**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 управления интерактивной интернет-рекламой. Модуль интеграции и статистики»

Выполнил студент группы ИСТ-51

Игнатченко Т. С.

Руководитель работы

асс. каф. ПИТ Шварц А. Ю.

Допущен к защите

Протокол № от 2015 г.

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

Саратов 2015

****Реферат****

Отчет 73 страницы, 3 части, 29 рисунков, 18 таблиц, 22 источника.

Цель работы – повышение эффективности разработки презентаций на основе видео для пользователей, не обладающих навыками дизайна и программиста, а также, обеспечение полного контроля за дальнейшей поддержкой презентации, сбор статистики просмотров, подключение сторонних приложений.

В процессе выполнения работы было проведено детальное сравнение существующих систем и установлен факт отсутствие подобных решений на рынке SAAS систем. Выбор программного обеспечения определялся исходя из возможностей языка программирования и наличия соответствующих лицензий. Для создания программного обеспечение, было проведено исследование современных направлений и трендов в развитии веб технологий. По результатам исследования была спроектирована база данных, отображающая особенности предметной области, и разработано распределенное приложение, состоящее из слабосвязанных модулей, выполняющих запросы друг к другу посредством сообщений XML.

В результате проведенных работ был получен полнофункциональный программный продукт, отвечающий всем предъявленным требованиям. Пояснительная записка выполнена в текстовом редакторе Microsoft Office Word 2013. Файл, содержащий пояснительную записку, имеет название: "Диплом Игнатченко.doc"

Оглавление

[Введение 5](#_Toc419725756)

[1 Анализ предметной области 6](#_Toc419725757)

[1.1 Экосистема RTB 6](#_Toc419725758)

[1.2 Видео реклама 8](#_Toc419725759)

[2 Обзор существующих решений 10](#_Toc419725760)

[2.1 BannerFlow 10](#_Toc419725761)

[2.2 Prezi 12](#_Toc419725762)

[3 Практическая реализация системы 14](#_Toc419725763)

[3.1 Функциональные возможности 14](#_Toc419725764)

[3.2 Входные и выходные данные 15](#_Toc419725765)

[3.3 Схема работы приложения 16](#_Toc419725766)

[3.4 Архитектура системы 17](#_Toc419725767)

[3.4.1 Слой взаимодействия с БД 19](#_Toc419725768)

[3.4.2 Слой бизнес логики 22](#_Toc419725769)

[3.4.3 Слой представления данных 28](#_Toc419725770)

[3.5 Используемые технологии 31](#_Toc419725771)

[3.5.1 Net Framework 32](#_Toc419725772)

[3.5.2 Язык программирования C# 35](#_Toc419725773)

[3.5.3 Фреймворк ASP.NET 38](#_Toc419725774)

[3.5.4 ASP.NET WebApi 48](#_Toc419725775)

[3.6 Классы программного обеспечения 52](#_Toc419725776)

[3.7 Проектирование базы данных системы 57](#_Toc419725777)

[3.8 Описание графического интерфейса 63](#_Toc419725778)

[3.8.1 Экран список презентаций 64](#_Toc419725779)

[3.8.2 Экран «A\B тестирование» 65](#_Toc419725780)

[3.8.3 Графики 67](#_Toc419725781)

[3.9 Аппаратное обеспечение, необходимое для запуска системы 69](#_Toc419725782)

[4 Заключение 70](#_Toc419725783)

[5 Список литературы 72](#_Toc419725784)

# 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 приложение, которое 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.

# Анализ предметной области

Рынок маркетинга стремительно осваивает веб технологии, позволяющие применить автоматизированные приемы доведения рекламы до пользователей. В настоящее время флагманской технологией в веб - маркетинге является RTB.

## Экосистема RTB

RTB (Real Time Bidding)[20] — технология закупки медийной рекламы посредством программируемых онлайн-аукционов. РТБ фокусируется непосредственно на показах целевым посетителям, а не планированию резервов рекламных площадей на определенных сайтах. Каждый показ выкупается за доли секунды — во время загрузки страницы — система RTB мгновенно проводит аукцион. В результате лучшее предложение от рекламодателей появляется на глазах пользователя, которому оно наиболее интересно.

RTB реклама делает общение между брендом и покупателем приятнее для обеих сторон. Пользователи видят действительно подходящую рекламу, а рекламодатели получают недостижимые ранее показатели эффективности. Личные данные пользователей при этом в безопасности: все RTB-платформы осуществляют таргетинг на основании предоставленных DMP анонимных профилей.

Рекламодатель заранее вводит в Trading Desk свои требования к ЦА (социально-демографические показатели, поведенческие критерии) и задает ограничения по бюджету и стоимости показа или клика. В ходе кампании Trading Desk задействует соответствующий RTB блок — DMP, SSP, Ad Networks, Ad Exchanges, — осуществляя множество ставок перед каждым показом.

Дальше система работает следующим образом:

* пользователь ввел в строку браузера название сайта, подключенного к рекламной RTB сети;
* сайт перед загрузкой страницы отправляет сети, к которой он подключен, запрос на показ баннера, а также данные о пользователе (не ФИО и адрес проживания, а доступные для таргетинга данные);
* рекламная сеть передает данные о пользователе в SSP;
* SSP классифицирует запрос по максимальному числу параметров: площадка на которую зашел пользователь, данные о пользователе, время захода, данные по ретаргетингу;
* SSP выставляет торг — информацию о сайте и пользователе системам рекламодателей. При желании SSP закупает дополнительные данные о пользователе у DMP;
* Trading Desk, используя DSP, передает информацию о ставке SSP;
* SSP обрабатывает информацию от всех рекламодателей, сделавших ставку;
* выбирает самую высокую;
* снижает ее до второй цены и осуществляет показ по цене, превышающей вторую по величине ставку на 1 цент;
* получает объявление от победителя и передает его браузеру пользователя в ответ на запрос рекламного объявления;
* загружается страничка с полученным объявлением.

Первые RTB площадки появились в России в 2011 году.

Однако, далеко не каждая площадка предоставляет редактор презентаций. Кроме того, созданную в стороннем редакторе презентацию не всегда можно легко встроить в любую площадку.

Применение видео-презентаций совместно с RTB площадками позволит предоставлять пользователю нужные ему презентации ненавязчиво, но с максимальным вовлечением.

## Видео реклама

В настоящее время, По опросам eMarketer, почти 67% американских маркетологов в 2009 году намерены сфокусировать свои маркетинговые бюджеты на онлайн-видеорекламе. Среди опрошенных этот вид рекламы в полтора раза популярнее, чем реклама в социальных сетях, в два раза популярнее продвижения через поисковики и в три раза популярнее традиционных баннеров [21] .

Сейчас существует несколько видов использования видеоплеера на сайте как рекламного носителя:

• реклама перед видеороликами (pre-roll);

• реклама поверх видеороликов (тизеры, overlay);

• реклама в окне до старта видеоролика;

• контекстная реклама в видеороликах;

• реклама в окне после видеороликов.

Наиболее часто используются форматы «Реклама перед видеороликами» и «Реклама поверх видеороликов».

Реклама перед видеороликами запускается после того, как пользователь нажимает кнопку «Play», чтобы посмотреть ролик на сайте. До показа основного ролика пользователь видит рекламу, по окончании которой уже запускается желаемый ролик (на некоторых сайтах рекламу можно пропускать). Реклама перед роликом — исключительно эффективный формат, так как в этом случае пользователь просто не может рекламу не увидеть. Для снижения навязчивости рекламы владельцы многих сайтов ограничивают показ рекламы и настраивают частоту показа рекламных сообщений (например, не более одного ролика на один IP-адрес пользователя в сутки).

Более того, объединение ведущих интернет рекламистов IAB (Interactive Advertising Bureau) разработало унифицированные форматы и интерфейсы, принятые многими хостингами, с помощью которых стороннее видео может в полной мере взаимодействовать с API хостинг сайта, что позволяет максимально полно интегрировать рекламное видео в окружающую среду. Согласно исследованию IAB, 63% взрослой аудитории в США [22] предпочитает просмотр телевизионных передач через трансляции в интернете, что делает аудиторию интернет видео рекламы гораздо более широкой чем традиционной телевизионной. Более того, это позволяет собрать гораздо большое количество данных о пользователях.

Подводя итоги вышесказанного, проанализировав ситуацию на рынке, было принято решение разработать программное обеспечение, обеспечивающее создание, тестирование и анализ видео, совместимых с RTB платформами и видео хостингами.

# Обзор существующих решений

## BannerFlow

BannerFlow.com 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 (Рис.1), 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.

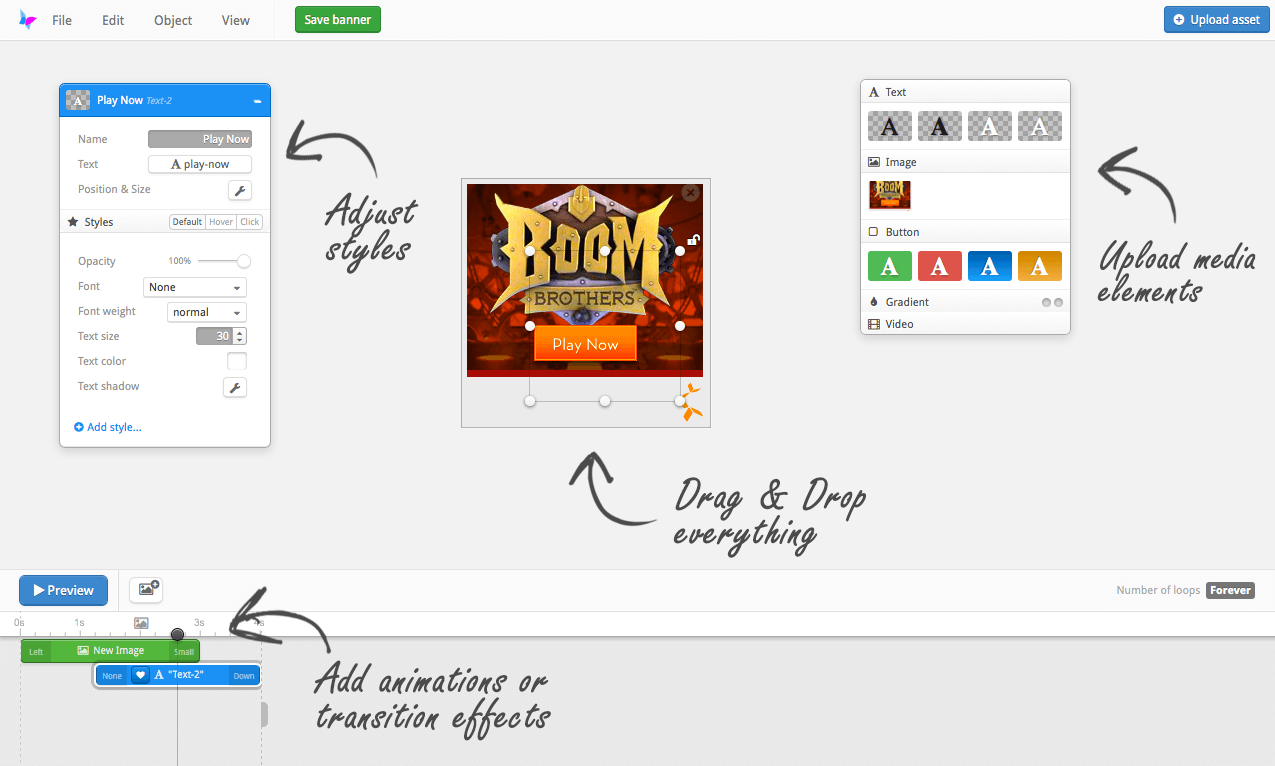


Рис. 1 Редактор BannerFlow.

Платформа позволяет

* сохранять весь 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. И получать статистику по его итогам. На рисунке 2 представлен пример статистики.

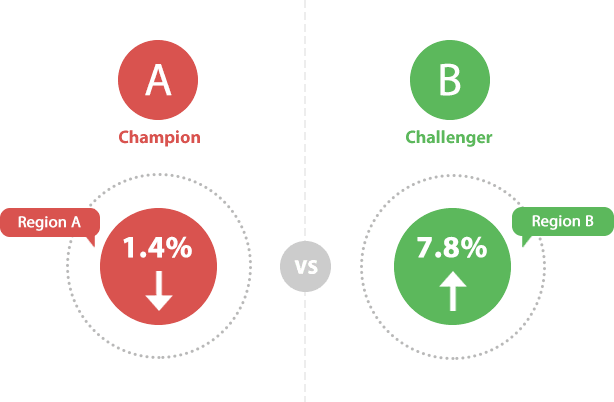


Рис. 2 Диаграмма результатов A/B тестирования в Bannerflow.

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 [1] – онлайн приложение для создания презентаций, которое позволяет создавать онлайн презентации, воспроизводимые 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.

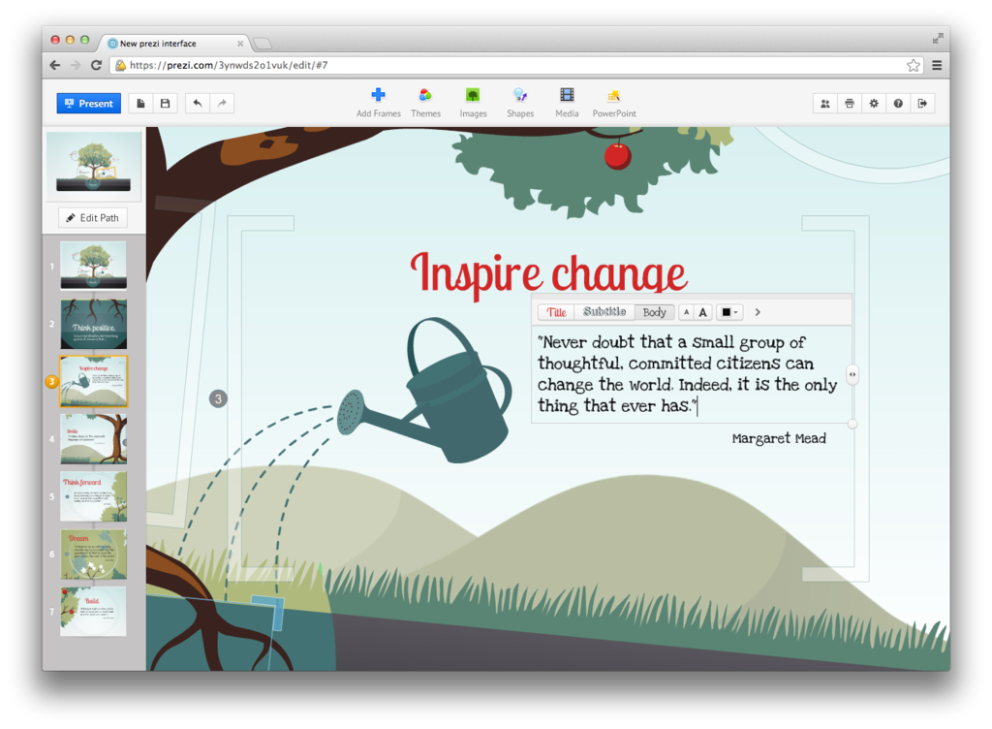


Рис. 3 Окно редактора Prezi.

Prezi обладает следующими характерными особенностями:

* 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

Разрабатываемое программное обеспечение 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, 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, полученные до которой данные не учитываются;
  + дата, полученные после которой данные не учитываются;
  + шаг времени;
  + номер презентации;
* данные для формирования отчета:
  + дата, полученные до которой данные не учитываются;
  + дата, полученные после которой данные не учитываются;
  + фильтр;
  + тип отчета.

Система формирует следующие выходные данные:

* ID загруженных сущностей;
* список презентаций;
* сущность каждой презентации;
* собранная статистика;
* отчеты, сформированные по запросу пользователя.

## Схема работы приложения

Пользователь авторизуется в системе, переходит на модуль создания презентации, 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 поддержки 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.

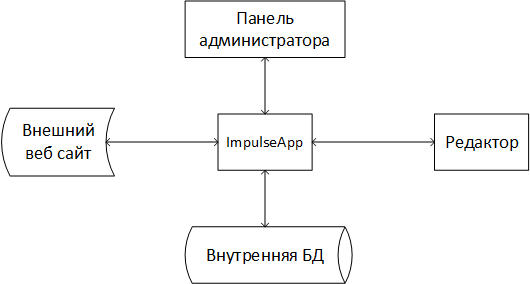


Рис. 4 Общая схема приложения.

## 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.

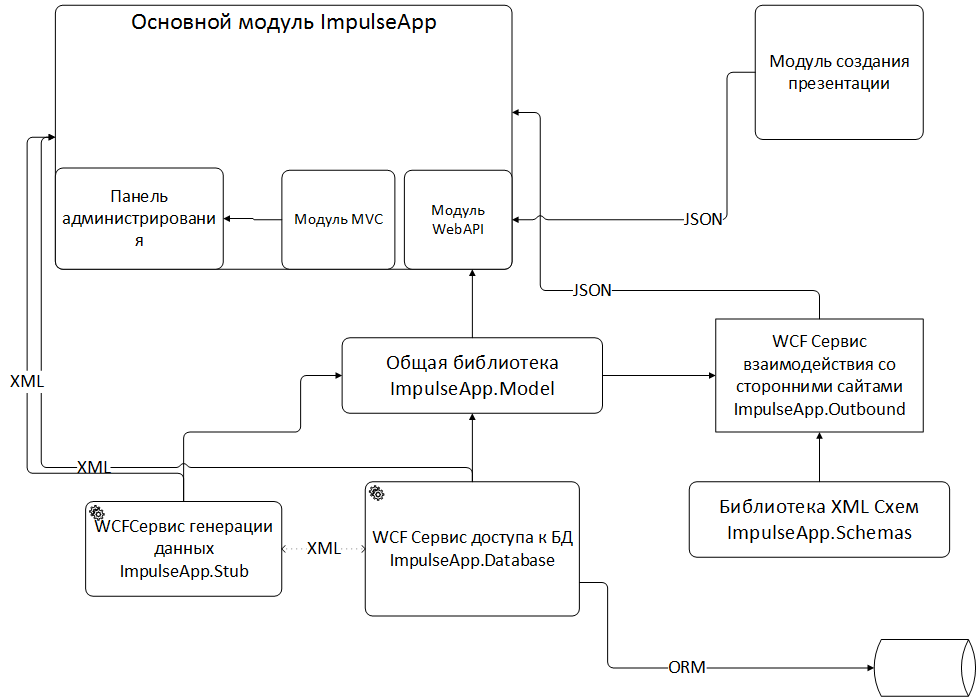


Рис. 5 Схема модулей и формат сообщений.

* 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 так называемое «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 создавать дополнительные DTO, был добавлен специальный служебный 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 класс модуля ImpulseApp.Database является класс DBService, реализующий интерфейс IDBService. Список методов интерфейса приведен в таблице 1.

Таблица 1 методы сервиса ImpulseApp.Database.

|  |  |
| --- | --- |
| 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 |

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.

### 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 2

Таблица 2 Поля модели SimpleAdModel.

|  |  |  |
| --- | --- | --- |
| 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 |

Далее, приведены прототипы на Javascript, согласно которым должны хранится данные на использующем серверное API редакторе.

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

Таблица 3 Поля класса AdState.

|  |  |  |
| --- | --- | --- |
| 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 |

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 4

Таблица 4 Поля класса NodeLink.

|  |  |  |
| --- | --- | --- |
| 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 |

Прототип на JS:

var NodeLink = function () { };

NodeLink.prototype = {

V1: 0,

V2: 0,

T: 0

};

**VideoUnit**

Поля представлены в таблице 5

Таблица 5 Поля класса VideoUnit.

|  |  |  |
| --- | --- | --- |
| 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 |

**SimpleAdModelDTO**

Упрощенная сущность SimpleAdModel без циклических зависимостей для публичной видимости. Поля представлены в таблице 6

Таблица 6 Поля класса SimpleAdModelDTO.

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

**AdStateDTO**

Упрощенная сущность слайдов для публичной видимости. Поля класса представлены в таблице 7

Таблица 7 Поля класса AdStateDTO.

|  |  |  |
| --- | --- | --- |
| 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 |

#### 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 HTM-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.

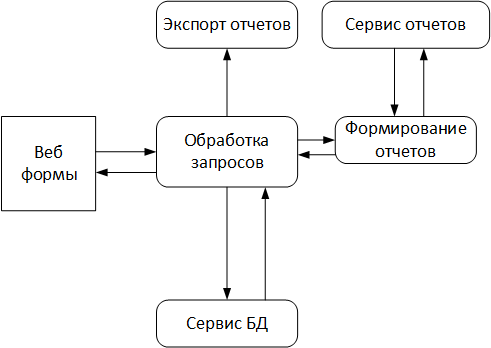


Рис. 6 Декомпозиция модуля Mvc.

#### Модуль ImpulseApp.Outbound

Данный модуль выполняет преобразование стандартной модели презентацию в рекламу формата VAST3, котороую 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, даты, полученные в рамках которых данные, учитываются при формировании выборки. Все запросы, в том числе на формирование графиков подразумевают возможность получения статистики по нескольким презентациям для сравнения.

Разработаны следующие шаблоны:

* статистика по браузерам;
* статистика по кликам;
* статистика по переходам.

#### Модуль ImpulseApp.Stub

Данный модуль используется для тестирования новых компонентов во время работы, а также для проведения нагрузочного тестирования. В данном модуле реализован алгоритм, генерирующий просмотры за определенный период в основном для тестирования модуля статистики.

Также, в этом модули находятся заглушки внешних систем для эмуляции доступа при отсутствии аккаунта.

### Слой представления данных

На данном слое, помимо HTML страниц, сгенерированных на движке Razor, находятся модули на JS, инкапсулирующую логику воспроизведения видео, сбора статистики, отрисовки диаграмм.

#### Модули mpls.statistics.helper.chartjs.js и mpls.statistics.helper.highchart.js

Данные модули предоставляют собой обертки, инкапсулирующую общую логику работы с JavaScript библиотеками chart.js и highchart.js, позволяя использовать несколько строк кода для формирования любых графиков в рамках текущей системы.

Инициализация графиков осуществляется через функцию init(chartType, chartText, legend), где chartType – тип графика, chartText – заголовок, legend – названия осей координат(необязательно для некоторых типов графиков). На странице должен быть DOM элемент типа div с атрибутом id = ‘chart-container’. Для корректной работы, у этого элемента также должен быть атрибут data-url, у которго значения равно адресу метода в модуле ImpulseApp.Stat, возвращающего данные в формате Json. Описание модели данных приведено в таблице 8

Таблица 8 Описание модели StatChartJs.cs.

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип | Описание |
| labels | List<string> | Список признаков |
| data | List<string> | Список значений |
| name | string | Имя графика (обычно переопределяется на странице html) |
| color | string | Цвет в формате rgb (необязательный параметр для highcharts) |

#### Модуль mpls.statistics.js

Данный модуль осуществляет сбор статистики взаимодействия пользователя с презентацией и отправляет ее на сервер. В модуле используются модели данных Click, Activity, AdSession, описанные в разделе 3.4.2.1 «Модуль ImpulseApp.Api». Модуль сохраняет взаимодействие только с теми DOM элементами, у которых атрибут class принимает значение “ mpls-tracked ”. По умолчанию такой атрибут стоит у всех элементов управления презентацией. Кроме того, данный модуль считывает и сохраняет данные со следующих атрибутов DOM элементов:

* При нажатии на кнопку перехода:
  + **data-current-id** – id текущего видео;
  + **data-next-id** – id следующего видео;
  + **data-next-time** – изначальное время следующего видео;
  + **data-action** – тип действия, осуществляемого элементом;
  + **class –** класс DOM элемента;
* При переходе на следующий слайд:
  + **data-name –** название текущего состояния (слайда);
  + текущее время.

Для сохранения данных в БД используется метод /api/stat/session компонента ImpulseApp.Api. Подробнее процесс формирования объекта с информацией о взаимодействии представлен на рисунке 7.

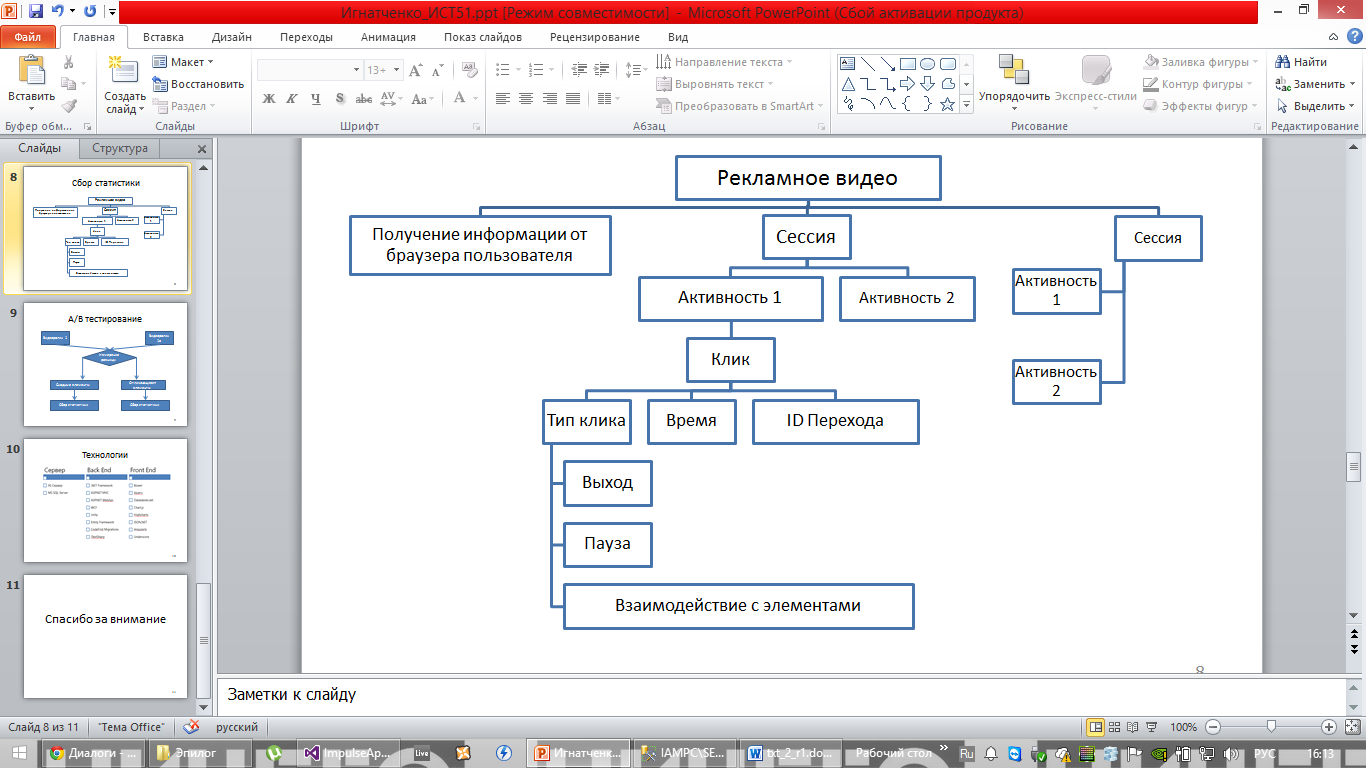


Рис. 7 Сбор статистики.

#### Модуль mpls.video.play.js

Данный модуль преобразовывает модель презентации, получаемую с сервера в DOM элементы на странице. Модуль получает с сервера объект типа SimpleAdModel (см. таблицу 2). На основе полученных состояний загружаются слайды с видео, на которые в ближайшее время может быть осуществлен переход. Обращение происходит к REST методу /api/ad.

#### Модули angular.js

На angular js написан основной функционал, реализующийся на клиентской части приложения. Список модулей следующий:

* mplsApp.js – основной модуль приложения, управляет загрузкой и связывается остальные контроллеры и модули;
* mplsDirectives.js – список директив приложения. Директивы Следующие:
  + adInfoblock – директива, определяющая блок отображения презентации. На вход идет scope родительского контроллера;
  + shortUrlFilter – возвращает отфильтрованные и сгруппированные по короткому адресу презентации;
* mplsServices.js– список сервисов приложения. Директивы Следующие:
  + SpinnerService– сервис, управляющий индикатором загрузки, появляющимся при переходе между представлениями;
  + ServerQueryService– Сервис, позволяющий осуществлять запросу к api сервера;
* AbTestController.js – контроллер, управляющий созданием AB тестирования приложения;
* AdController.js– контроллер, управляющий списком презентаций;
* GraphController.js– контроллер, управляющий созданием, просмотром графиков функций;
* HomeController.js– контроллер, управляющий основными страницами приложения;
* ModalController.js– контроллер, управляющий модальными окнами приложения;
* TableController.js– контроллер, управляющий отчетами и таблицами.

## Используемые технологии

Основными критериями 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.

В таблице 9 приведен список используемых технологий для каждого слоя приложения

Таблица 9 Ипользуемые технологии.

|  |  |  |
| --- | --- | --- |
| **Сервер** | **Back End** | **Front End** |
| IIS Сервер | .NET Framework | Bower |
| MS SQL Server | ASP.NET MVC | Jquery |
|  | ASP.NET WebApi | Datatables.net |
|  | WCF | Chart.js |
|  | Unity | Highcharts |
|  | Entity Framework | JSON.NET |
|  | CodeFirst Migrations | AngularJs |
|  | iTextSharp | Underscore |

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

### Net Framework

.NET Framework [2] – 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 [3]:

* 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.

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. 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 в 2000 году 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#[4]:

* 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 интернационализации;
* C# пригоден для разработки приложений для встраиваемых систем, а также для разработки как для сложных операционных систем, так и для выполняющих специфические функции;
* 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 10

Таблица 10 развитие языка C#.

|  |  |  |
| --- | --- | --- |
| Версия | Дата релиза | Новые особенности |
| C# 2.0[5] | Ноябрь 2005 | обобщенные типы;  разделенные типы;  анонимные методы;  итераторы;  типы Nullable;  геттеры и Сеттеры;  делегаты;  статические классы. |
| C# 3.0[6] | 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[7] | Апрель 2010 | динамическое связывание;  именованные и необязательные аргументы;  ковариация и контравариация в универсальных шаблонах;  встраиваемые типы взаимодействия. |
| C# 5.0[8] | Август 2012 | асинхронные методы и ключевые слова await, Async;  информация о вызывающем объекте |
| C# 6.0[9] | 2015 | компилятор как сервис;  импортирование статических классов в пространство имен;  фильтры исключений;  await в блоке catch\finally;  автоматическая инициализация свойств;  стандартные значения для только get свойств;  null-Условный оператор “?.”;  интерполяция строк;  определение свойств и методов в lambda выражениях;  оператор nameof;  инициализация Dictionary |

Отличительными особенностями языка являются[10]:

* 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[11]. 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.

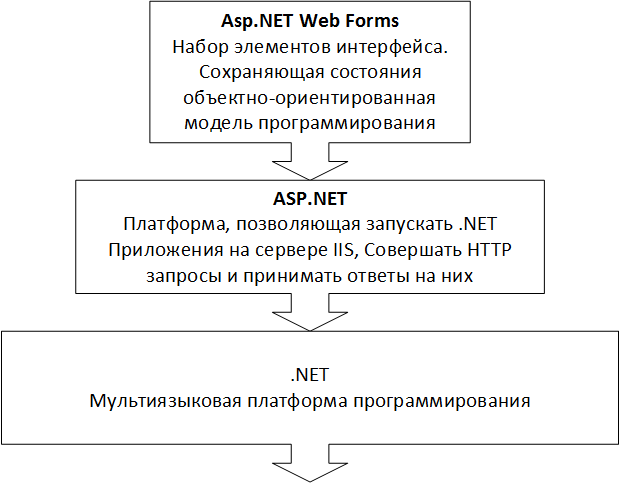


Рис. 9 Стек технологий ASP.NET WebForms.

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 10.



Рис. 10 Типичное взаимодействие компонентов паттерна MVC.

Хотя паттерн был создан в 1978 году, он приобрел широкую популярность сейчас при разработке веб приложений по следующим причинам[12]

* 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.

ASP.NET MVC Framework состоит из множества независимых компонентов, например, системы маршрутизации, движка представлений, фабрики 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 контроллера и метода в нем.

ASP.NET MVC поддерживает 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[13]. 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[14]

* Инициализация приложения

Как и любое приложение ASP.NET, при старте запускается метод в файле Global.asax Application\_Start() (как можно увидеть на рисунке 11).

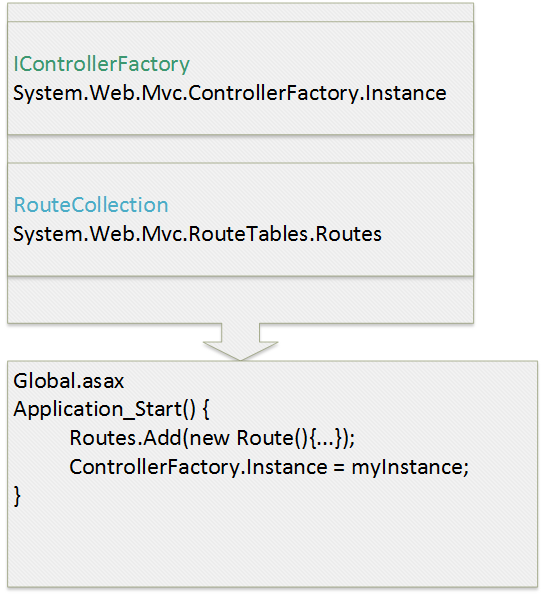


Рис. 11 Инициализация приложения.

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.

* 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 12 представлена наглядная схема вышеописанного процесса

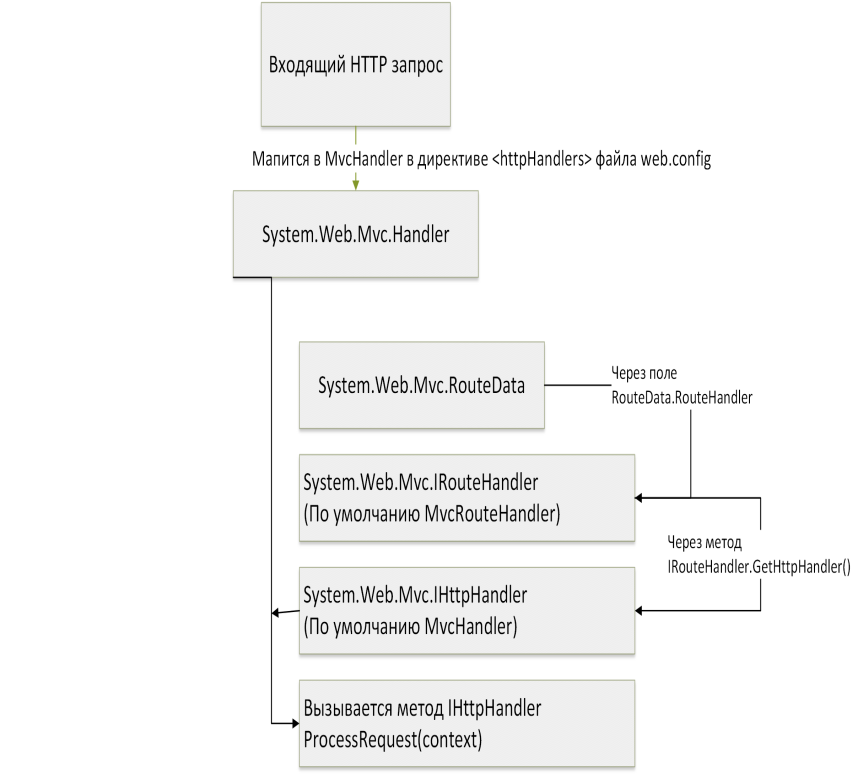


Рис. 12 Маршрутизация.

* Создание экземпляра и инициализация контроллера

Как видно на схеме, представленной на рисунке 13, 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.

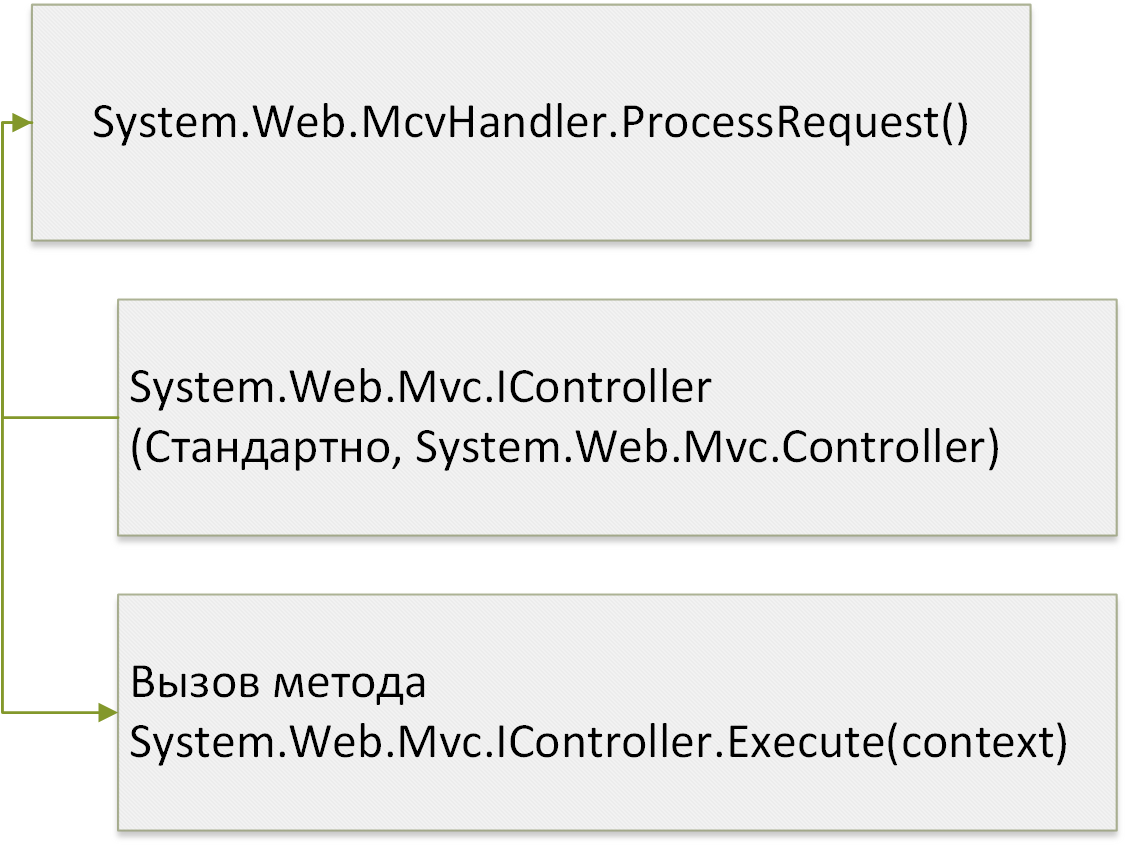


Рис. 13 создание экземпляра и инициализация контроллера.

* 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 14

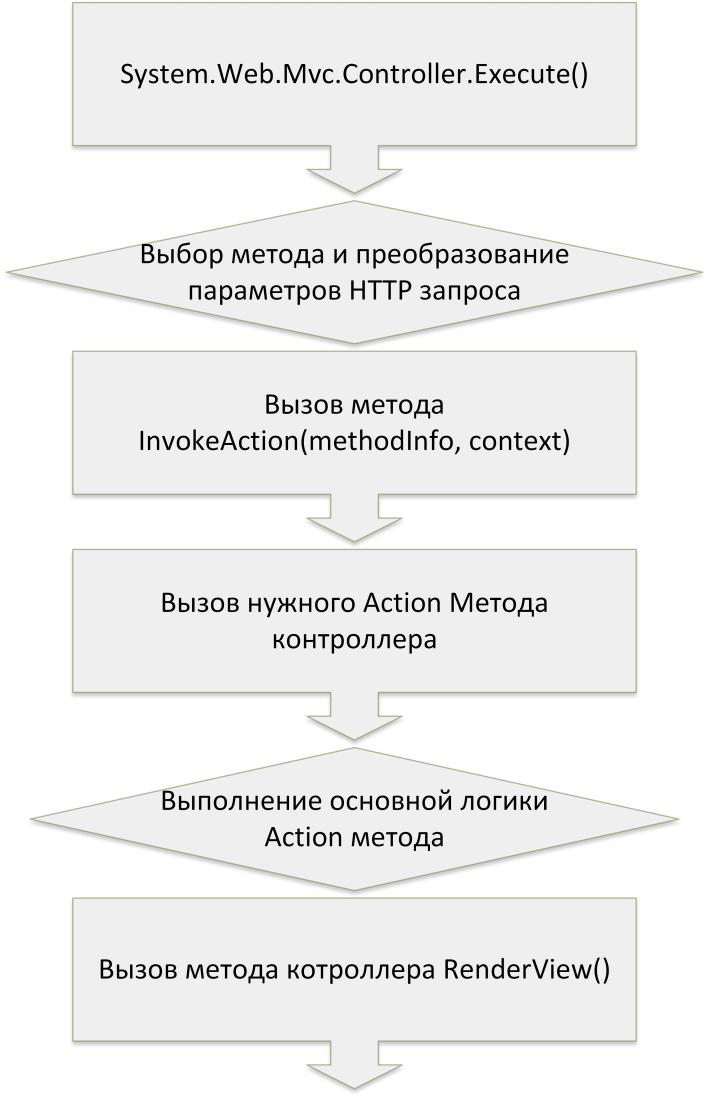


Рис. 14 Выбор требуемого метода контроллера.

* 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 15

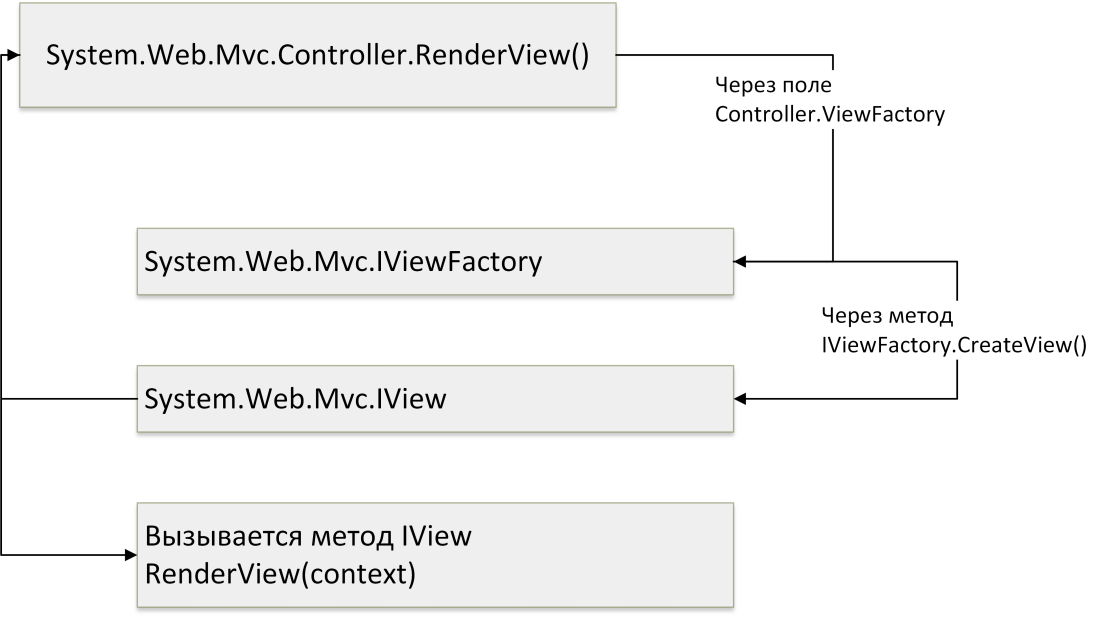


Рис. 15 Создание экземпляра и отрисовка представления.

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 10 и на рисунке 16. ASP.NET Web Api изначально разрабатывался как сервис WCF, но затем, стало ясно, что Web Api является больше веб фреймворком, чем сервис фреймворком.

Таблица 11 Сравнение HTTP и SOAP.

|  |  |  |
| --- | --- | --- |
| 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 |

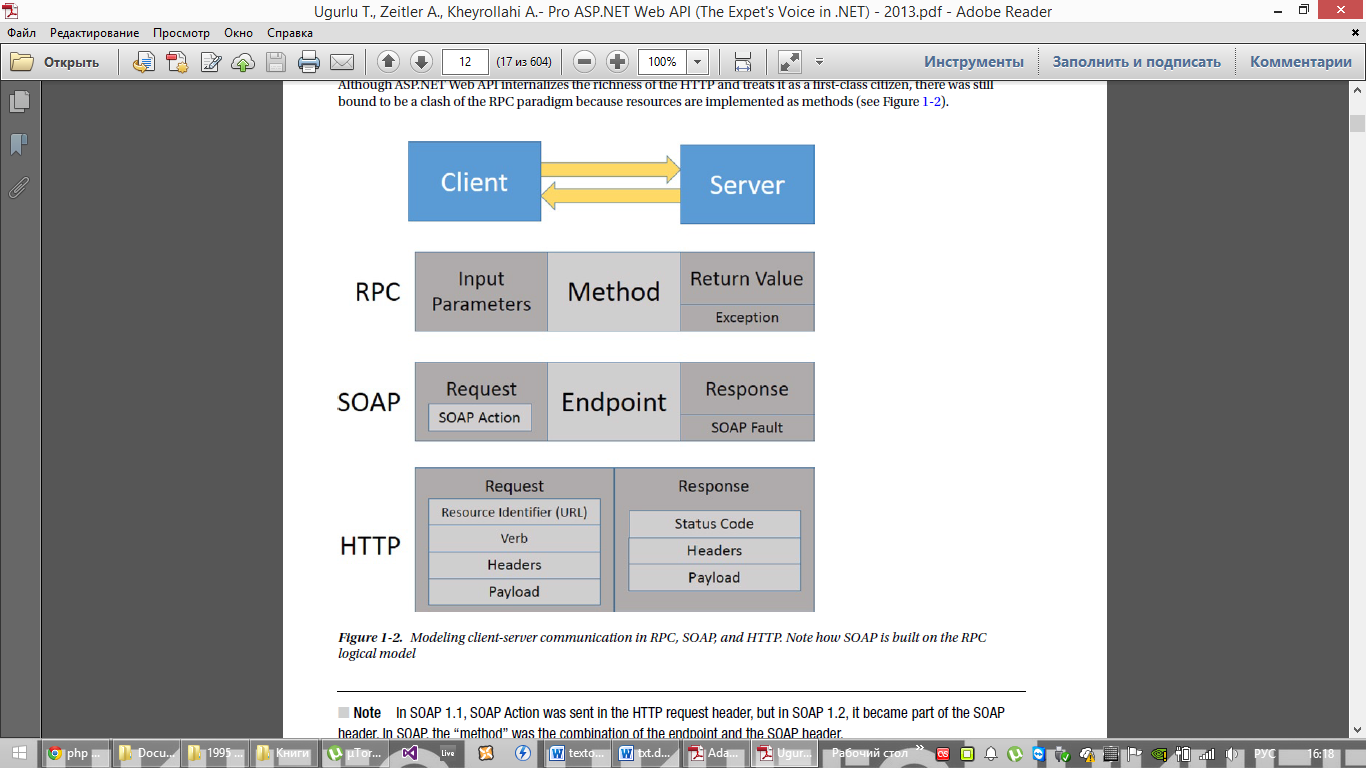


Рис. 16 Сравнение протоколов.

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 17, 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.

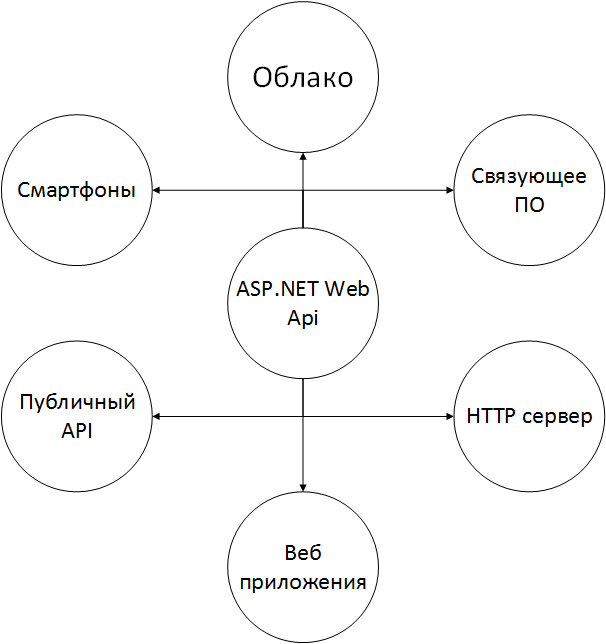


Рис. 17 Варианты использования ASP.NET WebApi.

В настоящее время REST API переживает взлет[15]. 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[16]. Стремительное развитие портативных устройств, 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)[17]. 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[18], 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.

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.

## Классы программного обеспечения

Диаграмма классов подмодуля WebAPI представлена на рисунке 18

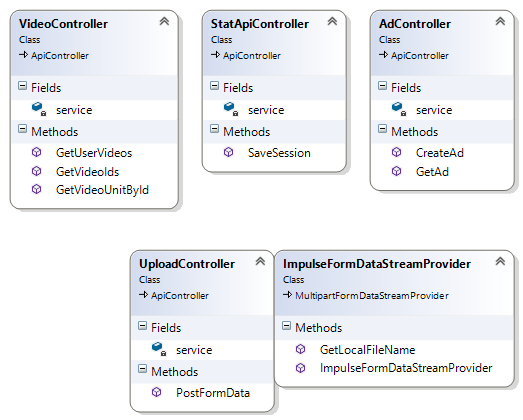


Рис. 18 Диаграмма классов ImpulseApp.Web API.

**AdController** – контроллер с методами взаимодействия с презентацией.

**StatApiController** – контроллер, позволяющий сохранять собранную статистику на сервер.

**UploadController** – контроллер, позволяющий загружать медиа элементы.

**VideoController** – контроллер, позволяющий взаимодействовать с видео.

**ImpulseFormDataStreamProvider –** класс, переопределяющий стандартный System.Net.Http.MultipartFormDataStreamProvider для корректного отображения имени файла

Диаграмма классов подмодулей ImpulseApp.Mvc и ImpulseApp.Stat представлена на рисунке 19.

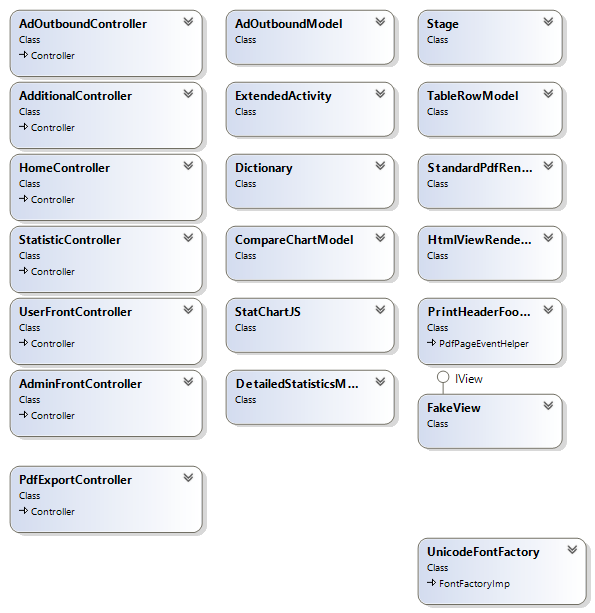


Рис. 19 Диаграмма классов ImpulseApp.

**AdOutboundController** – контроллер, управляющий отображением презентации на внешнем сайте.

**AdditionalController** – контроллер, управляющий частичными представлениями, использующимися в других представлениях.

HomeController – стандартный контроллер, управляющий переходами по пунктам меню.

**StatisticController** – контроллер, выполняющий обработку и вывод на представление статистики.

**UserFrontController** – контроллер, управляющий представлениями, доступными авторизованному пользователю.

**AdminFrontController** – контроллер, управляющий представлениями, доступными администратору.

**PdfExportController** – контроллер, управляющий экспортом в PDF.

**AdOutboundModel –** модель представления на внешнем сайте.

**ExtendedActivity –** расширенная модель модели Activity.

**Dictionary –** класс, хранящий константы и перечислимые типы.

**CompareChartModel –** класс, содержащий коллекцию объектов класса StatChartJs.

**StartChartJs** – модель, описывающая графики.

**DetailedStatisticsModel** – модель, описывающая детализированную статистику.

**Stage –** модель, описывающая name/value в обощенной таблице для экспорта.

**TableRowModel –** модель, описывающая строку в обобщенной таблице для экспорта.

**StandartPdfRenderer** – главный класс, выполняющий преобразование HTML кода в PDF.

**HtmlViewRenderer** – класс, выполняющий загрузку html кода по переданной в аргументы конструктора ссылке.

**PrintHeaderFooter** – класс добавляющий метки на каждую pdf страницу.

**FakeView** – заглушка представления.

**UnicodeFontFactory –** класс, выполняющий преобразование кодировки в UTF-8 и преобразовывающий текст в соответствующий шрифт.

Классы модуля ImpulseApp.Models представлены на рисунке 20.

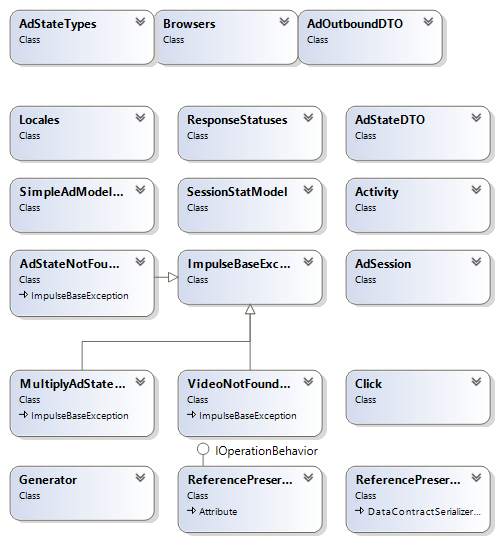


Рис. 20 Классы Impulse.Model.

**AdStateTypes –** константытипов слайдов.

**Browsers** – константы браузеров.

**AdOutboundDTO** – Data Transfer Object для передачи в представление презентации на внешнем сайте.

**Locales** – класс, содержащий константы локалей.

**ResponseStatuses** – класс, содержащий константы статусов ответа (ERROR, SUCCESS, WARNING).

**AdStateDTO** - Data Transfer Object, основанный на классе AdState.

**SimpleAdModelDTO** - Data Transfer Object, основанный на классе SimpleAdModel.

**SessionStatModel** – модель, хранящая аггрегированные данные статистики по сессии.

**Activity** – модель, хранящая данные о статистике в рамках слайда.

**AdSession** – модель, хранящая данные о статистике в рамках сессии.

**Click** – модель, хранящая данные о клике.

**Generator** – класс для генерации различных случайных имен.

## Проектирование базы данных системы

База данных – основа большинства информационных систем, фундамент на которой строится вся остальная бизнес логика приложения. От качества построения базы данных зависит качество остального приложения, сложность его программирования, количество ошибок, безопасность. Обычно, слой базы данных наиболее требователен к ресурсам, поэтому важна оптимизация данного компонента, как никакого другого. Основным требованием к структуре, это соответствие ее третьей нормальной форме, основными критериями которой являются[19]

* отсутствие повторяющихся строк;
* отсутствие зависимости от порядка строк в таблице;
* отсутствие зависимости от порядка столбцов в таблице;
* отсутствие скрытых компонентов;
* пересечения кортежей и атрибутов единичны;
* полная зависимость от первичных ключей;
* ни один неключевой атрибут не находится в транзитивной функциональной зависимости от потенциального ключа.

База данных должна хранить информацию, представляющих следующие объекты:

* объекты, представляющие структуру презентации:
  + слайды;
  + общие параметры;
  + элементы управления;
  + медиа элементы;
  + параметры структуры и переходов.
* объекты, представляющие статистические данные:
  + информация о пользователях;
  + информация о взаимодействиях;
  + информация о хостинг сайтах;
  + информация о DOM объектах, с которыми осуществляется взаимодействие.

Так как база данных была разработана с помощью технологии Code First, целостность была достигнута с помощью аннотаций к полям классов. Структура сгенерированной базы данных представлена на рисунке 21.



Рис. 21 Структура БД.

Далее каждая таблица базы данных будет описана отдельно.

**Таблица SimpleAdModel**

Данная таблица представляет собой базовую модель презентации. Описание атрибутов представлено в таблице 11.

Таблица 12 SimpleAdModel.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | Целочисленный | Идентификатор |
| Name | Текстовый | Имя |
| ShortUrlKey | Текстовый | Короткий адрес |
| HtmlSource | Текстовый | Общий html слой |
| UserId | Текстовый | Идентификатор пользователя |
| DateTime | Дата | Время создания |
| JsSource | Текстовый | Пользовательский JS код |
| HtmlStartSource | Текстовый | Начальный html слой |
| HtmlEndSource | Текстовый | Конечный html слой |

**Таблица Activities**

Данная таблица представляет собой описание взаимодействий в рамках одного слайда с презентации. Описание атрибутов представлено в таблице 12.

Таблица 13 Activities.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | Целочисленный | Идентификатор |
| StartTime | Дата | Начало просмотра слайда |
| EndTime | Дата | Окончание просмотра слайда |
| SessionId | Целочисленный | Id сессии |
| CurrentStateName | Текстовый | Имя слайда |

**Таблица AdSessions**

Данная таблица представляет собой описание статистики в рамках одного просмотра презентации. Описание полей представлено в таблице 13.

Таблица 14 AdSessions.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| SessionId | Целочисленный | Идентификатор |
| DateTimeStart | Дата | Начало просмотра презентации |
| DateTimeEnd | Дата | Окончание просмотра презентации |
| AdId | Целочисленный | Id презентации |
| ActiveMilliseconds | Целочисленный | Имя слайда |
| UserIp | Текстовый | IP адрес пользователя |
| UserLocation | Текстовый | Местонахождение пользователя, вычисленное на основе IP |
| UserLocale | Текстовый | Язык браузера пользователя |
| UserBrowser | Текстовый | Название браузера |

**Таблица Clicks**

Данная таблица содержит информацию о кликах пользователя. Описание атрибутов приведено в таблице 14.

Таблица 15 Clicks.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | Целочисленный | Идентификатор |
| ActivityId | Целочисленный | Идентификатор активности |
| ClickZone | Текстовый | Класс элемента, на который воздействовал пользователь |
| ClickTime | Дата | Время клика |
| ClickType | Текстовый | Тип клика |
| ClickCurrentStage | Целочисленный | Номер текущего слайда |
| ClickNextStage | Целочисленный | Следующий слайд |
| ClickText | Текстовый | Дополнительная информация |
| ClickNextTime | Целочисленный | Время следующего слайда |
| ClickStamp | Текстовый | Идетнтификатор, состоящий из номера следующего слайда и времени следующего слайда |

**Таблица AdStates**

Данная таблица содержит информацию о слайдах презентации. Данные об атрибутах приведены в таблице 15.

Таблица 16 AdStates.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | Целочисленный | Идентификатор |
| Type | Целочисленный | Тип слайда |
| Name | Текстовый | Имя слайда |
| AdId | Целочисленный | Идентификатор презентации |
| EndTime | Целочисленный | Конечная секунда видео, до которого идет воспроизведение |
| VideoUnitId | Целочисленный | Идентификатор текущего видео |
| ChainedHtml | Целочисленный | Связанный код разметки |
| IsFullPlay | Логический | Флаг, проигрывать ли видео полностью |
| IsStart | Логический | Флаг, обозначающий, является ли слайд стартовым |
| IsEnd | Логический | Флаг, обозначающий, является ли слайд конечным |

**Таблица VideoUnits**

Таблица, описывающая видео файл. Описание атрибутов приведено в таблице 16.

Таблица 17 VideoUnit.

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| Id | Целочисленный | Идентификатор |
| DateLoaded | Дата | Дата загрузки |
| Name | Текстовый | Имя видео |
| GeneratedName | Целочисленный | Сгенерированное имя |
| UserName | Целочисленный | Имя пользователя, загрузившего видео |
| Length | Целочисленный | Длина видео |
| FullPath | Целочисленный | Путь до файла |
| MimeType | Логический | Тип видео |

## Описание графического интерфейса

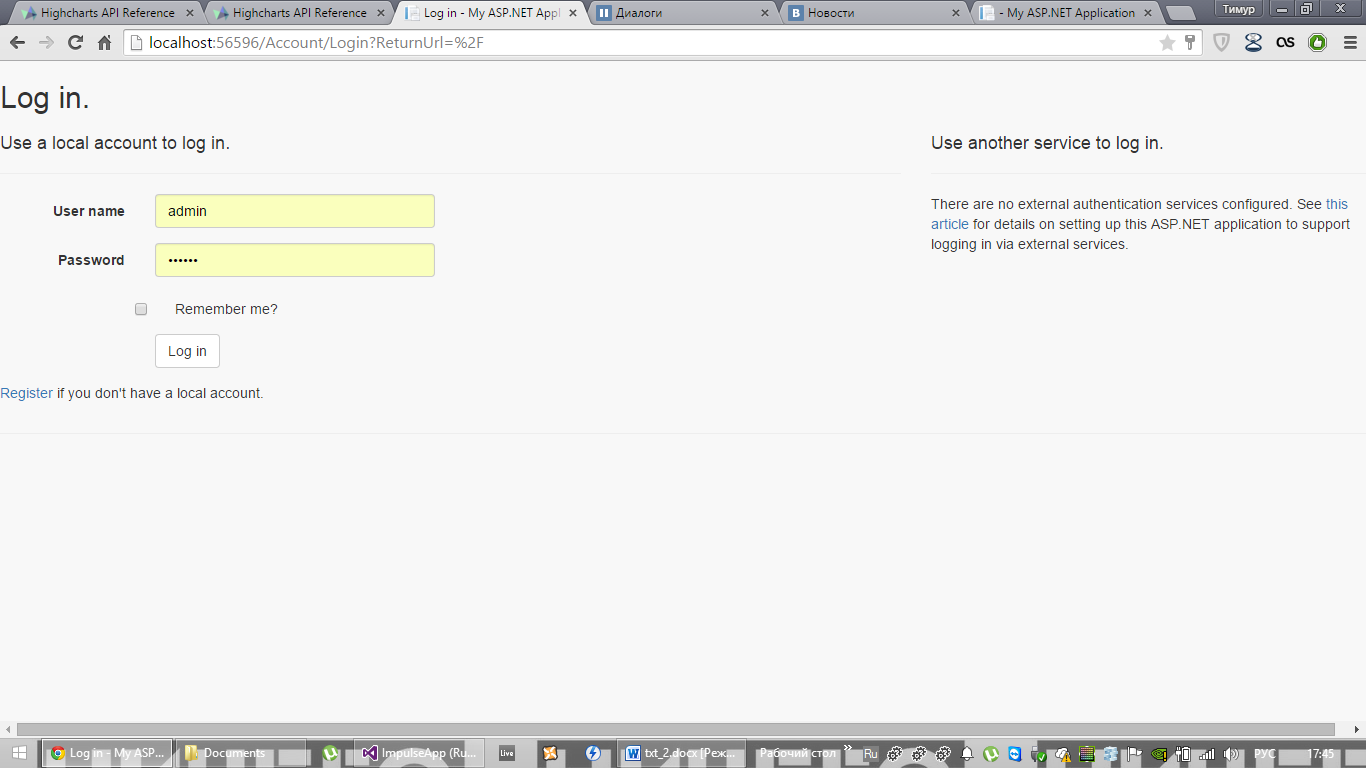
Весь основной функционал приложения доступен исключительно зарегистрированным пользователям, поэтому при входе в консоль администрирования, пользователь видит окно логина, представленное на рисунке 

Рис. 22 Форма входа.

Слева пользователь видит меню, из которого осуществляется доступ к различным компонентам системы. Внешний вид меню представлен на рисунке

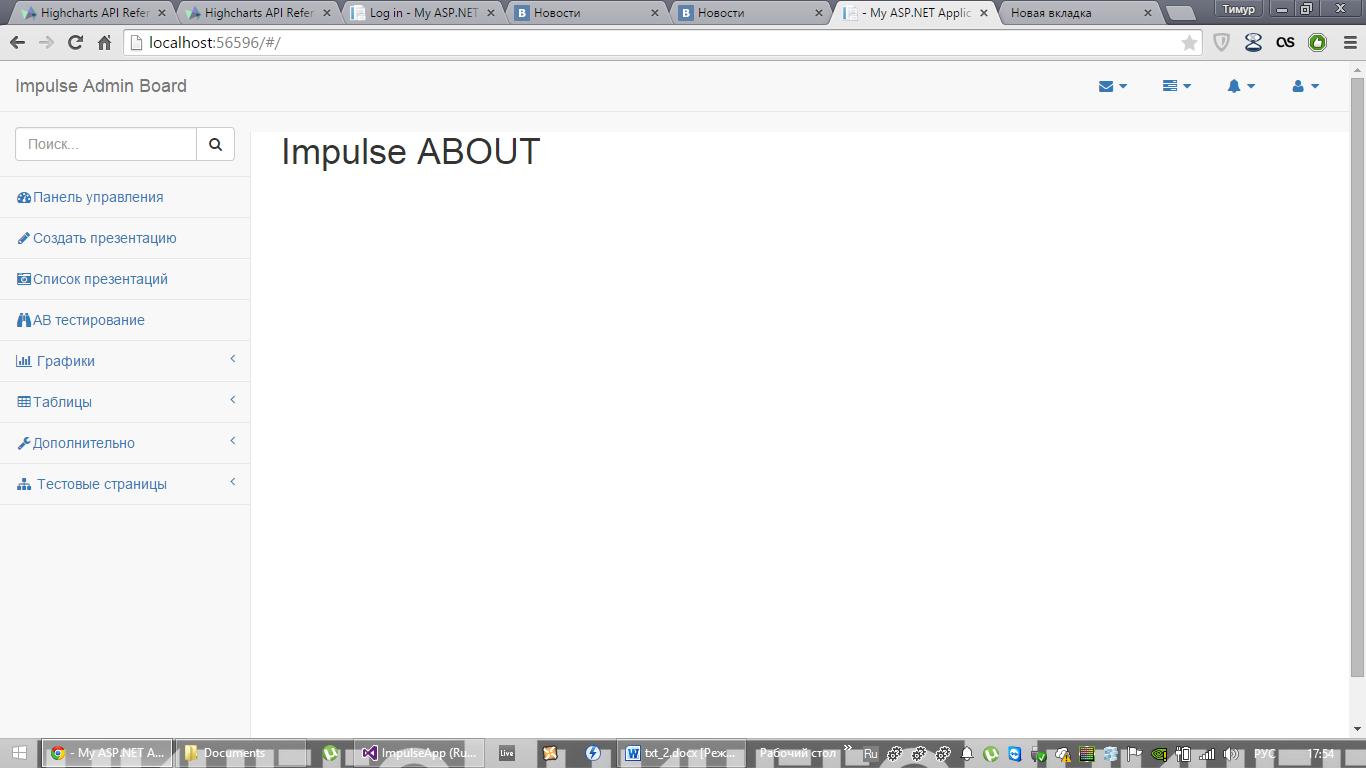


Рис. 23 Внешний вид меню.

При клике по пунктом меню, все соответствующие экраны открываются на той же странице без полного обновления, следуя принципам Single Page Application. Тем не менее, адресная строка, представляет собой url вида «host/#/path/to/page».

### Экран список презентаций

На данном экране отображается полный список презентаций пользователя. Внешний вид экрана представлен на рисунке 24

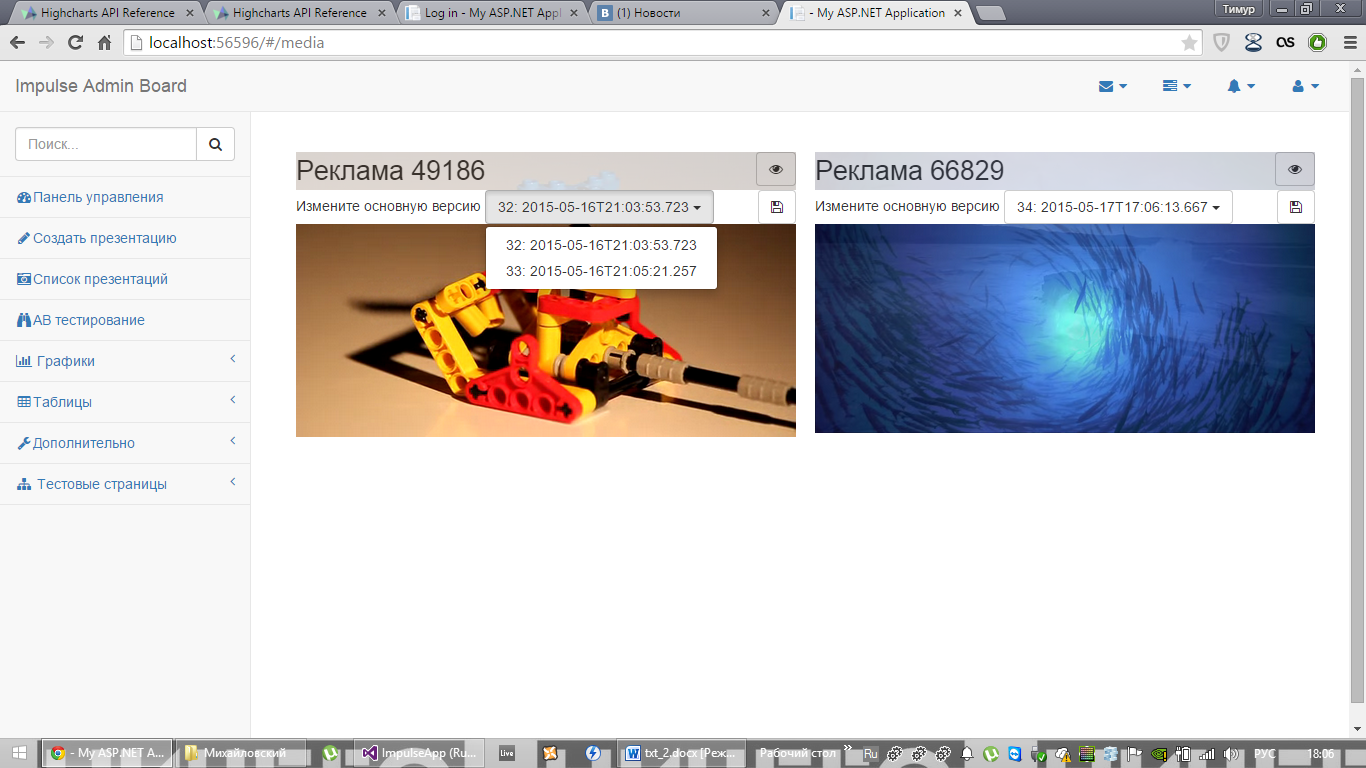


Рис. 24 Список презентаций.

На данном экране отображается картинка превью и имеется возможность выбора показа основного варианта по ссылке. При клике по значку глаза, открывается модальное окно с предпросмотром презентации. При этом статистики при данном просмотре собираться не будет. Внешний вид экрана предпросмотра представлен на рисунке 25.

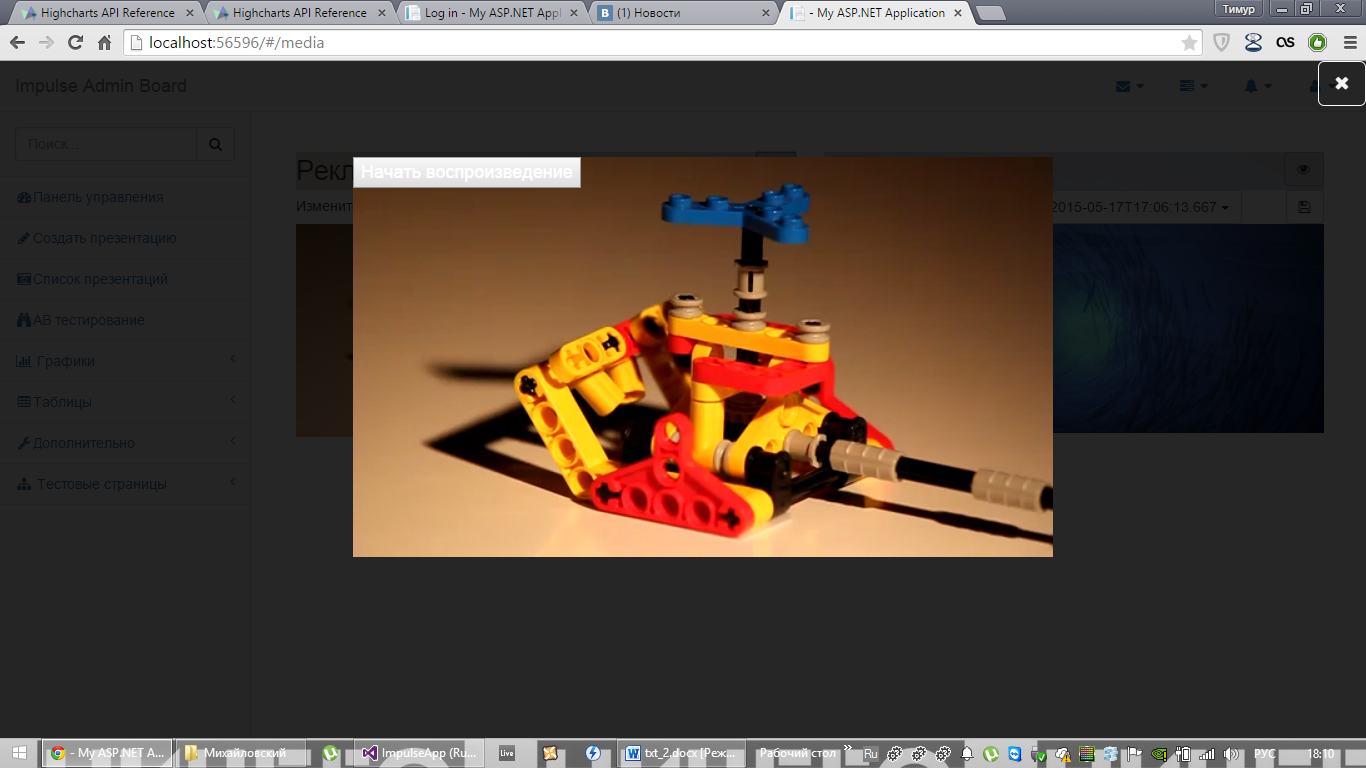


Рис. 25 Предпросмотр презентации.

### Экран «A\B тестирование»

Данный экран представляет собой набор элементов, необходимых при проведении сравнительного тестирования презентаций. Тестировать можно как два варианта одной презентации, так и совершенно разные презентации. Внешний вид экрана представлен на рисунке 26.

В блоках «Презентация A» и «Презентация B» производится выборка соответствующих презентаций и их вариантов. При смене значения верхнего выпадающего списка, представляющего собой список презентаций, в нижний выпадающий список автоматически подгружаются все варианты данных презентаций. При этом, выбор одного варианта валидируется и кнопка «Создать» автоматически блокируется. Внешний вид данного экрана с сообщениями валидации представлен на рисунке 27.

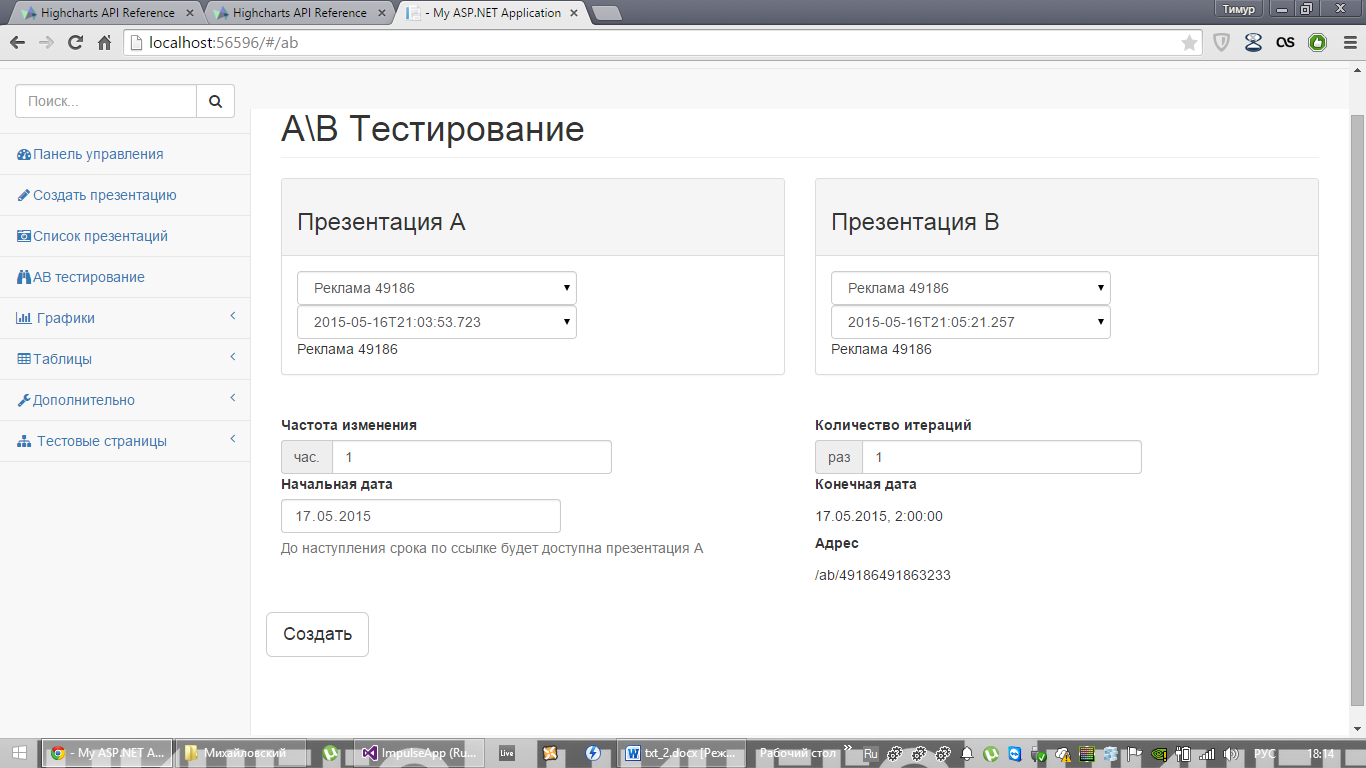


Рис. 26 A/B тестирование.

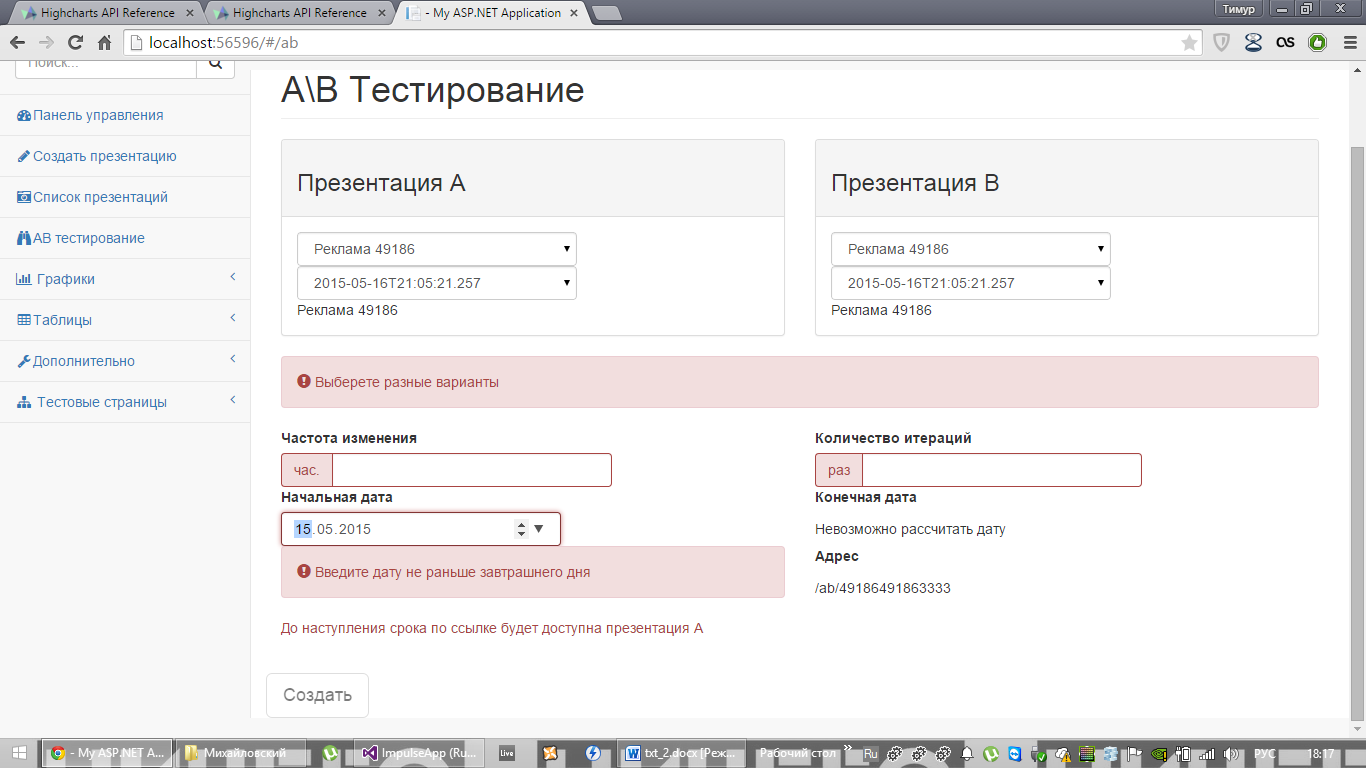


Рис. 27 Экран A/B тестирования с сообщениями валидации.

Предоставляется возможность настроить Частоту изменения – количество времени, которая будет держаться каждая презентация и количество итераций – количество смен презентаций. Таким образом, исходя из этих данных, рассчитывается конечная дата AB тестирования. Тестирование начинается с 00:00 начальной даты. По адресу будет показываться текущая презентация.

### Графики

Все графики формируются с помощью двух библиотек, либо highcharts, либо, chart.js. Для некоторых видов статистически представлений возможно загрузить сравнительные графики для одинакового промежутка времени. Например, как это представлено на рисунке 27

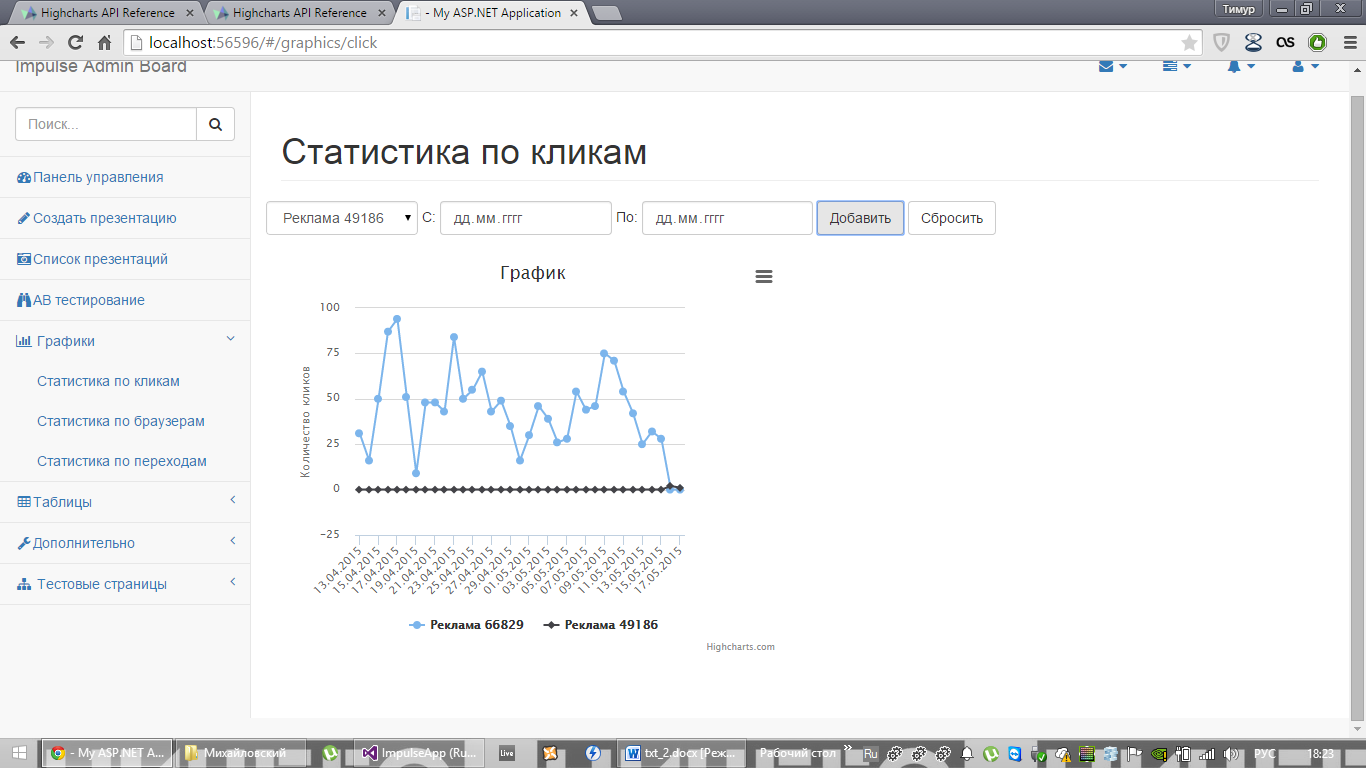


Рис. 28 Сравнительный анализ двух презентаций.

Графики, формирующиеся с помощью Chart.js имеют особую структуру. Это представлено на рисунке 29

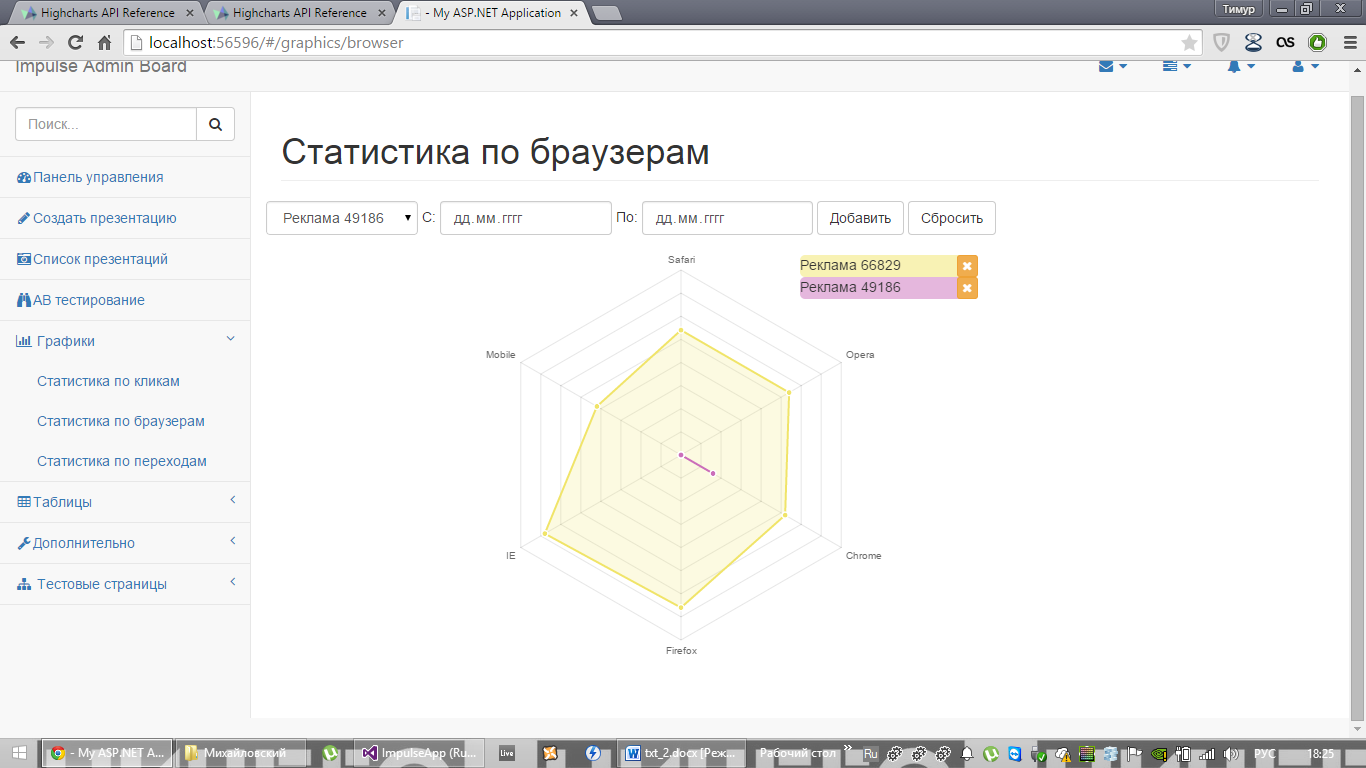


Рис. 29 График, сформированный с помощью chart.js.

Основным графиком является воронка просмотров, позволяющая определить, какой процент пользователей и насколько далеко посматривает слайды презентации. Внешний вид воронки просмотров представлен на рисунке 30

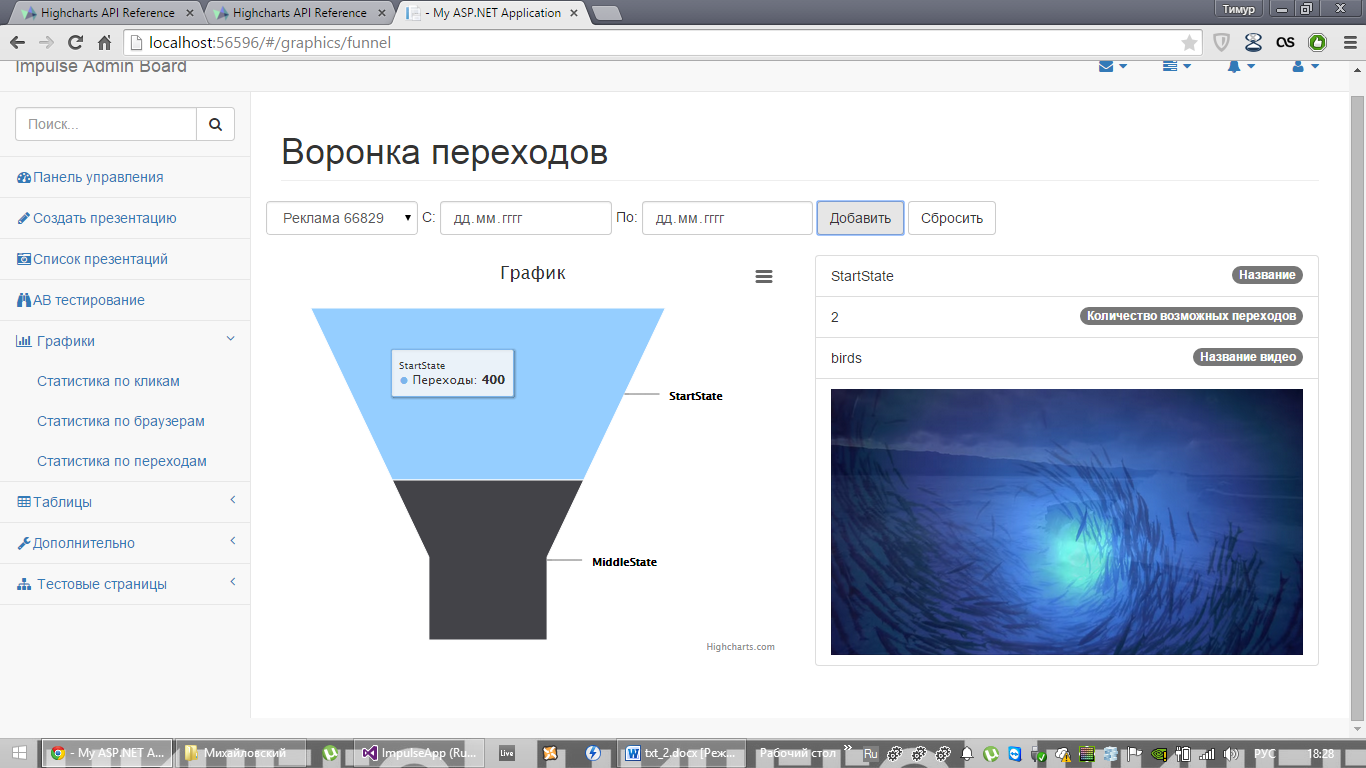


Рис. 30 Воронка переходов.

## Аппаратное обеспечение, необходимое для запуска системы

Для внедрения разработанной системы необходим сервер базы данных и сервер приложений. Ниже приведены требования к аппаратной части этих серверов более подробно:

Минимальные технические характеристики сервера БД:

* процессор – 3 ГГц;
* объем оперативной памяти –8 Гб;
* дисковая подсистема –250 Гб;
* клавиатура – Windows совместимая.

Минимальные технические характеристики сервера приложений:

* процессор –3 ГГц;
* объем оперативной памяти –10 Гб;
* дисковая подсистема – 50 Гб;
* клавиатура – windows совместимая.

Технические характеристики клиентских ПК приведены в таблице

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Windows** | **Mac** | **Linux** |
| **ОС** | * Windows XP SP 2+ * Windows Vista * Windows 7 * Windows 8 | Mac OS X 10.6 | Ubuntu 12.04+ Debian 7+ OpenSuSE 12.2+ Fedora Linux 17 |
| **Процессор** | Intel Pentium 4 | Intel | Intel Pentium 4 |
| **RAM** | 512 MB | | |

Таблица 18 Технические характеристики

# Заключение

В результате работы надо дипломным проектом была разработана система автоматизации и управления презентациями и рекламными кампаниями «Impulse». Были поэтапно пройден жизненный цикл проекта от концепции до полной реализации, включая некоторое тестирование. Важно отметить, что система согласована с другим проектом – редактором презентаций и рекламных проектов, поэтому был также получен опыт долговременной коллаборации при создании интегрированных систем. При разработке был применен принцип SCRUM. В соответствии с которым каждые 2 недели разрабатывался некоторый законченный функционал или модуль. В итоге разработанная система удовлетворяет как базовым требованиям вроде безопасности, так и другим важным требованиям:

* актуальность, так как на рынке отсутствуют системы простой полуавтоматизированной разработки презентаций, основанной на видео, совмещенной со сбором статистики;
* удобство использования, так как работа с веб приложением ничем не отличается от работы с обычным настольным приложением, что достигнуто максимально возможным количеством асинхронных запросов;
* независимость от конкретного места, ведь неважно где была начата разработка, все хранится в БД, к которой есть доступ из панели администрирования, доступной по сети Интернет;
* независимость от платформы, важно только наличие браузера, поддерживающего JavaScript, версия стандарта которого равна или выше того, на котором написан функционал презентационного слоя приложения;
* современный внешний вид, соответствующий актуальным трендам веб-дизайна, что повысит лояльность пользователей сервиса.

Разработка была осуществлена с помощью IDE Visual Studio, языков программирования C# 5.0, Javascript и SQL и платформы .NET Framework. 4.5. Были применены знания предметной области, концептуального моделирования, проектирования БД и разработки ПО.

# Список литературы

1. Официальный сайт Prezi [Электронный ресурс]. – Электрон. текстовые дан. – URL: prezi.com
2. Сайт Microsoft Developers Network [Электронный ресурс]. – Электрон. текстовые дан. – URL: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx
3. Nagel C. Professional C# 5.0 and .NET 4.5.1 (Programmer to Programmer) / Nagel C., Glynn J., Skinner M – Wiley 340c.
4. Сайт ECMA [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf
5. What's new in the C# 2.0 Language and Compiler"- Burrows, Chris - [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://msdn.microsoft.com/en-us/library/7cz8t42e(v=vs.80).aspx
6. "Overview of C# 3.0". - Hejlsberg, Anders - [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://msdn.microsoft.com/en-us/library/bb308966.aspx
7. "C# 4.0 - New C# Features in the .NET Framework 4"- Burrows, Chris - [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://msdn.microsoft.com/en-us/magazine/ff796223.aspx
8. Future directions for C# and Visual Basic. - Hejlsberg, Anders - [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://channel9.msdn.com/Events/BUILD/BUILD2011/TOOL-816T
9. Language feature implementation status. - [Электронный ресурс]. – Электрон. текстовые дан. – URL: https://github.com/dotnet/roslyn/wiki/Languages-features-in-C%23-6-and-VB-14
10. Wikipedia GitHub - [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://en.wikipedia.org/wiki/C\_Sharp\_(programming\_language)#cite\_ref-roslyn\_github\_lang\_design\_43-0
11. Adam Freeman Pro ASP.NET. / Adam Freeman – Wiley/2014. – 768  с.
12. Professional ASP.NET MVC 4 / on Galloway, Phil Haack, Brad Wilson, K. Scott Allen – WILEY/2014 – 538c.
13. Сайт Microsoft Developers Network [Электронный ресурс]. – Электрон. текстовые дан. – URL: https://msdn.microsoft.com/en-us/library/cc668201.aspx
14. Stevens Anderson Blog ASP.NET MVC Pipeline Lifecycle [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://blog.stevensanderson.com/2007/11/20/aspnet-mvc-pipeline-lifecycle/
15. API Library Programmable Web [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://www.programmableweb.com/apis/directory
16. OpBeat.com [Электронный ресурс]. – Электрон. текстовые дан. – URL: https://opbeat.com/blog/posts/the-changing-face-of-web-development/
17. Wikipedia [Электронный ресурс]. – Электрон. текстовые дан. – URL: https://en.wikipedia.org/wiki/Single-page\_application
18. Amazon Web Services [Электронный ресурс]. – Электрон. текстовые дан. – URL: http://aws.amazon.com/what-is-aws/
19. Kent, William. «A Simple Guide to Five Normal Forms in Relational Database Theory», Communications of the ACM 26 (2), Feb. 1983, pp. 120—125.
20. rtb-media [Электронный ресурс]. – Электрон. текстовые дан. – http://rtb-media.ru/wiki/
21. emarketer.com [Электронный ресурс]. – Электрон. текстовые дан. – <http://www.emarketer.com/Article.aspx?id=1006848>
22. IAB [Электронный ресурс]. – Электрон. текстовые дан. – http://www.iab.net/ODV2015