**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Саратовский государственный технический университет**

**имени Гагарина Ю.А.»**

Факультет Международный факультет прикладных информационных технологий

Специальность Информационные системы и технологии

Кафедра Прикладные информационные технологии

**ДИПЛОМНАЯ РАБОТА**

«Разработка программного обеспечения»

Выполнил студент группы ИСТ-51

Игнатченко Т. С.

Руководитель работы к т.н., ассистент

каф. ПИТ Шварц А. Ю.

Допущен к защите

Протокол № 00 от 00 июня 2015 г.

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долинина О.Н.

Саратов 2015

# Введение

На этапе становления бизнеса и в условиях конкуренции необходимо захватить внимание клиента при первом его контакте с контентом в сети интернет. Для привлечения клиента, следует наглядно и быстро показать преимущества продукта, услуги или компании в целом. Наиболее мощным инструментом для привлечения внимания клиента и взаимодействия с ним является интерактивная презентация.

Однако изготовление качественной интерактивной презентации в студии дизайна требует вложения больших средств и времени и наличия людей с соответствующими навыками. Что касается обычных презентаций, то они не позволяют взаимодействовать с пользователем и отслеживать его активность в рамках взаимодействия с презентацией.

Целью дипломной работы является создание презентации, которая позволит

* Создать интерактивную презентацию продукта и разместить в сети интернет
* Интегрировать презентацию с сайтом
* Встроить в презентацию форму обратной связи
* Отследить сценарий поведения пользователя
* Произвести A/B тестирование для увеличения эффективности презентации
* Преобразовать презентацию к стандартам видеорекламы для проведения рекламных кампаний
* Собирать данные пользователей для оптимизации рекламной кампании и увеличения конверсии

Для достижения целей работы необходимо выполнить следующие задачи:

* провести анализ рынка интернет рекламы;
* составить массив требований к системе;
* проанализировать аналоги;
* спроектировать архитектуру;
* написать алгоритмы поведения системы;
* спроектировать интерфейс системы;
* спроектировать базу данных системы;
* разработать подсистему взаимодействия с БД;
* разработать подсистему сбора статистики
* разработать подсистему, предоставляющую публичные методы
* разработать подсистему экспорта в сторонние форматы
* разработать подсистему отчетов
* разработать конечный вариант системы

# Обзор существующих решений

## BannerFlow

BannerFlow позволяет быстро без дополнительных навыков создавать рекламные кампании, использующие HTML5, которые корректно отображаются на большинстве экранах и браузерах, интегрируются с партнерскими и рекламными сетями.

Основа BannerFlow – WYSIWYG редактор (см. Рис.1 редактор BannerFlow), позволяющий без навыков программирования создавать баннеры, компилируемые в HTML5+CSS3 код, впоследствии отображаемые на веб сайтах с помощью встраиваемого кода, основанного на JavaScript, который можно экспортировать из редактора. Если реклама отображается браузером, который не поддерживает HTML5, баннер компилируется в GIF изображение.

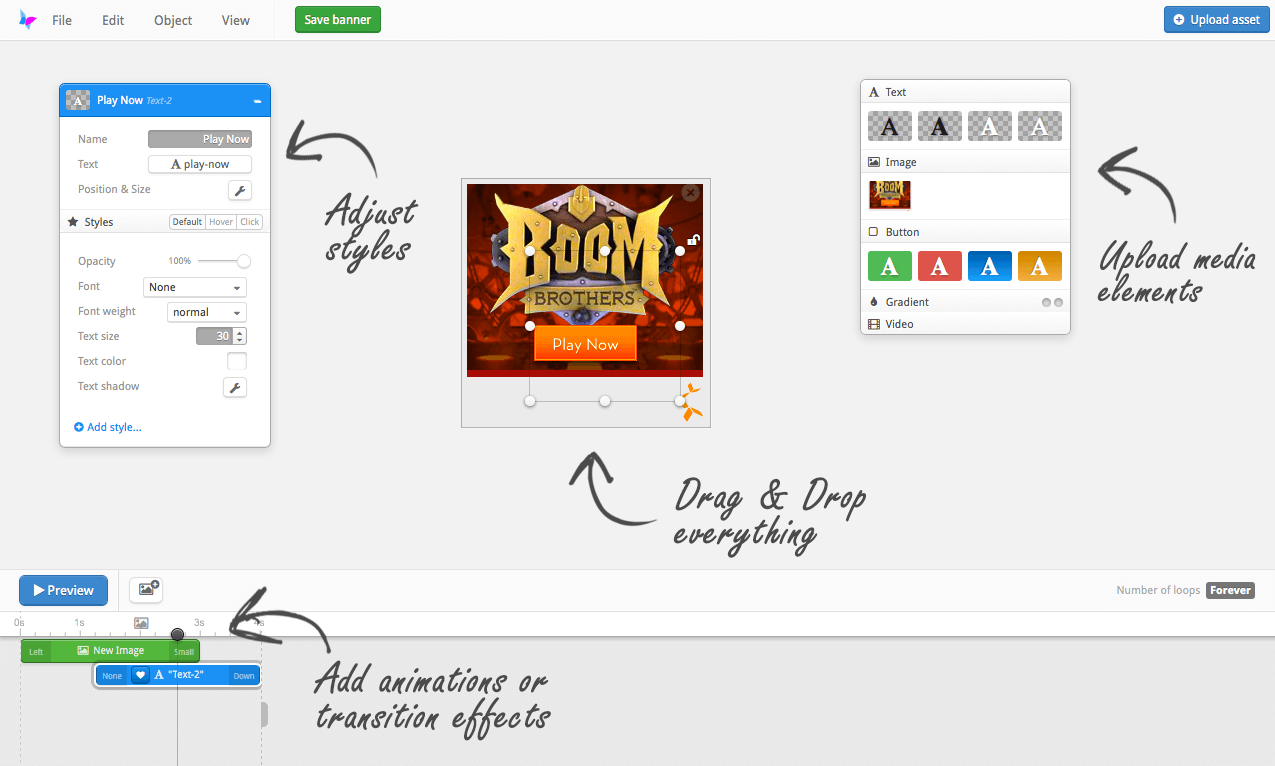


Рис. 1 редактор BannerFlow

Платформа позволяет

* Сохранять все загруженный медиа-контент в форматах PNG, SVG, JPEG в библиотеку и использовать его впоследствии при создании различных баннеров.
* Применять различные стили к компонентам, которые компилируются в CSS код. Например, цвет, тип границы, эффекты текста.
* Применять эффекты к событиям мыши.
* Присутствует шкала прогресса, на которой можно располагать очередность появления элементов, применять стили и эффекты анимации. Редактирование осуществляется в режиме drag’n’drop.
* Определять локализацию баннера. В зависимости от региона пользователя, отображается соответствующий текст, кроме того, возможно изменять стиль, элементы и анимацию.
* Подгружать динамический контент с любого сервера посредством XML, который может в себе содержать текст на любом языке, числа, ссылки на изображения. Таким образом, можно отображать последние заголовки новостей, или список товаров, которые подходят именно тому пользователю, который зашел на страницу.
* По окончании разработки, система генерирует код, который можно встроить в более чем 60 рекламных сетей.
  + Adconion
  + Adform
  + Adroll
  + DoubleClick DFA
  + EyeWonder
  + LinkShare
  + Google AdWords
  + Facebook Adverts
  + и т. д.
* Встроить поддержку Google Analytics для слежения за кликами пользователей
* Автоматически экспортировать баннеры в различные CMS
* Собирать полную статистику по показам и взаимодействиям
* Проводить A/B тестирование (представлено на рисунке 2)

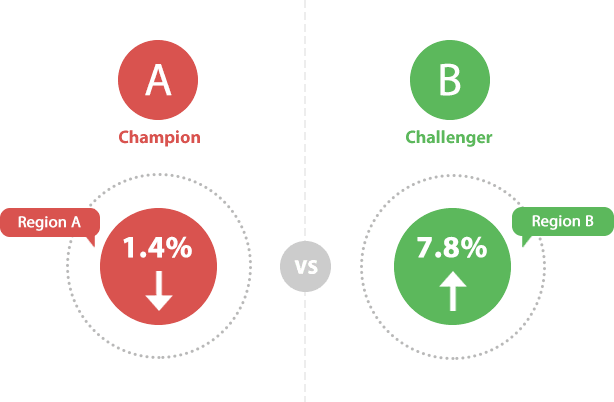


Рис. 2 A/B тестирование

BannerFlow - мощный и простой инструмент создания баннеров, использующий последние технологии и гибко подстраивающийся под различные среды, в которых запускается экземпляр баннера, но он заточен именно под рекламу и маркетинг, кроме того, на нем нельзя создавать большие презентации, использующие видео.

## Prezi

Prezi – онлайн приложение для создания презентаций, которое позволяет создавать онлайн презентации, которые могут быть воспроизведены на любом совместимом устройстве. В Prezi можно создавать, распространять и редактировать презентации как онлайн (окно редактора представлено на рисунке 3), так и в случае отсутствия подключения к сети интернет. Все презентации, которые были созданы, могут быть открыты в приложении Prezi, независимо от того, где и когда они создавались. Prezi предоставляет пакеты как для отдельных пользователей, так и для команд с разграничением прав и уровней доступа. Prezi также доступна как приложение для windows, которое синхронизируется с онлайн аккаунтом.

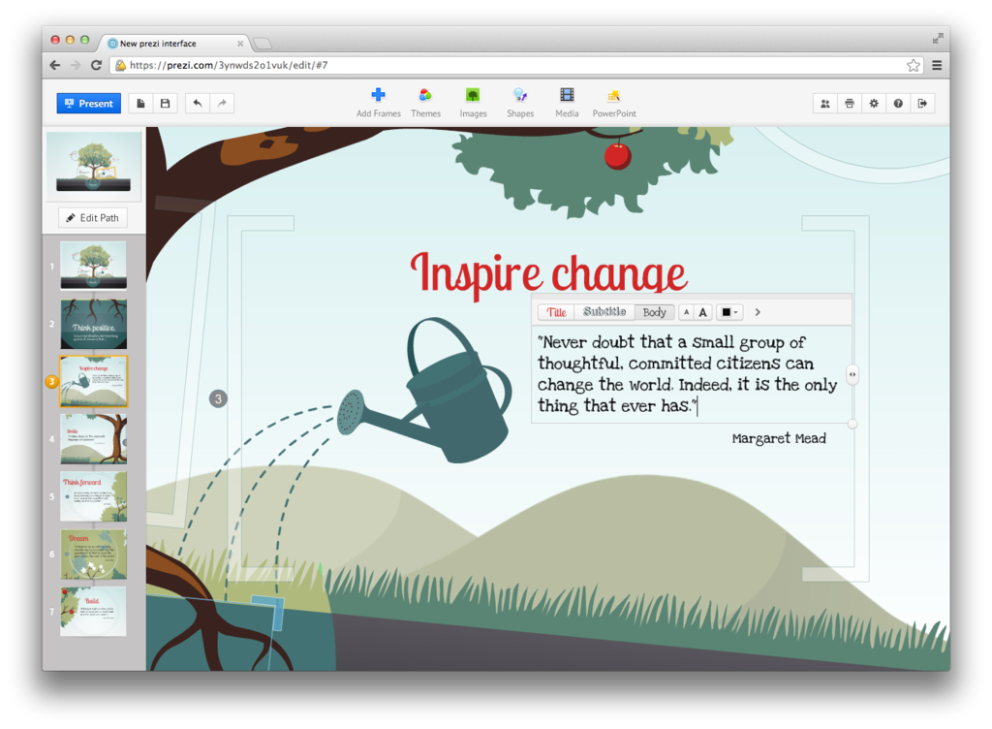


Рис. 3 окно редактора Prezi

Возможность Prezi

* Облачное приложение с возможностью коллаборации при создании презентаций
* Возможность одновременного редактирования презентации несколькими пользователями
* Импорт медиа объектов
* Масштабирование и панорамирование на одном слайде
* Общие папки
* Библиотека презентаций
* Размещение данных, как на компьютере, так и в облаке
* Приложение для iPad и iPhone
* Настройки прав доступа

Prezi является одним из популярнейших продуктов для создания презентаций, предоставляет широчайшие возможности редактирования, включая 3d и 2.5d переходы как между слайдами, так и внутри них, но отсутствует контроль за действиями пользователей при просмотре презентации. Структура презентаций исключительно линейна, кроме того, используется собственное приложение для воспроизведения.

# Практическая реализация системы

## Функциональные возможности

Сервис позволит:

* Создать интерактивную презентацию продукта и разместить в сети интернет. Создание презентации осуществляется с помощью стороннего редактора, который имеет доступ к API сервиса, который предоставляет методы, позволяющие загружать сущность презентации вместе со всеми медиа элементами в БД и файловое хранилище на сервере
* Интегрировать презентацию с сайтом.

Пользователю предоставляется возможность связать элементы управления на презентации с функционалом хостинг сайта, осуществлять вызов rest методов, посредством взаимодействия с презентацией

* Встроить в презентацию форму обратной связи.
* Отследить сценарий поведения пользователя.

Сервис отслеживает все действия пользователя, сохраняя данные в БД

* Произвести A/B тестирование для увеличения эффективности презентации.
* Преобразовать презентацию к стандартам видеорекламы для проведения рекламных кампаний. Существует описание стандартных рекламных интерфейсов, принятых IAB, к которым осуществляется преобразования в случае потребности в экспорте презентации на рекламную площадку.
* Формировать отчеты

Сервис формирует отчеты, как в виде таблиц, так и в виде интерактивных графиков, позволяя гибко настроить выборку с помощью простого интерфейса. Имеется возможность экспортировать в форматы PNG, JPEG, CSV, XLS, PDF и распечатать отчет прямо из браузера.

## Входные и выходные данные

Система принимает на вход следующие данные

* Данные аккаунта пользователя
* Сущность презентации
* Данные по активности пользователя
* Данные браузера пользователя
  + IP
  + Имя браузера
  + Локаль
  + Язык
* Запросы на формирование статистики
  + Дата начала
  + Дата окончания
  + Шаг времени
  + Номер презентации
* Данные для формирования отчета
  + Дата начала
  + Дата окончания
  + Фильтр
  + Тип отчета

Система формирует следующие выходные данные:

* ID загруженных сущностей
* Список презентаций
* Сущность каждой презентацию
* Собранная статистика
* Отчеты, сформированные по запросу пользователя

## Схема работы приложения

Пользователь авторизуется в системе, переходит на модуль создания презентацию, с помощью которого создает ее и отправляет на сервер. В процессе создания, с сервера с помощью api методов можно получить все медиа-элементы, загруженные пользователем до этого. Если в презентации содержатся новые медиа-элементы, они будут сохранены в библиотеке. При загрузке видео элемента, они автоматически трансформируется в формат Webm по причине наибольшей поддержке его в браузерах. После создания можно сгенерировать код для встраивания на панели управления. Также, есть возможность экспортировать в унифицированные форматы, разработанные IAB, а именно, VAST и VMAP. После добавления кода на страницу система начнет получать данные о взаимодействии пользователей с этой презентации. Можно настроить email уведомления по достижении каких-либо показателей, также, автоматически поступает уведомление модератору, у которого есть право удалить презентацию при наличии в ней запрещенного материала. Вся полученная информация доступна в виде настраиваемых графиков и табличных отчетов, которые можно экспортировать в форматы PNG, JPEG, CSV, PDF, скопировать в буфер обмена или распечатать прямо из браузера с отключением отображения дополнительных элементов на странице. При редактировании презентации есть возможность сохранить оба варианта и провести a/b тестирование, настроив периодичность показов того или иного варианта. Общая схема приложения представлена на рисунке 4.

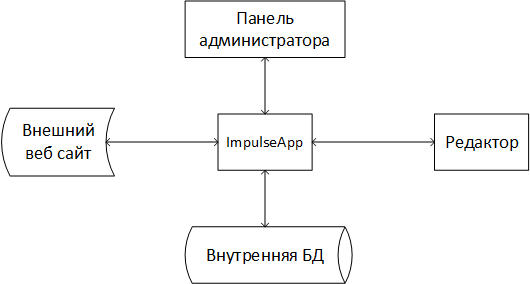


Рис. 4 общая схема приложения

## Архитектура системы

При разработке архитектуры основополагающим был принцип расширяемости и слабосвязанности. Так как в системе есть модули, взаимодействующие как с внешними системами, так и между собой, решено было построить взаимодействие между внутренними модулями на основе протокола SOAP, а взаимодействие с внешними системами, в том числе с редактором на основе REST.

На высшем уровне абстракции, модули представляют собой классическую трехслойную архитектуру. Схема представлена на рисунке 5.

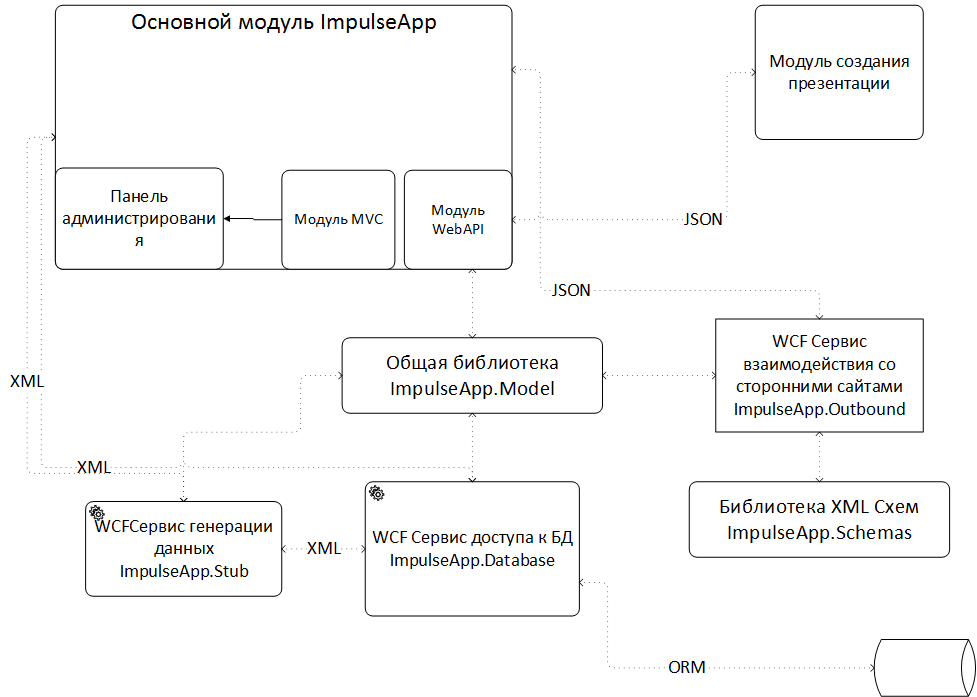


Рис. 5 Схема модулей и формат сообщений

* Модули, хранящие модели данных
  + ImpulseApp.Model
  + ImpulseApp.Schemas
* Слой доступа к данным
  + ImpulseApp.Database
* Слой бизнес логики
  + ImpulseApp
    - Подмодуль ImpulseApp.Api
    - Подмодуль ImpulseApp.Mvc
  + ImpulseApp.Outbound
  + ImpuleApp.Stub
  + ImpulseApp.Stats
* Слой представлений
  + Модуль JS mpls.statistics.helper.chartjs.js
  + Модуль JS mpls.statistics.helper.highchart.js
  + Модуль JS mpls.statistics.js
  + Модуль JS mpls.video.play.js

Далее, приводится более подробное описание каждого компонента архитектуры

### Слой взаимодействия с БД

Доступ к данным осуществляется с помощью ORM Entity Framework. Основной модуль представляет собой WCF сервис с набором методов. Данный модуль ссылается на библиотеку классов ImpulseApp.Model. Каждый класс является одновременно как DTO (Data Transfer Object), так и моделью ORM системы. Это избавляет от необходимости написания маппингов при получении моделей из сервиса.

При разработке БД использовался метод Code First, означающий, что были разработаны модели данных с необходимыми данными, а на основе них автоматичсеки была сгенерирована база данных. Список моделей следующий: (более подробное описание приведено в разделе «Описание базы данных»).

* Модели составляющих презентации.
  + **VideoUnit**

модель, инкапсулирующая информацию о видеофайле

* + **SimpleAdModel**

основная модель, инкапсулирующая данные о презентации

* + **AdState**

модель, характеризующая каждое состояние презентации

* + **NodeLink**

модель, описывающая граф переходов между состояниями объектов

* Модели статистики
  + **AdSession**

основная модель данных статистики

* + **Activity**

Модель хранящая данные о взаимодействии в рамках одного состояния

* + **Click**

Модель, хранящая данные о клике

Все модели являются полями в классе контекста, унаследованного от класса DbContext. В этом классе хранится описание всех таблиц, а также настроена конфигурация мапинга в базу данных, настроены параметры целостности.

Для обеспечения целостности БД и ее согласованности с ORM моделью, используется технология миграций. Весь процесс настроен в классе Сonfiguration, который унаследован от класса DbMigrationsConfiguration<ImpulseApp.Database.ImpulseContext>.

Так как для DbContext настроена «ленивое» связывание объектов, можно связать модель с коллекцией моделей, которые ссылаются на первую по внешнему ключу. Однако это влечет за собой ошибки припередаче данных по SOAP протоколу. Чтобы не писать дополнительные DTO, был добавлен класс ReferencePreservingDataContractFormatAttribute, который можно применить как атрибут к методу, формирующему xml сообщение. Этот класс реализует интерфейс IOperationBehavior и формирует специально настроенный объект типа DataContractSerializer для обхода ошибок при стандартной настройке сериализации.

Главным классом модуля ImpulseApp.Database является DBService, реализующий интерфейс IDBService. Список методов интерфейса приведен в таблице 1.

|  |  |
| --- | --- |
| Сигнатура метода | Описание |
| IEnumerable<SimpleAdModel> GetUserAds(string UserId) | Возвращает список всех презентаций по id пользователя |
| SimpleAdModel GetAdByUrl(string url); | Возвращает презентацию по короткому url |
| SimpleAdModel GetAdById(int id); | Возвращает презентацию по Id |
| IEnumerable<AdSession> GetSessionsByAdId(int AdId); | Возвращает статистику по Id презентации |
| string SaveAdSession(AdSession model, bool proceedToDB = true); | Сохраняет сессию, а рамках которой пользователь осуществлял взаимодействие с презентацией. Флаг proceedToDB здесь и далее сигнализирует о завершении транзакции. |
| IEnumerable<Activity> GetActivityBySessionId(int SessionId); | Возвращает все объекты типа Activity по Id сессии |
| string SaveActivity(Activity model, bool proceedToDB = true); | Сохраняет объект типа Activity в БД |
| IEnumerable<Click> GetClicksByActivityId(int ActivityId); | Получает все клики, сделанные в рамках сессии на одном слайде презентации |
| string SaveClick(Click model, bool proceedToDB = true); | Сохраняет данные о клике в БД |
| string SaveVideo(VideoUnit model); | Сохраняет данные о видео в БД |
| IEnumerable<VideoUnit> GetUserVideo(string UserName); | Получает все видео, загруженные пользователем |
| VideoUnit GetVideoById(string UserName, int Id); | Получает видео по Id и имени пользователя |
| AdState GetStateByAdIdAndVideoId(int adId, int videoId, string stateName); | Получает слайд презентации по id рекламы, id главного видео и названию |

Таблица 1 методы сервиса ImpulseApp.Database

Все методы могут генерировать исключение типа ImpulseBaseException. От него наследуются другие исключения более высокого уровня. Все исключения имеют поле \_message, в которое передается сообщение об ошибке. Все сообщения об ошибках хранятся в виде строковых ресурсов в файле ExceptionText.resx в модуле ImpulseApp.Models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Адрес | Параметры | Тип | Возвращаемые значения | Описание |
| /api/ad/create | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | model | SimpleAdModel | Смотри раздел  “Описание моделей” | | POST | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | status | number | Код HTTP ответа. В случае отсутствия ошибок = 201 (created). | | url | string | Полный адрес созданной рекламы | | Сохраняет в БД созданную рекламу |
| /api/ad | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | url | string | короткий url рекламы | | GET | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | model | SimpleAdModelDTO |  | | Получение рекламы по короткому url |
| /api/upload/video | MultiFormData | POST | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | video | VideoUnit |  | | Загрузка видео файла. Возвращается сформированный объект |
| /api/video/all | - | GET | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | videos | Array<VideoUnit> |  | | Возвращает все видео авторизованного пользователя |
| /api/video/ids | - | GET | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | ids | Array<Number> | Массив id видео данного пользователя | | Возвращает все id видео авторизованного пользователя |
| /api/video/{id} | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | id | int | короткий url рекламы | | GET | |  |  |  | | --- | --- | --- | | Переменная | Тип | Описание | | video | VideoUnit | Массив id видео данного пользователя | | Возвращает видео по id |

Таблица 2 api методы

### Слой бизнес логики

#### Модуль ImpulseApp.Api

В настоящий момент, этот модуль используется главным образом для связывания редактора и системы обработки. Он предоставляет несколько методов, представленных в таблице 2. Каждый метод взаимодействует с помощью следующих моделей данных

**SimpleAdModel**

Главная модель, описывающая презентацию. Поля представлены в таблице 3

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| Name | string | Название рекламы |
| HtmlStartSource | html string | Начальный html слой |
| HtmlEndSource | html string | Конечный html слой |
| AdStates | Array<AdState> | Массив слайдов |
| StateGraph | Array<NodeLink> | Список ребер |

Таблица 3 поля модели SimpleAdModel

Прототип на JS

var SimpleAdModel = function () { }

SimpleAdModel.prototype = {

Name: '',

HtmlStartSource: '',

HtmlEndSource: '',

AdStates: new Array(),

StateGraph: new Array()

};

**AdState**

Сущность видеослайда. Поля представлены в таблице 4

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| Name | string | Название слайда |
| EndTime | int | До какой секунды воспроизводится видео. Обрабатывается, если IsFullPlay==true |
| ChainedHtml | html string | Слой, показываемый при окончании воспроизведения |
| IsFullPlay | boolean | Флаг, указывающий проигрывать, ли видео до конца. |
| IsStart | boolean | Флаг, указывающий на начальный слайд |
| VideoUnitId | int | Id видео |
| IsEnd | boolean | Флаг, указываюший на завершающий слайд. Свойство может быть истинным как у 0 слайдов (при циклическом воспроизведении), так и у нескольких |

Таблица 4 поля класса AdState

Прототип на JS:

var AdState = function () { }

AdState.prototype = {

EndTime: -1,

ChainedHtml: '',

IsFullPlay: true,

IsStart: false,

VideoUnitId: 0,

IsEnd: false,

Name: ''

}

**NodeLink**

Сущность, описывающая переходы между слайдами. Список полей представлен в таблице 5

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| V1 | int | Id видео, с которого осуществляется переход |
| V2 | int | Id видео, на который осуществляется переход |
| T | int | Время видео, на которое осуществляется переход |

Таблица 5 поля класса NodeLink

Прототип на JS:

var NodeLink = function () { };

NodeLink.prototype = {

V1: 0,

V2: 0,

T: 0

};

**VideoUnit**

Поля представлены в таблице 6

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| Id | int |  |
| DateLoaded | date | Дата загрузки |
| Name | string | Имя видео, данное пользователем (По умолчанию DefaultName, пока не релизовано) |
| GeneratedName | string | Сгенерированное имя видео  (Хэш код от случайного Guid) |
| UserName | string | Логин пользователя, загрузившего видео |
| Length | int | Длина видео (в байтах) |
| FullPath | string | Путь до файла (виден только в редакторе, при воспроизведении, в целях безопасности, используется BLOB) |
| MimeType | string | MIME тип |

Таблица 6 поля класса VideoUnit

**SimpleAdModelDTO**

Упрощенная сущность SimpleAdModel без циклических зависимостей для публичной видимости. Поля представлены в таблице 7

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| HtmlStartSource | date | Начальный Html слой |
| Name | string | Название презентации |
| States | Array<AdStateDTO> | Массив слайдов |

Таблица 7 поля класса SimpleAdModelDTO

**AdStateDTO**

Упрощенная сущность слайдов для публичной видимости. Поля класса представлены в таблице 8

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип** | **Описание** |
| Name | string | Название слайда |
| EndTime | int | До какой секунды воспроизводится видео. Обрабатывается, если IsFullPlay==true |
| ChainedHtml | html string | Слой, показываемый при окончании воспроизведения |
| IsFullPlay | boolean | Флаг, указывающий проигрывать, ли видео до конца. |
| IsStart | boolean | Флаг, указывающий на начальный слайд |
| VideoUnitId | int | Id видео |
| IsEnd | boolean | Флаг, указываюший на завершающий слайд. Свойство может быть истинным как у 0 слайдов (при циклическом воспроизведении), так и у нескольких |

Таблица 8 поля класса AdStateDTO

#### Модуль ImpulseApp.Mvc

Данный модуль содержит в себе логику обработки запросов пользователя к формам и элементам управления на панели администрирования. Кроме, того данный модуль содержит в себе компонент, осуществляющий формирование отчетов, экспорта, преобразования из моделей данных конкретных отчетов в унифицированные и обратно, осуществляет генерацию ошибок пользователю, которые произошли в вызываемых модулях. Осуществляет маршрутизацию запросов, в том числе на модуль ImpulseApp.Api. Схема приведена на рисунке 6.

**Компонент формирования отчетов.**

Данный компонент выполняет преобразование из модели данных конкретного отчета в обобщенную модель данных отчета, которая выводится по запросу пользователя в обобщенном представлении. Все ответы возвращаются в формате Json, так что все запросы выполняются асинхронно, соответствующие функции запрограммированы в JavaScript модулях.

**Компонент экспорта отчетов**

Данный модуль выполняет преобразование Html разметки в формат Pdf. Нужная разметка при этом формируется на стороне клиента, так что данный модуль также выполняет удаление нежелательного кода и символов, что делает атаку сложной. Основа модуля – библиотека iTextSharp. В компонент также можно передать название CSS файла, который будет применен к экспортируемому html.

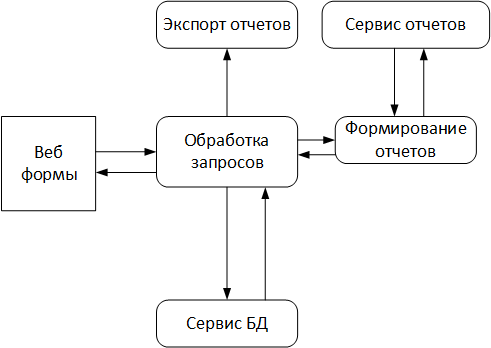


Рис. 6 Декомпозиция модуля Mvc

#### Модуль ImpulseApp.Outbound

Данный модуль выполняет преобразование стандартной модели презентации в рекламу формата VAST3, которую можно впоследствии импортировать в большинство рекламных сервисов. Все схемы хранятся в библиотеке ImpulseApp.Schemas, где также на основе этих схем при сборке приложения автоматически формируются C# классы.

#### Модуль ImpulseApp.Stat

Данный модуль возвращает данные из БД по запросу пользователя. Содержатся несколько шаблонных запросов, к которым можно применить параметры, например, дату начала и конца. Все запросы, в том числе на формирование графиков подразумевают возможность получения статистики по нескольким презентациям для сравнения. Ниже приведен список шаблонов

## Используемые технологии

Основные критерии для выбора технологий были следующие

* Простая разработка веб сервисов
* Возможность разработки слабосвязанного многомодульного приложения в рамках единой среды
* Наличие хостингов, которые поддерживают высоконагруженные системы
* Наличие документации и активного сообщества

Всем вышеописанным критериям удовлетворяет платформа .NET, которая и была выбрана в качестве основы приложения.

**Описание платформы**

.NET Framework – фреймворк, разработанный Microsoft и запускающийся преимущественно на Microsoft Windows. Он включает в себя большую библиотеку классов FCL и обеспечивает совместимость между несколькими языками программирования. Программы, написанные под .NET запускаются в специальной среде, называемой CLR, виртуальной машине, предоставляющей такие возможности, как обеспечение безопасности, управление памятью, обработка исключений. CLR и FCL вместе составляют .NET Framework.

CLR (общеязыковая исполняющая среда) можно представить как посредника, управляющего кодом во время исполнения. Она управляет памятью, выполнением потоков, выполнением кода, верифицирует безопасность кода, управляет компиляцией и другими системными службами. Эти особенности являются неотъемлемой частью управляемого кода, который выполняется в рамках общеязыковой среды выполнения.

Что касается безопасности, управляемым компонентам присваиваются различные степени безопасности, в зависимости от нескольких факторов, включая их источник (интернет, корпоративная сеть, локальный компьютер). Это означает, что управляемому компоненту может быть запрещено или, наоборот, разрешено выполнять некие действия, например, доступ к файлам, реестру и другим важным элементам.

Управляющая среда также обеспечивает надежность выполнения кода за счет реализации спецификации CTS (Стандартная система типов). CTS гарантирует безопасность типов, используемых в коде.

Фреймворк также увеличивает производительность разработки, позволяя создавать библиотеки, которые можно использовать в нескольких языках программирования, запускать на различных операционных системах, включая mac и Linux. Наряду с DLL, также присутствует поддержка COM компонентов.

Среда увеличивает производительность. Управляемый ею код никогда не интерпретируется. Технология, именуемая JIT (just in time) позволяет компилировать часть кода в нативный и выполнять его, только тогда, когда требуется часть функционала, описанная этим кодом.

Библиотека классов решает основные часто встречающиеся проблемы при разработке кода, инкапсулируя часто используемый разработчиками функционал.

Обобщая, можно привести следующие достоинства и особенности .NET Framework

* Объектно-ориентированное программирование

Как .NET Framework, так и используемый в разработке язык C# изначально основаны на принципах и концепциях объектно-ориентированного программирования

* Библиотека классов

Входящая в поставку библиотека предоставляет широкие возможности для разработчика, позволяет ему сократить свой код, используя уже разработанные алгоритмы и решения.

* Независимость от языка программирования

Неважно, на каком языке было написано приложение, будь то C++, J#, F#, Basic, C#, оно будет скомпилировано в промежуточный код. Таким образом, обеспечивается взаимодействие модулей, написанных на разных языках

* Поддержка динамических веб сайтов
* Эффективный доступ к данным в БД через ADO.NET
* Разделение кода

В .NET взамен стандартных библиотек DLL использована концепция сборок, в рамках которой проекты имеют доступ к public и protected коду других проектов

* Повышенная безопасность
* Поддержка веб служб

NET Framework можно использовать для разработки следующих приложений:

* Консольные приложения
* Приложения Windows Forms
* Приложения Windows Presentation Foundation
* Приложения ASP.NET
* Приложения Windows Communication Foundation
* Веб службы ASP.NET
* Бизнес процессы Windows Workflow Foundation

**Обоснование выбора платформы**

Прежде всего, выбор продиктован предыдущим опытом разработки, знанием основных технологий и фреймворков. Также, немаловажным является факт наличия большого сообщества и книг, в которых содержится абсолютно вся информация обо всех компонентах платформы, описаны решения часто встречающихся проблем. Также, ключевой недостаток NET Framework – отсутствие обратной совместимости в данном случае не играет роли, так как разработка начинается «с нуля».

**Язык программирования C#**

C# - язык программирования, возникший 15 лет назад и созданный под влиянием C++ и Java, позднее Haskell и других функциональных языков. Язык основан на стандартах ECMA и ISO. Крайняя версия C# 5.0 была выпущена в 2012 году. Стандарты ECMA, нашедшие реализацию в C#

* Язык и его расширения реализуют проверку исключений, выход за пределы массива, строгую типизацию, проверку неуничтоженных переменных и их утилизацию, динамическую сборку мусора.
* С# является объектно-ориентированным языком
* Язык предназначен для разработки программных компонентов, пригодных для развертывания в распределенных средах.
* Важна возможность конвертации в другой язык программирования исходного кода, написанного на C#, также, важен низкий входной порог для программистов, имеющий опыт работы с другими языками, особенно, C, C++.
* Поддержка интернационализация.
* C# пригоден для разработки приложений для встраиваемых систем, а также для разработки как для сложных операционных систем, так и для выполняющих специфические функции.
* Также, C# должен экономно расходовать ресурсы системы, на которой запущено написанное на данном языке программирования программное обеспечение, также, однако, приложения, написанные на языке C#, не могут конкурировать по производительности с приложениями, разработанными с использованием более низкоуровневых языков.

Язык C# очень быстро развивается в соответствии с трендами в программировании, что можно увидеть в таблице 1

|  |  |  |
| --- | --- | --- |
| Версия | Дата релиза | Новые особенности |
| C# 2.0 | Ноябрь 2005 | * Обобщенные типы * Разделенные типы * Анонимные методы * Итераторы * Типы Nullable * Геттеры и Сеттеры * Делегаты * Статические классы |
| C# 3.0 | Ноябрь 2007 | * Косвенно типизированные локальные переменные * Инициализаторы объектов и коллекций * Автоматически реализуемые свойства * Анонимные типы * Методы расширения * Лямбда выражения * Выражения запросов * Дерево выражений * Разделенные методы |
| C# 4.0 | Апрель 2010 | * Динамическое связывание * Именованные и необязательные аргументы * Ковариация и контравариация в универсальных шаблонах * Встриваемые типы взаимодействия |
| C# 5.0 | Август 2012 | * Асинхронные методы и ключевые слова await, async * Информация о вызывающем объекте |
| C# 6.0 | 2015 | * Компилятор как сервис * Импортирование статических классов в пространство имен * Фильтры исключений * await в блоке catch\finally * Автоматическая иниализация свойств * Стандартные значения для только get свойств * null-Условный оператор “?.” * Интерполяция строк * Определение свойств и методов в lambda выражениях * Оператор nameof * Инициализация Dictionary |

Таблица 9 развитие языка C#

Отличительными особенностями языка являются

* Переносимость
* Строгая типизация
* Мета программирование
* Методы и функции
* Свойства (get; set;)
* Пространства имен
* Доступ к памяти

Все блоки, в которых используется прямой доступ к памяти, должны быть помечены как небезопасные

* Только обрабатываемые исключения
* Полиморфизм
* Функциональное программирование

**Обоснование выбора языка**

Основным критерием выбора C# является наличие большего опыта программирования на нем, знание основных библиотек и функций, нежели на остальных языках, поддерживаемых платформой .NET

**Фреймворк ASP.NET**

Основной модуль приложения разработан в рамках фреймворка ASP.NET

ASP.NET (Active Server Pages) – фреймворк с открытым исходным кодом для создания динамических веб приложений. Изначально, ASP.NET был известен как WebForms, хотя, это компонент верхнего уровня стека технологий (Рисунок 4 стек технологий ASP.NET WebForms). В рамках создания этой платформы, была предпринята попытка скрыть непосредственную работу с HTTP и HTML за моделированием пользовательского интерфейса, как иерархии контролируемых сервером объектов. Каждый объект хранил и контролировал свое состояние через запросы, формировал HTML представление на клиентском браузере и связывал взаимодействие с HTML элементом с обработчиком на сервере. Основной идеей было создать ощущение при работе с WebForms, что программист работает с обычными формами Windows. Программистам не надо было думать о независимых HTTP запросах и ответах, работа облегчалась надстройкой в виде хранящих свое состояние объектов пользовательского интерфейса.

Недостатки ASP.NET WebForms

* Большой размер объектов, передаваемых через механизм View State
* Сложность в контроле над жизненным циклом страниц
* Ограниченный контроль над объектами через HTML
* Сложность в автоматическом тестировании

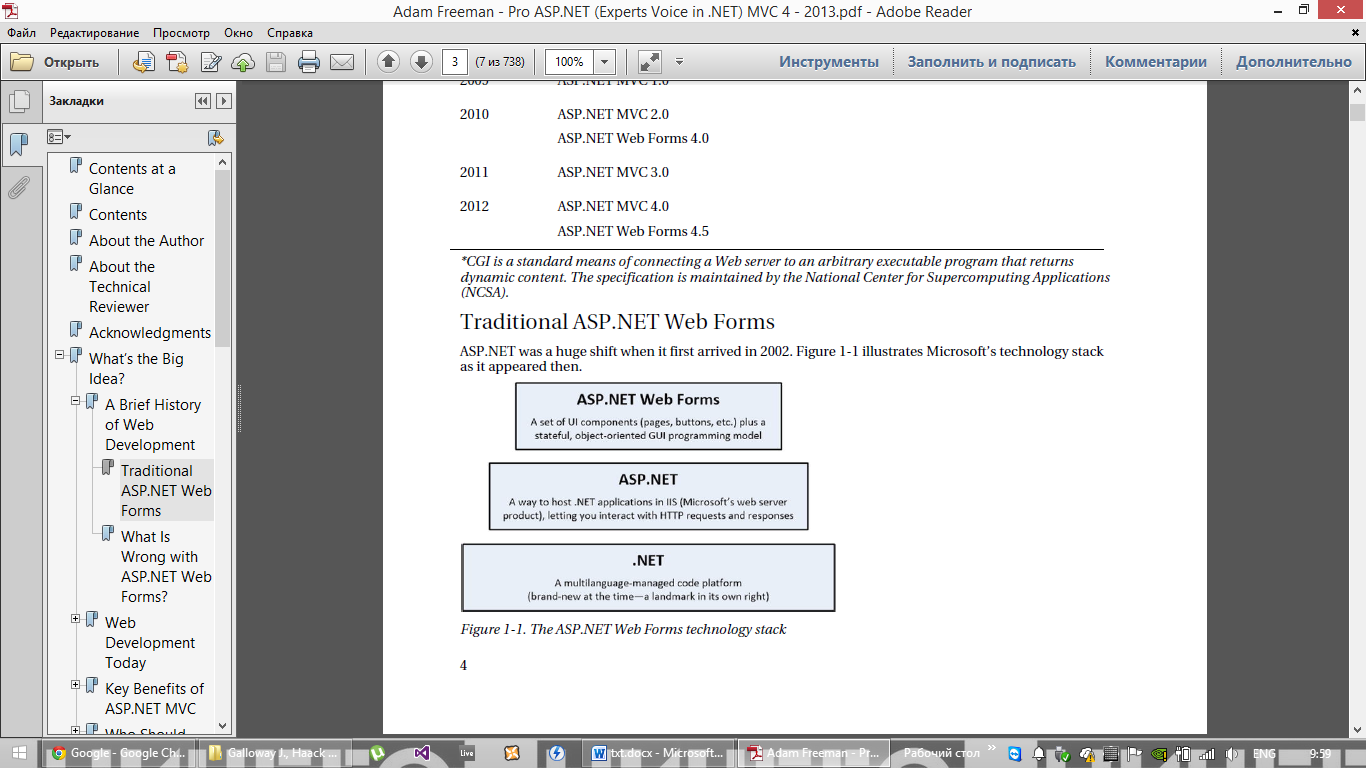


Рис. 7 стек технологий ASP.NET WebForms

Эти проблемы были решены с появлением ASP.NET MVC и ASP.NET Web Api, а решение с нестандартными CSHTML страницами было решено появлением движка Razor. MVC и WebApi являются слабосвязанными архитектурами, что сильно упростило написание автоматических тестов. Также, в настоящее время ASP.NET полностью интегрирован с новейшими технологиями .NET, такими как Entity Framework, LINQ, Async Task и т.д. Среди расширений ASP.NET стоит выделить

* ASP.NET MVC

Фреймворк, который использует паттерн model-view-controller для построения приложений

* ASP.NET Ajax

Модули, обеспечивающие простую имплементацию асинхронных веб страниц в рамках разработке в среде .NET

* ASP.NET Razor

технология, позволяющая разрабатывать веб страницы, используя html код и C# конструкции

* ASP.NET WebApi API

технология, позволяющая разрабатывать приложения в рамках REST архитектуры

Для разработки базового функционала панели администрирования использована технология ASP.NET MVC.

**Паттерн MVC**

Данный паттерн подразумевает разделение приложение на три взаимодействующие части: модель, представление и контроллер. Модель, центральный компонент системы, является точкой соприкосновения приложения и предметной области. Модель напрямую контролирует данные, логику и правила поведения приложения. Контроллер принимает входные данные и формирует на их основе команды, посылаемые в представление или модель. Представление может быть любым отображением информации, обрабатываемой приложением, например, диаграмма или таблица. Взаимодействие элементов показано на рисунке 5.

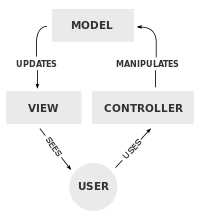


Рис. 8 типичное взаимодействие компонентов паттерна

Хотя, паттерн был создан в 1978 году, он приобрел широкую популярность сейчас при разработке веб приложений по следующим причинам

* Взаимодействие пользователя с приложением, основанном на данном паттерне, происходит по следующей схеме: пользователь совершает какое-либо действие, в ответ, приложение изменяет модель данных на сервере и возвращает пользователю представление, основанное на уже измененной модели. Затем, процесс повторяется вновь. Это очень хорошо подходит к веб приложениям, построенных на HTTP запросах и ответах.
* В рамках веб приложения необходимо использовать множество различных технологий: языки разметки и стилей, язык для описания логики работы сервера, СУБД и т.д., которые разделены на несколько слоев. Это очень хорошо укладывается в рамки концепции MVC.

В ASP.NET MVC Framework состоит из множества независимых компонентов, например, система маршрутизации, движок представлений, фабрика контроллеров. Взаимодействие между компонентами происходит в рамках соглашения, называемого convention over naming (Соглашения по именам), когда принадлежность представления, обрабатывающему его контроллеру и методу внутри него определяется по названию папки и html файла, соответствующего названию контроллер и методу в нем.

ASP.NET MVC позволяет чистую стандартизированную разметку представлений. В него встроена система вспомогательных HTML методов, формирующих стандартный XHTML код, которые в отличие от громоздких тегов WebForms, компактны и позволяют легко настроить css стили и html атрибуты. Сгенерированные страницы не содержат объекты состояния (View State), так что они на десятки килобайт меньше, чем те же страницы WebForms. Каждое представление привязано к модели данных, которая обрабатывается в соответствующем методе контроллера. В самой HTML разметке легко получить доступ к полям и методам объекта, привязанного к представлению.

Для того чтобы вызвать метод контроллера, пользователь должен ввести в адресной строке название контроллера без постфикса “Controller”, затем, название метода, после перечислить параметры метода как параметры GET запроса. Однако, ASP.NET MVC позволяет гибко настраивать маршрутизацию. Каждый обработчик маршрута привязан к какому-либо паттерну. Адрес запроса подставляется к каждому из паттернов, начиная с первого и обрабатывает первое попавшееся совпадение. Система маршрутизации сама не знает ничего о контроллерах и методах, она только приводит URL адрес к унифицированному виду и направляет запрос дальше по конвейеру. Например, на вход поступает адрес вида «http://site.com/abc/cde», таким образом, переменной controller присвоится значение «abc», а переменной address значение «cde», затем, эти переменные отправляются в следующий обработчик, называемый Controller Factory – класс, реализующий интерфейс IControllerFactory, который уже вызывает соответствующий метод в соответствующем контроллере.

Рассмотрим более подробно процесс обработки запросов в ASP.NET

* **Инициализация приложения**

Как и любое приложение ASP.NET, при старте запускается метод в файле Global.asax Application\_Start() (как можно увидеть на рисунке 6).

В этом методе можно добавить маршруты в статическую коллекцию RouteTable.RouteCollection. Можно написать собственный обработчик адресов, реализуя интерфейс IRouteHandler, или, использовать стандартный MvcRouteHandler. Также здесь можно указать фабрику контроллеров, созданную вручную, путем реализации интерфейса IControllerFactory.

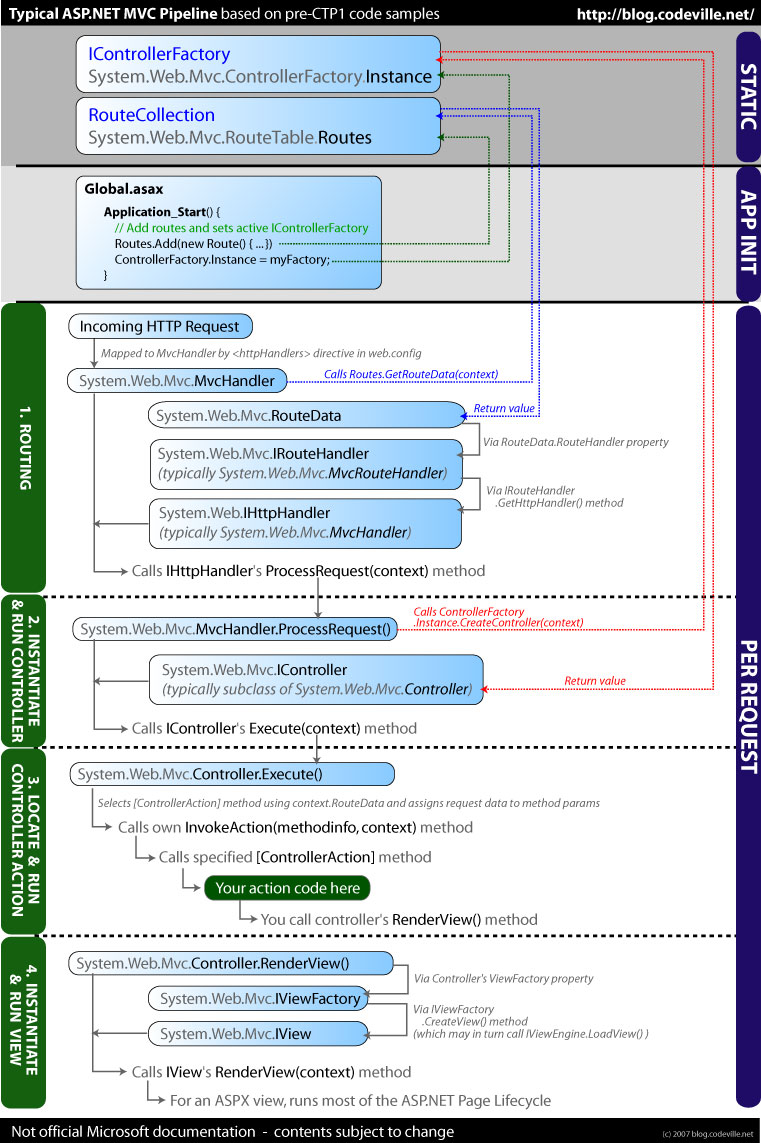


Рис. 9 Инициализация приложения

* **Маршрутизация**

Маршрутизатор – независимый компонент, который подбирает адрес, содержащийся в запросах, перехватываемым IHttpHandler, в соответствии с шаблоном URL. По стандарту, IHttpHandler реализует класс MvcHandler, который выступает в роли proxy к остальным обработчикам, описанным в таблице маршрутизации. Комбинация System.Web.Mvc.MvcHandler и System.Web.Mvc.UrlRoutingModule, на которые ссылается web.config дает ответственность за обработку входящих запросов MvcHandler.

Сначала, вызывается метод Routes.GetRouteData(), который сравнивает входящий запрос со списком маршрутов, которые были добавлены в методе Application\_start(). Выбирается нужный маршрут и инициализируется объект RouteData. Он ссылается на связанные объекты, реализующие IRouteHandler и IController. Далее, вызывается метод GetHttpHandler интерфейса IRouteHandler. GetHttpHandler возвращает объект IHttpHandler, в котором вызывается метод ProcessRequest(). На рисунке 7 представлена наглядная схема вышеописанного процесса

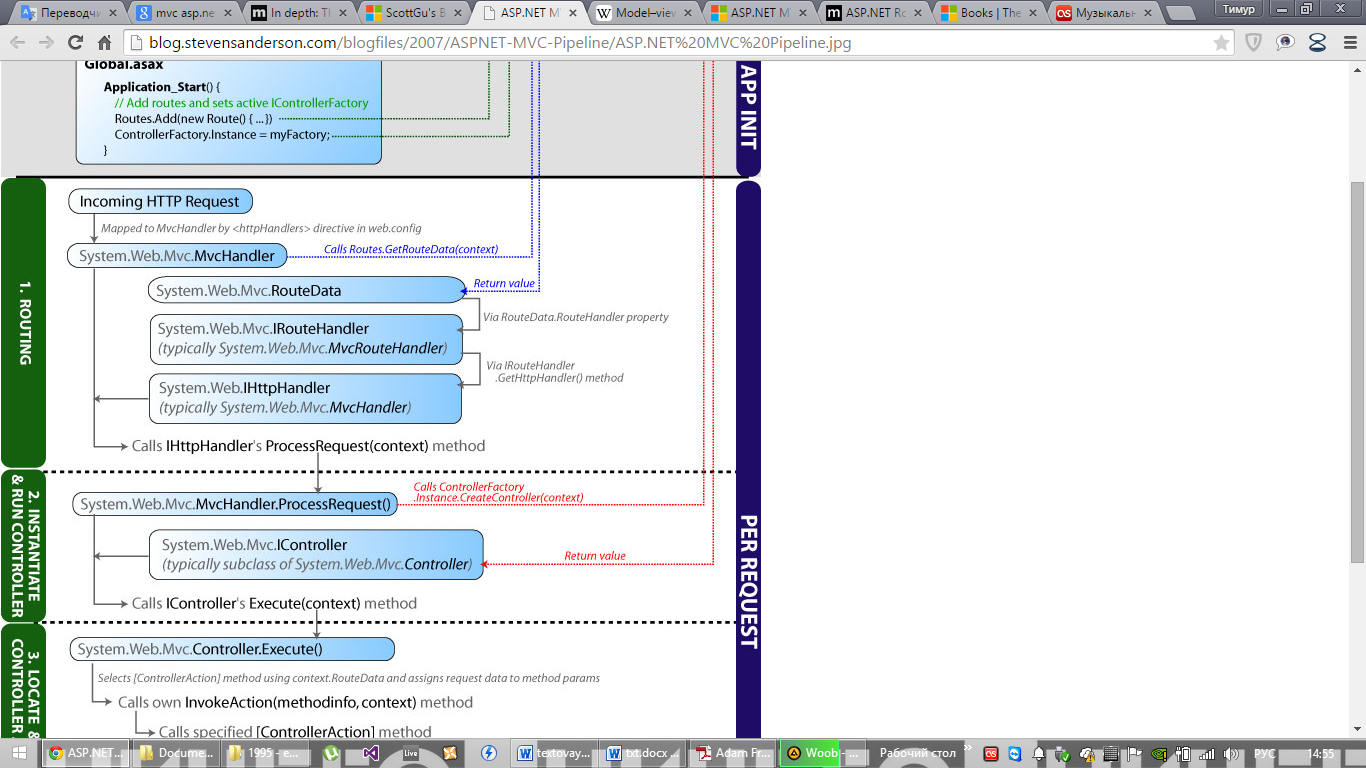


Рис. 10 Маршрутизация

* **Создание экземпляра и инициализация контроллера**

Как видно на схеме, представленной на рисунке 8, метод ProcessRequest() вызывает метод ControllerFactory.Instance.CreateController(), передавая контекстную информацию, включая тип контроллера, содержащийся в объекте RouteData.

IControllerFactory способна создавать экземпляр класса, реализующего интерфейс IController и возвращать вызывающему контроллер методу. Обычно, это подкласс класса Controller.

Затем, вызывается метод Execute() интерфейса IController.

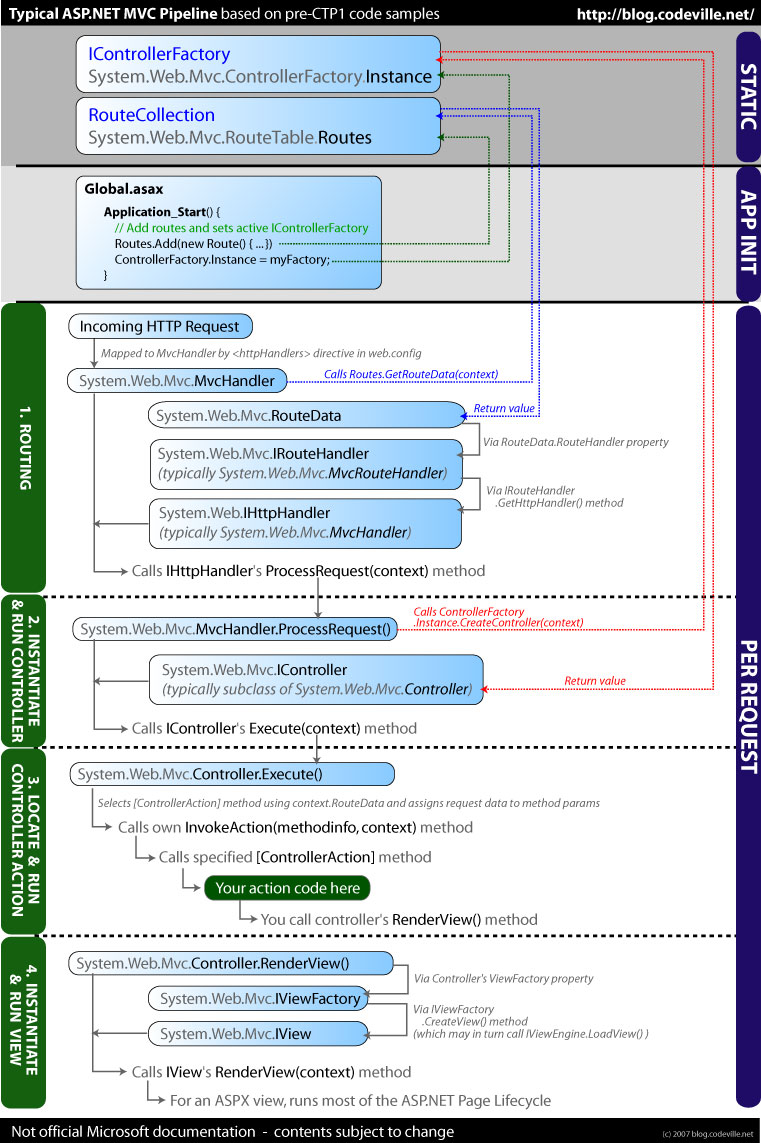


Рис. 11 создание экземпляра и инициализация контроллера

* **Выбор требуемого метода контроллера**

Метод Controller.Execute() использует объект RouteData и другую контекстную информацию, чтобы выбрать подходящий метод. До версии MVC3, этот метод должен был иметь атрибут [ControllerAction]. Также, он должен принимать и преобразовывать параметры входящего запроса в список параметров метода.

Контроллер вызывает его собственный метод InvokeAction(), передовая информацию о выбранном методе, который, в свою очередь, вызывает нужный метод. В данный момент выполняется код, написанный разработчиком веб сайта.

Далее, вызывается метод контроллера RenderView(). К этому времени инициализируется и заполняется данными объект ViewData, содержащий в себе также модель. Данный процесс наглядно представлен на рисунке 9

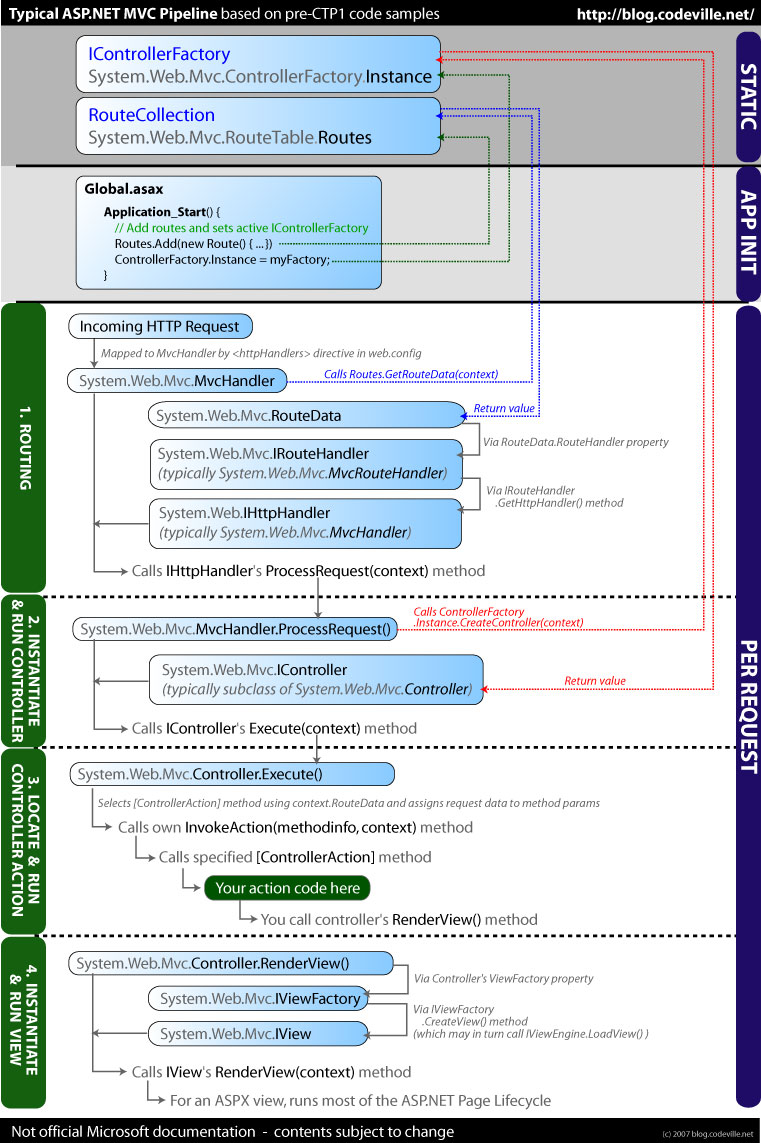


Рис. 12 Выбор требуемого метода контроллера

* **Создание экземпляра и отрисовка представления**

Подсистема представлений использует тот же паттерн «фабрика», что и система маршрутизации. У объекта Controller есть поле ViewFactory типа IViewFactory. Интерфейс IViewFactory определяет метод, называемый CreateView(), который принимает имя представления и другую контекстную информацию, используя которую, создается и возвращается экземпляр IView.

Контроллер теперь может вызвать метод IView.RenderView(), передавая необходимую контекстную информацию, которая включает объекты ViewData и IHttpResponse, в которые можно записать любые текстовые и двоичные данные и, далее, отправить в качестве ответа от сервера на запрос пользователя. Этим ответом может быть HTML, Json, файл, изображение, текст или другие двоичные данные. Данный процесс представлен на рисунке 10

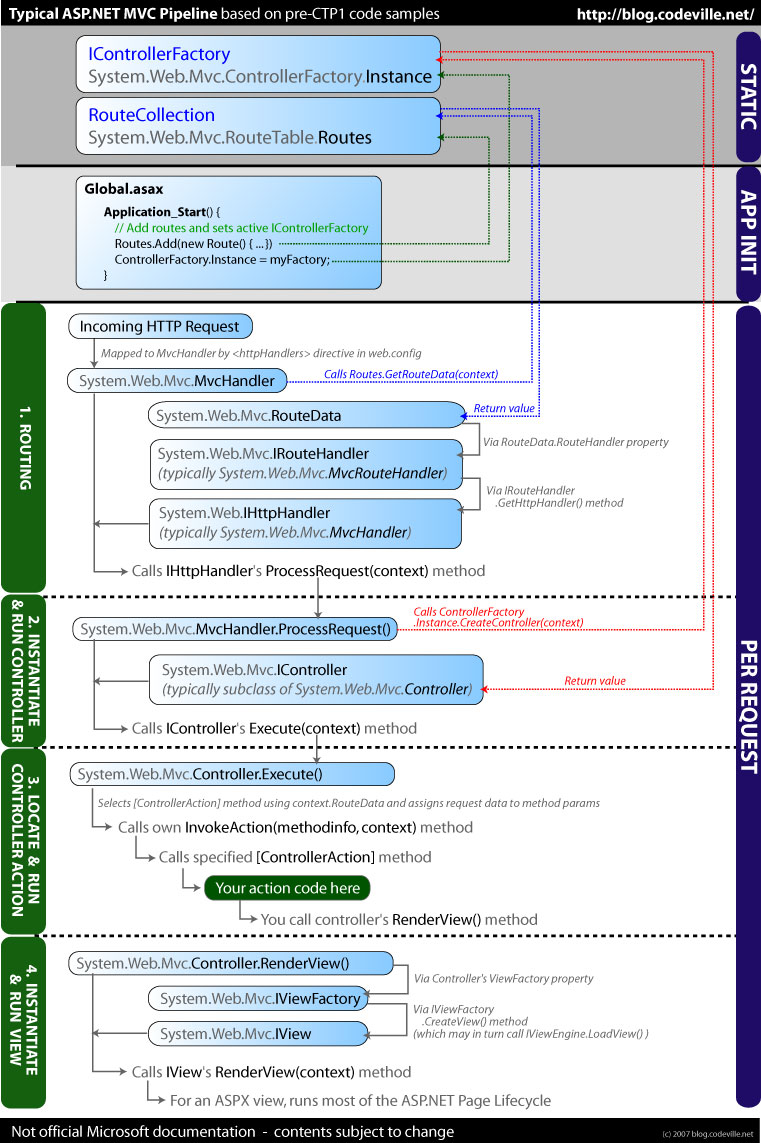


Рис. 13 Создание экземпляра и отрисовка представления

Таким образом, типовой процесс разработки таков:

* В Global.asax, добавить объект, обрабатывающий нужным алгоритмом входящий запрос и выделяя из него контроллер, метод контроллера и параметры метода.
* Добавить класс, расширяющий Controller, добавить в него нужный метод, обрабатывающий модель
* Добавить шаблон HTML, который отображает данные, хранящиеся в объекте модели

**Обоснование выбора технологии**

Так как на данной технологии реализован относительно небольшая часть приложения выбор был непринципиален, однако, основным доводом в пользу ASP.NET MVC является полная совместимость даже в рамках одного проекта с WebApi, а также, простое связывание с веб службами WCF. Кроме того, немаловажным следует признать наличие обширного опыта разработки на этой технологии.

**ASP.NET WebApi**

Методы, предполагающие асинхронное взаимодействие и обработку запросов от редактора реализованы на базе ASP.NET WebApi.

ASP.NET WebApi – фреймворк, реализовывающий спецификацию HTTP и может быть использован для построения HTTP сервисов, как альтернатив к SOAP сервисам и WCF сервисам, сравнительная характеристика которых приведена в таблице 1 и на рисунке 11. ASP.NET Web Api изначально разрабатывался как сервис WCF, но затем, стало ясно, что Web Api является больше веб фреймворком, чем сервис фреймворком.

|  |  |  |
| --- | --- | --- |
| Свойство | SOAP | HTTP |
| Фрагменты URL | Да, содержит конечные точки и методы | Да, может включать множество значений |
| Полезная нагрузка | Да, содержит обертку в видео заголовка и тела сообщения | Да |
| Контракт | Да | Нет |
| Строка URL запроса | Нет | Да |
| Код статуса | Нет | Да |
| Форматирование и описание типа объекта | Нет | Да, хранится в свойстве Accept |
| Кэширование | Нет | Да, реализовано в заголовках |
| Согласование | Нет | Да, реализовано в заголовках |
| Аутентификация | Нет | Да |
| Кодировка | Нет | Да |
| Частичные ответы | Нет | Да |

Таблица 10 сравнение HTTP и SOAP

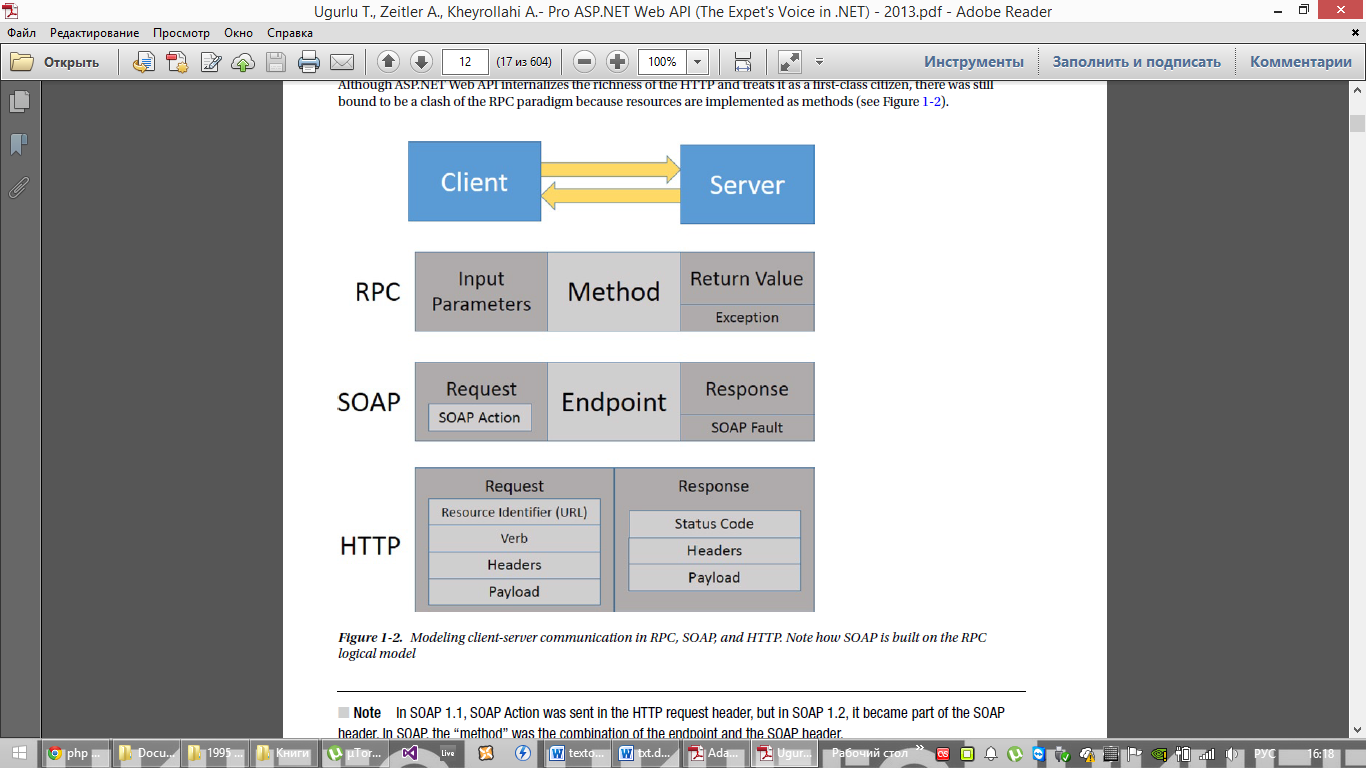


Рис. 14 Сравнение протоколов

Архитектурные особенности и преимущества ASP.NET WebApi:

* Полная асинхронность

Web API был спроектирован, используя модель программирования основанную на асинхронно выполняемых задачах.

* Отсутствие переменной HttpContext.Current

В ASP.NET MVC есть переменная HttpContext, обернутая в HttpContextBase. В Web API, в свою очередь все параметры запроса хранятся в коллекции «ключ-значение» Request.Properties, которая не является статичной в рамках сессии, в отличие от HttpContext и позволяет внедрять асинхронность.

* Проектирование аналогичных последовательностей HTTP запросов и ответов, как в библиотеке клиента, так и сервера, что позволяет упростить модель поведения за счет унификации.
* Возможность запускать приложение на серверах, отличных от IIS.
* Встроенное внедрение зависимостей.

Web API поддерживает любой фреймворк внедрения зависимостей через простой интерфейс.

* Простое написание автоматических тестов с возможностью полного покрытия.
* Класс HttpConfiguration

Контекст времени выполнения Web API абстрактен и представлен в классе HttpConfiguration. В этом классе представлены контекстные переменные, инициализирующиеся во время выполнения, в то же время они не являются статичными, что увеличивает тестируемость кода. Наиболее важные поля:

* + Маршруты
  + Перехватчики сообщений
  + Глобальные фильтры
  + Форматеры
  + Распознаватель зависимостей
  + Провайдеры связывания параметров

Как видно на рисунке 11, ASP.NET Web API может быть использован во множестве случаев. C одной стороны, Web API использует HTTP, нетребовательный к ресурсам. Таким образом, можно подключить к серверу маломощные устройства, например, смартфон. Также, с его помощью можно производить распределенные вычисления, как в облаке, так и в корпоративной сети. С другой стороны, он может быть использован для предоставления данных веб приложениям.

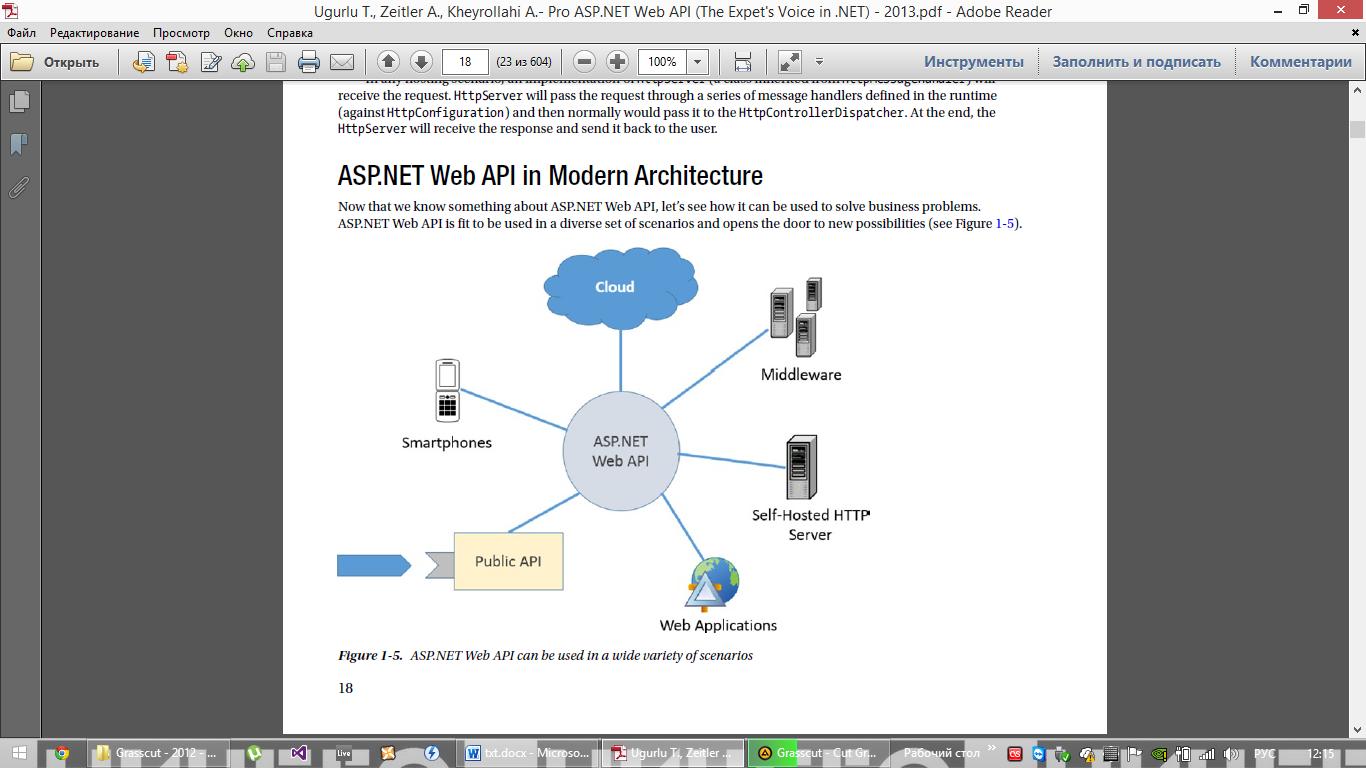


Рис. 15 Возможности использования ASP.NET Web API

В настоящее время REST API переживает взлет. Многие крупные компании делают возможным доступ к их продуктам по чистому HTTP. Такие сервисы и платформы, как Twitter, Amazon, Facebook, Netflix, Flickr, Tumblr, Google Maps предоставляют пользователям доступ к публичному API. Предоставление публичного API добавляет дополнительные возможности для расширения своего приложения за счет других программистов.

Кроме того, программирование веб приложений претерпело значительные изменения в последние годы. Стремительное развитие портативных устройств повлекло, а также появление новых техник программирования на JavaScript и изменения библиотек открыло путь к программированию клиентских приложений, инкапсулирующих значительную логику и использующих значительное количество ресурсов, а также перенос некоторых вычислений с сервера на клиент.

Последнее время приобрел значительную популярность такой тип приложений, как Single Page Application (SPA). Наиболее известные это Gmail (и другие сервисы google), TiddlyWiki, и т.д., хотя первый такой сайт появился в 2002 по адресу slashdotslash.com, как демонстрация нового подхода к дизайну веб приложений. В этом типе приложений сервер предоставляет только статичные HTML шаблоны и js код, а затем приложение получает данные с сервера, используя асинхронные вызовы посредством AJAX. Вся навигация и передача данных на сервер происходит без полной отправки данных все страницы. SPA обычно разрабатывается с помощью JS MVC фреймворка вроде Angular.js, Knockout.js или Backbone.js (также называемые библиотеками MVVM). ASP.NET Web API как раз подходит для создания приложений такого типа.

Также, ASP.NET Web API может использоваться для облачных вычислений. Как показала практика Amazon, который внедрил сервис AWS, использование REST решений в области облачных решений чрезвычайно эффективно. Также, REST архитектура применена в Microsoft Azure.

Как говорилось ранее, развитие смартфонов, повлекло за собой развитие более легковесного, в сравнении с SOAP, протокола HTTP. Множество приложений может быть охарактеризовано как обычный терминал, осуществляющий вызовы серверных методов посредством API.

**Обоснование выбора технологии**

Так как приложение тесно связано с редактором, который, тем не менее является обособленным SPA приложением, Web API пришелся как нельзя более кстати для взаимодействия между редактором и сервером. Кроме того, это позволит в дальнейшем, без переработки сервера разработать приложение для смартфона и легко добавлять другие сервисы, разработанные, в том числе, сторонними программистами.