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: Козяков А.И.

Руководитель практики

от университета: Бурый Я.А.

Руководитель практики

от предприятия: Ведмич В.В.

МИНСК 2017

**СОДЕРЖАНИЕ**

1. ЗАДАНИЕ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ 3
2. СТРУКТУРА ПРЕДПРИЯТИЯ 4
3. Обзор стека технологий используемых на проекте 8
   1. JavaScript 8
      1. Структура языка 8
      2. Область применения 10
   2. TypeScript 11
   3. SQL 14
      1. Описание 16
      2. Операторы 17
      3. Преимущества и недостатки 18
   4. Сторонние компоненты 19
      1. Библиотека Knockout.js 19
4. Выполнение технического задания 21
   1. Определение требований 21
5. Знакомство с системами контроля версий 39
6. Заключение 41
7. **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 — ООО «ИксБи Софтваре». 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.

1. **СТРУКТУРА ПРЕДПРИЯТИЯ**

Компания XB Software предоставляет услуги полного цикла по разработке программного обеспечения под заказ, а также разрабатывает и реализует собственные ИТ продукты. Компания основана в 2008 году. Офис располагается в Минске. В компании работают сертифицированные ИТ-специалисты: разработчики веб- и мобильных приложений, дизайнеры, бизнес-аналитики, менеджеры проектов, QA инженеры, маркетологи и менеджеры по продажам, и другие.

Компания создала собственные продукты, в том числе JavaScript-библиотеку виджетов Webix UI library, которая позволяет создавать передовые веб-приложения на базе HTML5.

Технологическая экспертиза XB Software включает в себя:

* + - 1. React.js;
      2. Marionette.js;
      3. Webix;
      4. Node.js;
      5. PHP / Laravel;
      6. .NET;
      7. Java.

В цикле разработки используются следующие базы данных:

* 1. Oracle;
  2. MS SQL;
  3. MySQL;
  4. PostgreSQL;
  5. MongoDB;
  6. Microsoft Access;
  7. SQLite;
  8. Redis

Также широко используются облачные решения, такие как: Amazon, JBoss, WebLogic, WebSphere и другие.

В работе также используются кроссплатформенные технологии, такие как PhoneGap, Webix JavaScript UI library, которые позволяют создавать пользовательские и бизнес-приложения для любых отраслей.

Кроме того, команда XB Software имеет существенный опыт работы с геоданными:

* 1. Google;
  2. Bing;
  3. OpenStreetMap;
  4. Yandex;
  5. Nokia.

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

1. Привлечение клиентов и укрепление лояльности;
2. Электронная коммерция;
3. Управление жизненным циклом продукции;
4. Взаимодействие с партнерами;
5. Формирование отчетности и бизнес-аналитика;
6. Бюджетирование и планирование;
7. Управление проектами;
8. Ведение учета и управление ресурсами (ERP) и другие.

За 9+ лет работы компания успешно выполнила более 150 проектов для своих заказчиков, а также успешно реализовала собственные продукты для клиентов из более чем 65 стран мира. Большинство клиентов XB Software находятся в США, Канаде, Великобритании и Европейском Союзе и представляют различные отрасли экономики:

* 1. Промышленность;
  2. Финансы;
  3. Медицина;
  4. Образование;
  5. Социальные сети.

Компания постоянно растет и развивается. Об это свидетельствует отчет за последний год работы:

Рисунок 2.1. Отчет о работе компании за последние год

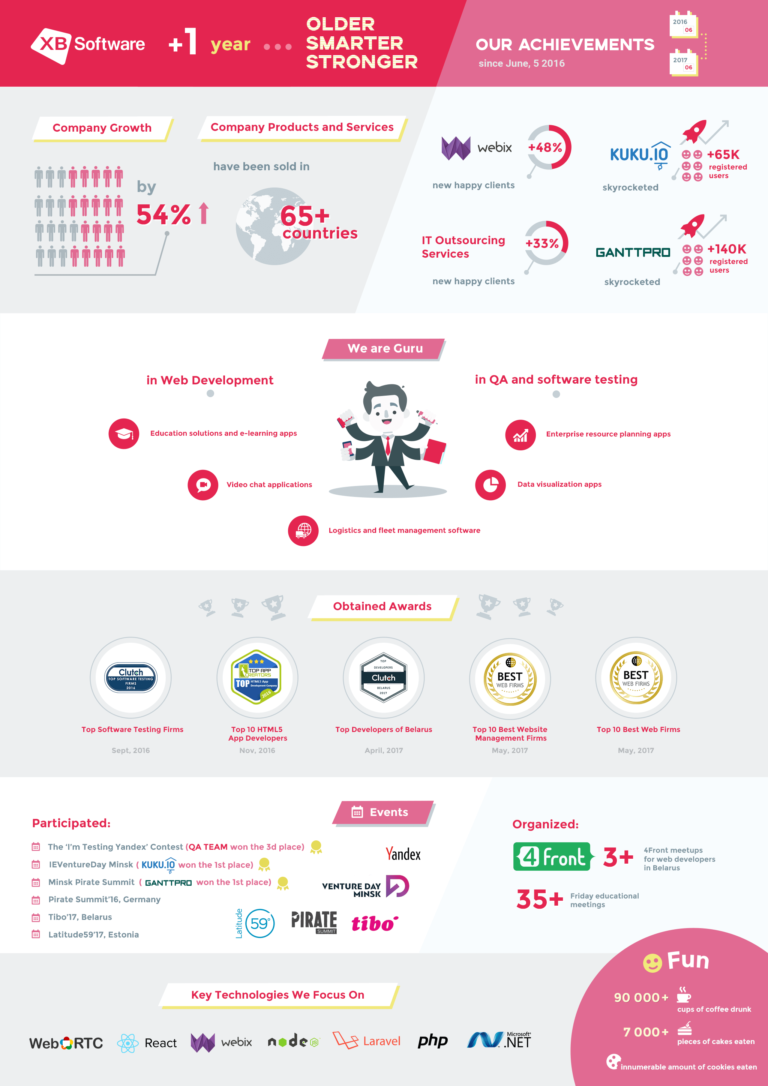
Необходимо отметить, что численность сотрудников постоянно увеличивается — это вызвано расширением компании. Регулярно организуются курсы по целенаправленному подбору персонала: как для сотрудников компании, так и для внешних кандидатов. Также огромное значение руководство компании уделяет вопросам повышения квалификации своих сотрудников. Это достигается путём организации тематических и проблемных курсов и семинаров для обмена опытом, а также пятничных лекция на которых всегда рассказывают новую информация из мира IT.

Рис. 2.2 Управленческая структура предприятия

Компания получила следующие награды в своей отрасли:

* 1. Clutch.co - Топ компаний-разработчиков из Беларуси, 2017;
  2. Топ 10 лучших компаний-разработчиков сайтов, 2017;
  3. Топ 10 лучших веб-компаний, 2017;
  4. Clutch.co - Топ компаний по тестированию ПО, 2016;
  5. Топ 10 компаний по разработке приложений на HTML5, 2016;
  6. Clutch.co - Топ 15 компаний-разработчиков из Беларуси, 2015.

1. Обзор стека технологий используемых на проекте
   1. JavaScript

Прототипно-ориентированный [сценарный](http://ru.wikipedia.org/wiki/%25D0%25A1%25D1%2586%25D0%25B5%25D0%25BD%25D0%25B0%25D1%2580%25D0%25BD%25D1%258B%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA) [язык программирования](http://ru.wikipedia.org/wiki/%25D0%25AF%25D0%25B7%25D1%258B%25D0%25BA_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D1%258F). Является [диалектом](http://ru.wikipedia.org/wiki/%25D0%2594%25D0%25B8%25D0%25B0%25D0%25BB%25D0%25B5%25D0%25BA%25D1%2582_(%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5)) 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/%25D0%259A%25D0%25BE%25D0%25BC%25D0%25BF%25D1%258C%25D1%258E%25D1%2582%25D0%25B5%25D1%2580%25D0%25BD%25D0%25B0%25D1%258F_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B0). Наиболее широкое применение находит в [браузерах](http://ru.wikipedia.org/wiki/%25D0%2591%25D1%2580%25D0%25B0%25D1%2583%25D0%25B7%25D0%25B5%25D1%2580) как язык сценариев для придания [интерактивности](http://ru.wikipedia.org/wiki/%25D0%2598%25D0%25BD%25D1%2582%25D0%25B5%25D1%2580%25D0%25B0%25D0%25BA%25D1%2582%25D0%25B8%25D0%25B2%25D0%25BD%25D0%25BE%25D1%2581%25D1%2582%25D1%258C) [веб-страницам](http://ru.wikipedia.org/wiki/%25D0%2592%25D0%25B5%25D0%25B1-%25D1%2581%25D1%2582%25D1%2580%25D0%25B0%25D0%25BD%25D0%25B8%25D1%2586%25D0%25B0).

Основные архитектурные черты: [динамическая типизация](http://ru.wikipedia.org/wiki/%25D0%2594%25D0%25B8%25D0%25BD%25D0%25B0%25D0%25BC%25D0%25B8%25D1%2587%25D0%25B5%25D1%2581%25D0%25BA%25D0%25B0%25D1%258F_%25D1%2582%25D0%25B8%25D0%25BF%25D0%25B8%25D0%25B7%25D0%25B0%25D1%2586%25D0%25B8%25D1%258F), слабая типизация, автоматическое управление памятью, [прототипное программирование](http://ru.wikipedia.org/wiki/%25D0%259F%25D1%2580%25D0%25BE%25D1%2582%25D0%25BE%25D1%2582%25D0%25B8%25D0%25BF%25D0%25BD%25D0%25BE%25D0%25B5_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5), функции как [объекты первого класса](http://ru.wikipedia.org/wiki/%25D0%259E%25D0%25B1%25D1%258A%25D0%25B5%25D0%25BA%25D1%2582_%25D0%25BF%25D0%25B5%25D1%2580%25D0%25B2%25D0%25BE%25D0%25B3%25D0%25BE_%25D0%25BA%25D0%25BB%25D0%25B0%25D1%2581%25D1%2581%25D0%25B0).

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/%25D0%25BF%25D0%25BE%25D0%25BB%25D1%258C%25D0%25B7%25D0%25BE%25D0%25B2%25D0%25B0%25D1%2582%25D0%25B5%25D0%25BB%25D1%258C-%25D0%25BD%25D0%25B5%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2581%25D1%2582). 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/%25D0%25A2%25D0%25BE%25D0%25B2%25D0%25B0%25D1%2580%25D0%25BD%25D1%258B%25D0%25B9_%25D0%25B7%25D0%25BD%25D0%25B0%25D0%25BA) компании [Oracle Corporation](http://ru.wikipedia.org/wiki/Oracle_Corporation).

* + 1. 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/%25D0%2597%25D0%25B0%25D1%2580%25D0%25B5%25D0%25B7%25D0%25B5%25D1%2580%25D0%25B2%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25BD%25D0%25BE%25D0%25B5_%25D1%2581%25D0%25BB%25D0%25BE%25D0%25B2%25D0%25BE) слова, [операторы](http://ru.wikipedia.org/wiki/%25D0%259E%25D0%25BF%25D0%25B5%25D1%2580%25D0%25B0%25D1%2582%25D0%25BE%25D1%2580_(%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5)), объекты, [регулярные выражения](http://ru.wikipedia.org/wiki/%25D0%25A0%25D0%25B5%25D0%25B3%25D1%2583%25D0%25BB%25D1%258F%25D1%2580%25D0%25BD%25D1%258B%25D0%25B5_%25D0%25B2%25D1%258B%25D1%2580%25D0%25B0%25D0%25B6%25D0%25B5%25D0%25BD%25D0%25B8%25D1%258F), 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/%25D0%25A4%25D1%2580%25D0%25B5%25D0%25B9%25D0%25BC_(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/%25D0%259C%25D0%25BE%25D0%25BD%25D0%25B8%25D1%2582%25D0%25BE%25D1%2580_(%25D1%2583%25D1%2581%25D1%2582%25D1%2580%25D0%25BE%25D0%25B9%25D1%2581%25D1%2582%25D0%25B2%25D0%25BE)),
* 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/%25D0%2598%25D0%25BD%25D1%2582%25D0%25B5%25D1%2580%25D1%2584%25D0%25B5%25D0%25B9%25D1%2581_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D1%258F_%25D0%25BF%25D1%2580%25D0%25B8%25D0%25BB%25D0%25BE%25D0%25B6%25D0%25B5%25D0%25BD%25D0%25B8%25D0%25B9) 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.
  + 1. 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.

* 1. TypeScript

Несомненно JavaScript хорош, но в нем есть ряд недостатков, особенно когда дело касается разработки крупного проекта. Именно этой целью и задали разработчики Microsoft - и вот уже в 2014 году миру был представлена первая версия TypeScript(далее TS).

В чем особенность данного языка? TS по сути является тем же JavaScript-ом, но, в отличии от последнего, в нем есть:

1. Строгая типизаци;
2. Интерфейсы;
3. Возможность компиляции в любую из версий JS.

3.2.1 Строгая типизация

Строгая типизация важна. Она позволяет сократить количество ошибок в коде еще на этапе разработки. Всегда приятно знать параметры какого типа нужно передавать в функцию. Мы значительно сокращаем время разработки, потому что нам не нужно заходить внутрь функции, чтобы понять какие же параметры в нее нужно передавать. Это также полезно, когда в команде появляется новые разработчик. Пирог вхождения в проект для него значительно сокращается, потому читаемость кода великолепна. Да и вероятность ошибки сделанной новичком значительно сокращается, что не может не радовать.

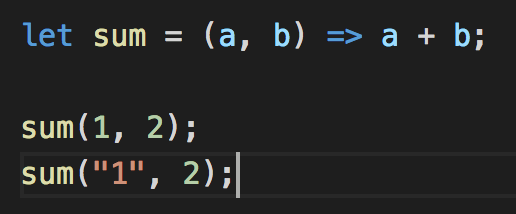
Напишем функцию сложения двух числе на JS:

Рисунок 3.2.1.1 Функция сложения двух чисел на JS

Как видно из примера она довольно проста, принимает два параметра и отдает их сумму. Чуть ниже на рисунке представлены вызовы данной функции. Как видно, в качестве аргумента функции мы можем передать переменную любого типа. Но если представить, что функция будет иметь больший размер и точно зависеть от типа данных, переданных на ее вход, то вероятность ошибки при вызове данной функции довольно велика.

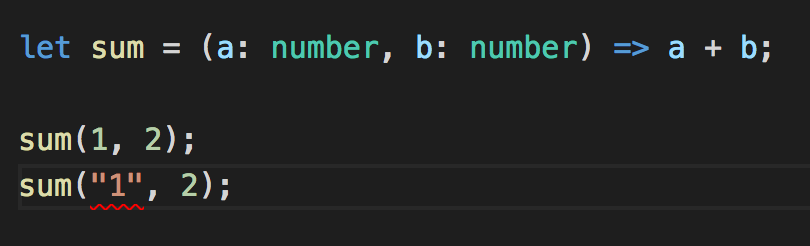
Рассмотрим пример написанный на TS:

Рисунок 3.2.1.1 Функция сложения двух чисел на TS

Как видно, к предыдущему примеру только дописали типы входных переменных. Но даже сейчас не компилирую данный файл, файловый редактор подчеркнул второй вызов функции, помечая не валидный вызов функции. По-моему это здорово. Когда часть работы при написании кода можно переложить на рабочую машину и сосредоточится над более важными вещами.

3.2.2 Интерфейсы

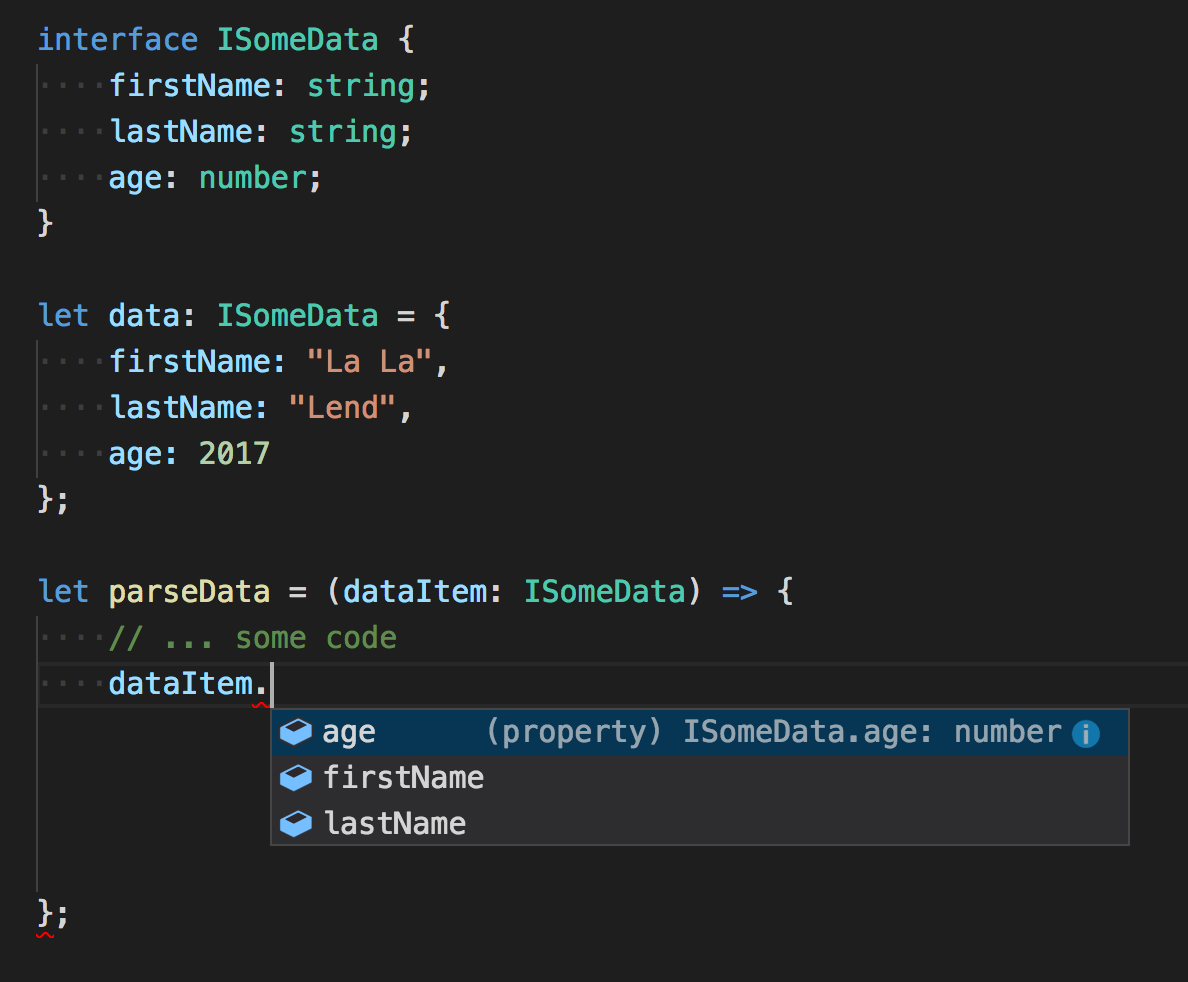
Второй важной характеристикой TS является наличие интерфейсов. Для чего же они нужны? В разработке очень часто работает с некоторыми структурами данных, а когда эту структуру данных можно описать и использовать во всем проекте, то легче становиться дышать всей команде. Приведем пример:

Рисунок 3.2.2.1 Пример использования интерфейса

Как видно из рисунка выше, создание и использование интерфейсов ничем не отличается от использования обычных типов, ведь они, по сути, и являются пользовательским типом данных.

Их можно свободно использовать в качестве типа для переменной или для типа аргумента функции. Самым большим плюсом является то, что, используя интерфейсы, мы помогаем текстовому редактору подсказывать нам все его свойства. На примере видно всплывающее окно текстового редактора, которое показывает все возможные свойства данного объекта.

3.2.2 Компиляция в JS

Как рассказывалось выше, браузер не понимает TS, но он понимает JS. Именно поэтому мы вынуждены компилировать один язык в другой. В последние два года идет прогрессивное обновление версий JavaScript. В него добавляются новые возможности, которые улучшают и повышают его функциональность. Но в тоже время, реальность такова, что мы вынуждены поддерживать поддержку новых возможностей JS в старых браузерах.

В данном вопросе TS умеет компилировать даже самую последнюю версию JS в более ранние версии. Это здорово, ведь как только проект перестает поддерживать старый браузер, мы можем безболезненно, поменяв один параметр компиляции, получить наш код в более современной версии.

* 1. SQL

SQL (structured query language — «структурированный язык запросов») — [формальный](http://ru.wikipedia.org/wiki/%25D0%25A4%25D0%25BE%25D1%2580%25D0%25BC%25D0%25B0%25D0%25BB%25D1%258C%25D0%25BD%25D1%258B%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA) непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной [реляционной базе данных](http://ru.wikipedia.org/wiki/%25D0%25A0%25D0%25B5%25D0%25BB%25D1%258F%25D1%2586%25D0%25B8%25D0%25BE%25D0%25BD%25D0%25BD%25D1%258B%25D0%25B5_%25D0%25B1%25D0%25B0%25D0%25B7%25D1%258B_%25D0%25B4%25D0%25B0%25D0%25BD%25D0%25BD%25D1%258B%25D1%2585), управляемой соответствующей системой управления базами данных ([СУБД](http://ru.wikipedia.org/wiki/%25D0%25A1%25D0%25A3%25D0%2591%25D0%2594)). SQL основывается на [исчислении кортежей](http://ru.wikipedia.org/wiki/%25D0%2598%25D1%2581%25D1%2587%25D0%25B8%25D1%2581%25D0%25BB%25D0%25B5%25D0%25BD%25D0%25B8%25D0%25B5_%25D0%25BA%25D0%25BE%25D1%2580%25D1%2582%25D0%25B5%25D0%25B6%25D0%25B5%25D0%25B9).

SQL является, прежде всего, [информационно-логическим языком](http://ru.wikipedia.org/wiki/%25D0%2598%25D0%25BD%25D1%2584%25D0%25BE%25D1%2580%25D0%25BC%25D0%25B0%25D1%2586%25D0%25B8%25D0%25BE%25D0%25BD%25D0%25BD%25D1%258B%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA), предназначенным для описания, изменения и извлечения данных, хранимых в [реляционных базах данных](http://ru.wikipedia.org/wiki/%25D0%25A0%25D0%25B5%25D0%25BB%25D1%258F%25D1%2586%25D0%25B8%25D0%25BE%25D0%25BD%25D0%25BD%25D1%258B%25D0%25B5_%25D0%25B1%25D0%25B0%25D0%25B7%25D1%258B_%25D0%25B4%25D0%25B0%25D0%25BD%25D0%25BD%25D1%258B%25D1%2585). SQL можно назвать [языком программирования](http://ru.wikipedia.org/wiki/%25D0%25AF%25D0%25B7%25D1%258B%25D0%25BA_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D1%258F), при этом он не является [Тьюринг-полным](http://ru.wikipedia.org/wiki/%25D0%259F%25D0%25BE%25D0%25BB%25D0%25BD%25D0%25BE%25D1%2582%25D0%25B0_%25D0%25BF%25D0%25BE_%25D0%25A2%25D1%258C%25D1%258E%25D1%2580%25D0%25B8%25D0%25BD%25D0%25B3%25D1%2583), 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/%25D0%2591%25D0%25B0%25D0%25B7%25D0%25B0_%25D0%25B4%25D0%25B0%25D0%25BD%25D0%25BD%25D1%258B%25D1%2585) 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.

* + 1. Описание

Язык SQL представляет собой совокупность

* операторов,
* инструкций,
* и вычисляемых функций.
  + 1. Операторы

Согласно общепринятому [стилю программирования](http://ru.wikipedia.org/wiki/%25D0%25A1%25D1%2582%25D0%25B8%25D0%25BB%25D1%258C_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D1%258F), операторы (и другие зарезервированные слова) в SQL обычно рекомендуется писать [прописными буквами](http://ru.wikipedia.org/wiki/%25D0%2597%25D0%25B0%25D0%25B3%25D0%25BB%25D0%25B0%25D0%25B2%25D0%25BD%25D1%258B%25D0%25B5_%25D0%25B1%25D1%2583%25D0%25BA%25D0%25B2%25D1%258B).

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/%25D0%259F%25D1%2580%25D0%25B5%25D0%25B4%25D1%2581%25D1%2582%25D0%25B0%25D0%25B2%25D0%25BB%25D0%25B5%25D0%25BD%25D0%25B8%25D0%25B5_(%25D0%25B1%25D0%25B0%25D0%25B7%25D1%258B_%25D0%25B4%25D0%25B0%25D0%25BD%25D0%25BD%25D1%258B%25D1%2585)), 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/%25D0%25A2%25D1%2580%25D0%25B0%25D0%25BD%25D0%25B7%25D0%25B0%25D0%25BA%25D1%2586%25D0%25B8%25D1%258F_(%25D0%25B8%25D0%25BD%25D1%2584%25D0%25BE%25D1%2580%25D0%25BC%25D0%25B0%25D1%2582%25D0%25B8%25D0%25BA%25D0%25B0)) (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.
  + 1. 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/%25D0%2594%25D0%25BE%25D0%25BA%25D1%2583%25D0%25BC%25D0%25B5%25D0%25BD%25D1%2582%25D0%25BE%25D0%25BE%25D0%25B1%25D0%25BE%25D1%2580%25D0%25BE%25D1%2582) [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/%25D0%259A%25D0%25BE%25D0%25B4%25D0%25B4,_%25D0%25AD%25D0%25B4%25D0%25B3%25D0%25B0%25D1%2580), [Кристофер Дейт](http://ru.wikipedia.org/wiki/%25D0%2594%25D0%25B5%25D0%25B9%25D1%2582,_%25D0%259A%25D1%2580%25D0%25B8%25D1%2581%25D1%2582%25D0%25BE%25D1%2584%25D0%25B5%25D1%2580) 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.
  1. Сторонние компоненты
     1. 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.

1. Выполнение технического задания
   1. 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/%25D0%259A%25D0%25BE%25D0%25BC%25D0%25BF%25D0%25BE%25D0%25BD%25D0%25B5%25D0%25BD%25D1%2582%25D0%25BD%25D0%25BE-%25D0%25BE%25D1%2580%25D0%25B8%25D0%25B5%25D0%25BD%25D1%2582%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25BD%25D0%25BE%25D0%25B5_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5). Является специфичной формой «[инверсии управления](http://ru.wikipedia.org/wiki/%25D0%2598%25D0%25BD%25D0%25B2%25D0%25B5%25D1%2580%25D1%2581%25D0%25B8%25D1%258F_%25D1%2583%25D0%25BF%25D1%2580%25D0%25B0%25D0%25B2%25D0%25BB%25D0%25B5%25D0%25BD%25D0%25B8%25D1%258F)» ([англ.](http://ru.wikipedia.org/wiki/%25D0%2590%25D0%25BD%25D0%25B3%25D0%25BB%25D0%25B8%25D0%25B9%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA) Inversion of control), где изменение порядка связи осуществляется путём получения необходимой зависимости.

Условно, если [объекту](http://ru.wikipedia.org/wiki/%25D0%259E%25D0%25B1%25D1%258A%25D0%25B5%25D0%25BA%25D1%2582_(%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5)) нужно получить доступ к определенному [сервису](http://ru.wikipedia.org/w/index.php?title=%25D0%25A1%25D0%25B5%25D1%2580%25D0%25B2%25D0%25B8%25D1%2581_(%25D0%25B0%25D1%2580%25D1%2585%25D0%25B8%25D1%2582%25D0%25B5%25D0%25BA%25D1%2582%25D1%2583%25D1%2580%25D0%25B0_%25D1%2581%25D0%25B8%25D1%2581%25D1%2582%25D0%25B5%25D0%25BC)&action=edit&redlink=1), объект берет на себя ответственность за доступ к этому сервису: он или получает прямую[ссылку](http://ru.wikipedia.org/wiki/%25D0%25A1%25D1%2581%25D1%258B%25D0%25BB%25D0%25BA%25D0%25B0_(%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5)) 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/%25D0%25A4%25D0%25B0%25D0%25B9%25D0%25BB_%25D0%25BA%25D0%25BE%25D0%25BD%25D1%2584%25D0%25B8%25D0%25B3%25D1%2583%25D1%2580%25D0%25B0%25D1%2586%25D0%25B8%25D0%25B8), без изменений в объектах, которые этот сервис используют. Это особенно полезно в [юнит-тестировании](http://ru.wikipedia.org/wiki/%25D0%25AE%25D0%25BD%25D0%25B8%25D1%2582-%25D1%2582%25D0%25B5%25D1%2581%25D1%2582%25D0%25B8%25D1%2580%25D0%25BE%25D0%25B2%25D0%25B0%25D0%25BD%25D0%25B8%25D0%25B5), потому что вставить реализацию «[заглушки](http://ru.wikipedia.org/wiki/Mock-%25D0%25BE%25D0%25B1%25D1%258A%25D0%25B5%25D0%25BA%25D1%2582)» 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/%25D0%25A1%25D1%2580%25D0%25B5%25D0%25B4%25D0%25B0_%25D1%2580%25D0%25B0%25D0%25B7%25D1%2580%25D0%25B0%25D0%25B1%25D0%25BE%25D1%2582%25D0%25BA%25D0%25B8_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25BD%25D0%25BE%25D0%25B3%25D0%25BE_%25D0%25BE%25D0%25B1%25D0%25B5%25D1%2581%25D0%25BF%25D0%25B5%25D1%2587%25D0%25B5%25D0%25BD%25D0%25B8%25D1%258F), 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 };

}

}

}

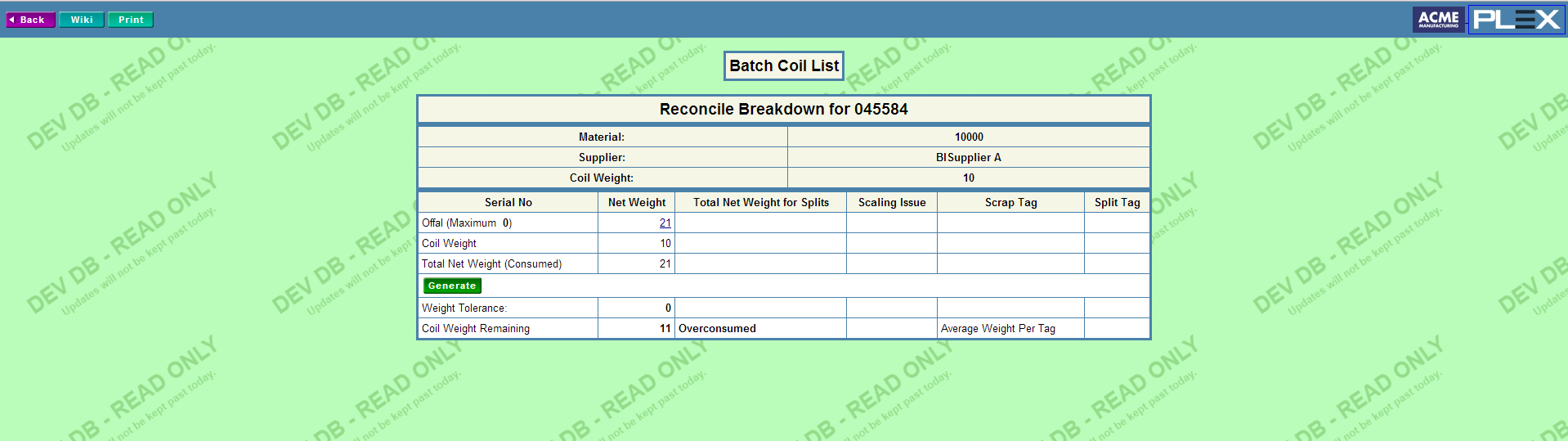


Рис. 4.1 Пример страницы до мигрирования

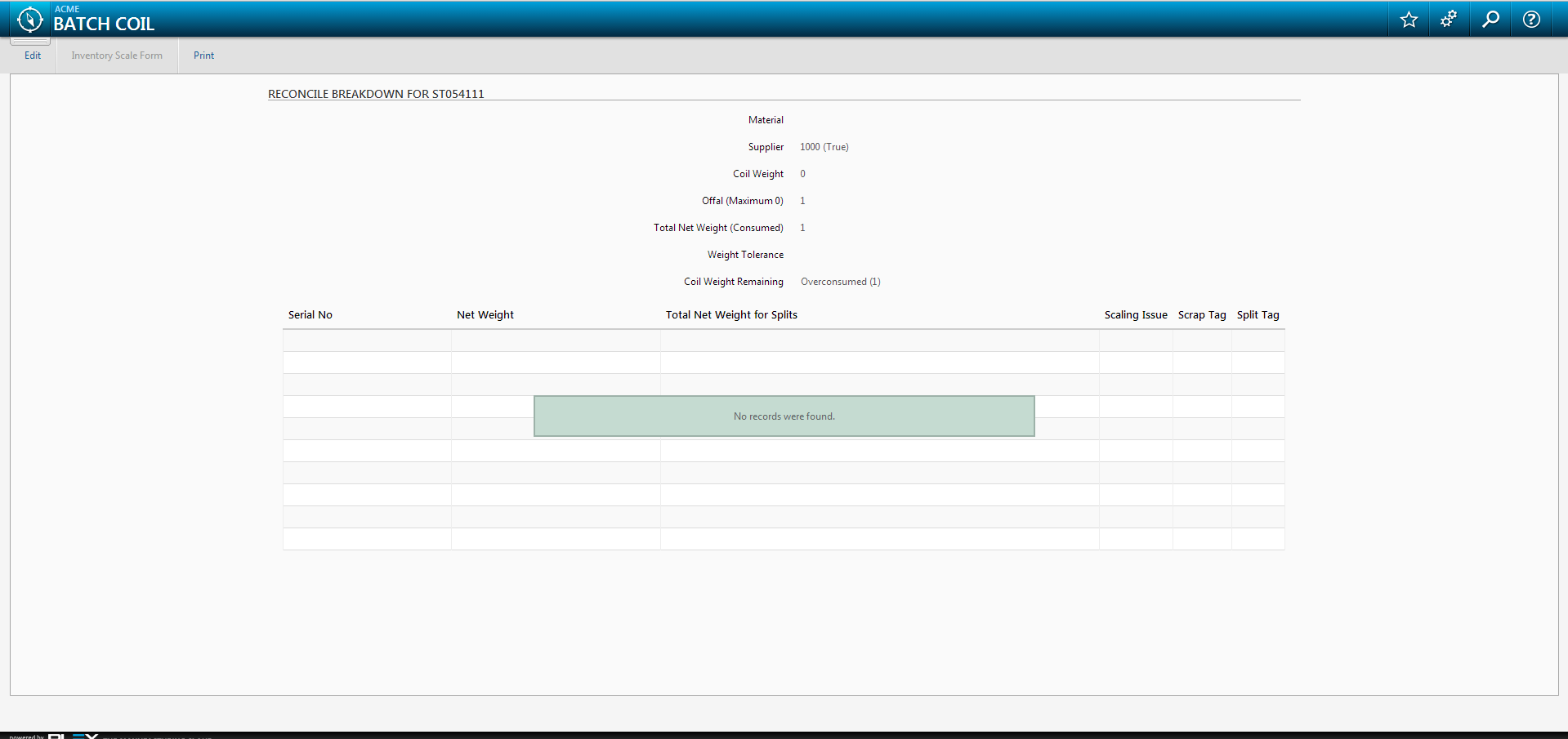


Рис. 4.2 Пример страницы после мигрирования

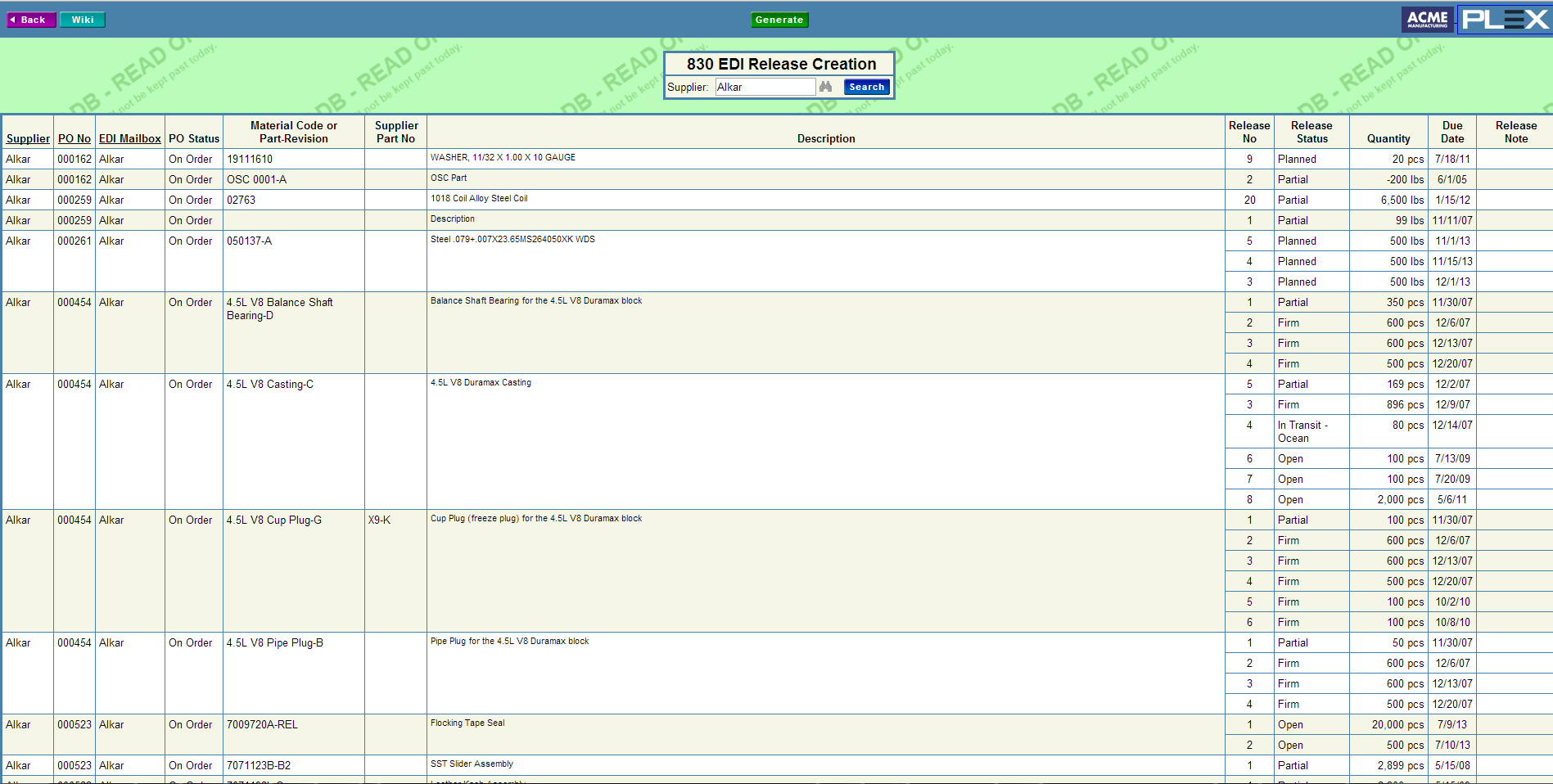


Рис. 4.3 Пример страницы до мигрирования

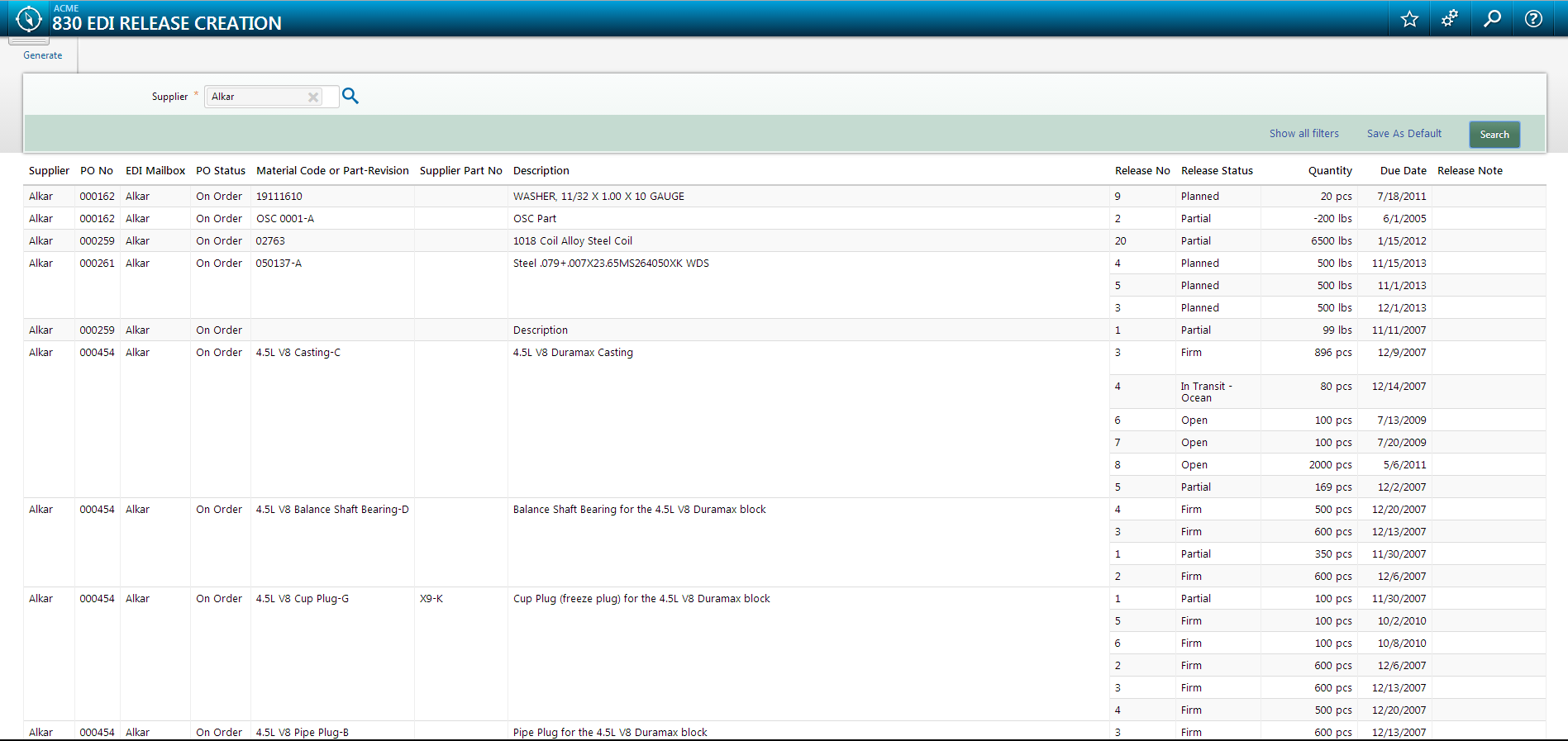


Рис. 4.4 Пример страницы после мигрирования

1. 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/%25D0%25A1%25D0%25B8%25D1%2581%25D1%2582%25D0%25B5%25D0%25BC%25D0%25B0_%25D1%2583%25D0%25BF%25D1%2580%25D0%25B0%25D0%25B2%25D0%25BB%25D0%25B5%25D0%25BD%25D0%25B8%25D1%258F_%25D0%25B2%25D0%25B5%25D1%2580%25D1%2581%25D0%25B8%25D1%258F%25D0%25BC%25D0%25B8), сбор данных, построение [отчетов](http://ru.wikipedia.org/wiki/%25D0%2594%25D0%25BE%25D0%25BA%25D0%25BB%25D0%25B0%25D0%25B4), отслеживание статусов и изменений по проекту и предназначенное для совместной работы над проектами по [разработке программного обеспечения](http://ru.wikipedia.org/wiki/%25D0%25A0%25D0%25B0%25D0%25B7%25D1%2580%25D0%25B0%25D0%25B1%25D0%25BE%25D1%2582%25D0%25BA%25D0%25B0_%25D0%25BF%25D1%2580%25D0%25BE%25D0%25B3%25D1%2580%25D0%25B0%25D0%25BC%25D0%25BC%25D0%25BD%25D0%25BE%25D0%25B3%25D0%25BE_%25D0%25BE%25D0%25B1%25D0%25B5%25D1%2581%25D0%25BF%25D0%25B5%25D1%2587%25D0%25B5%25D0%25BD%25D0%25B8%25D1%258F). 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).

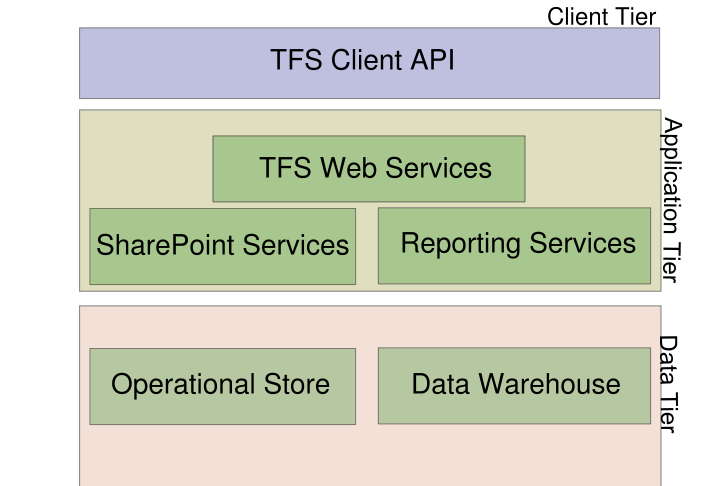


Рис 5.1 Архитектура TFS

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/%25D0%2598%25D0%25BD%25D1%2582%25D0%25B5%25D1%2580%25D1%2584%25D0%25B5%25D0%25B9%25D1%2581_%25D0%25BF%25D0%25BE%25D0%25BB%25D1%258C%25D0%25B7%25D0%25BE%25D0%25B2%25D0%25B0%25D1%2582%25D0%25B5%25D0%25BB%25D1%258F), но предоставляет [веб-сервисы](http://ru.wikipedia.org/wiki/%25D0%2592%25D0%25B5%25D0%25B1-%25D1%2581%25D0%25BB%25D1%2583%25D0%25B6%25D0%25B1%25D0%25B0), 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.

1. 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.