ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»

МЕЖДУНАРОДНЫЙ ФАКУЛЬТЕТ ПРИКЛАДНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Информационные системы и технологии»

Отчет по учебной практике

|  |  |
| --- | --- |
| Место прохождения  практики | Кафедра ИСТ |
| Время прохождения практики | с 22 июня 2015г. по 19 июля 2015г. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | ФИО | Подпись | Дата |
| Выполнил студент группы | Апсаликов М.Ю. |  |  |
| Руководитель от кафедры | Вагарина Н.С. |  |  |
| Руководитель от предприятия | Вагарина Н.С. |  |  |

|  |  |
| --- | --- |
| Оценка руководителя практики от предприятия |  |
| Итоговая оценка по защите результатов деятельности на практике |  |

Саратов 2015

# Содержание

[Содержание 2](#_Toc424853595)

[Введение 3](#_Toc424853596)

[1. Анализ проблемы использования Semantic Web. 4](#_Toc424853597)

[2. Обзор технологий 7](#_Toc424853598)

[3. Аналогичные программные продукты 10](#_Toc424853599)

[4. Систематизация требований 11](#_Toc424853600)

[Заключение 13](#_Toc424853601)

[Список использованной литературы 14](#_Toc424853602)

# Введение

В настоящее время делаются попытки поиска новых методов и подходов к созданию, предоставлению и обработке информации в связи с тем, что объем информации в Web колоссален. Один из подходов для решения этой задачи – применение технологий Semantic Web. Концепция Semantic Web была принята и продвигается Консорциумом W3C (World Wide Web Consortium), который внедряет и разрабатывает стандарты для World Wide Web. Данная концепция является стеком технологий, существующих над Всемирной паутиной (World Wide Web), который позволяет размещать информацию в сети Интернет в виде, пригодном для машинной обработки[1].

В настоящее время технологии Семантического Веба, также известного как Web 3.0, практически не используются в коммерческих проектах[2]. Это связано, в первую очередь, со значительным увеличением стоимости и времени разработки.

В выпускной квалификационной работе планируется частично решить эту проблему путем создания библиотеки для генерации семантических данных. Целью данной работы является определение требований к библиотеке. Для выполнения этой цели требуется выполнить следующие задачи:

* Проанализировать проблему и предложить решение;
* Выполнить краткий обзор планирующихся к использованию в решении технологий и обоснование их использования;
* Найти и описать аналогичные программные продукты;
* Систематизировать и обобщить требования к функционалу библиотеки.

# Анализ проблемы использования Semantic Web.

Базовая модель семантического Web включает следующие компоненты (в порядке повышения уровня абстракции):

* URI;
* XML;
* RDF;
* RDF Schema;
* OWL;
* OWL Schema;
* SPARQL;
* WSDL.

Ключевыми технологиями являются RDF и OWL.

RDF – это модель представления данных, которая позволяет описать любую информацию в виде триплетов: субъект – предикат – объект[3]. Субъект представляет собой сущность, которую необходимо описать, предикат – свойство субъекта, объект – значение этого свойства. Эта модель имеет несколько сериализаций, также известных как диалекты. Некоторые из них реализованы на основе популярных форматов сериализации. Рекомендованным диалектом является RDF/XML, который разрабатывался на основе XML. Также имеет большую популярность RDF/JSON, основанный на не менее популярном формате сериализации JSON. Существуют и упрощенные сериализации, такие как N-Triples, Turtle и другие. В чистом виде RDF-документ представляет собой коллекцию триплетов.

Технология OWL, являясь более высоким уровнем абстракции над RDF, описывает онтологии. Онтология – это формализированное описание общепринятого понимания некоторой предметной области, с помощью которого могут общаться как люди, так и компьютерные системы. Это система понятий предметной области которая представляется набором сущностей, соединенных различными отношениями. Онтологии имеют формальную структуру, поэтому их автоматизированная обработка имеет низкую ресурсозатратность. Онтология состоит из:

* Понятия – общие категории, которые могут быть упорядочены иерархически. Понятия описывают группы индивидуальных сущностей, объединенных по наличию общих свойств. Аналогом в объектно-ориентированном программировании является класс.
* Отношения. Самый распространенный тип – отношения категоризации, другими словами присвоение определенной категории.
* Аксиомы – это утверждения, вводимые в неизменяемом виде. Другие утвержения в онтологии могут быть выведены их аксиом. Они нужны для осуществления логических выводов из данных онтологий.
* Отдельные экземпляры – аналог объекта класса в объектно-ориентированном программировании. То есть конкретные элементы конкретной категории.

Основная цель Семантического Веба – это возможность семантического поиска информации, которую предоставляет логический вывод. Web 3.0, как и Web 2.0 работают на протоколах HTTP/HTTPS и в качестве идентификатора ресурсов URI (Uniform Resource Identifier).

Как было описано выше, в Семантическом Веб данные предоставлены в виде триплетов, предоставляемых в формате RDF. Однако, Web 2.0 предоставляет информацию человекочитаемыми технологиями, такими как HTML, CSS и JavaScript. Технологии Семантической и Всемирной паутин разные. Это логично, потому что цель одной – предоставить информацию алгоритмам, а другой – отобразить её для человека. Это означает, что если заказчику программного обеспечения нужен продукт, реализующий Семантические технологии, то при для его разработки потребуется написать два разных слоя представления: один для человека, а другой - машиночитаемый. Данный факт значительно увеличивает время и стоимость разработки. Как следствие, компании отказываются от использования Семантических технологий в своих проектах.

В качестве решения этой проблемы предлагается вариант, когда код хотя бы одного из слоёв представления будет генерироваться автоматически. С пользовательским интерфейсом в коммерческих приложения это впринципе невозможно, потому что бизнес всегда будет диктовать новые требования и постоянно модифицировать HTML, CSS и JavaScript код. А вот с семантическим предоставлением информации – это возможно. Так как и модель базы данных, и семантическое хранилище являются машиночитаемыми.

Теоретически, данную идею можно реализовать, как минимум, двумя путями. Первый – это использовать его как шаблон проектирования. Иными словами, переписывать для каждого приложения отдельно. Очевидно, что это жизнеспособное, но не самое лучшее решение. Вероятно, этот подход уже используется в каких-либо проектах. Второй способ – создание библиотеки под определенную платформу. Этот вариант реализации предпочтительнее, так как позволяет написать код один раз для всех проектов на этой платформе, которые пожелают его использовать.

# Обзор технологий

Давайте взглянем на современную веб-резработку и её стандарты[4]. На первом месте по использованию в качестве Web-платформы на сегодняшний день находится PHP. Он занимает 81.9% рынка. Однако, стоит учитывать специфику данного языка программирования. Данный язык разрабатывался для создания домашних страниц, о чем явно говорит его название: Personal Home Page. PHP очень редко используется в Enterprise-проектах в силу своей ненадежности. Более того, язык имеет существенные минусы, не позволяющие построить хорошую архитектуру Web-приложения, а это немаловажный фактор использования крупных коммерческих проектах.

Среди оставшихся технологий, самой популярной является ASP.NET. Она на втором месте по использован ию во всей Всемирной паутине и занимает 16.9% рынка. Как следствие, можно сделать вывод, что это самая распространенная Web-платформа в крупных коммерческих проектах. Библиотека ASP.NET является самым мощным фреймворком для построения Web-приложений на текущий момент. Она содержит в себе такие технологии, как ASP.NET WebForms, ASP.NET MVC и ASP.NET WebAPI. Последние две технологии являеются самыми современными на текущий момент. Для большинства запускаемых проектов приложений под платформу .NET выбирают эти библиотеки.

Также в качестве положительного решения в пользу использования платформы .NET можно указать богатые возможности языка C#, а именно:

* Большое количество «синтаксического сахара» в языке, который значительно увеличивает скорость разработки приложения;
* Удобство компиляции компонентов;
* Улучшенная скорость выполнения кода в связи с тем, что CLR оптимизирует IL код под различные архитектуры процессоров;
* Удобство отладки, предоставленное IDE Microsoft Visual Studio 2013;
* Технология LINQ, позволяющая работать эффективно обрабатывать данные из коллекций;
* Наличие библиотеки с открытым исходным кодом dotNetRdf для работы с семантическими данными[5];
* Наличие средств для ускорения разработки, таких как ReSharper.

Отдельно следует упомянуть здесь такое преимущество C# перед PHP и большинством других платформ, как метапрограммирование. В нем C# имеет самые мощные возможности на рынке и с этим может конкурировать только Java. Только эти две платформы предоставляют полноценные возможности рефлексии, которые позволяют восстановить исходный код с точностью до названий переменных. Это связано с тем, что обе платформы используют виртуальные машины: для Java – это JVM (Java Virtual Machine), а для C# - CLR (Common Language Runtime). Использование рефлексии в разрабатываемой библиотеке, несмотря на ее негативные моменты, такие как производительность и риски нарушения целостности данных, значительно упростит код приложений, написанных на этой библиотеке. Подробнее о планируемом использовании рефлексии можно прочитать в главе 4.

Для работы с базами данных в программных продуктах, написанных на C#, в том числе и коммерческих, часто используется ORM (Object-Relational Mapping) Entity Framework. Это библиотека с открытым исходным кодом, разработанная Microsoft, которая позволяет работать с базами данных используя три подхода: Database First, Model First и Code First. Database First позволяет по готовой базе сгенерировать модель, Model First – по готовой модели сгенерировать базу. Эти два подхода считаются устаревшими, так как имеют недостаток: при изменении схемы базы данных необходимо модель пересоздавать. Таким образом, актуален сейчас только один подход Code First. При данном подходе модель собирается из классов C#. Это позволяет отследить изменения модели и применять их (по возможности) автоматически, либо с помощью миграций.

Вследствие всего вышеуказанного, в коммерческих проектах очень часто можно встретить связку ASP.NET + Entity Framework. Более того, разработка логики на этих технологиях занимает в среднем меньше времени по сравнению с аналогичными технологиями на рынке.

В качестве еще одного варианта можно рассмотреть платформу Java. Она также широко используется в коммерческой разработке. Также, как было сказано ранее, имеет богатые возможности рефлексии. Однако, её распространенность в Web равна 3%, что значительно меньше, чем у .NET. Также, у нее меньше языковых возможностей и нет единого популярного ORM-фреймворка, который использовался бы на большинстве проектов.

В результате данного анализа для разрабатываемого решения был выбран технологический стек .NET. Стоит отметить, что для Java также можно создать аналогичную с разрабатываемой библиотеку, однако разработка на этой платформе внесет дополнительные риски в силу вышеописанных причин.

# Аналогичные программные продукты

В силу малой распространенности Semantic Web среди коммерческих проектов, под платформу .NET не было найдено ни одного аналогичного продукта. Существуют решения только в виде шаблонов проектирования.

В силу того, что аналогов для платформы .NET было не найдено, был произведен поиск аналогичных программных продуктов для всех платформ. В результате этого поиска была найдена единственная библиотека SuRF или SurfRDF[6]. Она написана на языке Python и соответственно может быть использована только для решений на этом языке. Исходный код проекта находится в свободном доступе на GitHub и доступен по лицензии BSD[7].

Функциональность этой библиотеки заключается в следующем: она производит конвертацию сущностей из кода в семантические данные на основе метаданных.

Результат анализа рынка аналогичных библиотек ожидаем в связи с тем, что, во-первых, технологии Семантического Веб в настоящее время не распространены повсеместно, а находятся в начале своего развития. Во-вторых, ORM фреймворки распространены только в пределах своих платформ. Это снижает вероятность существования подобного фреймворка.

# Систематизация требований

Итого, требуется разработать библиотеку под платформу .NET. Библиотека должна быть интергрирована с Entity Framework и ASP.NET WebAPI. Библиотека будет состоять из следующих модулей:

* Реализация шаблона проектирования «Репозиторий»;
* Базовый контроллер WebAPI;
* Базовый контроллер семантического сервиса;
* Интерфейсы для контроллеров;
* Конвертер данных;
* Менеджер конфигурации.

Рассмотрим подробнее каждый из модулей. В коммерческих проектах для работы с базами данных используется шаблон проектирования «Репозиторий»[8]. Его суть в том, что он унифицирует доступ к сущностям. В данном случае его необходимо реализовать в библиотеке в связи с необходимостью унификации доступа к данным из контроллеров. Это позволит избежать дублирования кода. Как небольшое дополнение, реализация паттерна в библиотеке позволит опустить его реализацию в самом коммерческом приложении, что снизит затраченные на приложение ресурсы. Шаблон проектирования «Репозиторий» планируется создать средствами обобщенных классов.

Базовые контроллеры WebAPI и семантического сервиса необходимо реализовать для того, чтобы приложение по-умолчанию поддерживало базовые операции работы с сущностями. Разработчики приложения унаследуют свои контроллеры от базового и получат всю базовую функциональность, не написав ни одной строки кода. Контроллеры напрямую планируется связать с соответствующим репозиторием при помощи обобщенных классов или рефлексии.

Интерфейсы для контроллеров WebAPI и семантического сервиса планируется создать в целях возможности юнит-тестирования. Это необязательное решение, однако оно может стать ключевым для выбора использования данной библиотеки в коммерческом проекте.

Конвертер данных – это ядро библиотеки. Необходим для конвертации сущностей в семантические данные для последующего предоставления клиентам. Однако, в отношениях баз данных есть только две сущности, но нет аналога предиката. Эта проблема может быть решена с помощью аттрибутов. Модуль будет считывать значение аттрибута и генерировать готовый к использованию триплет. Таким образом, рабочий процесс конвертации сущности в триплет будет выглядеть следующим образом: субъект равен сущности, к которой было произведено обращение; предикат равен информации из аттрибута, который необходимо заполнить разработчику; объект равен сущности, с которой субъект связан. Возможны несколько вариантов, если разработчик не укажет предикат. Самый простой вариант – это предотвращение компиляции приложения. Также возможно сгенерировать уникальный предикат по умолчанию. В разрабатываемой библиотеке планируется реализовать оба варианта и выбирать из них на основе конфигурационного файла. Конвертер данных планируется реализовать при помощи рефлексии.

Стоит отдельно разработать модуль конфигурации. Так как библиотеку планируется реализовать для реального использования в коммерческих проектах, то необходимо уделить расширяемости большое внимание. Модуль конфигурации должен считывать определенную секцию из файла конфигурации и применять соответствующие настройки как только конфигурация была изменена. Это также может стать решающим фактором в пользу использования данной библиотеки в коммерческом проекте.

# Заключение

В данной работе был выполнен обзор, анализ, систематизация и обобщение изученной научно-технической информации, а также сделаны следующие выводы:

* Семантический Веб имеет проблему дублирования кода, которую можно решить с помощью генерации семантического слоя представления;
* Это можно сделать путем создания библиотеки под определенную платформу, которую смогут использовать в коммерческих проектах;
* Для создания такой библиотеки наиболее перспективной является платформа .NET.

Также, в данной работе были исследованы аналогичные программные средства. По результатам анализа был найден всего один программный продукт, написанный под другую платформу.

Были систематизированы требования к приложению, оно было условно разбито на предполагаемые модули, было описано их назначение и варианты реализации.

Предполагается, что библиотека будет иметь востребованность среди проектов на платформе .NET. В любом случае, реализация данной библиотеки является одним шагом вперед к повсеместному использованию технологий Семантического Веб.

# Список использованной литературы

1. Berners-Lee, T. The Semantic Web. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities [Электронный ресурс] / T. Berners-Lee, J. Hendler, O. Lassila. Режим доступа: http://jeckle.de/files/tblSW.pdf (дата обращения 11.07.2015).
2. Kashyap V. Real World Semantic Web Applications / V. Kashyap, L. Shklar. – IOS Press, 2002. – 197 c.
3. Klyne, G. Resource Description Framework (RDF): Concepts and Abstract Syntax [Электронный ресурс] /G. Klyne, J. Carroll. − Режим доступа: http://www.w3.org/TR/rdf-concepts/ (11.07.2015).
4. World Wide Web Technology Surveys [Электронный ресурс] / W3Techs. – Режим доступа: http://w3techs.com/technologies/overview/programming\_language/all/ (11.07.2015)
5. Официальный сайт библиотеки dotNetRdf [Электронный ресурс]. – Режим доступа: http://www.dotnetrdf.org (дата обращения 13.06.2014)
6. Официальный сайт библиотеки SuRF [Электронный ресурс]. – Режим доступа: https://code.google.com/p/surfrdf/ (дата обращения 13.06.2014)
7. The BSD 3-Clause License [Электронный ресурс] / Open Source Initiative - Режим доступа: http://opensource.org/licenses/BSD-3-Clause (дата обращения 13.06.2014).
8. Microsoft Development Network: The Repository Pattern [Электронный ресурс] / MSDN. Режим доступа - https://msdn.microsoft.com/en-us/library/ff649690.aspx (11.07.2015).