Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

**ОТЧЕТ**

по производственной практике

Студент группы 350531: Рогатка А.А.

Руководитель практики

от университета: Бурый Я.А.

Руководитель практики

от предприятия: Ведмич В.В.

МИНСК 2017

**СОДЕРЖАНИЕ**

[1 Задание по производственной практике 3](#_Toc391415565)

[2 Структура предприятия 4](#_Toc391415566)

[3 Обзор стека технологий используемых на проекте 10](#_Toc391415567)

[3.1 .Net Framework 10](#_Toc391415568)

[3.1.1 C# 11](#_Toc391415569)

[3.1.2 ASP.NET MVC 13](#_Toc391415570)

[3.2 JavaScript 15](#_Toc391415571)

[3.2.1 Структура языка 15](#_Toc391415572)

[3.2.2 Область применения 17](#_Toc391415573)

[3.3 SQL 19](#_Toc391415574)

[3.3.1 Описание 21](#_Toc391415575)

[3.3.2 Операторы 21](#_Toc391415576)

[3.3.3 Преимущества и недостатки 22](#_Toc391415577)

[3.4 Сторонние компоненты 24](#_Toc391415578)

[3.4.1 Библиотека Knockout.js 24](#_Toc391415579)

[3.5 VBScript 25](#_Toc391415580)

[4 Выполнение технического задания 28](#_Toc391415581)

[4.1 Определение требований 28](#_Toc391415582)

[5 Знакомство с системами контроля версий 39](#_Toc391415583)

[6 Заключение 41](#_Toc391415584)

# **ЗАДАНИЕ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Заданием по производственной практике является:

1. Ознакомление со структурой предприятия. Организация управления
2. Ознакомление с системами планирования, организации труда и формами его оплаты на предприятии
3. Изучение основных информационных потоков и документооборота на предприятии
4. Этапы прохождения продукции на предприятии, от заключения договора на его производство до сдачи заказчику (проект, создание спецификаций, ...)
5. Изучить вопросы тестирования и сертификации продукции

Место прохождения технологической практики — ИООО «ЭПАМ Системз». Руководитель практики от предприятия — Кострома Н.И. B соответствии с указаниями руководителя технологической практики от вуза Иванова Н.Н. были определены цели и сформулированы основные задачи.

Дата прибытия на практику – 08.06.2013.

Дата окончания практики – 29.06.2013.

Результатом прохождения практики является оценка руководителя практики, характеристика, а также отзыв руководителей практики от вуза.

# **СТРУКТУРА ПРЕДПРИЯТИЯ**

Компания «EPAM Systems» — ведущий разработчик программного обеспечения в Центральной и Восточной Европе.

Созданная в 1993 году, сегодня она имеет 17 представительств в 8 странах мира, в штате более 10 000 высококвалифицированных специалистов и компания продолжает стабильный рост. Выполняя проекты для крупнейших корпораций и сотрудничая с ведущими мировыми разработчиками программного обеспечения, «EPAM Systems» приобрела уникальный опыт в таких областях как:

* разработка по заказам крупнейших производителей программного обеспечения для систем корпоративного планирования (ERP), управления жизненным циклом изделий (PLM); корпоративных информационных порталов (EIP), систем управления отношениями с клиентами (CRM), серверов интеграции приложений (EAI), систем управления контентом, систем управления знаниями (KMS);
* разработка приложений, соответствующих требованиям новейших сервис-ориентированных архитектур;
* создание и развертывание электронных систем управления закупками и сбытом;
* построение порталов крупных предприятий и холдингов с развитыми средствами анализа данных и управления знаниями;
* интеграция приложений в распределенных системах (в том числе насчитывающих сотни производственных площадок, сотни унаследованных приложений и десятки ERP-систем), проектирование, консолидация и настройка корпоративных справочников и каталогов;
* внедрение ERP, PLM, CRM, SCM решений и систем аналитики, стратегического планирования и бюджетирования в ряде отраслей;
* анализ инфраструктуры и информационных ресурсов, проектирование и реинжиниринг бизнес-процессов, управление проектами модернизации и развития информационных систем.

EPAM обладает обширным списком престижных клиентов по всему миру, среди которых многие члены списка Fortune 500 (рейтинг 500 крупнейших мировых компаний, критерием составления которого служит выручка) и ведущие компании-разработчики программного обеспечения.

Компания с успехом внедрила свои решения в различных областях промышленности, включая:

* страхование и финансы;
* программное обеспечение;
* телекоммуникации и высокие технологии;
* розничная торговля и потребительские товары;
* туризм и индустрия развлечений;
* СМИ, исследования и дистанционное образование;
* производство, транспорт и энергетика.

Выручка компании за 2012 год составила $433,8 млн долл. Среди клиентов EPAM более 100 компаний, в их число входят, по собственным данным EPAM, такие крупные компании, как Google, UBS, Coca-Cola, Oracle, Adidas, Viacom, «Ренессанс-Капитал», Thomson Reuters, Ситибанк, Wolters Kluwer, Barclays Capital, Schlumberger и Halliburton, а также SAP, IHG, Expedia и многие другие. Также «EPAM Systems» стала первым в истории из региона Центральной и Восточной Европы поставщиком услуг в сфере ИТ, разместившим свои акции на Нью-Йоркской фондовой бирже.

EPAM Systems занимает ведущие позиции среди аутсорсинговых компаний в ежегодном рейтинге "Global Outsourcing", составленном Международной ассоциацией профессионалов в области аутсорсинга (IAOP). Компания, в частности, входит в число лидеров – поставщиков услуг для организаций, производящих программное и аппаратное обеспечение, и компаний сферы развлечений и медиа-бизнеса. EPAM также входит в список лучших мировых поставщиков услуг "2011 Global Services", составленном журналом Global Services и компанией NeoAdvisory: третий год подряд EPAM награждается в номинации "Top Outsourced Product Engineering Vendors".

В настоящий момент в штате компании более 10 000 специалистов, выполняющих проекты для крупных заказчиков в более чем 30 странах мира. Наиболее крупные центры разработки находятся в Беларуси и Украине. Также отделения компании расположены в США, Канаде, Венгрии, Украине, Великобритании, Германии, Казахстане, Польше, Швеции и Швейцарии.

Рис. 2.1. Центры разработки

Специалисты компании разрабатывают решения, обеспечивающие поддержку актуальных задач заказчика, в числе которых:

* привлечение клиентов и укрепление лояльности
* электронная коммерция
* управление жизненным циклом продукции
* взаимодействие с партнерами
* формирование отчетности и бизнес-аналитика
* бюджетирование и планирование
* управление проектами
* ведение учета и управление ресурсами (ERP) и другие.

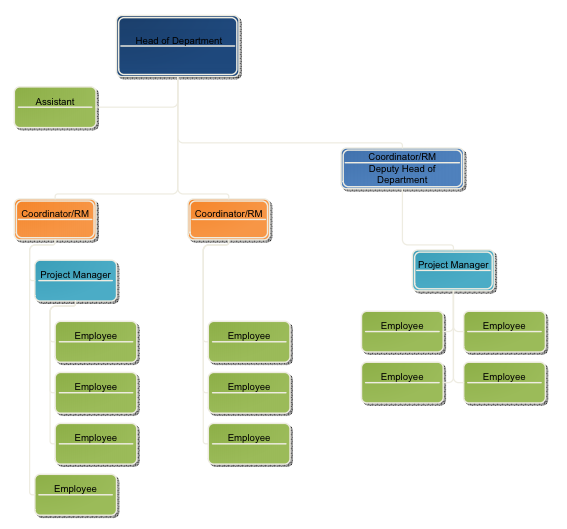


Рис. 2.2 Пример управленческой структуры предприятия

Необходимо отметить, что численность сотрудников постоянно увеличивается — это вызвано расширением компании. Регулярно организуются курсы по целенаправленному подбору персонала: как для сотрудников компании, так и для внешних кандидатов. Также огромное значение руководство компании уделяет вопросам повышения квалификации своих сотрудников. Это достигается путём организации тематических и проблемных курсов и семинаров (как на базе компании, так и в специализированных учебных центрах), обмена опытом и т.д.

Компания «EPAM Systems» успешно не единожды прошла международную сертификацию соответствия 4-му уровню CMMI — Capability Maturity Model Integration, показывающему уровень совершенствования процессов в организации. Во всем мире лишь порядка 140 компаний обладают сертификатами CMMI 4 и 5 уровня, что говорит о жестких требованиях, предъявляемых к организациям. EPAM первой в Европе была сертифицирована по CMMI Level 4 и на сегодняшний день является единственной в странах СНГ компанией, имеющей не один, а несколько сертифицированных по CMMI центров разработки программного обеспечения.

Соответствие стандарту CMMI означает, что компания на международном уровне признается надежным и эффективным разработчиком программного обеспечения, а также поставщиком услуг в области ИТ. В процессе сдачи "экзамена" по CMMI организация совершенствует свои бизнес-процессы и качество разработки ПО, что позволяет обеспечить стабильно высокое качество услуг и служит одной из основ для повышения конкурентоспособности и дальнейшего развития компании.

# Обзор стека технологий используемых на проекте

## .Net Framework

NET Framework — [программная платформа](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#.D0.BA.D1.80.D0.BE.D1.81.D1.81.D0.BF.D0.BB.D0.B0.D1.82.D1.84.D0.BE.D1.80.D0.BC.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BF.D1.80.D0.BE.D0.B3.D1.80.D0.B0.D0.BC.D0.BC.D0.BD.D0.BE.D0.B5_.D0.BE.D0.B1.D0.B5.D1.81.D0.BF.D0.B5.D1.87.D0.B5.D0.BD.D0.B8.D0.B5), выпущенная компанией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) в [2002 году](http://ru.wikipedia.org/wiki/2002_%D0%B3%D0%BE%D0%B4). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](http://ru.wikipedia.org/wiki/Common_Language_Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Считается, что платформа .NET Framework явилась ответом компании [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) на набравшую к тому времени большую популярность платформу [Java](http://ru.wikipedia.org/wiki/Java) компании [Sun Microsystems](http://ru.wikipedia.org/wiki/Sun_Microsystems) (ныне принадлежит [Oracle](http://ru.wikipedia.org/wiki/Oracle)).

Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows), существуют независимые проекты (прежде всего это [Mono](http://ru.wikipedia.org/wiki/Mono) и[Portable.NET](http://ru.wikipedia.org/wiki/Portable.NET)), позволяющие запускать программы .NET на некоторых других операционных системах.

Существует много различных версий .Net FrameWork,в данном случае использовался .NET Framework 4.5[

Основными отличиями от предыдущих версий являются:

* Возможность уменьшения количества перезапусков системы путем обнаружения и закрытия приложений платформы .NET Framework версии 4 во время развертывания.
* Поддержка массивов, размер которых превышает 2 гигабайта (ГБ) на 64-разрядных платформах.
* Улучшенная производительность благодаря фоновой сборке мусора для серверов. При использовании серверной сборки мусора в .NET Framework 4.5 фоновая сборка мусора включается автоматически.
* Фоновая компиляция по требованию (JIT), которая опционально доступна на многоядерных процессорах для повышения производительности приложения.
* Возможность ограничить, как долго обработчик регулярных выражений будет пытаться разрешить регулярное выражение до истечения времени ожидания.
* Возможность определить культуру по умолчанию для домена приложения.
* Поддержка кодировки Юникод (UTF-16) в ﻿консоли.
* Поддержка управления версиями культурных данных сортировки и сравнения строк.
* Улучшенная производительность при извлечении ресурсов.
* Улучшения сжатия ZIP
* Возможность настраивать контекст отражения для переопределения поведения отражения по умолчанию
* Поддержка версии 2008 стандарта интернационализированных доменных имен в приложениях, когда класс используется в Windows 8
* Делегирование сравнения строк операционной системе, которая реализует Юникод 6.0, если платформа .NET Framework используется в Windows 8. При работе на других платформах платформа .NET Framework включает собственные сведения о сравнении строк, которые реализуют Юникод 5.xx.
* Возможность вычисления хэш-кода для строк на основе домена для каждого приложения.

### C#

C# (произносится сишарп) — [объектно-ориентированный](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](http://ru.wikipedia.org/wiki/1998)—[2001 годах](http://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров под руководством [Андерса Хейлсберга](http://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) в компании [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) как язык разработки приложений для платформы [Microsoft .NET Framework](http://ru.wikipedia.org/wiki/.NET_Framework) и впоследствии был стандартизирован как [ECMA](http://ru.wikipedia.org/wiki/ECMA)-334 и [ISO](http://ru.wikipedia.org/wiki/ISO)/[IEC](http://ru.wikipedia.org/wiki/IEC) 23270.

C# относится к семье языков с [C-подобным синтаксисом](http://ru.wikipedia.org/w/index.php?title=C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81&action=edit&redlink=1), из них его синтаксис наиболее близок к [C++](http://ru.wikipedia.org/wiki/C%2B%2B) и [Java](http://ru.wikipedia.org/wiki/Java). Язык имеет[статическую типизацию](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [перегрузку операторов](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](http://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [свойства](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типыи методы, [итераторы](http://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#.D0.93.D0.B5.D0.BD.D0.B5.D1.80.D0.B0.D1.82.D0.BE.D1.80.D1.8B), [анонимные функции](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](http://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](http://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](http://ru.wikipedia.org/wiki/C%2B%2B), [Pascal](http://ru.wikipedia.org/wiki/Pascal_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Модула](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](http://ru.wikipedia.org/wiki/Smalltalk) и в особенности [Java](http://ru.wikipedia.org/wiki/Java_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественное наследование интерфейсов](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#.D0.9C.D0.BD.D0.BE.D0.B6.D0.B5.D1.81.D1.82.D0.B2.D0.B5.D0.BD.D0.BD.D0.BE.D0.B5_.D0.BD.D0.B0.D1.81.D0.BB.D0.B5.D0.B4.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D0.B5_.D0.B8_.D1.80.D0.B5.D0.B0.D0.BB.D0.B8.D0.B7.D0.B0.D1.86.D0.B8.D1.8F_.D0.B8.D0.BD.D1.82.D0.B5.D1.80.D1.8)).

Основные особенности данного языка программирования:

C# разрабатывался как язык программирования прикладного уровня для [CLR](http://ru.wikipedia.org/wiki/Common_Language_Runtime) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает [BCL](http://ru.wikipedia.org/wiki/Base_Class_Library). Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим [.NET](http://ru.wikipedia.org/wiki/.NET_Framework)-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, [сборка мусора](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на [VB.NET](http://ru.wikipedia.org/wiki/Visual_Basic_.NET), [J#](http://ru.wikipedia.org/wiki/Visual_J%E2%99%AF) и др.

### ASP.NET MVC

ASP.NET MVC представляет собой платформу для создания сайтов и веб-приложений с использованием паттерна (или шаблона) MVC (model - view - controller).

Работа над новой платформой была начата в 2007 году, а в 2009 году появилась первая версия. В итоге к текущему моменту (2012 год) уже было выпущено 4 версии платформы, а сам фреймворк обрел большую популярность по всему миру благодаря своей гибкости и адаптивности.

Шаблон MVC, лежащий в основе новой платформы, подразумевает взаимодействие трех компонентов: контроллера (controller), модели (model) и представления (view). Что же представляют эти компоненты?

Контроллер (controller) представляет класс, с которого собственно и начинается работа приложения. Этот класс обеспечивает связь между моделью и представлением. Получая вводимые пользователем данные, контроллер исходя из внутренней логики при необходимости обращается к модели и генерирует соответствующее представление.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения - например, html-страница, через которую пользователь, зашедший на сайт, взаимодействует с веб-приложением.

Модель (model) представляет набор классов, описывающих логику используемых данных.

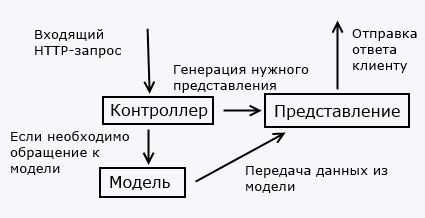


Рис 3.1 Схема взаимодействия

ASP.NET MVC является в некотором роде конкурентом для традиционных веб-форм и имеет по сравнению с ними следующие преимущества:

Разделение ответственности. В MVC приложение состоит из трех частей: контроллера, представления и модели, каждая из которых выполняет свои специфичные функции. В итоге приложение будет легче поддерживать модифицировать в будущем.

В силу разделения ответственности приложения mvc обладают лучшей тестируемостью. И мы можем тестировать отдельные компоненты независимо друг от друга.

Соответствие протоколу HTTP. Приложения MVC в отличие от веб-форм не поддерживают объекты состояния (ViewState). Ясность и простота платформы позволяют добиться большего контроля над работой приложения

Гибкость. Вы можете настраивать различные компоненты платформы по своему усмотрению. Изменять какие-либо части конвейера работы MVC или адаптировать его к своим нуждам и потребностям.

В то же время не стоит однозначно сбрасывать со счетов ASP.NET WebForms. Поскольку она также имеет свои сильные стороны, например, модель событий, которая будет ближе тем разработчикам, которые ранее занимались созданием клиентских приложений.

В традиционных веб-формах вы имеете контроль над разметкой и можете в реальном времени в визуальном редакторе Visual Studio увидеть, как будет выглядеть та или иная страница. При работе с MVC Visual Studio подобного не позволяет делать.

В любом случае вы вольны выбирать ту платформу, которая приходится вам больше по душе. И если у вас написаны объемные проекты с применением традиционных веб-форм, возможно, стоит продолжать с ними работать.

## JavaScript

[Прототипно-ориентированный](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [сценарный](http://ru.wikipedia.org/wiki/%D0%A1%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Является [диалектом](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%BB%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) языка [ECMAScript](http://ru.wikipedia.org/wiki/ECMAScript).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам [приложений](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0). Наиболее широкое применение находит в [браузерах](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) как язык сценариев для придания [интерактивности](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [веб-страницам](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0).

Основные архитектурные черты: [динамическая типизация](http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), слабая типизация, [автоматическое управление памятью](http://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0), [прототипное программирование](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), функции как [объекты первого класса](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D0%B2%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%B0).

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования [непрограммистами](http://ru.wiktionary.org/wiki/%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C-%D0%BD%D0%B5%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82). Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным [товарным знаком](http://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B2%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%B7%D0%BD%D0%B0%D0%BA) компании [Oracle Corporation](http://ru.wikipedia.org/wiki/Oracle_Corporation).

### Структура языка

Структурно JavaScript можно представить в виде объединения трёх чётко различимых друг от друга частейю

* ядро (ECMAScript),
* объектная модель браузера ([Browser Object Model или BOM](http://ru.wikipedia.org/w/index.php?title=Browser_Object_Model&action=edit&redlink=1) ),
* объектная модель документа ([Document Object Model или DOM](http://ru.wikipedia.org/wiki/Document_Object_Model)).

Если рассматривать JavaScript в отличных от браузера окружениях, то объектная модель браузера и объектная модель документа могут не поддерживаться.

Объектную модель документа иногда рассматривают как отдельную от JavaScript сущность, что согласуется с определением DOM как независимого от языка интерфейса документа. В противоположность этому ряд авторов находят BOM и DOM тесно взаимосвязанными.

ECMAScript не является браузерным языком и в нём не определяются методы ввода и вывода информации. Это, скорее, основа для построения скриптовых языков. Спецификация ECMAScript описывает типы данных, инструкции, ключевые и [зарезервированные](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D1%80%D0%B5%D0%B7%D0%B5%D1%80%D0%B2%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D1%81%D0%BB%D0%BE%D0%B2%D0%BE) слова, [операторы](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), объекты, [регулярные выражения](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F), не ограничивая авторов производных языков в расширении их новыми составляющими.

Объектная модель браузера — браузер-специфичная часть языка[[30]](http://ru.wikipedia.org/wiki/JavaScript#cite_note-chapmanBOM-35), являющаяся прослойкой между ядром и объектной моделью документа. Основное предназначение объектной модели браузера — управление окнами браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window, центральным объектом DOM. Объектная модель браузера на данный момент не стандартизирована, однако спецификация находится в разработке [WHATWG](http://ru.wikipedia.org/wiki/WHATWG).

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих сущностей.

* управление [фреймами](http://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC_(HTML)),
* поддержка задержки в исполнении кода и зацикливания с задержкой,
* системные диалоги,
* управление адресом открытой страницы,
* управление информацией о браузере,
* управление информацией о параметрах [монитора](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80_(%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)),
* ограниченное управление историей просмотра страниц,
* поддержка работы с [HTTP cookie](http://ru.wikipedia.org/wiki/HTTP_cookie).
* Объектная модель документа
* Основная статья: [Document Object Model](http://ru.wikipedia.org/wiki/Document_Object_Model)

Объектная модель документа — [интерфейс программирования приложений](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для HTML и [XML](http://ru.wikipedia.org/wiki/XML)-документов. Согласно DOM, документ (например, веб-страница) может быть представлен в виде дерева объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

* генерация и добавление узлов,
* получение узлов,
* изменение узлов,
* изменение связей между узлами,
* удаление узлов.

### Область применения

JavaScript используется в клиентской части веб-приложений: клиент-серверных программ, в котором клиентом является браузер, а сервером — веб-сервер, имеющих распределённую между сервером и клиентом логику. Обмен информацией в веб-приложениях происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами.

JavaScript используется в [AJAX](http://ru.wikipedia.org/wiki/AJAX), популярном подходе к построению интерактивных пользовательских интерфейсов веб-приложений, заключающемся в «фоновом» асинхронном обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и интерфейс веб-приложения становится быстрее, чем это происходит при традиционном подходе (без применения AJAX).

JavaScript широко используется в [браузерных операционных системах](http://ru.wikipedia.org/wiki/WebOS). Так, например, исходный код [IndraDesktop WebOS](http://ru.wikipedia.org/w/index.php?title=IndraDesktop_WebOS&action=edit&redlink=1) на 75 % состоит из JavaScript, код браузерной операционной системы [IntOS](http://ru.wikipedia.org/w/index.php?title=IntOS&action=edit&redlink=1) — на 70 %. Доля JavaScript в исходном коде [eyeOS](http://ru.wikipedia.org/wiki/EyeOS) — 5 %, однако и в рамках этой операционной системы JavaScript играет важную роль, участвуя в визуализации на клиенте и являясь необходимым механизмом для коммуницирования клиента и сервера.

Пользовательские скрипты в браузере — это программы, написанные на JavaScript, выполняемые в браузере пользователя при загрузке страницы. Они позволяют автоматически заполнять формы, переформатировать страницы, скрывать нежелательное содержимое и встраивать желательное для отображения содержимое, изменять поведение клиентской части веб-приложений, добавлять элементы управления на страницу и т. д.

Для управления пользовательскими скриптами в [Mozilla Firefox](http://ru.wikipedia.org/wiki/Mozilla_Firefox) используется расширение [Greasemonkey](http://ru.wikipedia.org/wiki/Greasemonkey); [Opera](http://ru.wikipedia.org/wiki/Opera) и [Google Chrome](http://ru.wikipedia.org/wiki/Google_Chrome) предоставляют средства поддержки пользовательских скриптов и возможности для выполнения ряда скриптов Greasemonkey.

Приложения, написанные на JavaScript, могут исполняться на серверах, использующих [Java](http://ru.wikipedia.org/wiki/Java) 6 и более поздних версий. Это обстоятельство используется для построения серверных приложений, позволяющих обрабатывать JavaScript на стороне сервера.

Помимо Java 6, существует ряд платформ, использующих существующие движки (интерпретаторы) JavaScript для исполнения серверных приложений. (Как правило, речь идёт о повторном использовании движков, ранее созданных для исполнения кода JavaScript в браузерах WWW.)

## SQL

SQL (structured query language — «структурированный язык запросов») — [формальный](http://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной [реляционной базе данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей системой управления базами данных ([СУБД](http://ru.wikipedia.org/wiki/%D0%A1%D0%A3%D0%91%D0%94)). SQL основывается на [исчислении кортежей](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BA%D0%BE%D1%80%D1%82%D0%B5%D0%B6%D0%B5%D0%B9).

SQL является, прежде всего, [информационно-логическим языком](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA), предназначенным для описания, изменения и извлечения данных, хранимых в [реляционных базах данных](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). SQL можно назвать [языком программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), при этом он не является [Тьюринг-полным](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B0_%D0%BF%D0%BE_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D1%83), но вместе с тем стандарт языка спецификацией [SQL/PSM](http://ru.wikipedia.org/wiki/SQL/PSM) предусматривает возможность его [процедурных расширений](http://ru.wikipedia.org/wiki/SQL#.D0.9F.D1.80.D0.BE.D1.86.D0.B5.D0.B4.D1.83.D1.80.D0.BD.D1.8B.D0.B5_.D1.80.D0.B0.D1.81.D1.88.D0.B8.D1.80.D0.B5.D0.BD.D0.B8.D1.8F).

Изначально SQL был основным способом работы пользователя с [базой данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования.

При всех своих изменениях SQL остаётся единственным механизмом связи между прикладным программным обеспечением и базой данных. В то же время современные СУБД, а также информационные системы, использующие СУБД, предоставляют пользователю развитые средства визуального построения запросов.

Каждое предложение SQL — это либо запрос данных из базы, либо обращение к базе данных, которое приводит к изменению данных в базе. В соответствии с тем, какие изменения происходят в базе данных, различают следующие типы запросов:

запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта);

* запросы на получение данных;
* запросы на добавление новых данных (записей);
* запросы на удаление данных;
* обращения к СУБД.

Основным объектом хранения реляционной базы данных является таблица, поэтому все SQL-запросы — это операции над таблицами. В соответствии с этим, запросы делятся на:

* запросы, оперирующие самими таблицами (создание и изменение таблиц);
* запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Каждая таблица описывается в виде перечисления своих полей (столбцов таблицы) с указанием

* типа хранимых в каждом поле значений;
* связей между таблицами (задание первичных и вторичных ключей);
* информации, необходимой для построения индексов.

Запросы первого типа в свою очередь делятся на запросы, предназначенные для создания в базе данных новых таблиц, и на запросы, предназначенные для изменения уже существующих таблиц. Запросы второго типа оперируют со строками, и их можно разделить на запросы следующего вида:

* вставка новой строки;
* изменение значений полей строки или набора строк;
* удаление строки или набора строк.

Самый главный вид запроса — это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трёх операций:

* просмотреть полученный набор;
* изменить все записи набора;
* удалить все записи набора.

Таким образом, использование SQL сводится, по сути, к формированию всевозможных выборок строк и совершению операций над всеми записями, входящими в набор.

### Описание

Язык SQL представляет собой совокупность

* операторов,
* инструкций,
* и вычисляемых функций.

### Операторы

Согласно общепринятому [стилю программирования](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D0%BB%D1%8C_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), операторы (и другие зарезервированные слова) в SQL обычно рекомендуется писать [прописными буквами](http://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D1%8B%D0%B5_%D0%B1%D1%83%D0%BA%D0%B2%D1%8B).

Операторы SQL делятся на:

* операторы определения данных (Data Definition Language, [DDL](http://ru.wikipedia.org/wiki/Data_Definition_Language)):
* [CREATE](http://ru.wikipedia.org/wiki/Create_(SQL)) создает объект БД (саму базу, таблицу, [представление](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_(%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)), пользователя и т. д.),
* [ALTER](http://ru.wikipedia.org/w/index.php?title=Alter_(SQL)&action=edit&redlink=1) изменяет объект,
* [DROP](http://ru.wikipedia.org/w/index.php?title=Drop_(SQL)&action=edit&redlink=1) удаляет объект;
* операторы манипуляции данными (Data Manipulation Language, [DML](http://ru.wikipedia.org/wiki/Data_Manipulation_Language)):
* [SELECT](http://ru.wikipedia.org/wiki/Select_(SQL)) считывает данные, удовлетворяющие заданным условиям,
* [INSERT](http://ru.wikipedia.org/wiki/Insert_(SQL)) добавляет новые данные,
* [UPDATE](http://ru.wikipedia.org/wiki/Update_(SQL)) изменяет существующие данные,
* [DELETE](http://ru.wikipedia.org/wiki/Delete_(SQL)) удаляет данные;
* операторы определения доступа к данным (Data Control Language, [DCL](http://ru.wikipedia.org/wiki/Data_Control_Language)):
* [GRANT](http://ru.wikipedia.org/w/index.php?title=Grant_(SQL)&action=edit&redlink=1) предоставляет пользователю (группе) разрешения на определенные операции с объектом,
* [REVOKE](http://ru.wikipedia.org/w/index.php?title=Revoke_(SQL)&action=edit&redlink=1) отзывает ранее выданные разрешения,
* [DENY](http://ru.wikipedia.org/w/index.php?title=Deny_(SQL)&action=edit&redlink=1) задает запрет, имеющий приоритет над разрешением;
* операторы управления [транзакциями](http://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) (Transaction Control Language, [TCL](http://ru.wikipedia.org/wiki/TCL_(SQL))):
* [COMMIT](http://ru.wikipedia.org/wiki/Commit_(SQL)) применяет транзакцию,
* [ROLLBACK](http://ru.wikipedia.org/wiki/Rollback_(SQL)) откатывает все изменения, сделанные в контексте текущей транзакции,
* [SAVEPOINT](http://ru.wikipedia.org/w/index.php?title=Savepoint_(SQL)&action=edit&redlink=1) делит транзакцию на более мелкие участки.

### Преимущества и недостатки

Несмотря на наличие диалектов и различий в синтаксисе, в большинстве своём тексты SQL-запросов, содержащие DDL и DML, могут быть достаточно легко перенесены из одной СУБД в другую. Существуют системы, разработчики которых изначально ориентировались на применение по меньшей мере нескольких СУБД (например: система электронного [документооборота](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82) [Documentum](http://ru.wikipedia.org/w/index.php?title=Documentum&action=edit&redlink=1) может работать как с [Oracle Database](http://ru.wikipedia.org/wiki/Oracle_Database), так и с [Microsoft SQL Server](http://ru.wikipedia.org/wiki/Microsoft_SQL_Server) и [DB2](http://ru.wikipedia.org/wiki/DB2)). Естественно, что при применении некоторых специфичных для реализации возможностей такой переносимости добиться уже очень трудно.

Наличие стандартов и набора тестов для выявления совместимости и соответствия конкретной реализации SQL общепринятому стандарту только способствует «стабилизации» языка. Правда, стоит обратить внимание, что сам по себе стандарт местами чересчур формализован и раздут в размерах (например, базовая часть стандарта SQL:2003 состоит из более 1300 страниц текста).

С помощью SQL программист описывает только то, какие данные нужно извлечь или модифицировать. То, каким образом это сделать, решает СУБД непосредственно при обработке SQL-запроса. Однако не стоит думать, что это полностью универсальный принцип — программист описывает набор данных для выборки или модификации, однако ему при этом полезно представлять, как СУБД будет разбирать текст его запроса. Чем сложнее сконструирован запрос, тем больше он допускает вариантов написания, различных по скорости выполнения, но одинаковых по итоговому набору данных.

Создатели реляционной модели данных [Эдгар Кодд](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4%D0%B4,_%D0%AD%D0%B4%D0%B3%D0%B0%D1%80), [Кристофер Дейт](http://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D1%82,_%D0%9A%D1%80%D0%B8%D1%81%D1%82%D0%BE%D1%84%D0%B5%D1%80) и их сторонники указывают на то, что SQL не является истинно реляционным языком. В частности, они указывают на следующие проблемы SQL[[6]](http://ru.wikipedia.org/wiki/SQL#cite_note-6):

* повторяющиеся строки;
* неопределённые значения (nulls);
* явное указание порядка колонок слева направо;
* колонки без имени и дублирующиеся имена колонок;
* отсутствие поддержки свойства «=»;
* использование указателей;
* высокая избыточность.

## Сторонние компоненты

### Библиотека Knockout.js

Сегодня связывание с данными — одна из наиболее популярных функций в разработке, и JavaScript-библиотека Knockout предоставляет такие функции при разработке с применением HTML и JavaScript. Простота синтаксиса декларативного связывания и беспроблемная интеграция с шаблонами разделения обязанностей (separation patterns), такими как Model-View-ViewModel (MVVM), значительно упрощают распространенные задачи передачи и приема данных, в то же время облегчая поддержку и совершенствование кода. В этой новой рубрике я расскажу о ситуациях, в которых применение Knockout просто идеально, объясню, как приступить к работе с ней, и продемонстрирую использование ее фундаментальных средств. Примеры кода, которые можно скачать по указанной в конце статьи ссылке, демонстрируют, как использовать декларативное связывание, создавать различные типы связующих объектов и писать ориентированный на данные JavaScript-код, который следует хорошим шаблонам разделения обязанностей вроде MVVM.

Knockout, разработанная Стивом Сандерсоном (Steve Sanderson), — небольшая библиотека JavaScript с открытым исходным кодом и лицензией MIT. На сайте [Knockoutjs.com](http://knockoutjs.com/) ведется обновляемый список браузеров, которые поддерживает Knockout (в настоящее время поддерживаются все основные браузеры, в том числе Internet Explorer 6+, Firefox 2+, Chrome, Opera и Safari). Чтобы приступить к разработке с использованием Knockout, вам потребуется несколько важных ресурсов. Начните со скачивания самой свежей версии Knockout (2.0.0 на данный момент) по ссылке [bit.ly/scmtAi](http://bit.ly/scmtAi) и добавьте ссылку на нее в свой проект. Однако, если вы используете Visual Studio 2010, настоятельно рекомендую вам предварительно установить NuGet Package Manager — расширение Visual Studio. И скачивать Knockout (а также любые другие нужные вам библиотеки) только с его помощью, потому что он обеспечивает управление версиями и уведомляет вас о появлении новых версий. NuGet скачает Knockout и поместит два JavaScript-файла в папку scripts вашего проекта. Для производственных целей рекомендуется файл с более коротким именем (knockout-x.y.z.js, где x.y.z — основной, вспомогательный номера и номер ревизии). Имеется также файл knockout-x.y.x-debug.js, содержащий исходный код Knockout в форме, читаемой человеком. Советую ссылаться на этот файл при изучении Knockout и в процессе отладки.

## VBScript

Язык на коором написаны страницы для мигрирования. Рассмотрим краткие сведения об эом языке

Visual Basic Scripting Edition (обычно просто VBScript) — [скриптовый](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), интерпретируемый компонентом [Windows Script Host](http://ru.wikipedia.org/wiki/Windows_Script_Host). Он широко используется при создании скриптов в [операционных системах](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) семейства [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows).

VBScript был создан компанией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) как замена устаревшему пакетному языку, интерпретируемому приложением [command.com](http://ru.wikipedia.org/wiki/Command.com).

Скрипты на языке VBScript чаще всего применяются в следующих областях, использующих продукцию Microsoft:

* автоматизация [администрирования](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) систем Windows;
* [серверный](http://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) программный код в [страницах](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) [ASP](http://ru.wikipedia.org/wiki/Active_Server_Pages);
* [клиентские](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9)) скрипты в [браузере](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) [Internet Explorer](http://ru.wikipedia.org/wiki/Microsoft_Internet_Explorer).

Такой тип сценариев обычно используется для:

* создания сложных сценариев;
* использования объектов из других приложений и библиотек;
* скрытия окон в ходе выполнения сценария;
* шифрования логики сценария.

В основном VBS-сценарии применяются для обработки данных, управления системой, работы с учетными записями пользователей и компьютеров, взаимодействия с офисными приложениями, работы с базами данных и прочих сложных задач.

VBS-сценарий — это обычный текстовый файл с расширением \*.vbs, который легко править в блокноте, а запускать на исполнение — двойным щелчком мыши или вызовом по имени в консоли.

Сценарии не [компилируются](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), а [интерпретируются](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B0%D1%82%D0%BE%D1%80). То есть для обработки скрипта в системе должен присутствовать интерпретатор языка VBS, и таких интерпретаторов в Windows два: оконный WScript и консольный CScript, оба интерпретатора это [Windows Script Host](http://ru.wikipedia.org/wiki/Windows_Script_Host) (WSH).

В Visual Basic работают следующие правила:

* длина строки не ограничена;
* регистр символов не учитывается;
* количество пробелов между параметрами не учитывается;
* строку команды можно разрывать, а на месте разрыва нужно вставлять символ " \_ ";
* максимальная длина имени переменной 255 символов;
* комментарии обозначаются символом " ' ";
* несколько строк можно объединять в одну, предварительно разделив их символом " : ".

VBScript, как и их родитель — язык Visual Basic, является объектно-ориентированным языком программирования, то есть основной концепцией является понятие объектов и классов.

Класс — это тип, описывающий устройство объектов. Объект подразумевает под собой нечто, что обладает определённым поведением и способом представления, объект — это экземпляр класса. Класс можно сравнить с чертежом, согласно которому создаются объекты. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.

Все объекты, с которыми работает Windows Script Host, имеют методы и свойства. Чтобы обратиться к методу, необходимо указать объект, а через точку - метод с необходимыми параметрами.

Аналогичная ситуация со свойствами, но свойства можно как назначать, так и считывать в переменные и другие свойства, правда, следует учитывать тип данных переменных и свойств, иначе сценарий выдаст ошибку несовместимости типов данных.

# Выполнение технического задания

## Определение требований

Основной задачей являлось перевод веб-страниц на новые стандарты и на новые технологии, используя современные подходы и паттерны проектирования. При миграции страниц использовался подход MVC. Рассмотрим ключевые идеи этого подхода.

Шаблон MVC, лежащий в основе новой платформы, подразумевает взаимодействие трех компонентов: контроллера (controller), модели (model) и представления (view).

Контроллер (controller) представляет класс, с которого собственно и начинается работа приложения. Этот класс обеспечивает связь между моделью и представлением. Получая вводимые пользователем данные, контроллер исходя из внутренней логики при необходимости обращается к модели и генерирует соответствующее представление.

Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения - например, html-страница, через которую пользователь, зашедший на сайт, взаимодействует с веб-приложением.

Модель (model) представляет набор классов, описывающих логику используемых данных.

Общую схему взаимодействия упрощенно можно представить следующим образом:

Рассмотрим пример контроллера на примере одной из мигрированных страниц

using System.Threading.Tasks;

using System.Web.Mvc;

using Cloud.UI.EDI.EDISystem.EDIProductionBatch.Actions;

using Cloud.UI.EDI.EDISystem.EDIProductionBatch.Models;

using Models.StoredProcedures.Part;

using Web.Mvc;

namespace Cloud.UI.EDI.Controllers

{

public class EdiProductionBatchController : BaseController

{

private readonly ViewBatchOutOfBalanceGridAction \_viewBatchGridAction;

private readonly UpdateBatchOutOfBalanceAction \_updateBatchAction;

private readonly ViewBatchOutOfBalanceFormAction \_viewBatchAction;

private readonly ViewReconcileTagListGridAction \_viewReconcileGridAction;

private readonly ViewReconcileTagGridAction \_viewBatchCoilListAction;

private readonly GenerateReconcileTagAction \_generateReconcileTagAction;

public EdiProductionBatchController(

ViewBatchOutOfBalanceGridAction viewBatchGridAction,

UpdateBatchOutOfBalanceAction updateBatchAction,

ViewBatchOutOfBalanceFormAction viewBatchAction,

ViewReconcileTagListGridAction viewReconcileGridAction,

ViewReconcileTagGridAction viewBatchCoilListAction,

GenerateReconcileTagAction generateReconcileTagAction)

{

\_viewBatchGridAction = viewBatchGridAction;

\_updateBatchAction = updateBatchAction;

\_viewBatchAction = viewBatchAction;

\_viewReconcileGridAction = viewReconcileGridAction;

\_viewBatchCoilListAction = viewBatchCoilListAction;

\_generateReconcileTagAction = generateReconcileTagAction;

}

public async Task<ActionResult> ReconcileTagList()

{

return View(await \_viewReconcileGridAction.ProcessAsync());

}

[HttpPost]

public async Task<ActionResult> UpdateBatch(OutOfBalanceUpdateRequest request)

{

return Json(await \_updateBatchAction.ProcessAsync(request));

}

public async Task<ActionResult> ViewForm(BatchOutOfBalanceParamsModel request)

{

return View(await \_viewBatchAction.ProcessAsync(request));

}

public async Task<ActionResult> Index()

{

return View(await \_viewBatchGridAction.ProcessAsync());

}

public async Task<ActionResult> BatchCoilList(ReconcileTagParamsModel request)

{

return View(awaitviewBatchCoilListAction.ProcessAsync(request));

}

[HttpPost]

public async Task<ActionResult> GenerateBatch(ReconcileTagParamsModel request)

{

returnJson(await\_generateReconcileTagAction.ProcessAsync(request));

}

}

}

Можно заметить что в данном случае применен механизм Внедрения зависимости

Внедрение зависимости (Dependency injection, DI) — процесс предоставления внешней зависимости [программному компоненту](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Является специфичной формой «[инверсии управления](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F)» ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Inversion of control), где изменение порядка связи осуществляется путём получения необходимой зависимости.

Условно, если [объекту](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) нужно получить доступ к определенному [сервису](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%80%D0%B2%D0%B8%D1%81_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC)&action=edit&redlink=1), объект берет на себя ответственность за доступ к этому сервису: он или получает прямую[ссылку](http://ru.wikipedia.org/wiki/%D0%A1%D1%81%D1%8B%D0%BB%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) на местонахождение сервиса, или обращается к известному «сервис-локатору» и запрашивает ссылку на реализацию определенного типа сервиса. Используя же внедрение зависимости, объект просто предоставляет свойство, которое в состоянии хранить ссылку на нужный тип сервиса; и когда объект создается, ссылка на реализацию нужного типа сервиса автоматически вставляется в это свойство (поле), используя средства среды.

Внедрение зависимости более гибко, потому что становится легче создавать альтернативные реализации данного типа сервиса, а потом указывать, какая именно реализация должна быть использована в, например, [конфигурационном файле](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB_%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D0%B8), без изменений в объектах, которые этот сервис используют. Это особенно полезно в [юнит-тестировании](http://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D1%82-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), потому что вставить реализацию «[заглушки](http://ru.wikipedia.org/wiki/Mock-%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82)» сервиса в тестируемый объект очень просто.

С другой стороны, излишнее использование внедрения зависимостей может сделать приложения более сложными и трудными в сопровождении: так как для понимания поведения программы программисту необходимо смотреть не только в исходный код, а еще и в конфигурацию, а конфигурация, как правило, невидима для [IDE](http://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), которые поддерживают анализ ссылок и рефакторинг, если явно не указана поддержка фреймворков с внедрениями зависимостей.

Рассмотрим пример модели :

namespace Cloud.UI.EDI.EDISystem.SupplierEDI.ReleaseCreation.Models

{

public class ReleaseCreationGridRowModel : ReleasesForEDIGetResponseRow

{

public ReleaseCreationGridRowModel()

{

}

public ReleaseCreationGridRowModel(ReleasesForEDIGetResponseRow row)

: base(row)

{

this.LastReceiptQuantityDefault = LastReceiptQuantity.HasValue ? LastReceiptQuantity.Value : default(decimal);

this.TotalReceivedDefault = TotalReceived.HasValue ? TotalReceived.Value : default(decimal);

this.ReceivedDefault = Received.HasValue ? Received.Value : default(decimal);

this.ReleaseQuantityBalanceDefault = ReleaseQuantityBalance.HasValue ? ReleaseQuantityBalance.Value : default(decimal);

this.ReleaseQuantityDefault = ReleaseQuantity.HasValue ? ReleaseQuantity.Value : default(decimal);

}

public decimal LastReceiptQuantityDefault { get; set; }

public decimal TotalReceivedDefault { get; set; }

public decimal ReceivedDefault { get; set; }

public decimal ReleaseQuantityBalanceDefault { get; set; }

public decimal ReleaseQuantityDefault { get; set; }

}}

Пример одного из экшнов, вызываемых в контроллере

using System.Linq;

using System.Threading.Tasks;

using Cloud.UI.EDI.EDISystem.SupplierEDI.ReleaseCreation.Models;

using Cloud.UI.EDI.EDISystem.SupplierEDI.ReleaseCreation.Validation;

using Models.StoredProcedures.Purchasing;

using Services.DataSource;

using TypeHelpers.Enumerable;

using Web.Mvc;

using Web.UI.Actions;

using Web.UI.Actions.Results;

using Web.UI.Builders.Validation;

namespace Cloud.UI.EDI.EDISystem.SupplierEDI.ReleaseCreation.Actions

{

public class SearchReleaseCreationAction : IDataAction<ReleasesForEDIGetRequest, ReleaseCreationGridModel>

{

private readonly IApplicationContext \_applicationContext;

private readonly IDataSourceInvoker \_dataSourceInvoker;

private readonly IValidationModelBuilderFactory \_validationBuilderFactory;

public SearchReleaseCreationAction(

IDataSourceInvoker dataSourceInvoker,

IApplicationContext applicationContext,

IValidationModelBuilderFactory validationBuilderFactory)

{

\_dataSourceInvoker = dataSourceInvoker;

\_applicationContext = applicationContext;

\_validationBuilderFactory = validationBuilderFactory;

}

public async Task<IDataResult<ReleasesForEDIGetRequest, ReleaseCreationGridModel>> ProcessAsync(ReleasesForEDIGetRequest request)

{

request.CustomerNo = \_applicationContext.CurrentCustomer.PCN;

var validationModel = \_validationBuilderFactory.Create<SearchReleaseCreationValidationBuilder>().Build();

var validationResult = await validationModel.ValidateModel(request);

if (!validationResult.Success)

{

return new DataResult<ReleasesForEDIGetRequest, ReleaseCreationGridModel>

{

ValidationResult = validationResult

};

}

var data = new ReleaseCreationGridModel();

ReleasesForEDIGetResponse responseData = await \_dataSourceInvoker.InvokeAsync<ReleasesForEDIGetRequest, ReleasesForEDIGetResponse>(request);

if (responseData != null && responseData.Rows != null)

{

data.Rows.AddRange(responseData.Rows.Select(row => new ReleaseCreationGridRowModel(row)));

}

return new DataResult<ReleasesForEDIGetRequest, ReleaseCreationGridModel> { Data = data };

}

}

}

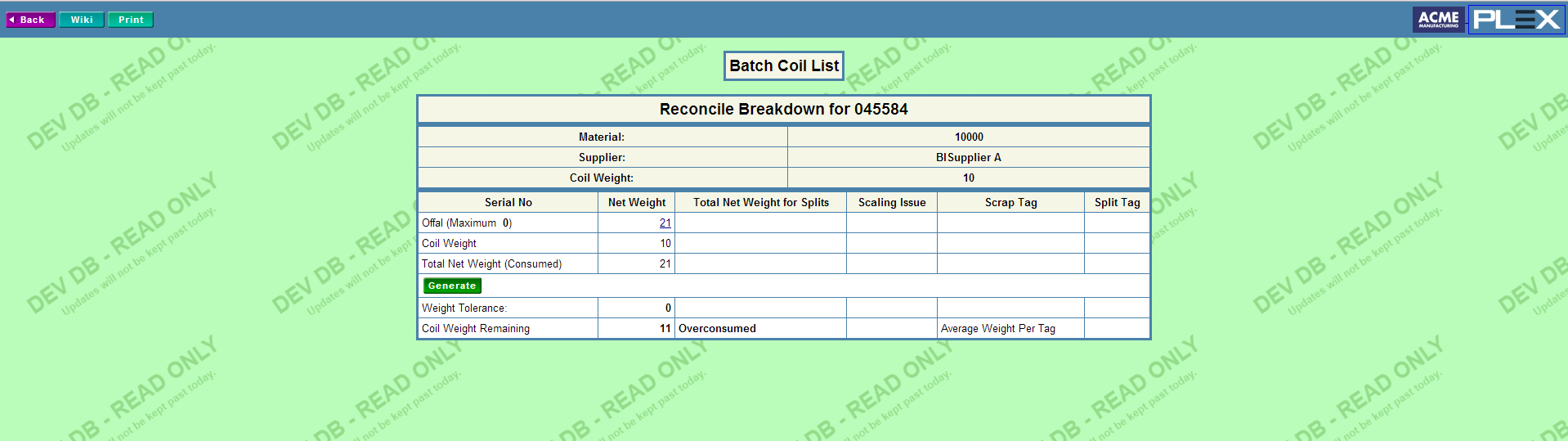


Рис. 4.1 Пример страницы до мигрирования

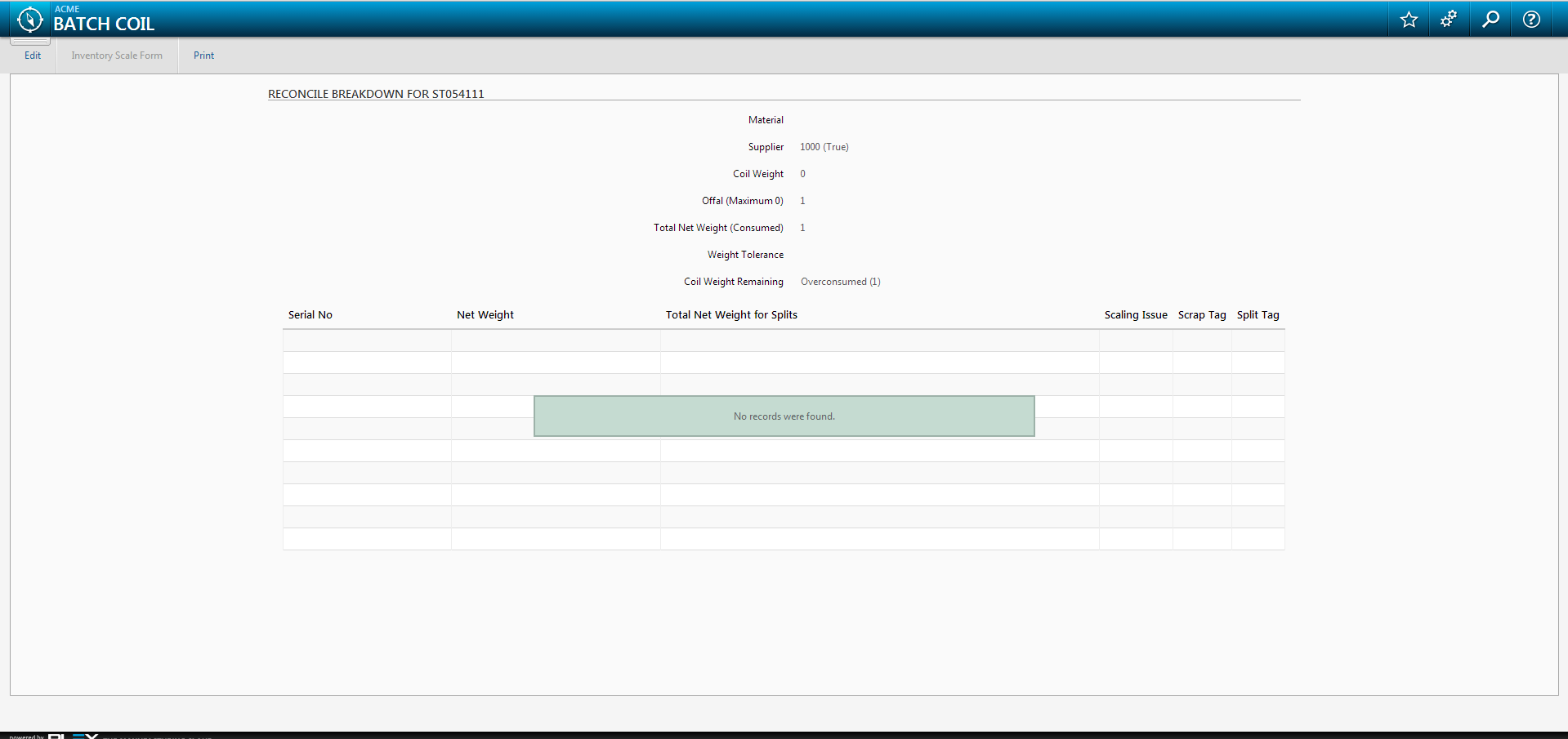


Рис. 4.2 Пример страницы после мигрирования

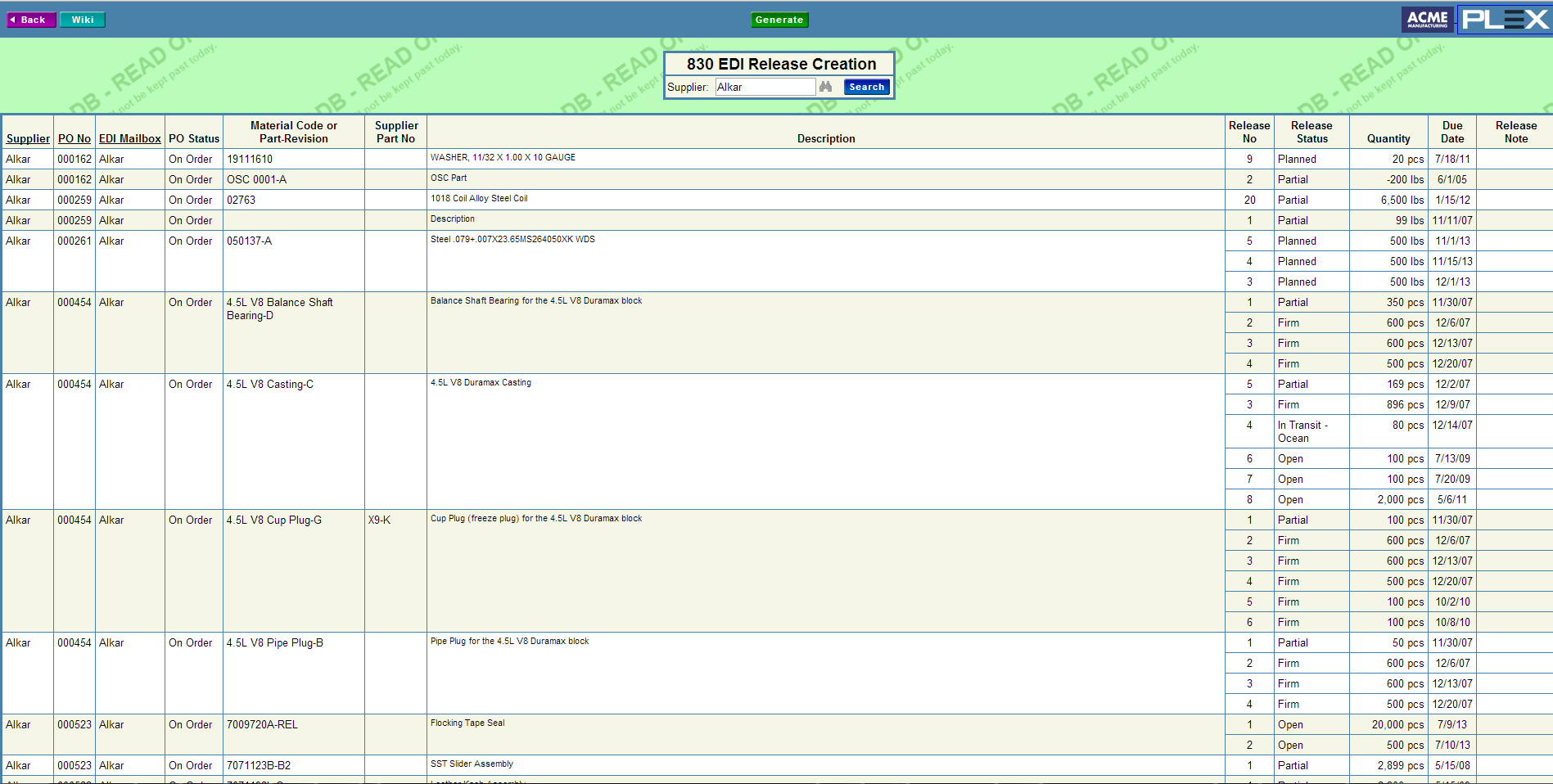


Рис. 4.3 Пример страницы до мигрирования

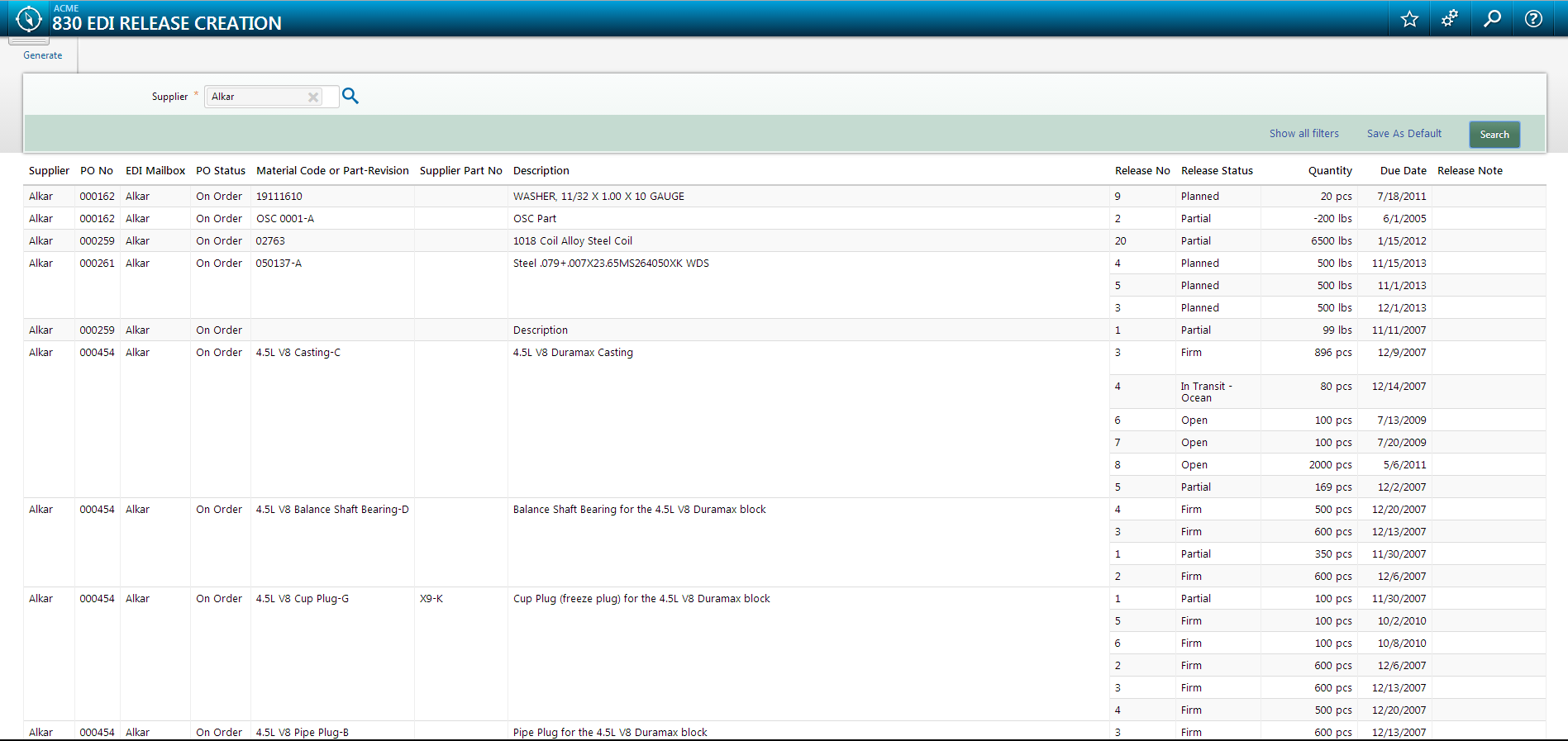


Рис. 4.4 Пример страницы после мигрирования

# Знакомство с системами контроля версий

В ходе прохождения практики была изучена система контроля версий Team Foundation Server.

Team Foundation Server (сокр. TFS) — продукт корпорации [Microsoft](http://ru.wikipedia.org/wiki/Microsoft), представляющий собой комплексное решение, объединяющее в себе [систему управления версиями](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8), сбор данных, построение [отчетов](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D0%BB%D0%B0%D0%B4), отслеживание статусов и изменений по проекту и предназначенное для совместной работы над проектами по [разработке программного обеспечения](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F). Данный продукт доступен как в виде отдельного приложения, так и в виде серверной платформы для [Visual Studio Team System](http://ru.wikipedia.org/wiki/Visual_Studio_Team_System) (VSTS).

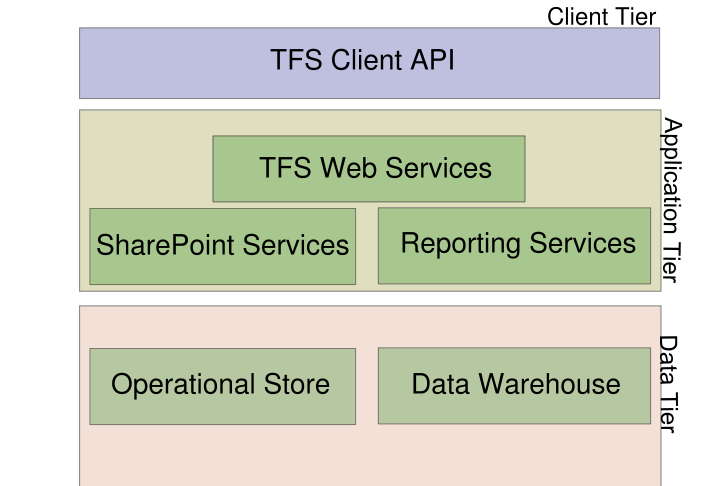


Рис 5.1 Архитектура TFS

Team Foundation Server работает по трёхуровневой архитектуре: клиентский уровень, прикладной уровень и уровень данных.

Клиентский уровень используется для создания и управления проектами, а также для доступа к хранимым и управляемым элементам проекта. На этом уровне TFS не содержит никаких [пользовательских интерфейсов](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), но предоставляет [веб-сервисы](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0), которые могут быть использованы клиентскими приложениями для самостоятельной интеграции в функциональность TFS. Эти веб-сервисы используются такими приложениями, как [Visual Studio Team System](http://ru.wikipedia.org/wiki/Visual_Studio_Team_System) для применения TFS в качестве серверной инфраструктуры хранилища информации или выделенного TFS управления приложениями, наподобие включенного приложения Team Foundation Client. Сами веб-сервисы находятся на прикладном уровне.

Прикладной уровень также включает в себя веб-портал и репозиторий (хранилище) документации, поддерживаемые [Windows SharePoint Services](http://ru.wikipedia.org/wiki/Windows_SharePoint_Services). Веб-портал, называемый Team Project Portal (портал командного проекта), выступает в роли центра взаимодействия для проектов, управляемых TFS. Репозиторий документов используется как для элементов проекта, так и для отслеживания ревизий (документирование изменений), а также для накопления и обработки данных и генерации отчетов.

Уровень данных, основывающийся в первую очередь на установленном [SQL Server 2005 Standard Edition](http://ru.wikipedia.org/wiki/Microsoft_SQL_Server), обеспечивает сервисы постоянного хранения данных для репозитория документов. Уровень данных и уровень приложений могут существовать на различных физических или виртуальных серверах при использовании [Windows Server 2003](http://ru.wikipedia.org/wiki/Windows_Server_2003) или более специализированных версий. Уровень данных не взаимодействует с клиентским уровнем напрямую, только через прикладной уровень.

# Заключение

В результате прохождения технологической практики были решены задачи, поставленные руководителем практики от предприятия ИООО «ЭПАМ Системз»: изучение стека технологий (MVC 4, Knockout.js, , Redis, C#5,.NET Framework 4.5, MVVM,JavaScript,SQL, VBScript). А также миграция веб страниц на новые технологии и по новым стандартам. Было проведено тестирование модуля.

Была изучена необходимая техническая документация, патентные и литературные источники, действующие стандарты, технические условия, положения и инструкции по разработке, наладке и эксплуатации, методы и технологии проектирования.

В соответствии с программой практики и индивидуальным заданием был осуществлен сбор, анализ, систематизация информации по теме задания и его непосредственное выполнение.

Также был подготовлен настоящий отчет и дневник с характеристикой руководителя от предприятия.

В ходе работы были получены навыки работы на предприятии, командной работы, общения с заказчиком. В ходе прохождения практики была изучена система контроля версий Team Foundation Server.