МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

МОДЕЛЬ И ПРОТОТИП ПРОМЫШЛЕННОЙ СИСТЕМЫ ОПЕРАТИВНОЙ ОБРАБОТКИ ГЕОЛОГИЧЕСКИХ ДАННЫХ ПО СТАНДАРТУ WITSML

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 01.04.02 Прикладная математика и информатика
факультета компьютерных наук и информационных технологий
Шехмаметьева Рустама Рамильевича

Научный руководитель:
д.ф-м.н. _____ Андрейченко Д.К.
подпись, дата

Зав. кафедрой:
д.ф.-м.н. _____ Андрейченко Д.К.
подпись, дата

Саратов 2019

ВВЕДЕНИЕ

Актуальность темы. В геологической промышленности международным стандартом для передачи данных является WITSML. На данный момент для него практически нет на рынке каких-либо средств для визуализации и оперативной обработки данных в этом формате. Вместе с тем, для оперативного управления процессами добычи и транспортировки нефтегазовых продуктов требуется распределённая система визуализации и оперативной обработки данных, способная работать на достаточно широком классе устройств: от мобильных устройств до достаточно мощных рабочих станций. Разработка ПО с данными возможностями может быть выполнено на основе платформ .NET и JavaScript. Однако, обе платформы не предусматривают стандартных средств работы с данными в формате WITSML.

Цель бакалаврской работы – моделирование системы оперативной обработки и визуализации геологических данных, состоящей из хранилища, куда поступают эти данные, оперативного обработчика и службы визуализации.

Поставленная цель определила **следующие задачи**:

1. Рассмотреть стандарт WITSML
2. Разработать службу обработки данных, поступающих в этом формате
3. Разработать прототип хранилища WITSML данных
4. Разработать службу визуализации данных

Практическая значимость бакалаврской работы.

Структура и объём работы. Магистерская работа состоит из введения, 2 разделов, заключения, списка использованных источников и 4 приложений. Общий объем работы – 77 страниц, из них 52 страниц – основное содержание, включая 5 рисунков, список использованных источников информации – 36 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Работа разделена на два раздела.

Первый раздел «Теоретическая часть» посвящён теоретическим основам работы. В нём рассказывается о компонентах и инструментарии для их разработки**.** Он разделён на три подраздела.

Первый подраздел посвящён разработке прототипа хранилища данных. В нём рассказывается о стандарте WITSML, описываются требования к прототипу хранилища. Описаны основы теории разработки SOAP служб и архитектуры, основанной на них, их построения и использования. Рассказывается о двухуровневой архитектуре «клиент-сервер» и базе данных MS SQL, описываются достоинства и недостатки данной архитектуры.

Основные определения первого подраздела:

WITSML (Wellsite Information Transfer Standard Markup Language) – стандарт на основе XML для передачи геофизических и технических данных между предприятиями, занимающимися нефтяной промышленностью. Стандарт возник вследствие требований индустрии, нуждающейся в стандартизированном единообразном способе обмена геофизическиой и технической информацией между различными нефтяными, энергетическими, газовыми компаниями, буревыми подрядчиками, регулирующими службами и другими.

WCF – это фреймворк для построения сервисоориентированных приложений на основе SOAP протокола. С помощью WCF, можно отправлять данные как асинхронные сообщения из одной конечной точки службы в другую. Конечная точка службы может входить в постоянно доступную службу, размещаемую в IIS, или представлять службу, размещаемую в приложении. Конечная точка может быть клиентом службы, которая запрашивает данные от конечной точки службы. Сообщения могут представлять одиночный символ или одно слово, отправляемое в формате XML, или иметь вид сложного потока двоичных данных. Следствием

применение стандартов WS является то, что WCF позволяет создавать ориентированные на службы приложения. Сервисноориентированная архитектура (SOA) подразумевает применение веб-служб для отправки и получения данных. Общим преимуществом служб является слабая связанность вместо жесткой запрограммированности для различных приложений. Слабая связь означает, что любой клиент, созданный на любой платформе, может подключаться к любой службе при условии, что выполняются необходимые контракты. WCF реализует современные отраслевые стандарты для веб-службы взаимодействие.

Обмен сообщениями выполняется по одному из нескольких шаблонов. Чаще всего используется шаблон «запрос-ответ», когда одна конечная точка запрашивает данные от другой конечной точки. Вторая конечная точка отвечает. Существуют и другие шаблоны, например одностороннее сообщение, когда одна конечная точка отправляет сообщение, не ожидая ответа. Более сложным является шаблон дуплексного обмена, когда две конечные точки устанавливают соединение и отправляют данные в обоих направлениях подобно программе обмена мгновенными сообщениями

Двухуровневая модель фактически является результатом распределения пяти указанных функций между двумя процессами, которые выполняются на двух платформах: на клиенте и на сервере. В чистом виде почти никакая модель не существует, однако рассмотрим наиболее характерные особенности каждой двухуровневой модели. В архитектуре «выделенный сервер базы данных» средства управления базой данных и база данных размещены на машине-сервере (DB-сервер). В такой модели база данных хранится на сервере. На сервере же находится ядро СУБД. На клиенте располагается презентационная логика и бизнес-логика приложения. Клиент обращается к серверу с запросами на языке SQL.

MS SQL Server. Это одна из самых распространенных СУБД, используемых в крупных организациях. Она использует реляционную модель

данных. MS SQL имеет следующие возможности: Поддержка структурированных и неструктурированных (XML) данных. Replication Services : репликация данных для распределённых и мобильных приложений обработки данных, высокая доступность систем, масштабируемый параллелизм со вторичными хранилищами данных для отчётных решений предприятия и интеграция с разнородными системами, включая существующие базы данных Oracle. Integration Services: возможности извлечения, преобразования и загрузки для хранилищ данных и интеграции данных в масштабе предприятия. Analysis Services : аналитическая обработка в реальном времени (OLAP) для быстрого, сложного анализа больших и смешанных наборов данных, использующая многомерное хранение. Reporting Services : исчерпывающее решение для создания, управления и доставки как традиционных бумажных отчётов, так и интерактивных, основанных на технологии WWW отчётов. Хранилище данных представляет собой базу данных, которая представляет собой набор таблиц из типизированных столбцов. SQL Server поддерживает различные типы данных, включая основные, такие как Integer, Float, Decimal, Char, Varchar, двоичный, Text и другие.

Второй подраздел посвящён разработке службы оперативной обработки данных. В нём рассказывается о разработке REST служб с использованием ASP.NET WebAPI, использование ASP.NET SignalR для обеспечения доступа к данным через протокол websocket. Описываются требования к службе оперативной обработке данных. Описываются теоретические основы аутентификации и авторизации с использованием OAuth и OWIN.

Основные определения второго подраздела:

ASP.NET Web API – фреймворк, основанный на ASP.NET, для построения REST-сервисов на основе .NET Framework. REST-архитектура предполагает применение следующих основных методов или типов запросов

HTTP для взаимодействия с сервером: GET POST PUT DELETE Этот фреймворк позволяет построить приложение, использующее паттерн MVC (Model View Controller). Каждый компонент паттерна выполняет определённую задачу: Controller (контроллер) – часть, с которой взаимодействует пользователь. Контроллер принимает входные данные от пользователя через web запрос, отправляет их в модель, получает ответ от модели, возвращает ответ пользователю. В ASP.NET Web API контроллер – это класс, наследующийся от класса ApiController. Методы этого класса возвращают IHttpActionResult: результат обработки действия пользователя. Это HTTP статус и тело ответа. View (представление) – часть, ответственная за отображение данных пользователю. В ASP.NET Web API нет View как такового, им выступает возвращаемый ответ на запрос. Model (модель) – часть ответственная за управление данными, бизнеслогикой и логикой приложения. Является своего рода «окном» для доступа к бизнес-логике.

ASP.NET SignalR – библиотека для ASP.NET, позволяющая добавить приложениям возможность отправлять данные соединённым к серверу клиентам в реальном времени, то есть в тот же момент, когда они становятся доступны. Типичным примером подобной функциональности «в реальном времени» может служить приложение-чат. Клиенты отправляют сообщения друг другу через сервер и видят эти сообщения в тот же момент, когда они были отправлены. SignalR предоставляет лёгкое в использовании API для создания RPC вызововов (Remote Procedure Call) от сервера к клиенту и обратно. Другими словами, клиент (в данном случае JavaScript, хотя существует клиент под .NET) может вызвать процедуру с сервера, а сервер может вызвать процедуру на стороне клиента.

OWIN (Open Web Interface for .NET) – интерфейс для взаимодействия между .NET сервером и веб-приложениями. Целью OWIN является отделение сервера и приложения, что позволяет разработать простые модули для .NET веб разработки. Katana представляет собой набор компонентов

OWIN, разработанных и опубликованных Microsoft. Эти компоненты включают в себя компоненты инфраструктуры (хосты и серверы), а также функциональным компоненты, такие как модули авторизации, аутентификации и привязанные к платформе SignalR и Asp.Net Web API. Разработка Katana преследовала 3 основные цели : Portable. Компоненты должны легко заменяться новыми компонентами по мере их выхода. Сюда также входит разработка компонент для различных серверов. Идея заключается в том, что сторонние платформы могут работать на платформе Microsoft и наоборот, приложения ASP.NET (OWIN) могут работать на серверах других платформ. Модульность. Компоненты должны быть небольшими узконаправленными. Тем самым позволяя разработчику выбрать какие компоненты использовать, а какие нет. Масштабируемость. Так как компоненты маленькие - они требует меньше вычислительных ресурсов при работе. Взаимозаменяемые компоненты низкого уровня означают, что они могут быть заменены сразу как только будут доступны новые сервисы с лучшей производительностью.

OAuth – открытый протокол авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей логин и пароль. В OAuth 1.0 и OAuth 2.0 используются три вида полномочий: учётные данные клиента, временные учётные данные и токены.

Учётные данные клиента используются для проверки подлинности клиента. Сервер может предоставлять клиентам специальные услуги, такие как регулирование свободного доступа или предоставление владельцу ресурса более подробной информации о клиентах, пытающихся получить доступ к защищённым ресурсам. В некоторых случаях учётные данные клиента ненадёжны и могут быть использованы только в информационных целях, например, в настольных приложениях. Токен используется вместо имени и пароля владельца ресурса. Владелец ресурса не раскрывает свои

учётные данные клиенту, а разрешает серверу выдавать клиенту токен — специальный класс учётных данных, предоставляющий клиенту некоторые ограниченные возможности. Клиент использует токен для доступа к защищённому ресурсу, не зная при этом пароля владельца ресурсов. Токен состоит из цифровой подписи, обычно (но не всегда) случайного набора букв и цифр, который является уникальным и криптографически стойким, и из ключа для защиты токена от использования посторонними лицами. Токен ограничен по полномочиям и продолжительности и может быть отозван в любой момент владельцем ресурса, при этом не затрагивает токенов, выданных другим клиентам

Третий раздел посвящён службе визуализации. В нём описываются требования к службе визуализации, рассказывается о разработке пользовательских интерфейсов с использованием фреймворка React.js и теоретические основы построения графиков с помощью библиотеки D3. Рассказывается о возможностях React и особенностях его работы.

Основные определения подраздела:

React – популярная на данный момент библиотека для построения пользовательских интерфейсов, разработанная компанией Facebook. Она привнесла в web front-end разработку идеи декларативного программирования и компонентного построения приложений. К её особенностям можно отнести то, что она является полностью открытой (распространяется под MIT лицензией, которая позволяет безвозмездное использование, свободное изменение, копирование, распространение, сублицензирование), поддерживается крупной компанией и собрала вокруг себя большое сообщество. Также вокруг React’а выросла своя экосистема – и это не только большое количество сторонних библиотек и готовых компонентов, но также и версии React для разработки кроссплатформенных мобильных приложений, приложений для виртуальной реальности (React Native, React VR соответственно).

D3.js представляет библиотеку на языке JavaScript для обработки и визуализации данных. Само название D3 расшифровывается как Data-Driven Documents и как бы делает упор на управление данными, хотя ключевой функциональностью библиотеки являются мощные возможности для их визуализации. Библиотека D3.js основана прежде всего на использовании JavaScript, SVG и CSS в противовес другим подобным библиотекам, которые вместо SVG используют элемент canvas и его возможности. Если стандартные механизмы рисования, например, элемент canvas, полагаются на пиксели, то svg использует векторы. Применение SVG позволяет создавать структуры с насыщенной графикой, обладающие анимацией и возможностями взаимодействия. По сравнению с пиксельной графикой SVG обладает рядом преимуществ. В частности, SVG основан на xml, что делает его более читабельным. Кроме того, код SVG более легковесный по сравнению файлами изображений.

Библиотека D3.js основана прежде всего на использовании JavaScript, SVG и CSS в противовес другим подобным библиотекам, которые вместо SVG используют элемент canvas и его возможности. Если стандартные механизмы рисования, например, элемент canvas, полагаются на пиксели, то svg использует векторы. Применение SVG позволяет создавать структуры с насыщенной графикой, обладающие анимацией и возможностями взаимодействия.

Второй раздел «Практическая часть» посвящен реализации прототипа хранилища данных, службы оперативной обработки данных и службы визуализации данных.

Был разработан прототип хранилища данных, который позволяет хранить геофизические данные, которые можно передавать с использованием стандарта WITSML.

Была разработана служба оперативной обработки данных, которая позволяет получать данные в формате WITSML и обрабатывать их для

употребеления клиентом. Служба позволяет получать данные как в реальном времени, так и по запросу. Были реализованы абстракции для представления WITSML данных в виде программных объектов. Были созданы XML-парсеры для извлечения информации из данных, отправленных в формате WITSML. Был разработан сервис, позволяющий получать данные в REST формате или через протокол websocket.

Была разработана служба визуализации данных. Были разработаны React компоненты, позволяющие аутентифицировать пользователя и получать данные из службы оперативной обработки для визуализации. Были разработаны компоненты для отрисовки графиков на основе полученных данных с использованием библиотеки D3.

ЗАКЛЮЧЕНИЕ

Реализован прототип промышленной системы для оперативной обработки и визуализации геофизических данных на основе стандарта WITSML.

Архитектура системы, состоящей из хранилища, куда поступают геологические данные, оперативного обработчика и службы визуализации обеспечивает удобство пользователям за счёт возможности работы как на рабочих станциях, так и на мобильных устройствах.

Хранилище данных позволяет сохранять и получать данные для последующей обработки службой оперативной обработки данных.

Служба обработки данных, поступающих в формате WITSML, обеспечивает клиентам системы безопасный доступ к геофизическим данным как по запросу, так и в реальном времени.

Служба визуализации позволяет клиентам наглядно рассмотреть геофизические данные в виде набора графиков, показывающих пользователю требуемые меры данных.

Отдельные части магистерской работы были опубликованы на конференции:

IX научная конференция молодых учёных «Presenting Academic Achievements to the World», выпуск 8, 2019. ISSN 2306-3068

Основные источники информации:

1. energistics.org [Электронный ресурс] – Energistics Energy Standards | Current Standards. URL: http://www.energistics.org/drilling-completions-interventions/witsml-standards/current-standards (Дата обращения 20.12.2018)
2. Lowy J, Montgomery M – Programming WCF Services: Design and Build Maintainable Service-Oriented Systems – O'Reilly, 4th edition (2015)
3. Assaf W, West R – SQL Server 2017 Administration Inside Out – Microsoft Press, 1st edition (2018)
4. tools.ietf.org [Электронный ресурс] – Hypertext Transfer Protocol (Http/1.1): Semantics and Content. URL: https://tools.ietf.org/html/rfc7231 (Дата обращения 26.03.2019)
5. tools.ietf.org [Электронный ресурс] – The WebSocket Protocol. URL: https://tools.ietf.org/html/rfc6455 (Дата обращения 23.12.2017)
6. owin.org [Электронный ресурс] – OWIN: Open Web Server Interface for .NET Specification. URL: http://owin.org/html/spec/owin-1.0.html (Дата обращения 22.12.2017)
7. React [Электронный ресурс] / React - A JavaScript library for building user interfaces. URL: https://reactjs.org (Дата обращения 06.07.2017)
8. D3 [Электронный ресурс] / D3.js – Data-Driven Documents. URL: https://d3js.org (Дата обращения 06.07.2017)