МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ФГБО ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

(ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)»

ИЖЕНЕРНО-ЭКОНОМИЧЕСКИЙ ИНСТИТУТ

Кафедра «Прикладная и бизнес-информатика»

**ОТЧЕТ ПО КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Проектирование информационных систем»

Тема: «Разработка информационной системы анонсера»

Студент(ка) группы:. ИЭC–166–15 Титов В.А.

(Ф.И.О.)

Руководитель к.т.н., Петров С.А.

(уч. степень, звание, Ф.И.О.)

|  |  |  |
| --- | --- | --- |
| Сдана на проверку |  |  |
| Возвращена на доработку |  |  |
| Допущена к защите |  |  |
| Оценка |  |  |
|  |

Москва 2017

Оглавление

[1. Описание предметной области 3](#_Toc484347690)

[2. Обоснование выбора объекта автоматизации 5](#_Toc484347691)

[2.1. Назначение, цели создания ИСА 5](#_Toc484347692)

[2.2. Общее описание 5](#_Toc484347693)

[2.3. Требования к функциональным характеристикам. 5](#_Toc484347694)

[3. Описание выбранных для реализации технологий 6](#_Toc484347695)

[4. Проектирование информационной системы 13](#_Toc484347696)

[5. Реализация информационной системы 21](#_Toc484347697)

[5.1. Архитектура разработанного приложения. 21](#_Toc484347698)

[5.2. Доменные сущности. 23](#_Toc484347699)

[5.3. Реализация доступа к данным. 26](#_Toc484347700)

[5.4. Реализация представления. 30](#_Toc484347701)

[5.5. Демонстрация работы приложения. 37](#_Toc484347702)

[Заключение 39](#_Toc484347703)

[Литература 40](#_Toc484347704)

## 1. Описание предметной области

В качестве предметной области для информационной системы (ИС) была выбрана деятельность рекламного веб-сервиса для игровых серверов (далее анонсер) – предоставление услуг по рекламированию(анонсированию) серверов, вывод информации о добавленных серверах в удобной системе структурированных данных для упрощенного поиска серверов для потенциальных игроков. Основной деятельностью анонсера является продажа рекламного места на веб-сервисе. Каждый анонс имеет свои параметры, которые включают в себя название, дату открытия, тип анонса, ссылку на сайт игрового сервера и параметры сервера.

Информационная система – организационно-техническая система, которая предназначена для выполнения информационно-вычислительных работ или предоставления информационно-вычислительных работ или предоставления информационно-вычислительных услуг, удовлетворяющих потребности системы управления и ее пользователей - управленческого персонала, внешних пользователей путем использования и/или создания информационных продуктов. Информационные системы существуют в рамках системы управления и полностью подчинены целям функционирования этих систем.

Основные задачи информационных систем:

* создание единого информационного пространства для всех пользователей системы, где бы они ни находились, реализация территориально распределенной системы управления,
* разделение прав доступа к данным в соответствии с полномочиями пользователей,
* обеспечение одновременного доступа множества пользователей к одним и тем же данным,
* однократность ввода данных, возможность их многократного извлечения,
* унификация форматов сбора и хранения данных,
* сокращение времени задержки получения данных в местах их использования,
* сокращение времени анализа информации,
* реализация многомерного анализа данных;

Информационная система анонсера должна обеспечить:

1. Добавление новых анонсов, с параметрами, которые включают в себя название, дату открытия, тип анонса, ссылку на сайт игрового сервера и параметры сервера.
2. Регистрация пользователей на веб-сервисе для авторизации и упрощенного мониторинга за приобретенными услугами. При регистрации записываются данные пользователей, включающая в себя ФИО, Email и псевдоним. Регистрация в сервисе должна реализовывать подтверждение регистрации через Email.
3. Регистрация посетителей анонсера и вывод количества индивидуальных пользователей в раздел статистики веб-сервиса.
4. Регистрация всех сделанных пользователем покупок рекламы, включающая в себя дату и время покупки, количество дней до окончания действия услуги.
5. Предоставление услуг баннерной рекламы.
6. Упрощенную систему поиска анонсов по параметрам серверов, дате открытия и ключевым словам в названии сервера.

## 2. Обоснование выбора объекта автоматизации

## 2.1. Назначение, цели создания ИСА

Информационная система анонсера (далее ИСА). Система предназначена для накопления, упорядочивания, переработки и передачи сведений о рекламных анонсах игровых серверов.

Основными целями создания ИСА являются:

* обеспечить сбор и первичную обработку исходной информации, необходимой рекламирования игрового сервера;
* повысить уровень привлечения игроков на сервера клиентов;
* повысить привлекательность игровых серверов;
* привлечь постоянных и новых клиентов.

## 2.2. Общее описание

Информационная система представляет собой хранилище данных, которое обеспечивает сохранность, упорядоченность, поиск и полный доступ к сведениям о анонсах игровых серверов.

## 2.3. Требования к функциональным характеристикам.

Система должна обеспечить возможность выполнения перечисленных ниже свойств:

* гибкость – настраивается на любую структуру, для широкого круга деятельности;
* безопасность – ограничение общего доступа к заранее определенным ресурсам БД анонсера;
* целостность и доступность – анонсы из базы данных сервиса должны быть представлены в общепринятых форматах, защищенных от изменений;
* возможность поиска – фильтрация анонсов по параметрам для извлечения требуемых из базы данных.

## 3. Описание выбранных для реализации технологий

В качестве технологий для разработки системы выступают:

* .NET Framework;
* ASP.NET CORE;
* EntityFramework;
* Language Integrated Query (LINQ).

Весь код программы написан в среде Microsoft Visual Studio 2015 на языке программирования C#.

Microsoft Visual Studio 2015 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных.

По умолчанию Visual Studio обеспечивает поддержку C#, C и C++, JavaScript, F# и Visual Basic. Visual Studio хорошо работает и интегрируется со сторонними приложениями, например Unity и Apache Cordova.

C# – объектно-ориентированный язык программирования. Разработанный в 1998 – 2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .Net Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Pascal, Модула, Smalltalk и, в особенности, Java – C#, опираясь на практику использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, в C# в отличии от C++, не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#, как и всем другим .NET – ориентированным языкам, многие возможности, которых лишены «классические» языки программирования.

.NET Framework – программная платформа, выпущенная компанией Microsoft в 2002 году. Основной платформой является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Считается, что платформа .NET Framework явилась ответом компании Microsoft на набравшую к тому времени большую популярность платформу Java компании Sun Microsystems. Хотя, .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под ОС семейств Microsoft Windows, существуют независимые проекты, позволяющие запускать программы .NET на некоторых других ОС.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчиков за счет предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах. Вторым принципом стала ориентация на системы, работающие под управлением семейства ОС Microsoft Windows.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код CIL (Common Intermediate Language). Затем код либо исполняется виртуальной машиной CLR, либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в нее JIT-компилятор «на лету» (just in time) преобразует промежуточный байт-код в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также заботиться о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке Framework Class Library (FCL). В FCL входят классы Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation и другие.

Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода веб-приложений: от небольших веб-сайтов до крупных веб-порталов и веб-сервисов.

С одной стороны, ASP.NET Core является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET Core фактически означает революцию всей платформы, ее качественное изменение.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на [GitHub](https://github.com/aspnet/).

ASP.NET Core построен на основе кросс-платформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS X, Linux. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой. То есть мы можем запускать веб-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания веб-приложения можно использовать традиционный IIS, либо кросс-платформенный веб-сервер Kestrel.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер Nuget. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку System.Web.dll.

ASP.NET Core включает в себя фреймворк MVC, который объединяет функциональность MVC, Web API и Web Pages. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель ASP.NET Core MVC. А Web Forms полностью ушли в прошлое.

Кроме объединения вышеупомянутых технологий в одну модель в MVC был добавлен ряд дополнительных функций.

Одной из таких функций являются тэг-хелперы (tag helper), которые позволяют более органично соединять синтаксис html с кодом С#.

ASP.NET Core характеризуется расширяемостью. Фреймворк построен из набора относительно независимых компонентов. И мы можем либо использовать встроенную реализацию этих компонентов, либо расширить их с помощью механизма наследования, либо вовсе создать и применять свои компоненты со своим функционалом.

Также было упрощено управление зависимостями и конфигурирование проекта. Фреймворк теперь имеет свой легковесный контейнер для внедрения зависимостей, и больше нет необходимости применять сторонние контейнеры, такие как Autofac, Ninject. Хотя при желании их также можно продолжать использовать.

В качестве инструментария разработки мы можем использовать последние выпуски Visual Studio, начиная с версии Visual Studio 2015. Кроме того, мы можем создавать приложения в среде Visual Studio Code, которая является кросс-платформенной и может работать как на Windows, так и на Mac OS X и Linux.

При разработке в Visual Studio 2015/2017 проекты приложений имеют встроенную поддержку с такими популярными инструментами, как Bower, Grunt, Gulp, который позволяют управлять скриптами JavaScript и стилями CSS, автоматизировать и оптимизировать процесс веб-разработки.

Платформа Entity Framework представляет собой набор технологий ADO.NET, обеспечивающих разработку приложений, связанных с обработкой данных.  Архитекторам и разработчикам приложений, ориентированных на обработку данных, приходится учитывать необходимость достижения двух совершенно различных целей.  Они должны моделировать сущности, связи и логику решаемых бизнес-задач, а также работать с ядрами СУБД, используемыми для сохранения и получения данных.  Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода приложения.

Language Integrated Query (LINQ) — это технология, позволяющая выполнять запросы к источникам данных, используя языки программирования платформы .NET Framework и синтаксис, похожий на язык запросов SQL.

Впервые LINQ появился в Visual Studio 2008 и, постепенно развиваясь, поддерживал всё новые источники данных. На текущий момент, в зависимости от вида источника данных, выделяют несколько разновидностей LINQ:

* LINQ to Objects: запросы к коллекциям;
* LINQ to Entities: запросы к БД через Entity Framework;
* LINQ to Sql: запросы к MS SQL Server;
* LINQ to XML: запросы к XML файлам;
* LINQ to DataSet: запросы к объекту DataSet (не путать с DbSet, используемый EF);
* Parallel LINQ (PLINQ): выполнение параллельных запросов.

Используя, некоторые новые особенности языка, позволяет использовать SQL-подобный синтаксис непосредственно в коде программы, написанной, например, на языке C#:

* Анонимная типизация
* Методы расширения
* Лямбда-исчисления
* Дерево выражений
* Стандартные операторы языка запросов

Общие сведения о выражениях запросов:

1. Выражения запросов можно использовать для запроса и преобразования данных, полученных из любого источника данных, поддерживающего LINQ. Например, один и тот же запрос может извлекать данные из базы данных SQL и на выходе создавать поток XML.
2. Выражения запросов достаточно легко изучить, поскольку они во многом схожи с элементами языка C#.
3. Все переменные в выражении запросов имеют строгую типизацию, хотя во многих случаях не требуется явным образом указывать тип, поскольку компилятор его вычисляет.
4. Выполнение запроса не происходит до использования переменной запроса в операторе foreach.
5. При компиляции выражения запросов преобразуется в вызовы методов стандартных операторов запроса согласно правилам, указанным в спецификациях языка C#. Любой запрос, которые может быть выражен с помощью синтаксиса запросов, также может быть выражен с помощью синтаксиса методов. Однако синтаксис запросов в большинстве случаев понятнее и лаконичнее.
6. При создании запросов LINQ рекомендуется использовать синтаксис запросов везде, где это возможно, и синтаксис методов, если это необходимо. Между этими двумя видами не существует различий ни в семантике, ни в производительности. Выражения запросов просто более понятны, чем соответствующие им выражения, написанные с помощью синтаксиса методов.
7. Для некоторых операций запросов, таких как [Count<TSource>](https://msdn.microsoft.com/ru-ru/library/bb338038.aspx) иди [Max](https://msdn.microsoft.com/ru-ru/library/bb335614.aspx), нет соответствующих предложений в выражениях запросов, поэтому их необходимо создавать с помощью вызова методов. Синтаксис запросов и синтаксиса методов можно сочетать друг с другом различными способами.
8. В зависимости от типа, к которому применяется запрос, выражения запросов можно компилировать в деревья выражений или в делегаты. Запросы [IEnumerable<T>](https://msdn.microsoft.com/ru-ru/library/9eekhta0.aspx) компилируются в делегаты. Запросы [IQueryable](https://msdn.microsoft.com/ru-ru/library/system.linq.iqueryable.aspx) и [IQueryable<T>](https://msdn.microsoft.com/ru-ru/library/bb351562.aspx) компилируются в деревья выражений.

## 4. Проектирование информационной системы

Определение цели проекта – это основная часть при проектирование информационных систем. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

* требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
* требуемую пропускную способность системы;
* требуемое время реакции системы на запрос;
* безотказную работу системы в требуемом режиме, иными словами - готовность и доступность системы для обработки запросов пользователей;
* простоту эксплуатации и поддержки системы;
* необходимую безопасность.

Производительность является главным фактором, определяющим эффективность системы. Хорошее проектное решение служит основой высокопроизводительной системы.

Проектирование информационных систем охватывает три основные области:

* проектирование объектов данных, которые будут реализованы в базе данных;
* проектирование сервисов, информации, экранных представлений, которые будут обеспечивать выполнение запросов к данным;
* учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

В реальных условиях проектирование - это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

К любому проекту предъявляется ряд абсолютных требований, например, максимальное время разработки проекта, максимальные денежные вложения в проект и т.д. Одна из сложностей проектирования состоит в том, что оно не является такой структурированной задачей, как анализ требований к проекту или реализация того или иного проектного решения.

Жизненный цикл программного обеспечения представляет собой модель его создания и использования. Модель отражает его различные состояния, начиная с момента возникновения необходимости в данном ПО и заканчивая моментом его полного выхода из употребления у всех пользователей.

Известны следующие модели жизненного цикла:

* Каскадная модель. Переход на следующий этап означает полное завершение работ на предыдущем этапе.
* Поэтапная модель с промежуточным контролем. Разработка ПО ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоемкость процесса разработки по сравнению с каскадной моделью; время жизни каждого из этапов растягивается на весь период разработки.
* Спиральная модель. Особое внимание уделяется начальным этапам разработки - выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание некой версии продукта или какого-либо его компонента, при этом уточняются характеристики и цели проекта, определяется его качество и планируются работы следующего витка спирали.

Целью построения логической модели является получение графического представления логической структуры исследуемой предметной области. Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Сущности описывают объекты, являющиеся предметом деятельности предметной области, и субъекты, осуществляющие деятельность в рамках предметной области. Свойства объектов и субъектов реального мира описываются с помощью атрибутов. Взаимоотношения между сущностями иллюстрируются с помощью связей. Правила и ограничения взаимоотношений описываются с помощью свойств связей. Обычно связи определяют либо зависимости между сущностями, либо влияние одной сущности на другую.

Основные требования к содержанию модели:

1. Логическая модель должна отображать все сущности и связи, значимые для той цели, ради которой мы ее рисуем.

2. Все объекты модели (и сущности, и связи) должны быть именованы.

Именование сущностей и связей должно выполняться в терминах предметной области.

3. Для связей должна быть указана кратность (один — многие).

4. Для каждой связи должно быть указано направление чтения.

5. Для сущностей должны быть указаны как минимум основные атрибуты.

Ниже представлена диаграмма прецедентов информационной системы анонсера, которая отражает отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. На ней изображено три действующих лица — это обычный пользователь и наследующие от него администратор сервера(клиент) и администратор сервиса. Обычный пользователь имеет только возможность просмотра существующих в БД ИСА анонсов с пометкой публикации, остальные действующие лица наследуют эту способность, администратор сервиса может просматривать весь список добавленных анонсов в БД ИСА, так же эта способность включает в себя выставление пометок о публикации анонсов. Последний из действующих лиц это непосредственно администратор сервера(клиент) он может добавлять анонсы в ИСА, а также покупать рекламные баннеры для публикации на данном сервисе.

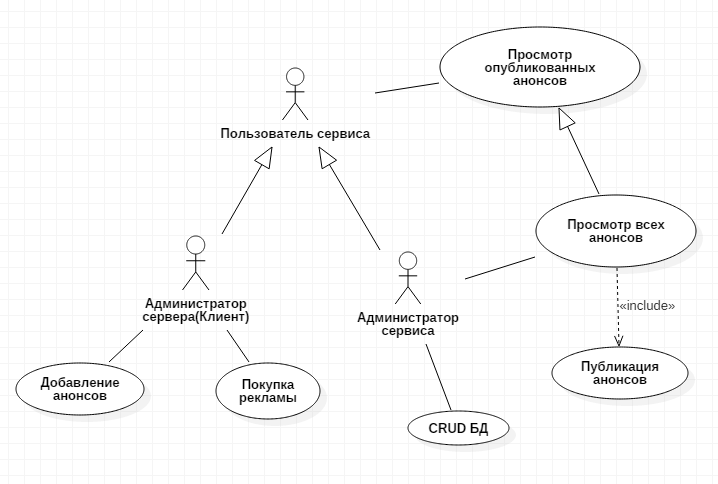


Рис. 1. Диаграмма прецедентов.

Ниже представлена диаграмма классов ИСА. Есть две доменные сущности, это модель анонса и модель пользователя.

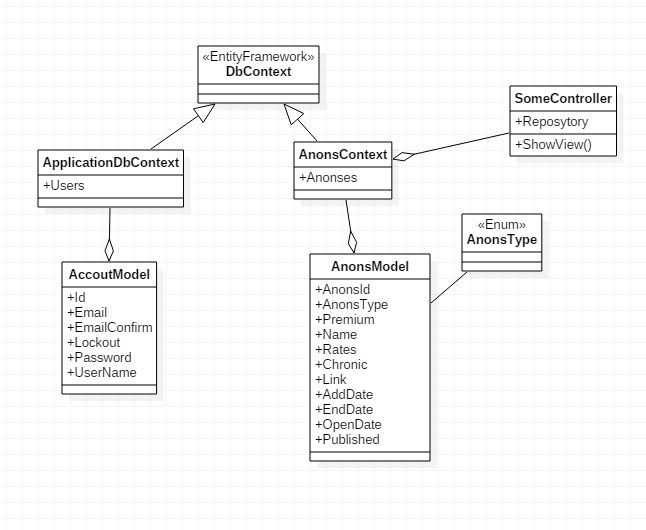


Рис. 2. Диаграмма классов.

Анонс имеет открытые свойства:

1. AnonsId – для идентификации анонса в БД ИСА.
2. AnonsType – тип анонса, для определения привилегированных анонсов.
3. Premium – буловая переменная означающая имеет ли анонс статус премиум.
4. Name – имя анонса.
5. Rates – рейты сервера для которого добавлен анонс.
6. Chronic – хроники сервера для которого добавлен анонс.
7. Link – ссылка на сайт игрового сервера.
8. AddDate – дата добавления анонса.
9. EndDate – дата окончания периода публикации анонса.
10. OpenDate – дата открытия игрового сервера.
11. Published – буловая переменная означающая опубликован анонс или нет.

Класс пользователя имеет атрибуты:

1. Id – для идентификации пользователя в БД ИСА.
2. Email – электронная почта пользователя.
3. EmailConfirm – буловая переменная для определения подтвердил ли пользователь регистрацию через электронную почту.
4. Lockout – закрытость.
5. Password – пароль от аккаунта пользователя.
6. UserName – имя пользователя.

Далее представлена диаграмма развёртывания ИСА. На диаграмме представлено физическое развертывание артефактов на узлах.

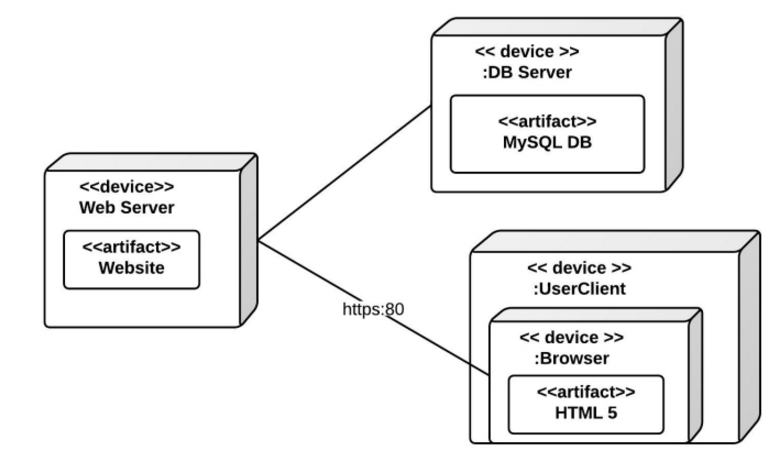


Рис. 3. Диаграмма развёртывания.

Так же представлена диаграмма последовательностей работы пользователя с веб сервисом. Для получения данных с сайта пользователь отправляет запрос в менеджер подключения с запросом просмотра интересующего его подбора серверов.

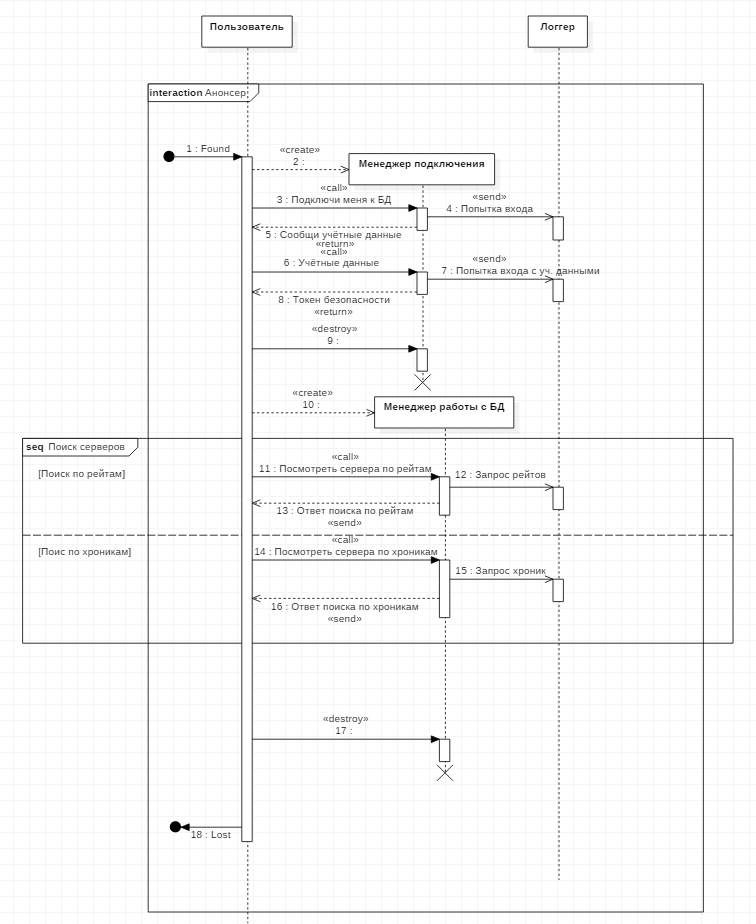


Рис. 4. Диаграмма последовательности.

Ниже представлена диаграмма вариантов использования, на которой показано разложение некоторой деятельности на её составные части. Диаграмма демонстрирует варианты работы с сайтом в зависимости от желания пользователя, а также показывает процесс действий от пользователя

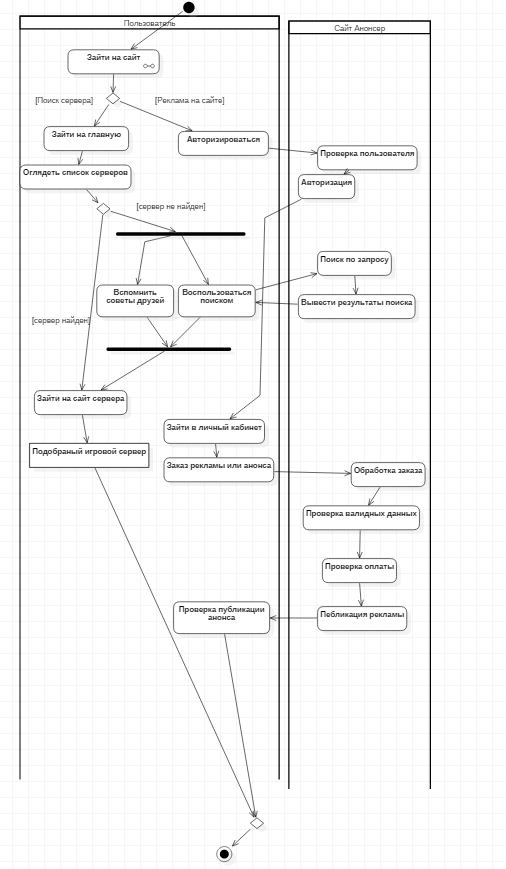


Рис. 5. Диаграмма вариантов использования.

## 5. Реализация информационной системы

## Архитектура разработанного приложения.

Для реализации данной программы я выбрал шаблон проектирования модель-представление-контроллер, сокращенно MVC. Данный шаблон предполагает разделение приложения на три основных компонента: модель, представление и контроллер.

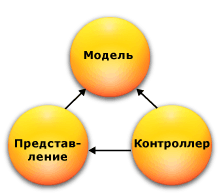


Рис. 6. Шаблон проектирования MVC.

В состав платформы MVC входят следующие компоненты.

* модели (Домен). Объекты моделей являются частями приложения, реализующими логику для домена данных приложения. Объекты моделей часто получают и сохраняют состояние модели в базе данных;
* представления. Представления служат для отображения пользовательского интерфейса приложения. Пользовательский интерфейс обычно создается на основе данных модели;
* контроллеры. Контроллеры осуществляют взаимодействие с пользователем, работу с моделью, а также выбор представления, отображающего пользовательский интерфейс. В приложении MVC представления только отображают данные, а контроллер обрабатывает вводимые данные и отвечает на действия пользователя. Например, контроллер может обрабатывать строковые значения запроса и передавать их в модель, которая может использовать эти значения для отправки запроса в базу данных.

Шаблон MVC позволяет создавать приложения, различные аспекты которых (логика ввода, бизнес-логика и логика интерфейса) разделены, но достаточно тесно взаимодействуют друг с другом. Эта схема указывает расположение каждого вида логики в приложении. Пользовательский интерфейс располагается в представлении. Логика ввода располагается в контроллере. Бизнес-логика находится в модели. Это разделение позволяет работать со сложными структурами при создании приложения, так как обеспечивает одновременную реализацию только одного аспекта. Например, разработчик может сконцентрироваться на создании представления отдельно от бизнес-логики.

Технология ASP.NET CORE позволяет реализовывать шаблон MVC встроенными инструментами, подключаемыми через библиотеку System.Web.Mvc.

При создании нашего приложения рассмотрим класс StartUp.cs, в методе *void ConfigureServices(IServiceCollection services)* мы видим подключение сервиса MVC.



Рис. 7. Подключение MVC сервисов в ASP.NET CORE.

Придерживаясь этого шаблона, мы разделяем программу на доменные данные, контроллеры и представления, при этом структура файлов делится по папкам.

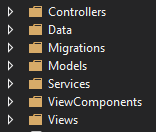


Рис. 7. Структура папок программы ИСА.

## Доменные сущности.

Для ИСА было реализовано две доменных сущности:

1. Пользователь.
2. Анонс.

Классы принадлежащие модели пользователя реализованными встроенными возможностями ASP.NET Identity Framework 2.0.

Основные преимущества использования ASP.NET Identity Framework 2.0

* использование единой системы авторизации как для MVC проекта, так и для WebAPI. Более того, можно на одном сайте (домене) настроить авторизации в зависимости от URL. Так например если все вызовы WebAPI начинаются с /api - для этой части сайта можно настроить проверку авторизации на основе токенов. Тогда как для основного сайта удобнее будет использовать Cookies;
* простота подключения;
* роли и авторизация на основе утверждений (Claims);
* возможность авторизации через социальные сети.

Фреймворк сам подключает нужные пакеты и на основе реализованных классов модели пользователя делает запись в базу данных создавая нужные таблицы.

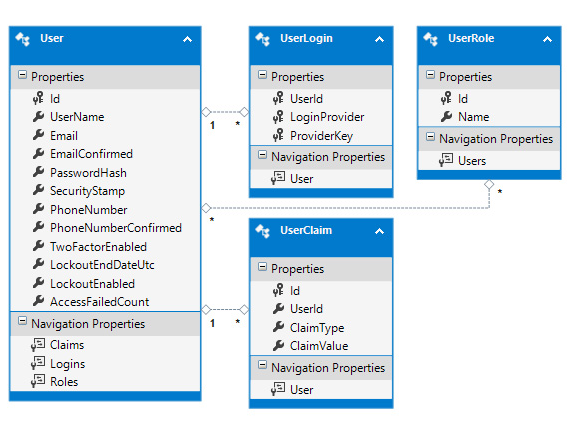


Рис. 8. Модель пользователя.

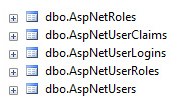


Рис. 9. Созданные таблицы в базе данных.

Модель анонса реализована на основе диаграммы классов (рис. 2) и представляет собой DTO объект, содержащий только поля для данных, не включая в себя поведения.

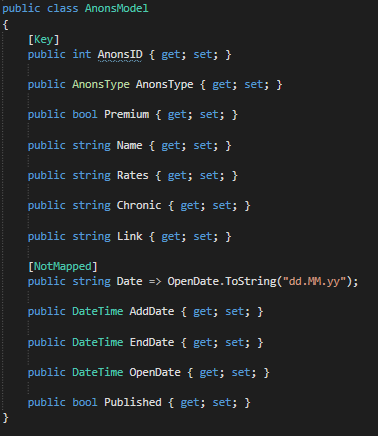


Рис. 10. Класс доменной модели анонса.

Над некоторыми полями есть атрибуты, поставленные для корректной работы с фреймворком доступа к данным. Атрибут [Key] означает первичный ключ в базе данных, а атрибут [NotMapped] означает что данное поле не учитывается при миграции данных в таблицу базы данных.

Тип анонса это один из вариантов из перечисления всех возможных типов анонса.

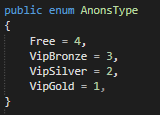


Рис. 11. Перечисление типов анонса.

Типы анонса:

1. Бесплатный(FREE) – анонсы не выделяются и имеют самый нижний уровень привилегий.
2. Бронзовый ВИП(VipBronze) – анонсы выделяются (бронзовый цвет), при сортировке на сайте всегда выше бесплатных.
3. Серебряный ВИП(VipSilver) – анонсы выделяются (серебряный цвет), при сортировке на сайте всегда выше бронзовых.
4. Золотой ВИП(VipGold) – анонсы выделяются (золотой цвет), при сортировке на сайте всегда выше всех.

## Реализация доступа к данным.

Для хранения данных я использую встроенные возможности Visual Studio 2015 – MS SQL. А для доступа к данным используется Entity Framework – которая предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL.

Для подключения я открыл окно управления NuGet-пакетами в окне поиска ввел слово "Entity" и выбрал пакет Entity Framework:

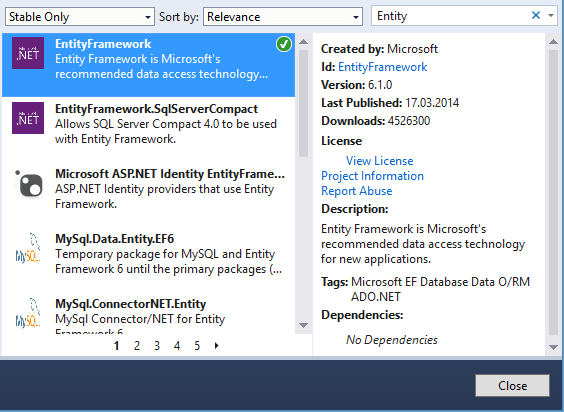


Рис. 12. Установка Entity Framework.

Во время разработки программы анонсера я использовал CodeFirst подход для реализации слоя доступа к данным. При CodeFirst подходе модель EDMX вообще не используется, и вы вручную настраиваете классы C# объектной модели (данный подход поддерживает как генерацию сущностных классов из существующей базы данных, так и создание базы данных из созданной вручную модели объектов C#). Очевидно, что это подходит для программистов, хорошо знакомых с синтаксисом C#.

Для Entity Framework есть Code First подход, где главным становится код, а не база.

Преимущества:

1. Никакого генерированного кода.
2. База — это снова просто хранилище.
3. Простейшая установка среды разработки.

Для реализации подхода CodeFirst был создан класс контекста данных анонса на основе доменной модели класса AnonsModel.cs

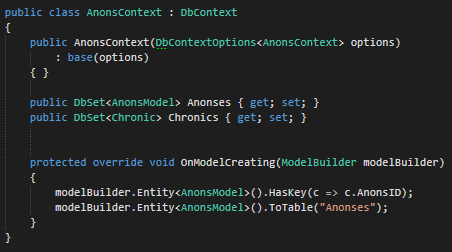


Рис. 13. Класс контекста данных анонса.

Основу функциональности Entity Framework составляют классы, находящиеся в пространстве имен System.Data.Entity. Среди всего набора классов этого пространства имен следует выделить следующие:

1. DbContext: определяет контекст данных, используемый для взаимодействия с базой данных.
2. DbModelBuilder: сопоставляет классы на языке C# с сущностями в базе данных.
3. DbSet/DbSet<TEntity>: представляет набор сущностей, хранящихся в базе данных.

В любом приложении, работающим с БД через Entity Framework, нам нужен будет контекст (класс производный от DbContext) и набор данных DbSet, через который мы сможем взаимодействовать с таблицами из БД. В данном случае таким контекстом является класс UserContext.

В конструкторе этого класса вызывается конструктор базового класса, в который передается строка "DbConnection" - это имя будущей строки подключения к базе данных. В принципе мы можем не использовать конструктор, тогда в этом случае строка подключения носила бы имя самого класса контекста данных.

И теперь нам надо установить подключение к базе данных. Для установки подключения обычно используется файл конфигурации приложения. В моем случае это файл appsettings.json, в него мы добавляем строку подключения.



Рис. 14. Строка подключения.

Далее через консоль вызываем команду миграции в базу данных которая автоматически создаст таблицу с полями, соответствующими классу AnonsModel.cs. Так как в программе используется система идентификации Identity 2.0, которая так же имеет свой контекст данных требуется указывать определенный контекст при вызове команды update-database.



Рис. 15. Команда миграции.

При старте программы если нет записей в базе данных был создан класс инициализации для заполнения тестовыми записями.

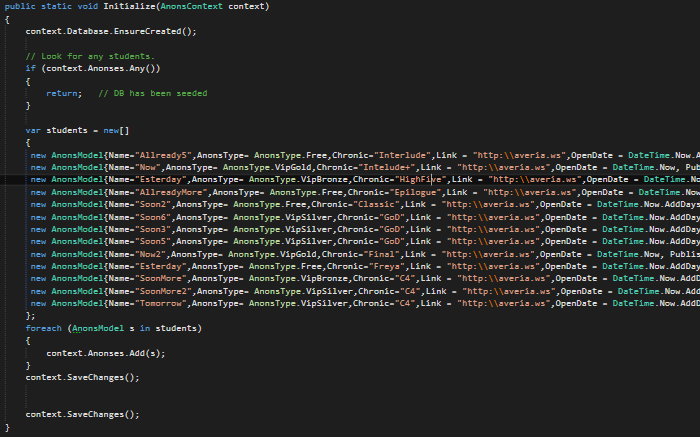


Рис. 16. Листинг инициализации базы данных.

Для доступа к записям из базы данных запрашиваю данные из переменной класса контекста данных по средствам LINQ запросов.

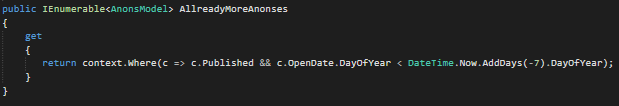


Рис. 17. Метод получения всех записей из базы данных у которых дата открытия была больше чем 7 дней назад.

## Реализация представления.

Для реализации представления используется движок Razor. Razor –интеллектуальный обработчик программного кода динамических веб-страниц на *ASP.NET*. Имеет простой, интуитивно понятный, синтаксис встраивания программного кода в веб-страницы. Razor это механизм визуализации (view engine) поддерживаемый NET*.*Framework в рамках ASP*.*NET и специально предназначен для создания веб-приложений.

При создании анонсера самое главное это главная страница потому что она несет основную информацию, находящуюся в информационной системе и больше всего интересную конечному пользователю. Для начала я создал главную основную страницу для шаблона со стандартным названием \_Layout.cshtml, которая будет служить телом веб-сайта анонсера. Этот файл имеет вид как обычная разметка html, но также позволяет использовать элементы с#.

В элементе Body файла \_Layout.cshtml пишется строка отображения конкретного контента. Информация о конкретном контенте приходит из контроллеров при каком-то действии со стороны пользователя.



Рис. 18. Листинг отображения контента.

При загрузке страницы будет посылаться через протокол HTTP GET запрос на контроллер Home, который в свою очередь вызывает метод Index запускающий асинхронное отображение View.

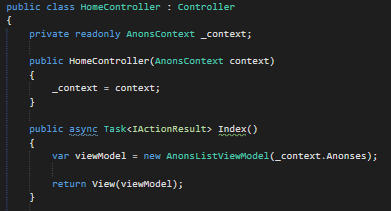


Рис. 19. Листинг контроллера Home.

Во время этого процесса на основе модели данных собирается разметка html и возвращается конечному пользователю в виде веб страницы. Для вывода разметки используются данные из класса AnonsListViewModel, который представляет собой доменные данные для отображения на странице.

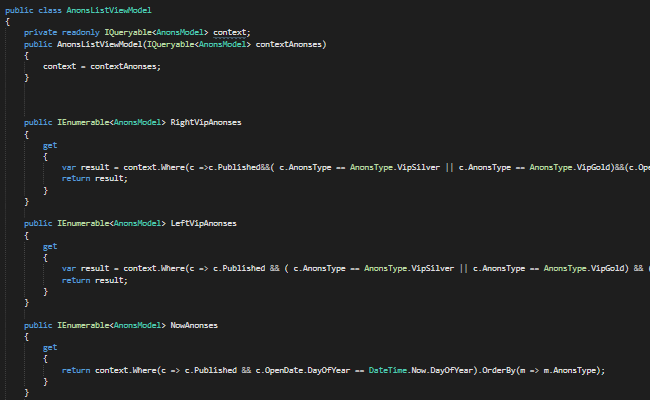


Рис. 20. Листинг класса AnonsListViewModel.

Класс AnonsListViewModel содержит поля перечислений, которые представляют собой поля из базы данных удовлетворяющие критериям вывода на экран.

Анонсы на главной странице разделены на несколько групп:

* VIP уже открылись;
* VIP скоро откроются;
* сегодня;
* вчера;
* завтра;
* скоро откроются;
  + ближайшие 7 дней;
  + через неделю и более.
* уже открылись.
  + предыдущие 7 дней;
  + неделю назад и более.

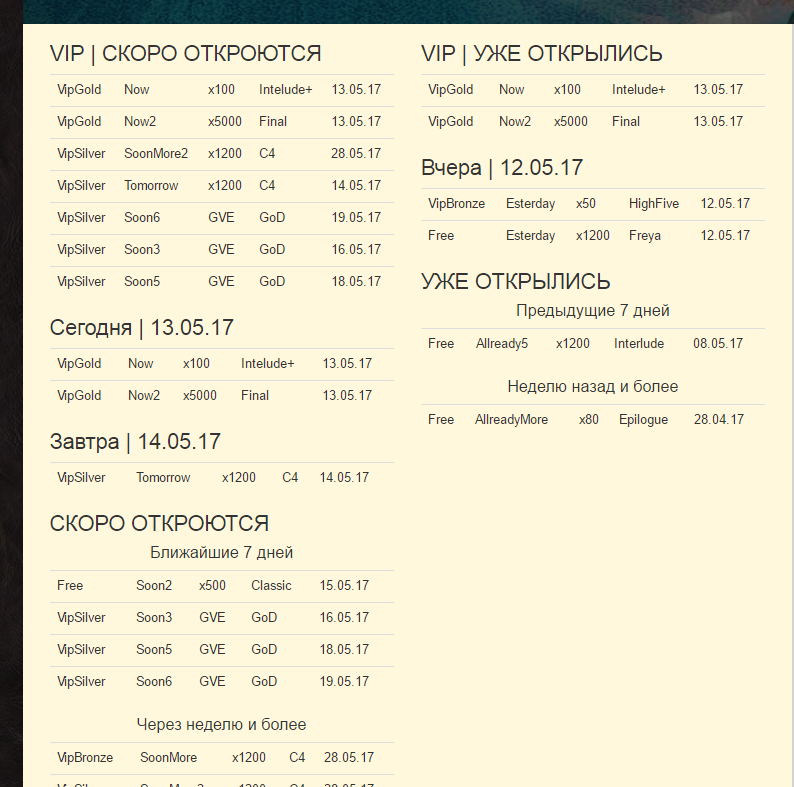


Рис. 21. Таблицы анонсов.

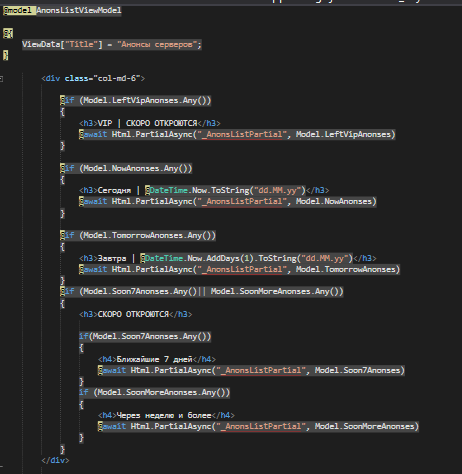
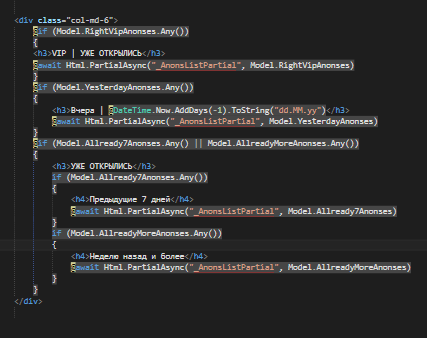
 

Рис. 22. Разметка контента таблиц анонсов на основе модели.

Так как код для вывода анонса на основе перечисления повторяется он был выделен в отдельный файл, который считается частью разметки и асинхронно загружается при вызове команды:

@await Html.PartialAsync("\_AnonsListPartial", Model.YesterdayAnonses)

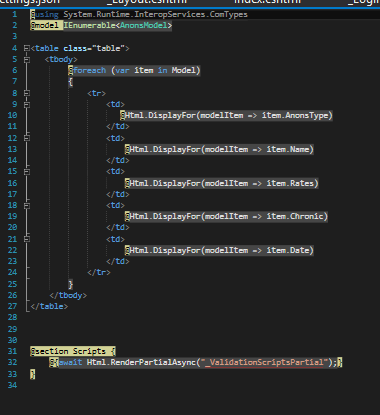


Рис. 23. Листинг разметки отображения перечислений анонсов.

Так же для удобства пользователя просматривать данные из ИСА было реализовано боковое меню поиска с вариантами поиска по хроникам и по рейтам. Так же панель с отображением всех хроник и информацией о количестве серверов, использующих эти хроники. Боковое меню реализовано с помощью компонентов отображения или View Component.

View Component представляет новую концепцию, которая появилась в ASP.NET MVC Core, и которая во многом схожа с частичными представлениями. При этом View Component предназначен для таких задач, которые, возможно, сложно решить с помощью одних частичных представлений, например, создание динамических меню, облако тегов, панель логина, корзина покупок и так далее.

View Component состоит из двух частей: класса на C# и частичного представления, которое вызывает методы этого класса. При этом View Component не может обрабатывать HTTP-запросы, а генерируемый им контент включается в код родительского представления, в котором вызывается компонент.

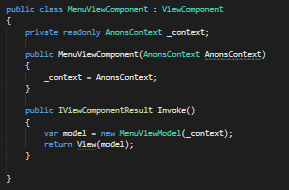


Рис. 24. Класс компонента представления меню.

На основе класса модели представления MenuViewModel строится разметка для отображения меню для поиска. Так как компонент не может обрабатывать HTTP запросы, все запросы перенаправляются на HomeController в перегруженный метод Search.

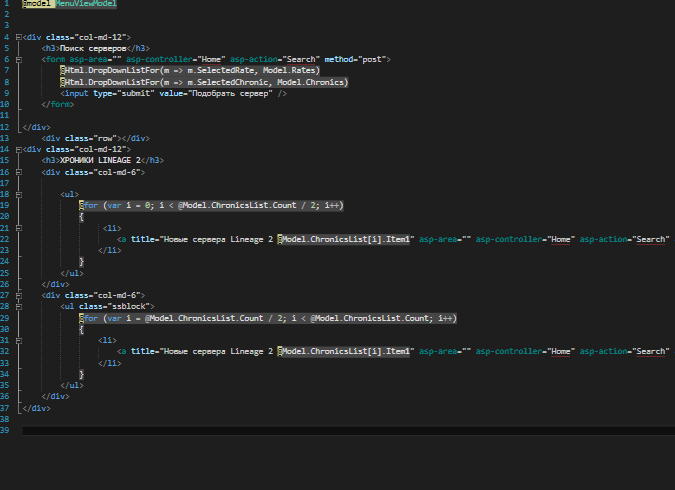


Рис. 25. Листинг разметки меню.

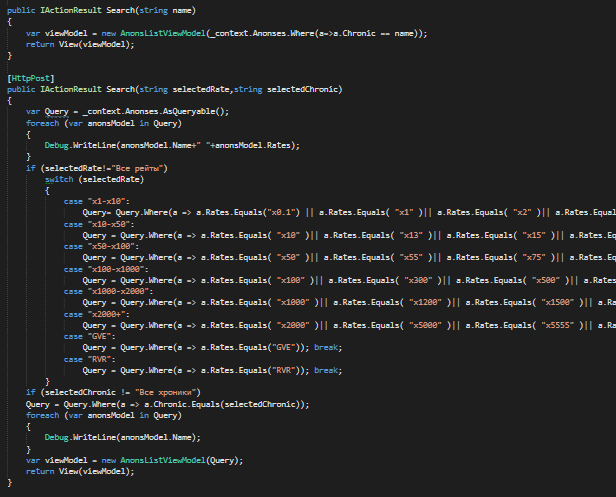


Рис. 26. Метод поиска контроллера Home.

При выборе хроник на панели меню отправится GET запрос на контроллер Home, и вызовется метод Search с параметром равным имени искомых хроник. На основе имени хроник происходит выборка из базы данных подходящих под условия анонсов и выводится на экран как новая html разметка.

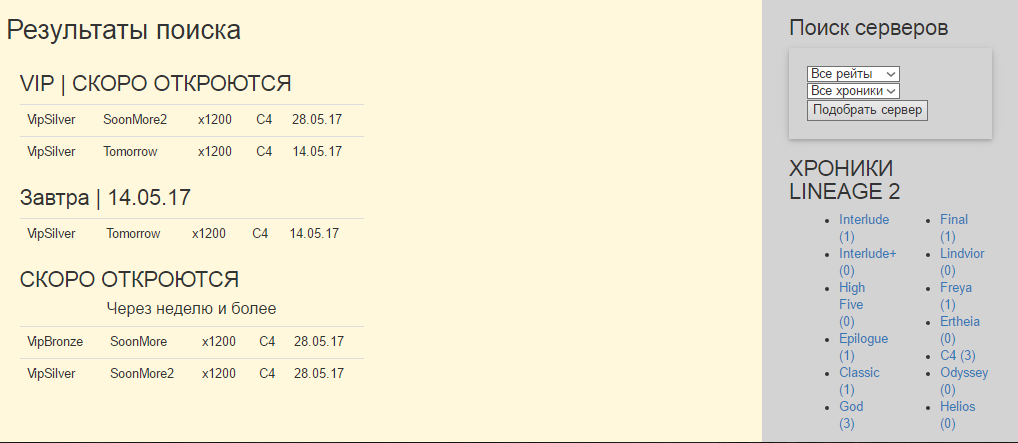


Рис. 27. Меню поиска, отображение после Search.

## Демонстрация работы приложения.

Запуск приложения осуществляется на локальном хосте IIS Express. После запуска открывается главная страница отображающая панель представления анонсов, а также меню поиска по анонсам. Страница снабжена рекламным слайдером и панелью навигации.

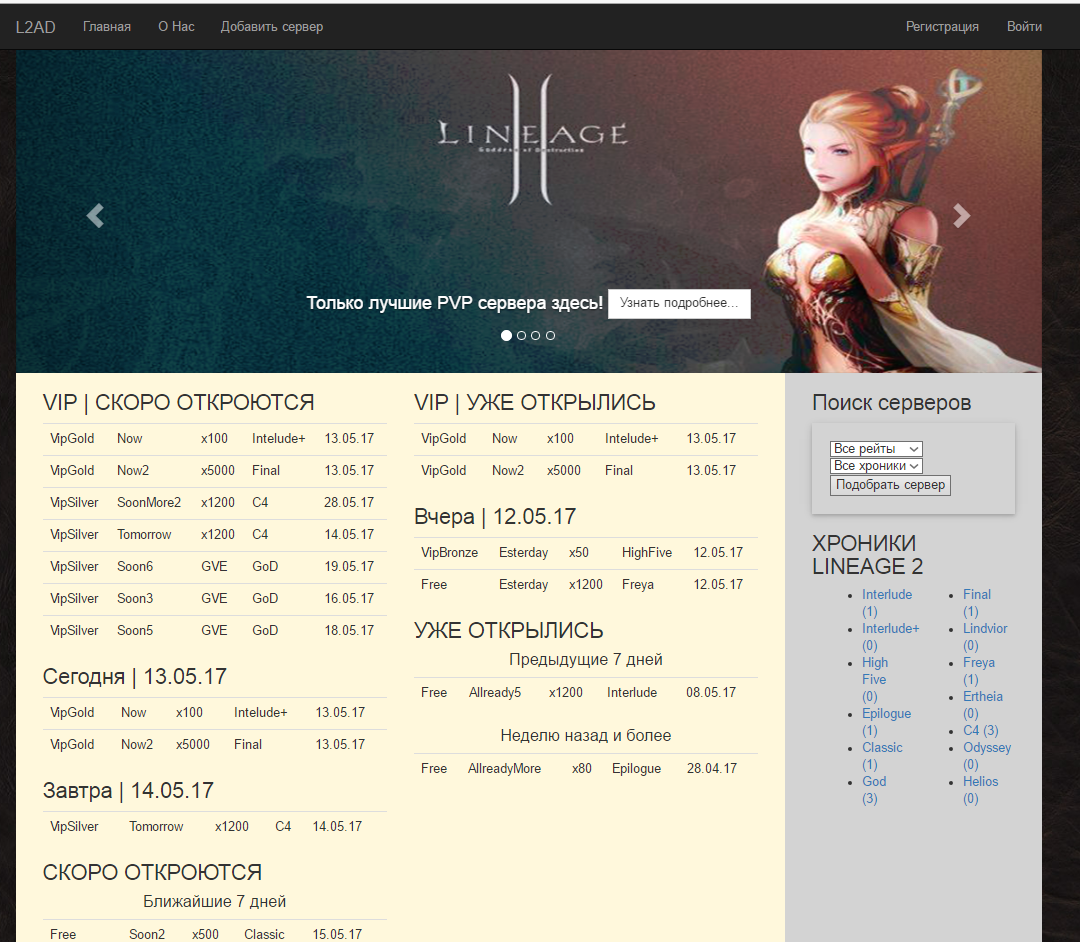


Рис. 28. Главная страница.

Пользователь ИСА сможет найти подходящий сервер воспользовавшись навигацией на панели поиска. При выборе интересующих серверов пользователю будет выдан список имеющихся в информационной системе серверов.

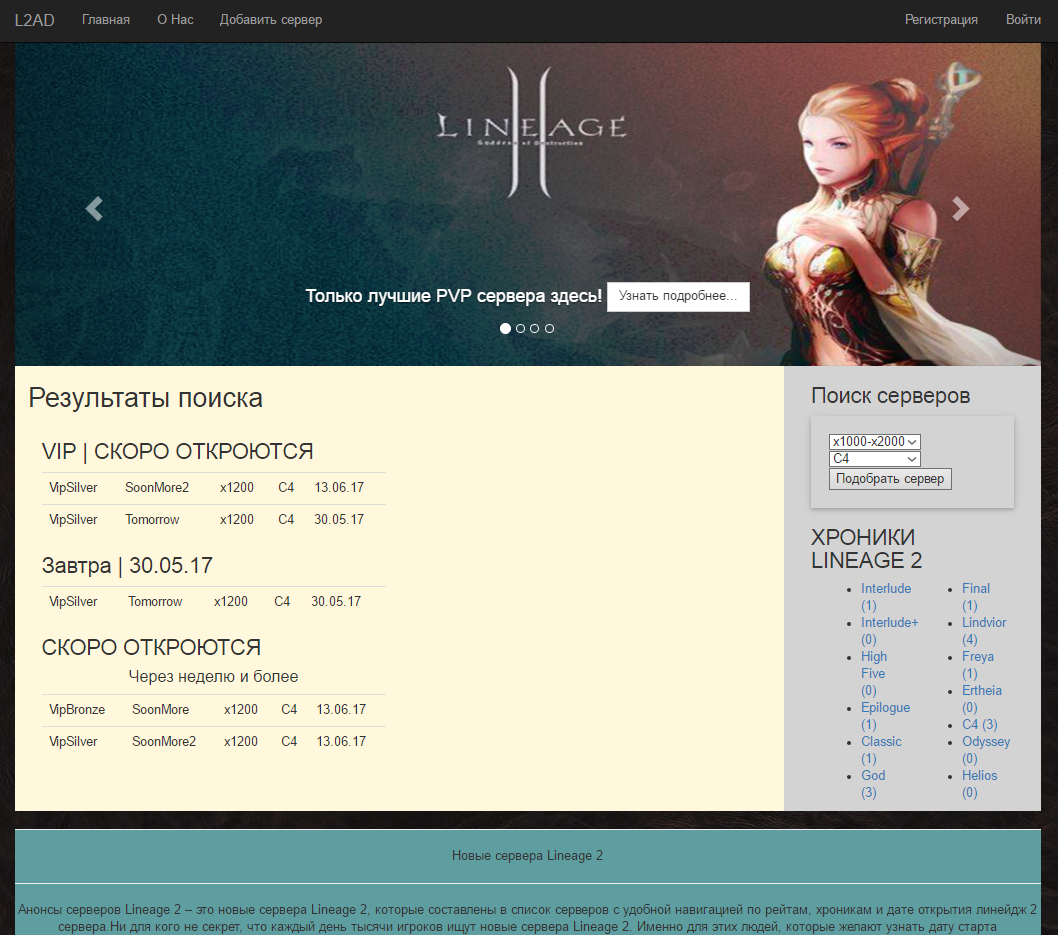


Рис. 29. Результат подбора серверов по запросу.

Для добавления нового анонса администратору игрового сервера достаточно перейти во вкладку «Добавить сервер» и заполнив поля для создания нового анонса нажать на кнопку добавления. При нажатии на контроллер добавления будет передан POST запрос с введенной информацией для модели. Во время добавления происходит автоматическая валидация введенной информации.

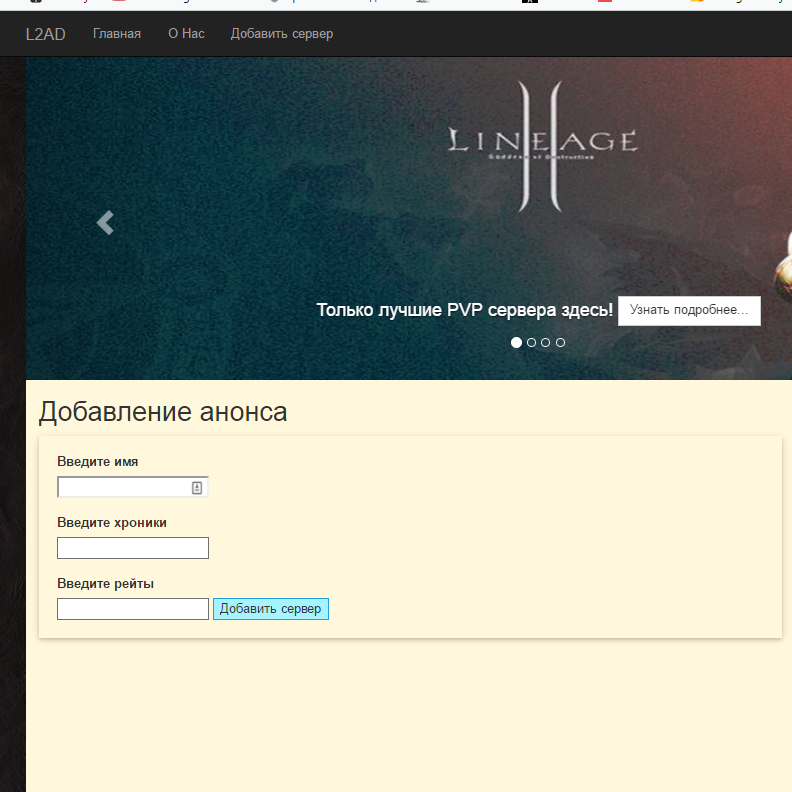


Рис. 30. Форма добавления нового анонса.

## Заключение

В ходе выполнения работы, была разработана информационная система серверного анонсера. Для нее было создано веб приложение, работающее через веб-браузер и позволяющее находить или выкладывать анонсы серверов для онлайн игр, так же система снабжена поиском по интересующим пользователя критериям и разными видами мультимедийной рекламы. Для реализации ИСА были изучены и использованы среда Microsoft Visual Studio, язык программирования C#, ASP.NET CORE, система управления реляционными базами данных Microsoft SQL Server, фреймворк EntityFramework, а также язык запросов LINQ.

## Литература

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. - СПб.: Питер, 2012.-928 с.: ил. ISBN 978-5-459-00297-3.
2. Julia Lerman. Programming Entity Framework, Second Edition. - Published by O’Reilly Media, 2010, - С. 914. ISBN: 978-0-596-80726-9.
3. LINQ. Карманный справочник: Пер. С англ. / Дж. Албахари, Б. Албахари. — СПб.: БХВ-Петербург, 2009. — 240 с.: ил. ISBN 978-5-9775-0317-4.
4. Дейт К. Дж. Введение в системы баз данных — 8-е изд. — М.: Вильямс, 2008. — 1328 с. — ISBN 978-5-8459-0788-2.
5. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования = Design Patterns: Elements of Reusable Object-Oriented Software. - СПб: [«Питер»](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%82%D0%B5%D1%80_%28%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE%29), 2007. — С. 366. ISBN 5-272-00355-1.
6. [Фаулер М.](https://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D1%83%D0%BB%D0%B5%D1%80,_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD), Бек К., Брант Д., Робертс Д., Апдайк У. Рефакторинг: улучшение существующего кода = Refactoring: Improving the Design of Existing Code (2000). — Спб: Символ-Плюс, 2009. — 432 с. [ISBN 5-93286-045-6](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B5%D0%B1%D0%BD%D0%B0%D1%8F:%D0%98%D1%81%D1%82%D0%BE%D1%87%D0%BD%D0%B8%D0%BA%D0%B8_%D0%BA%D0%BD%D0%B8%D0%B3/5932860456).
7. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. — М.: ДМК Пресс, 2006. — 496 с.: ил. ISBN 5-94074-334-X.
8. Стандарт ISO/IEC 2382:2015.
9. Загрузка среды разработки MS Visual Studio: <https://www.visualstudio.com/> .
10. Создание учётной записи Microsoft: <https://login.live.com/RU> .
11. Microsoft Developer Network: <https://msdn.microsoft.com/> .
12. Загрузка СУБД MS SQL Express: <https://www.microsoft.com/en-us/cloud-platform/sql-server> .
13. Документация по Entity Framework: <http://metanit.com/sharp/entityframework/> .