных ископаемых, встречающихся на территории Российской Федерации. Благодаря им, Россия занимает лидирующие позиции в мировом рынке источников топлива.

Актуальность выбранной темы дипломной работы не подвергается опровержению из-за недостатка и несовершенства существующего программного обеспечения в сфере нефтегазовой промышленности. Аналогов проекта, который рассматривается в данной дипломной работе, в России еще нет. В этом проекте используются разнообразные технические решения, которые направлены на максимально эффективную реализацию поставленных перед разработчиками задач.

Программное обеспечение, используемое промышленными компаниями и холдингами, которые специализируются на добыче полезных ископаемых, должно соответствовать большому количеству требований, например, таких как: надежность, стабильность, безопасность. Одним из примеров такого программного обеспечения является база данных WITSML, предоставляющая универсальный протокол для обмена и хранения данных о геологии и полезных ископаемых. Многие компании используют продукты, разработанные

штатными программистами, постоянно дорабатывая их до необходимого состояния, некоторые - заказывают чужие разработки.

Эффективное и подходящее программное обеспечение может значительно облегчить работу нефтегазовой промышленности благодаря системам мониторинга, отчетности и представления графиков. На настоящий момент используются системы, записывающие показатели при бурении скважин в реальном времени и сортирующие эти данные в цифровых базах.

Предмет исследования —- внутреннее устройство и реализация RESTFul API для мониторинга и отчетности в нефтегазовой отрасли. Описанный в этой дипломной работе проект будет использоваться на практике в промышленных компаниях. Он позволяет интегрировать различные сервисы, предоставляющие данные, в удобную для клиентского приложения форму. Включает в себя алгоритмы кэширования, сортировки и обработки информации, может генерировать ссылки на сторонние ресурсы. У заказчика данного программного обеспечения уже есть сервера и базы данных, содержащие необходимые данные. Также есть собственный C# клиент для их использования. Сервис, серверная часть которого рассматривается в данной дипломной работе, должен реализовывать функционал этого клиента. Этот сервис относится к программе импортозамещения.

Цель бакалаврской работы — рассмотрение основных проблем, на которые может натолкнуться разработчик программного обеспечения, необходимого для нефтегазовой отрасли. Анализ различных путей решения, выбор наилучшего, пример реализации с помощью конкретной технологии (C# ASP.NET 4.5).

Поставленная цель определила **следующие задачи** для реализации:

1. Суперрепозиторий — система кэширования данных скважин. Запрос данных показателей ствола скважины является самым частым запросом к разрабатываемому серверу. Кэширование данных стволов (значений мнемоник) позволяет избежать запроса к WITSML серверу, что повысит общую скорость и производительность сервера.
2. Кэширование метаданных скважин чрезвычайно необходимая функция, так как за счет нее можно резко повысить производительность сервера и клиента.
3. Экспорт данных стволов в файл формата xlsx.

4. Обработка данных от скважин, поступающих в реальном времени. Открыв вкладку с данными скважин по времени или по глубине, пользователь видит набор графиков, каждому из которых соответствует некоторая мнемоника. Если скважина в данный момент активна, и с нее поступают данные, разрабатываемый сервер должен посылать их на клиента с определенной периодичностью.

Методологические основы разработки RESTFul API для мониторинга и отчетности в нефтегазовой отрасли представлены в работах С. Макконнелла [1], Мартина Фаулера [2], Сергея Теплякова [3], К. Бэка [4], А. Дэвиса [5].

Практическая значимость бакалаврской работы. В ходе этой работы был разработан сервис для мониторинга и отчетности в нефтегазовой отрасли, который был внедрен и уже используется на практике.

Структура и объем работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и одного приложения. Общий объем работы — 52 страницы, из них 38 страниц — основное содержание, включая 2 рисунка, цифровой носитель в качестве приложения, список использованных источников информации — 30 наименований.

1 Подготовка теоретической основы

C# — объектно-ориентированный язык программирования, разработанный группой инженеров под руководством Андерса Хейлсберга и Скотта Вильтаумота. Он создавался с целью стать главным прикладным языком разработки под платформу Microsoft .NET Framework. C# обладает C-подобным синтаксисом, который больше всего похож на Java.

Разработчики C# подошли к новому языку максимально ответственно и постарались применить опыт использования других языков программирования. Они решили избежать множественного наследования классов, возможности низкоуровневых операций с памятью. Их использование в других языках, например таких как C++, приводило к фатальному количеству ошибок, на поиск которого затрачивался непомерный объем времени. Это заставляло компании нанимать колоссальное количество профессиональных программистов. За счет более удобного языка, который обладал бы сборщиком мусора (автоматической системой очистки памяти) и не позволял бы программисту совершить фатальную ошибку, можно было бы резко повысить эффективность и скорость разработки программного обеспечения. Именно к таким языкам программирования относится C#.

Платформа Microsoft .NET Framework была создана фирмой Microsoft для того, чтобы составить конкуренцию набиравшей в то время популярность платформе Java фирмы Sun Microsystems. Основное преимущество этой платформы перед другими является то, что программы, собранные из некого промежуточного языка, переводятся в промежуточный код, который теоретически может выполняться на любых платформах, в том числе, отличающихся от обычных настольных компьютеров.

Далее, в работе приведено описание программного обеспечения, которое уже используется заказчиком, и основных терминов.

Данные датчиков, установленных на оборудовании конкретного ствола, передаются на общий WITSML сервер. В свою очередь видеопотоки с камер, на тех стволах, где они есть, передаются в специальный сервер для видеопотоков. Таким образом, на предприятии у заказчика действуют одновременно несколько сервисов, содержащих используемые данные. Каждый из них имеет отдельную регистрацию, однако имена пользователей и пароли везде одинаковые.

Кроме этих серверов, у заказчика имеется разработанный внутри компании толстый клиент. Он представляет собой Windows Forms приложение, написанное с использованием C#. Он имеет функции просмотра и редактирования данных, которые содержатся на служебных сервисах. Это включает в себя просмотр данных и метаданных скважин, просмотр статусов, подключение к видеокамерам, получение документации, составление отчетов, экспорт данных скважин в различные форматы, отрисовка графиков. Это приложение активно используется на практике.

Проект MLServer представляет из себя веб-сервис, который предоставляет возможности мониторинга данных, хранящихся на служебных сервисах. В идеале, по возможности он должен полностью соответствовать толстому клиенту. Проект состоит из клиентской и серверной частей. Клиентская часть представляет собой веб-приложение, написанное с использованием фреймворка React. Клиент напрямую не взаимодействует с WITSML сервером, полностью перекладывая эту обязанность на разрабатываемый сервер.

На стороне заказчика работает WITSML база данных, которая предоставляет набор определенных SOAP запросов. Каждый запрос и ответ на него представляет из себя XML файл, в котором описаны требуемые для запроса аргументы или возвращенная сервером информация.

2 Анализ поставленных задач

Система кэширования и распределения запросов для основных данных стволов, проходящих через сервис, необходима для того, чтобы обеспечить достойную скорость ответа на запросы к разрабатываемому серверу.

Кэширование происходит в несколько этапов. Когда тонкий клиент отправляет на сервер запрос о получении данных ствола, он перенаправляется внутрь суперрепозитория. Диапазон этого запроса разделяется на пакеты, для каждого из которого внутри суперрепозитория хранится информация о том, какие данные о нем загружены. Для тех пакетов, которые не содержат никакой информации, то есть они не были запрошены раньше или были очищены, делается запрос к WITSML. Когда все пакеты готовы, данные с них собираются в специальную форму и возвращаются клиенту.

Кроме предоставления обычных данных стволов, сервер позволяет клиенту загружать метаданные скважин, в которых эти стволы находятся. Этот запрос выполняется клиентом перед самым началом работы, поэтому скорость ответа сервера является критичным параметром. Если сервер отвечает слишком медленно, пользователь вынужден ожидать подгрузки дерева и при этом он не может начать пользоваться всем сервисом. Уже после того как дерево загрузилось у клиента, пользователь может выбрать ствол и тип данных, которые он хочет посмотреть и начать работу.

У каждого пользователя в системе WITSML есть набор скважин, которые он имеет право просматривать. В разрабатываемом сервисе backend часть должна возвращать специально подготовленное дерево скважин и стволов, доступных текущему пользователю. Однако, в случае чрезмерно огромного количества скважин для пользователя проявилась проблема в виде слишком медленного ответа сервера (от 10 до 40 секунд, в зависимости от машины), что является неприемлемым.

Это происходило вследствие того, что когда пользователь запрашивал метаданные скважин, сервер каждый раз перезагружал список всех скважин для этого пользователя и метаданные для каждой скважины. Для пользователей с весьма внушительным количеством доступных скважин (более 1000), приходилось делать слишком большой объем запросов, что в свою очередь заставляло тонкий клиент ожидать пока сервер завершит сбор и обработку всех данных. Например, допустим с сервисом начал работать пользователь с 600

скважинами. Тогда, чтобы сгенерировать и вернуть клиенту дерево скважин, серверу придется сделать 601 запрос, один из которых будет для получения списка скважин, а остальные — для загрузки метаданных каждой отдельной скважины.

Одним из вариантов решений было запрашивать метаданные скважины один раз, и обновлять их полностью по истечении достаточно большого промежутка времени, например, часа. Однако у данного способа есть большие недостатки. Пользователи разрабатываемого сервиса достаточно часто меняют состояние какой-нибудь отдельной скважины или ствола. В случае, если метаданные скважин будут обновляться, например, раз в час, изменения состояний в WITSML сервере далеко не сразу будут отражаться на разрабатываемом сервере. Такое решение было реализовано командой разработчиков. Однако, после тестирования со стороны заказчика, оказалось, что такое решение неприемлемо, так как уменьшает удобство использования сервиса в практической работе. Необходимо, чтобы изменения появлялись на сервере хотя бы в течении десяти минут.

Таким образом, к дереву скважин предъявляются следующие требования:

— Время ответа сервера, адекватное количеству загружаемых скважин.
— В случае использование кэширования дерева скважин, актуальность возвращенной информации.

Самым логичным данной ситуации было бы использовать триггер со стороны WITSML, такой, чтобы при изменении состояний ствола или скважины, а также добавления или удаления их, соответствующее сообщение приходило на разрабатываемый сервер. Сервер при получении этого сообщения, изменял бы данные о состояний скважин внутри себя. Такое решение позволило бы сэкономить большое количество вычислительных ресурсов и сетевого трафика.

Однако препятствием для такого решения является то, что стандарт WITSML не поддерживает использование триггеров, что было выяснено в ходе общения с заказчиком. К сожалению, какой-либо способ обойти стандарт или улучшить его не представляется возможным. Поэтому необходимо найти отличное от этого решение.

Для сохранения быстродействия сервера и при этом актуальности дан-

ных лучший вариант обновлять скважины по порядку в фоновом режиме. При ответе на запрос клиента, сервер будет брать метаданные из готового набора скважин. Этот набор будет постепенно обновляться в фоновом режиме. С помощью такого решения можно добиться высокой скорости ответа и возвращать клиенту актуальные данные.

Во время начала работы над проектом в нем уже имелась система для работы с данными, приходящими в реальном времени. Однако эта система обладала внушительным количеством недочетов и недостатков.

Перед разработчиками стояла задача исправить следующие проблемы:

— Неправильная архитектура SignalR хаба приводила к тому, что переподключение к другим скважинам и стволам позволяло создавать бесконечное количество потоков на сервере, что приводило к его отключению.
— Неэффективная реализация запросов к WITSML серверу. Необходимо создать такую архитектуру, которая будет запрашивать данные в реальном времени с максимальной эффективностью, другими словами не создавать дублирующие запросы к WITSML серверу, если два и более пользователя подписались на одни и те же данные.

Когда пользователь заходит в новую вкладку, переключается между вкладками, открывает новую скважину, изменяет параметры рабочего окна, клиент отписывается от текущего ствола, и подключается к новому (или к тому же, но с новыми параметрами). Таким образом, тонкий клиент может быть подписан только на один источник данных в какой-то момент времени. Если сервер отправляет тонкому клиенту два потока данных одновременно, это может привести к сбоям в его работе.

Одним из способов представления данных стволов на разрабатываемом сервисе является Excel файл. Для этого были спроектировано специальное API позволяющее тонкому клиенту загружать данные стволов в виде файла с таблицей.

Предполагается, что экспорт будет производится для огромного объема данных. Загрузка и запись в таком случае будет затрачивать большое количество вычислительных ресурсов и происходить чрезмерно медленно. Поэтому требуется реализовать систему процессов экспорта, которые будут сообщать тонкому клиенту о своем состоянии. Пока сервер загружает данные и записывает их в файл, тонкий клиент будет получать информацию о прогрессе

процесса экспорта.

Так как платформой, под которую разрабатывается сервер, является .NET framework, было решено сначала попытаться использовать библиотеку Interop Excel. Эта библиотека предоставляет интерфейс для взаимодействия с Microsoft Office Excel. Однако такой способ обладает несколькими недостатками, которые делают его полностью не подходящим для практического использования. Во-первых, эта библиотека предоставляет чрезвычайно неудобный интерфейс для использования в нашем приложении. Для того, чтобы записать простейший Excel файл требуется огромное количество кода. Во-вторых, ее использование требует установки Microsoft Office Excel и специальной библиотеки для работы с Interop. Заказчик не может позволить себе устанавливать на каждую машину эти зависимости. В-третьих, Interop сервис при создании каждого файла запускает новое окно установленного Office Excel, в котором производит действия с файлом, а после закрывает его. Запуск и закрытие окна отнимают слишком много процессорного времени, что является неприемлемым.

Существует множество форматов, которые используется в приложении Excel. В данной работе нас интересует только формат xlsx, один из новых форматов таблиц. Он представляет собой определенную структуру XML файлов в ZIP архиве. Это облегчает нам задачу, так как для работы с этим форматом можно использовать сторонние библиотеки, что позволит напрямую записывать данные стволов в файл формата xlsx.

Для того, чтобы получить файл с экспортированными данными, клиенту необходимо воспроизвести определенную последовательность действий, в конце которой тонкий клиент получит файл.

1. Пользователь выбирает скважину и ствол, данные которого он собирается получить в виде Excel файла.
2. Пользователь в специальном графической оболочке выбирает параметры для выбора данных, которые ему нужны.
3. Тонкий клиент отправляет на сервер запрос о начале процесса экспорта и получает уникальный идентификатор для доступа к статусу и файлу.
4. Тонкий клиент периодически запрашивает с сервера статус процесса экспорта, который был ранее запущен. С помощью этого статуса клиент может выводить на экран информацию о том, через какое время будет получен результат.

5. После того, как клиент в конце ряда запросов статуса получит код завершения процесса и имя файла, генерируется ссылка для загрузки файла.
6. В случае, если экспорт прошел успешно, сервер возвращает файл клиенту по готовой ссылке.

Таким образом серверу необходимо предоставить API, состоящий из следующих запросов:

— Создание процесса экспорта, возвращение его идентификатора.
— Запрос, позволяющий клиенту проверить состояние процесса экспорта.
— Возвращение файла при успешном завершении.

ЗАКЛЮЧЕНИЕ

В ходе работы по разработке RESTFul API для мониторинга и отчетности в нефтегазовой отрасли мы решили большое количество разнообразных проблем, лишь часть из них были описаны в этой работе.

При решении задачи кэширования геологических данных стволов в работе показана эффективность метода разбиения данных на отдельные пакеты и сохранение состояния каждого отдельного пакета. Как показало наше исследование, этот метод обладает большой эффективностью и надежностью, позволяет значительно увеличить скорость ответа разрабатываемого сервера.

Одной из задач нашего исследования было реализовать экспорт данных стволов в Excel формат. На основании вышеприведенной аргументации можно сделать вывод о том, что, несмотря на использование .NET платформы, организовать работу с Excel файлам через Excel Interop не является выгодным решением. Намного лучше с точки зрения производительности и надежности использовать прямую запись в Excel файл. Мы показали, что при экспорте данных в файл можно добиться большой гибкости в работе. В ходе работы мы успешно решили поставленную задачу.

При решении задачи кэширования метаданных скважин в работе изучен способ, позволяющий с высокой эффективностью достигнуть поставленных заказчиком целей. Этот способ заключается в том, чтобы хранить на сервере копию метаданных скважин, постоянно обновляя эти данные в фоновом режиме. Это позволяет обеспечить актуальность хранимых на сервере метаданных, поскольку метаданные для 1600 скважин будут совершать полный цикл обновления в течении 5 минут. При этом обеспечение такой высокой взаимосвязи с WITSML сервисом не будет негативно отражаться на производительности сервера. При этом, загрузка метаданных скважин со стороны клиента с максимальным количеством доступных скважин, будет занимать не минуту, как в варианте без кэширования, а секунду. Мы приходим к выводу о том, что учитывая ограничения, которые касаются WITSML протокола, данный способ решения задачи является наилучшим из возможных.

Мы решили задачу создать систему, которая надежно и корректно предоставляет данные стволов в реальном времени за счет использования кодовой базы суперрепозитория. Используя методы многопоточного программирования и внутреннее устройство набора скважин можно добиться высокой ста-

бильности отправки данных. Протокол общения между тонким клиентом сервером, который мы реализовали в данной работе, позволяет избежать нарушений в работе даже при неправильных данных поступающих от WITSML.

Таким образом, мы решили все поставленные задачи в полном объеме. Мы проанализировали различные пути решения, выбрали наилучший, привели пример реализации с помощью C# и ASP.NET 4.5. В итоге, нами был разработан стабильный и надежный продукт, удовлетворяющий всем поставленным требованиям.

Практическая значимость разработки RESTFul API для мониторинга и отчетности в нефтегазовой отрасли состоит в важности программного обеспечения используемого компаниями, которые специализируется на добычи полезных ископаемых на территории Российской Федерации. Собственное программное обеспечение в этой сфере позволит нашей стране не зависеть от импортного.

Разработанный сервис может быть использован на практике как надежный, удобный и функциональный инструмент в работе с геологическими данными. Аналогов среди российских разработчиков на данный момент нет.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Макконнелл, С.* Совершенный код. Мастер-класс / С. Макконнелл. — БХВ-Петербург, 2017.

2 *Фаулер, М.* Рефакторинг. Улучшение проекта существующего кода / М. Фаулер. — Вильямс, 2017.

3 *Тепляков, С.* Паттерны проектирования на платформе .NET / С. Тепляков. — Питер, 2018.

4 *Бек, К.* Шаблоны реализации корпоративных приложений / К. Бек. — Вильямс, 2017.

5 *Дэвис, А.* Асинхронное программирование в C# 5.0 / А. Дэвис. — ДМК Пресс, 2018.