РЕФЕРАТ

Пояснювальна записка: 23 с., 9 рис., 10 джерел.

Метою роботи є проектування та розробка програмної системи підвищення якості обслуговування клієнтів.

Метод розробки - концептуальне і UML-моделювання предметної області, об'єктно- орієнтований підхід до розробки програмного продукту.

У процесі розробки були використані такі технології як Microsoft SQL Server, ADO.NET Entity Framework, T-SQL, LINQ to SQL, ASP.Net, CsQuery, JQuery Twitter bootstrap а також шаблон проектування MVC.

У результаті курсової роботи була спроектована база даних для сервер, був створений сервер додатків і веб сервер.

C#, ASP.NET MVC, MICROSOFT SQL SERVER, ENTITY FRAMEWORK, LINQ, HTML PARSING, .NET FRAMEWORK, CODE FIRST DB, MIGRATIONS, JQUERY, CSQUERY

The explanatory note contains: 23 p., 9 fig., 10 sources.

The work's goal is design and developing software system of improving client’s service experience.

Developing method – conceptual and UML modeling of the domain, object-oriented approach to developing software.

During developing were used such as technologies as Microsoft SQL Server, ADO.NET Entity Framework, T-SQL, LINQ to SQL, ASP.Net, CsQuery, JQuery Twitter bootstrap and also such as pattern design as MVC.

As a result of course work there are were developed database for server application, application server and web server were created.

C#, ASP.NET MVC, MICROSOFT SQL SERVER, ENTITY FRAMEWORK, LINQ, HTML PARSING, .NET FRAMEWORK, CODE FIRST DB, MIGRATIONS, JQUERY, CSQUERY

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc450500308)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ](#_Toc450500317) 6

[1.1 Анализ предметной области](#_Toc450500321) 8

[1.2 Описание функциональной структуры системы 1](#_Toc450500322)0

[1.3 Концептуальное моделирование предметной области 1](#_Toc450500323)1

[2. АНАЛИЗ АНАЛОГОВ](#_Toc450500309) 12

[3. НЕФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ](#_Toc450500317) 14

[4. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА](#_Toc450500317) 15

[4.1 Архитектура](#_Toc450500321) 15

[4.2 Технологии разработки 1](#_Toc450500322)7

[4.3 Программная реализация 1](#_Toc450500322)8

[ВЫВОДЫ 2](#_Toc450500325)0

[ПЕРЕЧЕНЬ ССЫЛОК 2](#_Toc450500326)1

[ПРИЛОЖЕНИЕ А 2](#_Toc450500308)2

ВВЕДЕНИЕ

Китайский рынок растет как никакой другой с большой скоростью. Каждый день появляются новые, никому не известные, интернет-магазины. Они привлекают покупателей со всего мира широким и постоянно пополняющимся ассортиментом товаров. При этом многие технические новинки появляются в Украине со значительной задержкой, либо не появляются вообще, что делает заказ некоторых товаров в Китае единственным доступным вариантом. Кроме того, стоимость доставки обычно невелика, или включается в стоимость товара, поэтому покупки из Китая обычно доставляются бесплатно по всему миру.

Основной недостаток глобальных торговых площадок в том, что в качестве продавцов там выступают сотни и тысячи различных поставщиков. Выбрать надёжного продавца сложно, а в случае не самых популярных товаров — просто невозможно. В этом ключе, надёжнее и проще покупать товар в самостоятельных интернет-магазинах Китая. Именно для этих целей и удобен данный сервис.

Сервис не является интернет-магазином. Выбрав конкретное предложение на этом сайте и нажав кнопку «В магазин» или логотип магазина, пользователь переходит на сайт интернет-магазина, где и осуществляется покупка. Каталог товаров должен обновляться в соответствии с поисковыми запросами пользователей, и поэтому всегда содержать актуальную информацию. Также изначально, в сервисе будут представлены наиболее популярные магазины, список которых можно расширять и дополнять в процессе разработки приложения.

Подобрать модель, соответствующую вкусам и потребностям, пользователей помогут умные фильтры по способам доставки, ценовому диапазону, категории товара и т.д., а также возможности сортировки товаров по различным критериям, таким как популярность товара, магазина, отзывы, и другим. При выборе следует использовать отзывы пользователей, которые размещены на странице каждого товара. Этим обеспечивается безопасность при покупке и получение дополнительной информации о заинтересовавшем товаре.

Стоит относиться с разумной осторожностью к предложениям с минимальной ценой. Всегда стоит помнить о том, что за достоверность информации отвечает только сам магазин.

Информация о товарах представлена в каталоге, а где и почем их можно купить – на странице каждого конкретного товара. Доступ ко всем разделам каталога осуществляется через главное меню. Для того чтобы шоппинг был не только простым, но и доставлял покупателю удовольствие, в каталогах может быть реализовано множество вспомогательных услуг: подбор и сравнение товаров в соответствии с их основными характеристиками, отзывы покупателей, акции магазинов.

Сервис поможет выбрать товар и магазин, но сам не является магазином. Поэтому, знакомясь с отзывами других покупателей нужно быть внимательным при выборе поставщика для приобретения товара. Перед оформлением заказа на сайте магазина, пользователь должен будет самостоятельно убедиться, что спецификация (характеристики) товара и условия поставки его устраивают.

Таким образом целью данной работы является проектирование и прототипирование программной реализация системы, которая позволит агрегировать и получать информацию о товарах в обработанном виде в соответствии с выбранными настройками. Система должна состоять из сервера приложений с веб сайтом, и быть понятной, надежной, отказоустойчивой в случае ошибки пользователя либо недоступности какого-либо из источников данных (магазина).

После прохождения инструктажа по технике безопасности в план практики входило проектирование и программная реализация прототипа системы, которая позволит искать, сортировать и фильтровать товары из нескольких китайских интернет-магазинов в одном месте. Практика проходила не предприятии «Институт радиофизики и электроники им. А. Я. Усикова НАН Украины». Более подробная информация о предприятии находится в приложении А данного отчета.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ

1.1 Анализ предметной области

1.1.1 Необходимость приложения в предметной области

Данное приложение даст вам широкий выбор различных принадлежностей и оборудования как для работы, так и развлечения. Это включает в себя последние модели смартфонов, планшетов, action камер, теле-приставок, дронов, компонентов к Arduino, а также техника для дома, приспособления для выживания, и другие вещи для гиков, большинство из которых не будут продаваться в Украине, либо будут, но появятся позже и будут стоить примерно на 25% дороже. А поиск по многим магазинам в одном месте позволит вам расширить выбор и сделать правильное решение, в виду того, что разные магазины часто работают с разными поставщиками оборудования, поэтому то что можно купить в одном магазине совсем не обязательно найдется в другом.

1.1.2 Облегчение работы человека

Современные технологии, используемые системой, обладают таким набором полезных свойств как высокая скорость работы с базой, возможность эффективной обработки большого количества информации, сохранение целостности данных, отказоустойчивость, и продолжают развиваться. Это позволяет автоматизировать многие действия и избежать возможных ошибок, допускаемых при ручных расчетах.

Также, использование облачных технологий и распределенных баз данных

предоставляет больше гибкости в развитии системы, чем использование альтернативных

методов хранения информации, а использование автоматизированных расчетов, алгоритмов сравнения и агрегации данных облегчает работу с информацией и упрощает её восприятие пользователем, что позволяет снизить количество возможных ошибок, и повышает качество обслуживания клиентов.

1.1.3 Данные для хранения

Проведя анализ предметной области, мы узнали, что базовым элементом в модели нашей системы является магазин, с такими полями как название, логотип, внутреннее название класса, адреса в интернете, которые будут использоваться для поиска, выбора категорий и перехода на главную страницу.

У каждого магазина должен быть список товаров, однако хранить все товары в нашей системе мы не будем. Поскольку мы собираем их с сайтов магазинов, и кроме технических сложностей копирования всей информации, возможны также случаи обновление описания, цены и другой информации, которые мы не сможем достаточно быстро отследить, и соответственно обновить информацию в нашем сервисе. Кроме того, предусмотреть все возможные параметры товаров с разных сайтов предусмотреть мы не можем, и если хранить их в реляционной СУБД, то база может получится либо слишком обобщенной, либо не нормализированной. Вместо этого в базе данных мы будем хранить лишь те товары, которые были добавлены пользователями в избранное.

Разумеется, данные об аккаунтах также будут хранится в базе для авторизации, и хранения настроек пользователей.

На рисунке 1.1 изображена предварительная ER диаграмма нашей базы данных. На ней видно, что пользователи системы могут сохранять много товаров в разделе избранное (связь Users 1:M FavoriteItem), которые могут относится к разным магазинам (связь с Shops). Магазины должны хранится в базе данных еще для того, чтобы можно было динамически добавлять магазины в систему в процессе её расширения. Таблица Categories предназначена для хранения список категорий в системе. Однако поскольку каждый магазин обладает собственным набором категорий, мы связываем одинаковые по смыслу категории магазина с категорией нашего сервиса в таблице CategoryToShopSpecificIdMapping.

В процессе разработки база данных может дополнятся дополнительными таблицами и полями в существующих таблицах. Для расширения базы в Entity Framework возможно использовать механизмы миграций, которые позволяют использовать Code First Migrations без наличия файла кода в проекте для каждого вносимого вами изменения. Не все изменения могут применяться автоматически, например, для переименований столбцов необходимо использовать миграцию на основе кода. Можно смешивать автоматические миграции и миграции на основе кода, но в сценариях командной разработки это делать не рекомендуется. Если вы входите в состав группы разработчиков, использующих систему управления версиями, необходимо применять либо только автоматическую миграцию, либо только миграцию на основе кода. С учетом ограничений автоматических миграций рекомендуется использовать в командных средах миграции на основе кода.

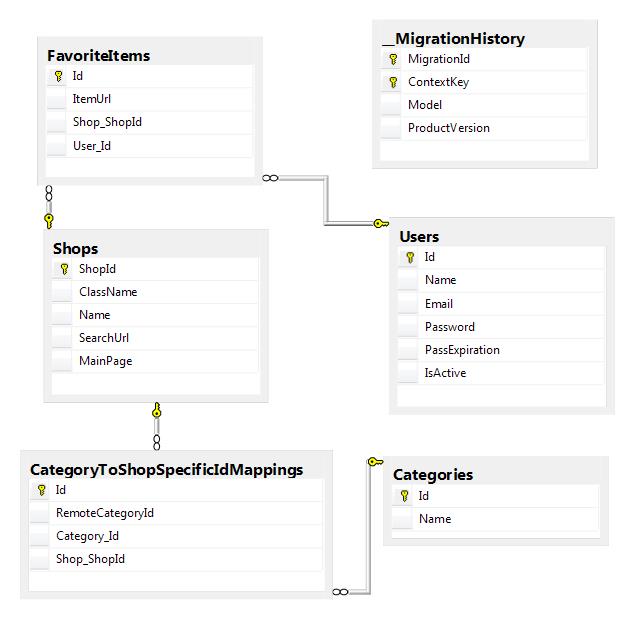


Рисунок 1.1 – ER диаграмма базы данных.

Таблица \_\_MigrationsHistory используется Code First Migrations для хранения информации о миграциях, которые были применены к базе данных. Эта таблица создается при применении первой миграции к базе данных. В Entity Framework 5 эта таблица была системной бри использовании Microsoft Sql Server. Однако в Entity Framework 6 эта таблица больше не отмечена как системная. Использование миграций поможет избежать многих проблем, если используется не одна база данных. Например, часто используют разные базы для разработки и рабочей системы, чтобы случайно все не испортить.

Все эти особенности хранения данных необходимо учитывать при моделировании и

построении архитектуры, а в дальнейшем при выборе программных средств реализации. В

случае неправильного проектирования могут возникнуть сложности с дальнейшей

разработкой, и логические проблемы при реализации отдельных модулей, или их

несовместимость.

1.2 Концептуальное моделирование предметной области

1.2.1 Описание функциональных потребностей пользователей

К функциональным потребностям пользователей относятся:

* поиск товаров по ключевым словам;
* фильтрация результатов поиска;
* сортировка результатов поиска;
* просмотр детальной информации о товаре;
* переход на страницу магазина для оформления заказа;
* сохранение товаров в раздел избранное;
* сравнение товаров по характеристикам;
* отправка описания найденного товаров на электронную почту.

1.3. Описание функциональной структуры системы

В данной системе пользователь будет иметь доступ к веб-приложению на компьютере

и/или планшете. Также в дальнейшем будет разработано клиентское приложение для платформы UWP.

В приложении имеется три основных раздела: список магазинов, поиск и избранное.

В это сложно поверить, но в разделе «список магазинов» будет, собственно, список магазинов. У каждого магазина можно посмотреть название, описание, логотип, список доступных категорий, а также возможен переход к его сайту.

В разделе избранное пользователь может отправить отчет о сохраненных товарах на почту, сравнить выбранные товары, просмотреть подробную информацию либо удалить. Что бы попасть в раздел избранное, необходимо быть авторизированным пользователем.

Ниже приведена диаграмма сценариев использования (диаграмма прецедентов) — диаграмма, отражающая отношения между актёрами (пользователями системы) и прецедентами (см. рисунок 1.2).

E:\Docs\use case.png

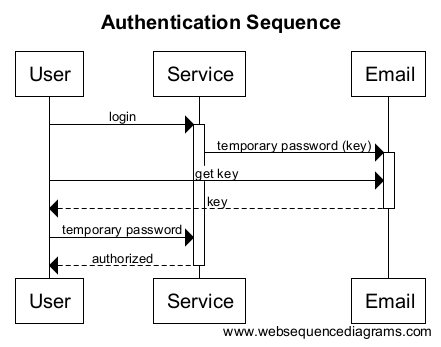
Рисунок 1.2 – Диаграмма сценариев использования.

Как можно увидеть на диаграмме, любой успешный сценарий заканчивается в узле «go to location» – переход пользователя на страницу магазина для осуществления покупки.

В раздел «поиск» можно попасть, введя ключевые слова в поле поиска в верхней части приложения (поле доступно с любой страницы), либо кликнув на одну из категорий в меню. При этом можно осуществить поиск по выбранной категории введя текст в поле поиска, либо наоборот, сузить область поиска выбрав категорию (см. рисунок 1). Для большего уточнения результатов поиска доступны фильтры, и сортировка. После того как пользователь найдет то что искал, он может просмотреть описание товара, и перейти на страницу магазина для покупки или связи с продавцом, либо сохранить его к себе в избранное.

Что бы упростить регистрацию пользователей в данном сервисе, было решено использовать авторизацию без пароля. Суть метода авторизации состоит в следующем: при авторизации пользователь указывает только свое имя и электронный адрес. Для входа в систему нужно ввести свой адрес, указанный при регистрации. После ввода адреса, на него приходит сообщение с одноразовым паролем и ссылкой, которые действительны небольшое количество времени.

Что бы лучше понять данный метод авторизации была нарисована диаграмма последовательности (см. рисунок 2).

Рисунок 2 – Диаграма последовательности авторизации клиента.

Для завершения входа пользователю нужно либо перейти по ссылке, и он попадет на страницу сервиса уже авторизированным, либо ввести пришедший пароль. Этот подход позволяет не только не использовать постоянных паролей, которые пользователь склонен забывать, но и увеличивает безопасность.

2 АНАЛИЗ АНАЛОГОВ

2.1 поиск аналогов

Перед проведением анализа аналогов был выполнен их поиск с использованием наиболее популярных поисковых систем в мире и русскоязычном сегменте интернета, таких как google (google.com) и яндекс (ya.ru). Для улучшения релевантности поисковых результатов использовались дополнительные настройки поиска.

По умолчанию слова и вообще любые введенные символы Google ищет по всем файлам на проиндексированных страницах. Ограничить область поиска можно по домену верхнего уровня, конкретному сайту или по месту расположения искомой последовательности в самих файлах. Для первых двух вариантов используется оператор site, после которого вводится имя домена или выбранного сайта. В третьем случае целый набор операторов позволяет искать информацию в служебных полях и метаданных. Например, allinurl отыщет заданное в теле самих ссылок, allinanchor — в тексте, снабженном тегом <a name>, allintitle — в заголовках страниц, allintext — в теле страниц. Для каждого оператора есть облегченная версия с более коротким названием без приставки all. Разница в том, что allinurl отыщет ссылки со всеми словами, а inurl — только с первым из них. Второе и последующие слова из запроса могут встречаться на веб-страницах где угодно. Оператор inurl тоже имеет отличия от другого схожего по смыслу — site.

Полученные результаты нередко засоряет какая-то лишняя информация. Чтобы не тратить время на ее просмотр, советую наложить на результаты поиска фильтр. Сделать это несложно. Надо лишь указать слова, которые нужно исключить, поставив перед ними знак «-», и включающие их страницы будут исключены из результатов поиска.

Лучший способ найти дружественные (и конкурирующие) сайты – спросить об этом Google. В ответ на модификатор related:<URL сайта> google выдаст сайты со схожей тематикой и контентом.

Можно использовать Google и для того, чтобы проверить популярность конкретного проекта. Так, модификатор link:<URL сайта> отобразит все страницы, которые ссылаются на этот ресурс. По умолчанию к каждому ключевому слову поисковая система применяет операцию логического «И», поэтому указав несколько магазинов, на которые ссылается искомый сайт можно его найти.

Google знает, что такое синонимы, поэтому чтобы в результаты вошли страницы не только с конкретным указанным словом, но и c его синонимами, нужно поставить перед ним знак «~».

Чтобы сделать акцент на одно или несколько ключевых слов, стоит поставить перед ними знак «+». Это поможет системе понять, какие из ключевых слов наиболее важные, и сформулировать результаты поиска более точно.

2.2 Описание аналогов

Что бы узнать больше о предметной области, был проведен анализ аналогов. Основные конкуренты в этом классе сервисов это PandaCheck (https://ru.pandacheck.com/), chinaprices.ru (<http://chinaprices.ru/>) и SearchSKU.ru (<http://searchsku.ru/>).

ПандаЧек – одна из самых больших поисковых машин для китайских интернет-магазинов. Здесь можно найти самые лучшие цены, новейшие купоны, а также советы и идеи при заказе из Китая. Под 'Найти' у есть возможность усовершенствовать Поиск. С '+' (плюсом) можно добавить, с '-' (минусом) удалить дополнительную информацию из поиска. Результаты будут автоматически сортированы по актуальности, но можно индивидуально выбрать вид сортирования – например по цене или виду доставки.

Как и указано в описании, сайт поддерживает наибольшее количество магазинов из всех конкурентов.



Рисунок 2.1 – Логотип PandaCheck.

Chinaprices.ru – сайт помогает находить товары в интернет-магазинах Китая, по самым выгодным ценам и с бесплатной доставкой большинства товаров. А поскольку большинство товаров производится в Китае, то поиск идёт по ассортименту именно проверенных китайских интернет-магазинов.

Довольно утомительно просматривать десятки сайтов в поиске нужного товара. С помощью ChinaPrices.ru возможно найти товар буквально за пару кликов мышки!

Возможности сайта:

* Поиск китайских товаров для самостоятельной покупки без посредников и грабительских наценок. Пользователи сами находите выгодные цены, сами покупают товары напрямую в Китае, сами получают товары на почте или используя курьерские службы доставки. На главной странице сайта, в распоряжении пользователя мощный поисковик товаров, с возможностью сортировки товаров по цене и с автоматическим переводом запросов на английский язык. Стоит выбирать товары только с бесплатной доставкой, или покупать товар партиями, чтобы сэкономить на стоимости доставки, а также:
* Использовать купоны китайских интернет-магазинов чтобы купить товар ещё дешевле!
* Подписаться на обновления блога о покупках в Китае, чтобы отслеживать скидки и распродажи самых популярных интернет-магазинов Китая, таких как Aliexpress, Dealextreme, Focalprice, Gearbest, Banggood и десятков других китайских магазинов.
* Отслеживать движение вашей посылки, чтобы всегда знать где посылка в данный момент времени.
* Читать инструкции по безопасной покупке товаров в нашем блоге, чтобы защитить себя от мошенничества и недобросовестных китайских продавцов. Можно узнать, как открыть диспут на возврат денег в случае неполучения товара в срок, или если товар пришёл повреждённым или не соответствует вашему заказу.

C:\Users\user\Desktop\mainlogo.gif

Рисунок 2.2 – Логотип Chinaprices

SearchSKU.ru – сайт создан для поиска товаров в китайских интернет-магазинах AliExpress, DX, TinyDeal и других. Большая часть сайта — это форум, на котором можно почитать или оставить отзывы, узнать в каких магазинах что лучше покупать, и много других различных обсуждений. Впрочем, это не является частью нашего сервиса, а такие форумы есть и за пределами подобных сервисов, поэтому считать это конкурентным преимуществом мы не будем.



Рисунок 2.3 – Логотип SearchSKU

Однако, вышеперечисленные системы имеют различные недостатки, описанные в следующем подразделе, а кроме этого их всех объединяет один фатальный недостаток общий для всех аналогичных систем: они были написаны не нами. В отличии от нашей системы, у конкурентов этот недостаток вероятно не будет исправлен в ближайших релизах.

2.1 Недостатки аналогов

То, что вышеперечисленными методами было найдено только три аналога, из которых по субъективным ощущениям только первые два пригодны к использованию показывает, что пока возможна разработка конкурентоспособного решения, однако для этого необходимо проанализировать недостатки аналогов и выявить их слабые стороны, чтобы избавиться от них в собственной реализации системы.

Отсутствие возможности посмотреть описание товара, не переходя на сайт магазина – пользователю может быть неудобно переходить на страницы разных магазинов, чтобы посмотреть основную информацию, а так как у различных магазинов интерфейс сильно отличается, это еще будет отнимать много времени.

Отсутствие функции сравнение товаров – на странице результатов отображается только название товара, картинка, цена, и у некоторых аналогов способ/стоимость доставки. Если пользователь еще не решил, что он собирается покупать, то этой информации недостаточно, и нужно углубляться в сайты магазинов, что лишает пользователей всех достоинств поиска в одном месте.

Отсутствие списка избранного. Если пользователь выбрал себе несколько подходящих товаров, но хочет решиться с финальным выбором позже, он не может сохранить их на сайте, а должен либо добавлять каждую страницу в закладки, либо использовать корзину на сайте магазина. При этом ему потребуется регистрироваться в каждом из магазинов, и запоминать в каких именно он это нашел, что не есть удобно.

Таким образом, вышеперечисленные недостатки делают сайты аналогов более подходящими для людей, которые уже определились с выбором товара, и теперь хотят решить в каком именно совершить покупку исходя из цены и метода доставки.

3 ПОСТАНОВКА ЗАДАЧИ

В результате проведения анализа аналогов была выявлена проблема, которая заключается в отсутствии на рынке системы, которая позволит пользователям удобно выбирать товары в китайских интернет магазинах.

Необходимо спроектировать и программно реализовать систему, которая позволит искать, сортировать и фильтровать товары из нескольких китайских интернет-магазинов в одном месте. Система должна быть понятной, надежной, отказоустойчивой в случае ошибки пользователя, а также спроектированной с возможностью расширения.

Перед началом разработки программной системы необходимо определиться со списком требований к ней. Требования обычно используются как средство коммуникации между различными заинтересованными лицами. Это означает, что требования должны быть просты и понятны для обычных пользователей и разработчиков. Один общий способ задокументировать требование — это написать утверждение о том, что должна сделать система. Ниже приведен список задач, предъявляемых к разрабатываемой информационной

системе и детальное описание к каждой из них:

* система должна предоставить возможность поиска товара по ключевым словам – при вводе в поле поиска слова, должна появиться страница, элементы которой каким-либо образом относятся к введенному ключевому слову. Например, содержат его в названии;
* система должна предоставить возможность фильтрации по различным критериям, таким как магазин, способ доставки, диапазон стоимости, категория, по которым будет осуществляться поиск. Т.е. на странице с результатами поиска должна быть область, в которой пользователь может включить или выключить критерий фильтрации;
* система должна предоставить возможность сортировки в найденных результатах по различным критериям как по возрастанию, так и по убыванию – на странице с результатами поиска должен быть элемент управления для выбора поля, по которому необходимо отсортировать результат;
* система должна поддерживать регистрацию и авторизацию пользователей. У зарегистрированных пользователей должна быть возможность добавить понравившийся товар в избранное, для последующего сравнения либо быстрого доступа позднее;
* система должна поддерживать сравнение параметров товаров, находящихся у пользователей в меню «Избранное» (см. выше). Для этого на каждом товаре в избранном должна отображаться кнопка «добавить к сравнению», и при выборе двух товаров должен осуществляться переход к странице сравнения;
* система должна уметь отправлять информацию о товаре на электронный адрес пользователя;
* система должна выдавать список всех поддерживаемых магазинов с возможностью быстрого перехода к самому магазину.

Определённые требования, по своей сути, не поддаются проверке, поэтому не были включены в данную постановку задачи. Они включают требования, которые говорят, что система никогда не должна или всегда должна показывать специфическое свойство. Надлежащее тестирование этих требований потребовало бы бесконечного цикла тестирования. Такие требования должны быть переопределены так, чтобы они стали поддающимися проверке. Как указано выше, все требования должны поддаваться проверке.

Нефункциональные требования, которые не поддаются проверке на программном уровне, все равно должны быть сохранены как документация намерений клиента. Такие требования к продукту могут быть преобразованы в требования к процессу.

3.1 Ограничения

Ограничения — условия, ограничивающие выбор возможных решений по реализации отдельных требований или их наборов. Они существенно ограничивают выбор средств, инструментов и стратегий при разработке внешнего вида и структуры (в том числе архитектуры) продукта или системы.

Ограничение в данную систему вводятся для осуществимости проекта – включаемое в спецификацию требование должно быть выполнимым при заданных ограничениях операционной среды. Осуществимость требований проверяется в процессе анализа осуществимости разработчиком. В частности, для нефункциональных требований проверяется возможность достижения указанных численных значений при существующих ограничениях, которые в случае с этой системой указаны ниже.

Данная система должна производить поиск по указанному в базе данных списку магазинов. Для первых версий приложения это будут наиболее популярные магазины AliExpress, GearBest и DealExtreme. При этом алгоритмы работы с каждым из магазинов должны быть отдельно запрограммированными в системе, и для добавления нового магазина будет требоваться дописывание отдельных модулей.

Система не должна хранить данные о всех товарах в своей базе данных, следовательно, если в момент поиска пользователем один из магазинов будет недоступен, то результаты с него не попадут в выборку.

Пользователь не может совершать покупок из системы, и должен перейти для этого на сайт магазина. При этом пользователь может перейти напрямую на страницу товара, а не искать его в нужном магазине заново.

В общем случае требования изменяются со временем. После того, как требования определены и одобрены, изменения должны попадать под контроль внесения изменений. Это происходит частично из-за сложности программного обеспечения и того факта, что пользователи не знают, что им нужно на самом деле либо из-за технических ограничений при реализации системы. Поэтому разработанное программное обеспечение по этим требованиям в конце может иметь характеристики несколько отличающиеся от заявленных.

4 АРХИТЕКТУРА СИСТЕМЫ И ВЫБОР ПРОЕКТНЫХ РЕШЕНИЙ

4.1 Архитектура

Для реализации проекта была выбрана платформа .Net компании Microsoft. Данный выбор был сделан из технологий .net, yii php framework и ruby on rails, которые сравнивались по нескольким критериям.

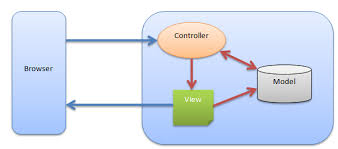
Скорость изучения – с учетом владения C# а также с бесплатными обучающими видео от Pluralsight на http://www.asp.net/mvc проблем должно быть не много. Функциональность – ещё не попадалась задача которую нельзя было решить на C# (попадались правда те которые элегантнее было бы решить на скритовом языке и те для решения которых в других языках были встроенные функции).

Создание API – в ­­­MVC 4 из коробки вы получаете отличный инструмент для построения API о чём можно прочитать в статье build restful apis with aspnet web api на сайте asp.net или RESTFul Api контроллеры в .NET MVC 4.

Возможность найти помощь по проблемам – не мало ресурсов таких как hashcode, StackOverflow, скорее зависит от специфической задачи. В конечном счете есть поисковики и очень редко бывает так, что никто не столкнулся с подобной проблемой.

Производительность с новыми обертками для асинхронности вроде async\await возможно значительно повысить производительность вебсайта, кроме того большая часть тормозов при работе с сайтами > 4000 пользователей это не грамотная работа с БД ну и сама БД.

Основой платформы 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. Для данной системы будет использоваться язык C# и ASP.NET MVC – фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller (см. рисунок 4.1 ).

 Рисунок 4.1 – Шаблон Model-view-controller.

Основная идея этого паттерна в том, что и контроллер и представление зависят от модели, но модель никак не зависит от этих двух компонент.

Для управления разметкой и вставками кода в представлении используется движок представлений. До версии MVC 5 использовались два движка:

Web Forms и Razor. Начиная с MVC 5 единственным движком, встроенным по умолчанию, является Razor. Движок WebForms использует файлы .aspx, а Razor - файлы .cshtml и .vbhtml для хранения кода представлений со вставками кода на языках C# и VB.NET соответственно. Основой синтаксиса Razor является знак @, после которого осуществляется переход к коду на языках C#/VB.NET. Также возможно и использование сторонних движков. Файлы представлений не являются стандартными статическими страницами с кодом html, а в процессе генерации контроллером ответа с использованием представлений компилируются в классы, из которых затем генерируется страница html.

При обработке запросов фреймворк ASP.NET MVC опирается на систему маршрутизации, которая сопоставляет все входящие запросы с определенными в системе маршрутами, которые указывают какой контроллер и метод должен обработать данный запрос. Встроенный маршрут по умолчанию предполагает трехзвенную структуру: контроллер/действие/параметр. Это позволяет создавать более структурированные сайты и делает адреса более читаемыми. В следующем примере показан стандартный маршрут, который генерируется при создании проекта в файле RoutesConfig.cs.

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

С таким маршрутом при заходе на сайт пользователь попадет на action Index, который расположен в контроллере Home, и вернет соответствующую страницу пользователю. По умолчанию эта страница будет сгенерирована с файла /View/Home/Index.cshtml.

Стоит также обратить внимание на опциональный параметр id, который позволяет использовать более красивый адрес /Home/Favorites/4/ вместо /Home/Favorites?id=4.

В более расширенном виде и с учетом особенностей платформы структура asp.net mvc приложения показана на рисунке 4.2

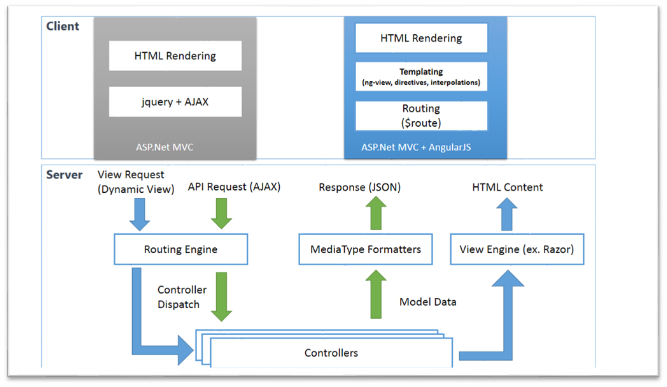


Рисунок 4.2 – Структура asp.net mvc приложения.

Поскольку для разработки приложения был выбран стек технологий Microsoft, в качестве системы управления базами данных больше всего подойдет MS SQL Server, поскольку для него в C# есть встроенная поддержка, и он больше всего подходит для ADO.NET Entity Framework.

EF должна работать со всеми СУБД, которые реализуют провайдер для ADO.NET. Но лучше всего EF работает как это несложно догадаться для SQL Server и SQL CE. Также возможно работать с Firebird через EF – явных ошибок не было замечано, но были вопросы по производительности. Также интеграцию для дизайнера VS нужно делать в ручном режиме и CodeFirst, который предполагается использовать для данной системы, для Firebird поддерживается. Приблизительно такая же картина как и для Firebird, но чуть лучше в плане автоматизации интеграции в VS - и для MySql. Провайдер EF для Firebird и MySql возможно использовать и в производственных решениях. Для SqlLite поддержка EF есть, но там еще версия dotNET 3.5 SP1. Postgresql поддерживается но тут надо понимать, что для MySQL и Postgres в EF 6 используются неофициальные провайдеры, не от MS, поэтому никто не гарантирует, что какой-то linq-запрос может не трансформировать в sql-запрос. А так, просто поменяв подключение в web.config можно запросто перейти от одной субд к другой, практически без изменения кода. В ORACLE и их ODP еЕсть движения в сторону EF, но пока в стадии бета и только 32-разрядный. Это что касательно из бесплатного. Из платных провайдеров на первых этапах разработки использовать ничего не будем.

Также для упрощения разработки в начале будет использоваться Microsoft SQL Server 2014 Express LocalDB — это режим выполнения SQL Server Express, предназначенный для разработчиков программ. При установке LocalDB выполняется копирование минимального набора файлов, необходимых для запуска компонента Компонент SQL Server Database Engine. После установки LocalDB разработчики могут установить соединение с помощью специальной строки подключения. При соединении необходимая инфраструктура SQL Server создается и запускается автоматически, что позволяет приложению использовать базу данных без выполнения сложной настройки, занимающей много времени. Средства разработчика позволяют использовать Компонент SQL Server Database Engine для написания и проверки кода Transact-SQL без необходимости управления полным экземпляром сервера SQL Server. Управление экземпляром SQL Server ExpressLocalDB осуществляется с помощью служебной программы SqlLocalDB.exe. SQL Server ExpressLocalDB следует использовать вместо устаревшей функции пользовательского экземпляра SQL Server Express.

4.2 Технологии разработки

Для разработки приложений использовались различные технологии, библиотеки и framework-и, такие как:

* Twitter Bootstrap — свободный набор инструментов для создания сайтов и веб- приложений. Включает в себя HTML и CSS шаблоны оформления для типографики, веб- форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения. Используется в пользовательском интерфейсе на веб сервере.
* CsQuery – использует портированный на C# validator.nu HTML парсер. Это тот же код, который используется в браузерном движке Gecko. Поэтому, CsQuery создаст такую же DOM структуру, как и любой основанный на Gecko браузер с такой же исходной разметки, и можно ожидать хороших результатов парсинга как с валидной, так и с невалидной разметки. CSS селекторы и jQuery упрощают манипуляции с HTML на клиенте, и не должно быть причин, по которым сделать то же самое на сервере будет сложнее.
* ADO.NET Entity Framework (EF) – объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.
* JQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.
* Microsoft SQL Server — это система анализа и управления реляционными базами данных в решениях электронной коммерции, производственных отраслей и хранилищ данных. SQL Server 2014 также предоставляет новые решения для аварийного восстановления, резервного копирования и гибридной архитектуры в Windows Azure, позволяя использовать существующие навыки работы с локальными функциями, опирающимися на возможности глобальных центров обработки данных Microsoft. Кроме того, в SQL Server 2014 используются новые возможности Windows Server 2012 и Windows Server 2012 R2, дающие несравненную масштабируемость для приложений баз данных в физических и виртуальных средах.

5 РАЗРАБОТКА СИСТЕМЫ

5.1 Программные решения

В самом начале разработки в среде Visual Studio был создан проект ASP.NET MVC, но без авторизации и с опцией empty project – эта опция указывает среде разработки, что нужно включать в проект только самые необходимые библиотеки, и создать структуру каталогов для MVC приложения. В результате мы получаем практически пустой проект, который не нагружен излишним на старте разработки функционалом, можно сразу запустить, а все необходимые библиотеки можно будет подключить по мере надобности вручную либо воспользоваться пакетным менеджером. Кроме этого был создан проект с типом class library, который предназначен для кода бизнес логики и проект c модульными тестами.

Сразу после создания проекта был создан локальный git репозиторий а также удаленный в GitHub (github.com). Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки/ Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой. Это позволяет работать над проектом с разных компьютеров, не задумываясь о том где находится какая версия, а также иметь резервную копию на тот случай, если что-то случится с диском (или другим устройством хранения информации) рабочего компьютера. В случае непредвиденных обстоятельств теряется лишь не добавленная в коммит информация, т.е. не больше дня работы. Для совершения коммита используется команда git commit, а затем выполняется git push для отправки изменений на удаленный сервер.

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

На рисунке 5.1 показан жизненный цикл файла в системе контроля версий. Кроме уже упомянутых состояний тут есть состояние staged. Оно необходимо для того, чтобы можно было добавить в коммит не все изменения, а только часть. Это может быть как отдельная часть функционала, так и изменения, которые не должны быть в удаленном репозитории. Например, строка подключения к базе данных может быть разной у разных разработчиков при использовании локальной БД для тестирования. Также может быть разная конфигурация сервера, LocalIIS или IIS Express.



Рисунок 5.1 – жизненный цикл файла в git

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

В созданный репозиторий был добавлен файл .gitignore – описание файлов которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках неотслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т.п.). В таком случае, можно создать файл .gitignore с перечислением шаблонов соответствующих таким файлам.

Репозиторий в github публично доступен по адресу <https://github.com/darksoullock/FindInCn> .

После подготовки необходимого окружения были созданы основные интерфейсы и модели. На рисунке 5.2 показана диаграмма классов для основных классов системы. В классе CnContext, наследуемом от DbContext, мы видим поля для доступа к таблицам базы данных. Примером класса для записи из таблицы является класс User на диаграмме. Также видим два интерфейса – IRemoteSearchItem и IRemoteShop, для магазина и результата поиска. Конкретными реализациями этих интерфейсов являются классы AliExpress и GearBest для IRemoteShop и GenericSearchItem для IRemoteSearchItemю

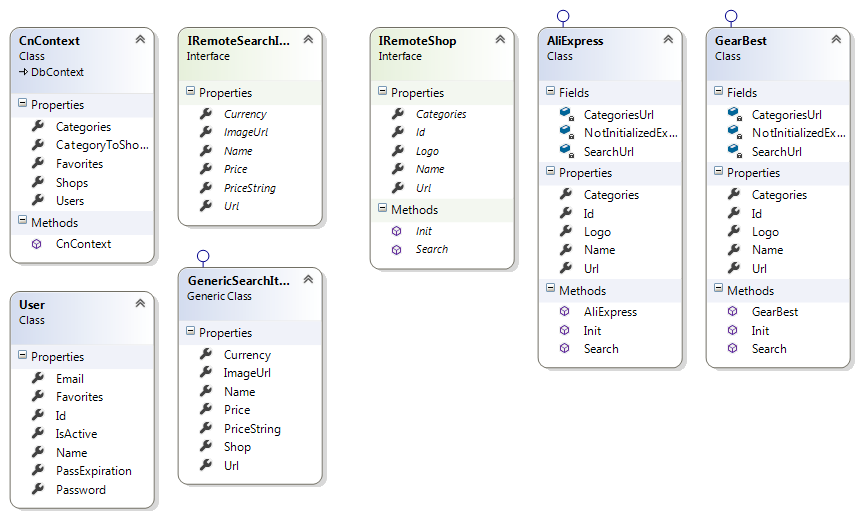


Рисунок 5.2 – Диаграмма классов.

В приложении был сначала написать код, описывающий классы-модели, а потом фреймворк автоматически создал БД по такому коду. Такой подход называется code first. Для определения связей между таблицами в классах пишутся свойства навигации, которые объявляются виртуальными. В результате включается функция отложенной загрузки Entity Framework. Отложенная загрузка означает, что содержимое этих свойств автоматически загружается из базы данных при попытке доступа к ним. После создания моделей необходимо определить производный контекст, который представляет собой сеанс с базой данных, позволяющий запрашивать и сохранять данные. Определяется контекст, который является производным от System.Data.Entity.DbContext и предоставляет типизированный DbSet<TEntity> для каждого класса нашей модели.

По соглашению базу данных создает DbContext. Если локальный экземпляр SQL Express доступен (установлен по умолчанию в Visual Studio 2010), то Code First создает базу данных на этом экземпляре. Если SQL Express недоступен, то Code First попытается использовать LocalDb (установлен по умолчанию в Visual Studio 2012). Базе данных присваивается полное имя производного контекста. Это соглашения по умолчанию. Имеются различные способы изменения базы данных, которую использует Code First.

Для доступа к данным в базе использовались LINQ запросы, как более безопасные чем sql по поношению к sql injection уязвимостям, кроме этого такие запросы могут быть проверены компилятором, что повышает стабильность системы, поскольку с явно некорректными запросами приложение даже не соберется.

LINQ to Entities обеспечивает поддержку LINQ при запросах к сущностям. Компонент позволяет разработчикам писать запросы к концептуальной модели Entity Framework на языке Visual Basic или Visual C#.Запросы к платформе Entity Framework представляются в виде дерева команд запроса, выполняемого на контексте объектов. Технология LINQ to Entities преобразует запросы Language-Integrated Queries (LINQ) в запросы в виде дерева команд, выполняет эти запросы на платформе Entity Framework и возвращает объекты, которые могут использоваться как платформой Entity Framework, так и технологией LINQ.

Для создания клиентской части были использованы языки HTML CSS и javascript. Для ускорения разработки и придания более гармоничного дизайна были использованы стили bootstrap. Для ускорения загрузки страницы, части, требующие времени, подгружаются с использованием ajax из библиотеки jquery. Это подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. Схема взаимодействия клиента с сервером с использованием ajax в сравнении с классической схемой показана на рисунке 5.3.

Действия с интерфейсом преобразуются в операции с элементами DOM (англ. Document Object Model), с помощью которых обрабатываются данные, доступные пользователю, в результате чего представление их изменяется. Здесь же производится обработка перемещений и щелчков мышью, а также нажатий клавиш. Каскадные таблицы стилей, или CSS (англ. Cascading Style Sheets), обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам. Объект XMLHttpRequest (или подобные механизмы) используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных.

Далее началась разработка парсеров для основных поддерживаемых магазинов. Вначале были созданы несколько регулярных выражений. Однако код быстро стал нечитаемым и плохо поддерживаемым, поэтому применять их нежелательно. Иногда возникает необходимость, так как парсеры, которые строят DOM, потребляют заметно больше ресурсов, чем Regex: они потребляют больше и процессорного времени, и памяти.

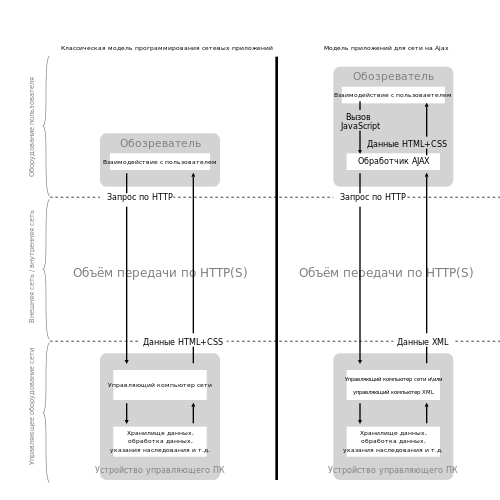


Рисунок 5.3 – схема взаимодействия клиента с сервером использованием ajax и без.

Если дошло до регулярных выражений, то нужно понимать, на них не получится построить на них универсальное и абсолютно надёжное решение. Однако если необходимо парсить конкретный сайт, то эта проблема может быть не так критична. Вместо регулярных выражений был выбран DOM парсер CsQuery, который черпал вдохновение у jQuery, для выбора элементов используется язык селекторов CSS. Названия методов скопированы практически один-в-один, то есть для программистов, знакомых с jQuery, изучение не составляет сложности.

5.2 Тестирование системы и её улучшение

Тестирование программного обеспечения — процесс исследования программного продукта, имеющий две различные цели: продемонстрировать, что программа соответствует требованиям и выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Тестировщики используют тестовые сценарии на разных уровнях: как в модульном, так и в интеграционном и системном тестировании. Тестовые сценарии, как правило, пишутся для проверки компонентов, в которых наиболее высока вероятность появления отказов или вовремя не найденная ошибка может быть дорогостоящей.

Примерами тестовых сценариев для данной системы могут быть такие:

* Доступность сайта. Шаги: Открыть сайт в браузере. Ожидаемый результат: главная страница сайта отображается. На странице отображается список магазинов и доступные категории на левой панели;
* Форма регистрации. Шаги: Открыть сайт в браузере. Нажать на кнопку регистрации. Ожидаемый результат: возле кнопки отображается всплывающее окно с полями «имя», «почта» и кнопкой «зарегистрироваться»;
* Регистрация. Шаги: Открыть сайт в браузере. Нажать на кнопку регистрации. Ввести имя и валидный адрес электронной почты. Нажать кнопку «зарегистрироваться». Перейти по ссылке в пришедшем письме. Ожидаемый результат: Письмо с подтверждением регистрации пришло на указанный адрес, и при переходе по ссылке отображается сообщение об успешном завершении регистрации.

Как правило, на каждую часть функционала пишутся свои тестовые сценарии, по которым впоследствии можно проверить правильность его реализации.

Из опыта разработки ПО известно, что повторное появление одних и тех же ошибок — случай достаточно частый. Иногда это происходит из-за слабой техники управления версиями или по причине человеческой ошибки при работе с системой управления версиями. Но настолько же часто решение проблемы бывает «недолго живущим»: после следующего изменения в программе решение перестаёт работать. И наконец, при переписывании какой-либо части кода часто всплывают те же ошибки, что были в предыдущей реализации. Поэтому считается хорошей практикой при исправлении ошибки создать тест на неё и регулярно прогонять его при последующих изменениях программы. Хотя регрессионное тестирование может быть выполнено и вручную, но чаще всего это делается с помощью специализированных программ, позволяющих выполнять все регрессионные тесты автоматически. В некоторых проектах даже используются инструменты для автоматического прогона регрессионных тестов через заданный интервал времени. Обычно это выполняется после каждой удачной компиляции (в небольших проектах) либо каждую ночь или каждую неделю.

Модульное тестирование, или юнит-тестирование (англ. unit testing) — процесс, позволяющий проверить на корректность отдельные модули исходного кода программы.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Модульное тестирование позже позволяет программистам проводить рефакторинг, будучи уверенными, что модуль по-прежнему работает корректно (регрессионное тестирование). Это поощряет программистов к изменениям кода, поскольку достаточно легко проверить, что код работает и после изменений.

Модульное тестирование помогает устранить сомнения по поводу отдельных модулей и может быть использовано для подхода к тестированию «снизу вверх»: сначала тестируя отдельные части программы, а затем программу в целом.

Для проведения модульного тестирования данного проекта использовался   
Unit Testing Framework – это встроенная в Visual Studio система тестирования, разрабатываемая Майкрософт, постоянно развивающаяся, и что немаловажно, она почти наверняка будет существовать все время, пока есть Visual Studio, чего нельзя сказать о стронних разработках. Другое преимущество – отличная интеграция в IDE и функция подсчета процента покрытия кода в программе. На рисунке 5.4 показана часть снимка экрана с test explorer в visual studio.

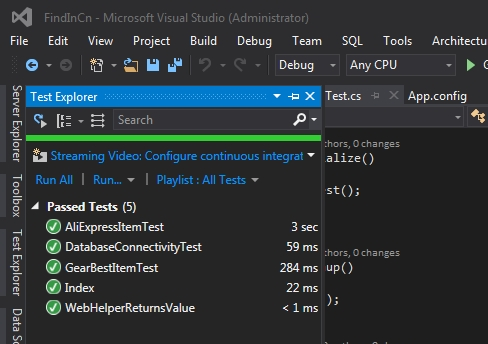


Рисунок 5.4 – test explorer.

В test explorer находится список тестов для проекта. Для нахождения тестов используются аттрибуты. Атрибут [TestClass] означает, что этот класс содержит тестовые методы, а [TestMethod] – что такой метод представляет собой конкретный метод. Зелеными в списке отображаются успешно пройденные тесты, оранжевыми – пропущенные, и красными – тесты, по каким-то причинам не прошедшие. Возле каждого теста пишется время его прохождения. Также тесты можно сгруппировать по разным категориям, например по тестируемым классам, прошедшие/непрошедшие, и т.д.

Для непрошедших тестов можно увидеть причину провала. Это может быть исключение, несоответствие ожидаемого результата с реальным, и т.д. На рисунке 5.5 показан непрошедший тест.

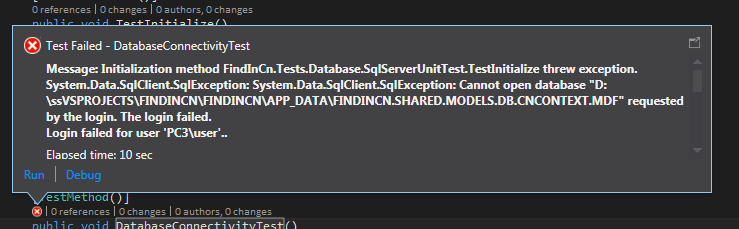


Рисунок 5.5 – failed test.

В данном случае тест не прошел из-за невозможности подключения к базе данных, и проблема решается заменой заведомо неверной строки подключения на правильную. В случае более непонятных проблем, возможно запустить дебаг для теста, и пошагово отследить его выполнение, чтобы найти ошибку в коде, либо в самом тесте.

Поскольку некоторые классы могут использовать другие классы, тестирование отдельного класса часто распространяется на связанные с ним. Например, класс пользуется базой данных; в ходе написания теста программист обнаруживает, что тесту приходится взаимодействовать с базой. Это ошибка, поскольку тест не должен выходить за границу класса. В результате разработчик абстрагируется от соединения с базой данных и реализует этот интерфейс, используя свой собственный mock-объект. Это приводит к менее связанному коду, минимизируя зависимости в системе.

5.5 Описание пользовательского интерфейса и работы с системой

При входе в систему пользователь попадает на главную страницу сайта. На ней расположены панель с категориями товаров, список поддерживаемых магазинов, при клике на который можно кликнуть, для получения более подробной информации о магазине и расширенного списка именно его категорий. На рисунке 5.6 показан снимок экрана с главной страницей сайта.

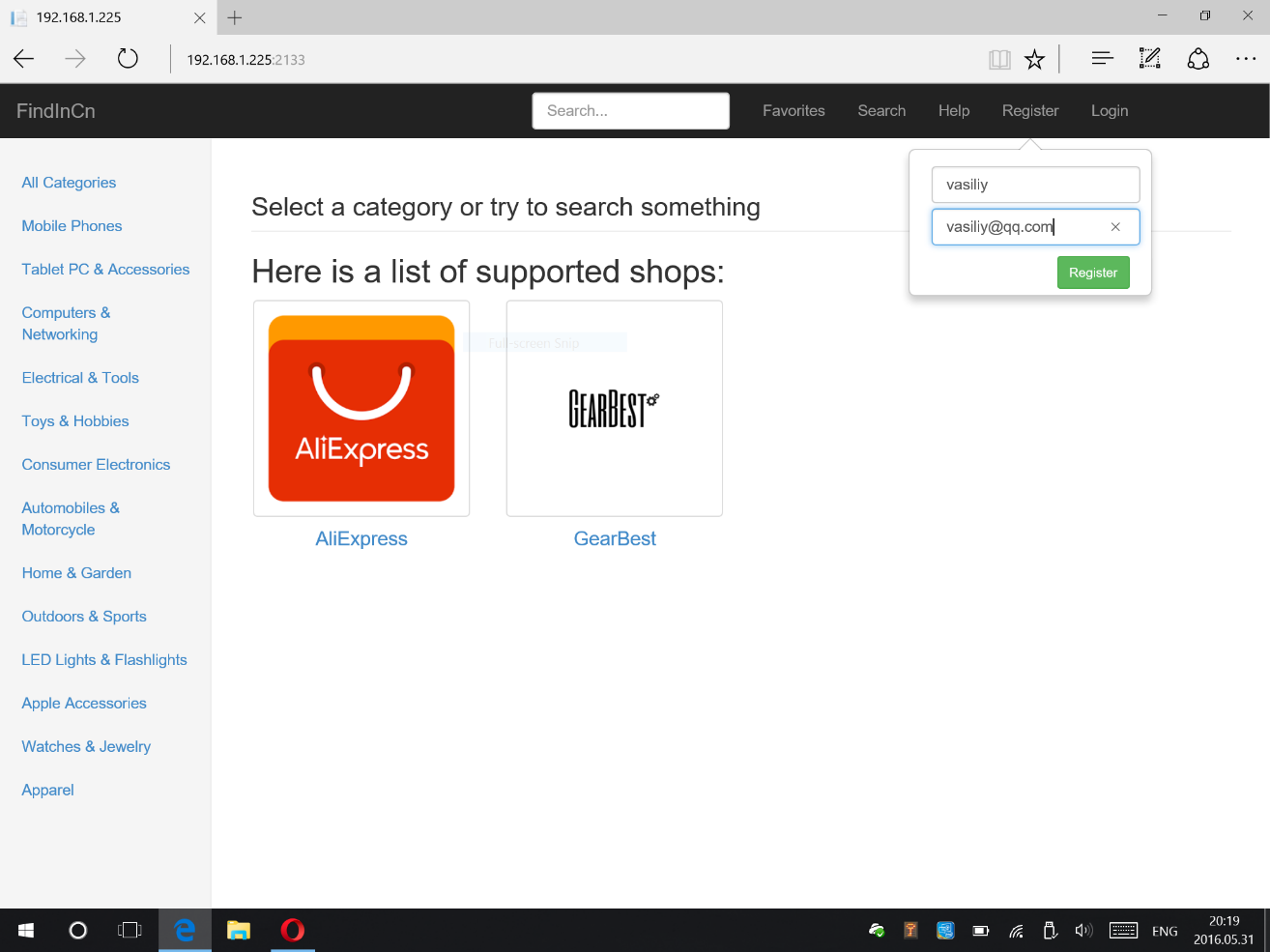


Рисунок 5.6 – Главная страница сайта.

Боковая панель реализована с использованием технологии grid system в bootstrap, которую используют для создания разметки страницы с помощью последовательности столбцов или строк, которые содержат необходимый контент. Строки должны находится внутри контейнеров с классом .container (фиксированной ширины) или .container-fluid (полноекранный). Также доступны предопределенные классы, например .row и .col-xs-4, которые помогают с более быстрым созданием разметки. В классе .col-xs-4 цифра означает ширину столбца, при этом 12 это вся ширина экрана. Часть xs sm md lg и lg обозначает размер экрана устройства, что позволяет создавать адаптивную верстку для разных размеров экрана разных классов устройств -- от мобильных устройств до широкоформатных мониторов. Тут xs означает extrasmall, md – medium, и так же остальные по аналогии, от соответствующих английских слов. Если в строке будет занято более 12 единиц измерения колонок, то лишние будут перенесены на следующую строку.

На верхней панели расположены кнопки избранного, поиска, регистрации, логина, поиска и поле для быстрого поиска. При нажатии на кнопку регистрации или логина появляется всплывающее окно, как показано на рисунке 5.7.

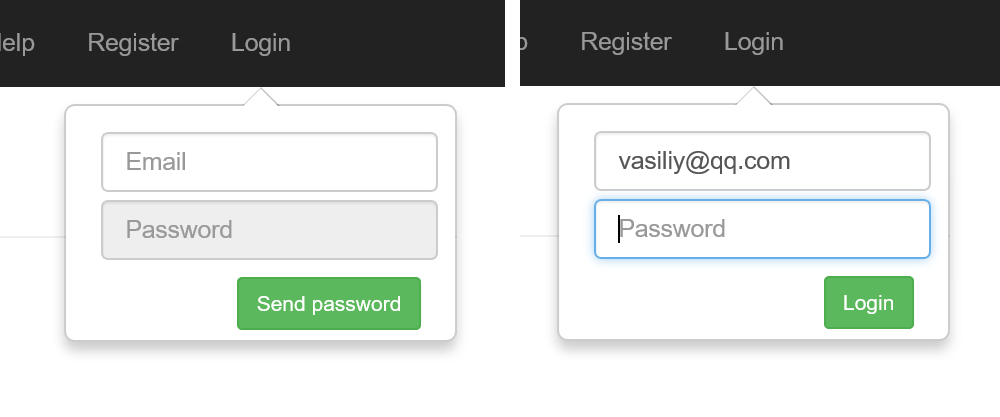


Рисунок 5.7 – Окно логина.

При этом переход на отдельную страницу не осуществляется. После ввода данных в окне регистрации, пользователю на почту приходит письмо с подтверждением, и из этого письма он должен перейти на сайт по ссылке.

В окне логина изначально доступно только поле email. После ввода почты и нажатии на кнопку, отправляется временный пароль, и разблокируется поле для его ввода. Также при этом меняется надпись на кнопке. Для всплывающего окна используется bootstrap popover. Также было проверено, что эти технологии работают с мобильными браузерами. На рисунке 5.8 показан скриншот с мобильного браузера safari, с открытой формой логина.

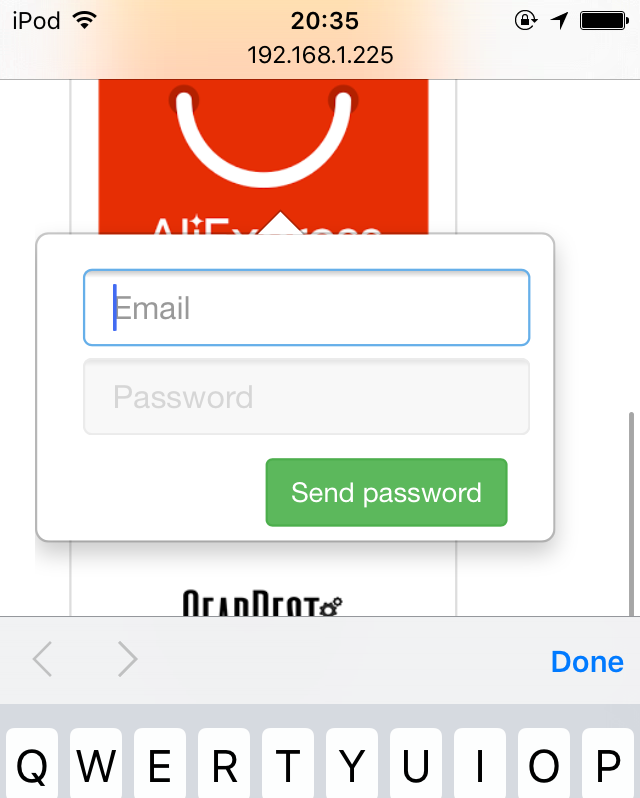


Рисунок 5.8 – Окно авторизации в мобильном браузере.

Следующий пример кода показывает функции асинхронной отправки данных на сервер, которая вызывается при нажатии на кнопку в обработчике события onclick.

function login() {

... //тут проверка правильности ввода

$.ajax({

method: "GET",

url: "/Account/AjaxLogin",

data: {

email: $('#email').val(),

password: $('#password').val()

},

error: function () { alert('error'); }

}).done(function (jsonString) {

var data = JSON.parse(jsonString);

if (data.status == 'sent') {

$('#dologin').html('Login');

$('#password').removeAttr('disabled');

}

else ... //продолжение

В данном примере функция JQuery отправляет запрос по адресу /Account/Login c параметрами email и password. После получения ответа, javascript код проверяет статус. Если был отправлен пароль, разблокируется поле для его ввода и меняется надпись на кнопке, и.т.д. Полную версию кода можно увидеть в приложении либо в репозитории.

Било также реализовано адаптивное поведение строки меню. На рисунке 5.9 показаны скриншоты с открытым и закрытым меню.

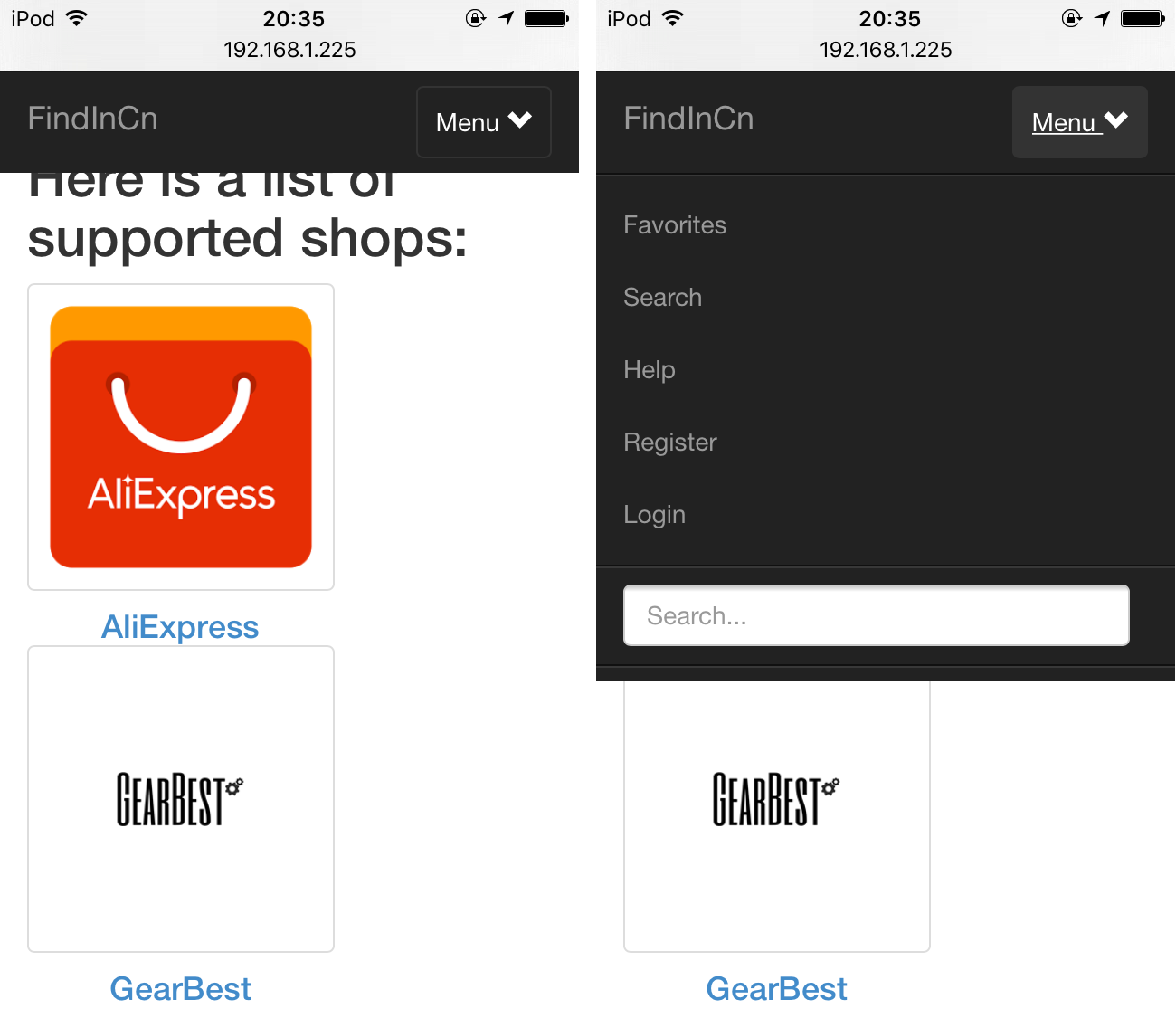


Рисунок 5.9 – Строка меню в мобильном браузере.

Для реализации такого поведения было создано два стиля – для ширины экрана меньше 768 пикселей (мобильные устройства), и больше. На маленьких экранах элементы меню превращаются в скрытый список, а также становится видимой кнопка сворачивания/разворачивания меню. На данной части кода показана часть разметки с кнопкой.

<div class="navbar-header">

<a class="navbar-brand" href="/">FindInCn</a>

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false">

<a style="color:white">Menu <span class="glyphicon glyphicon-chevron-down" aria-hidden="true"></span></a>

</button>

</div>

<div id="navbar" class="collapse navbar-collapse">

<ul class="nav navbar-nav navbar-right" style="margin-right:100px;">

Тут button – кнопка , в которой содержится надпись «меню» и иконка со стрелкой. Класс navbar-toggle указывает, что кнопка используется переключает панель.

При вводе в поле поиска ключевого слова, пользователь попадает на страницу результатов поиска. Поскольку загрузка всех данных и их агрегированное занимают какое-то время, сначала загружается страница, и асинхронно загружаются результаты.

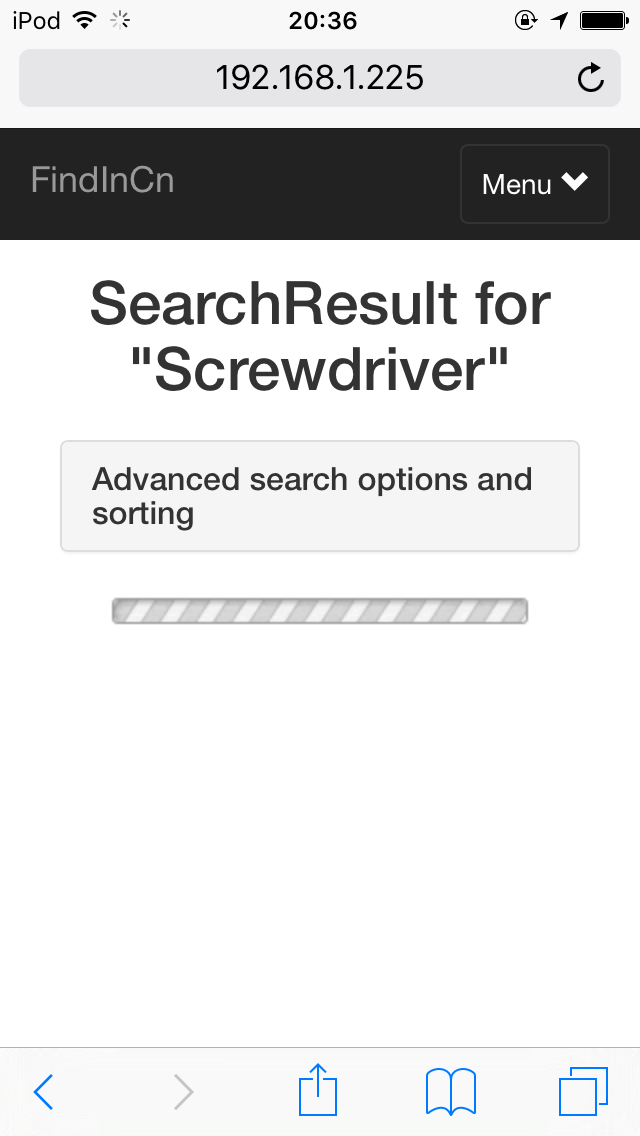


Рисунок 5.10 – Загрузка результатов поиска

На рисунке 5.10 показано окно с анимацией, которая показывается пользователю до окончания загрузки всех данных. На странице также есть раскрывающееся меню с дополнительными настройками поиска. После окончания загрузки, пользователь видит страницу с результатами поиска, показанную на рисунке 5.11.

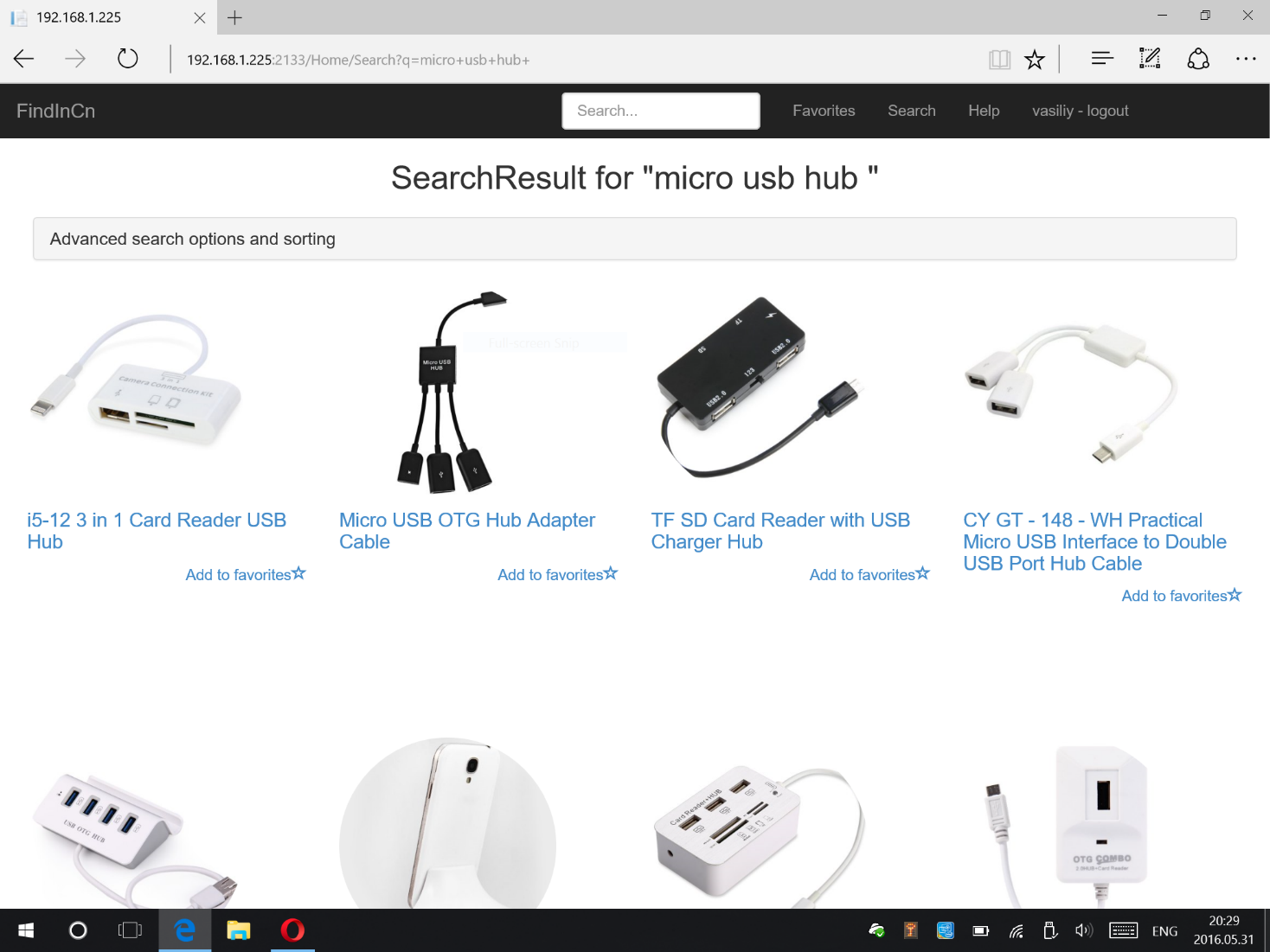


Рисунок 5.11 – Страница результатов поиска.

При нажатии на товар, пользователь попадет на страницу сайта, где он сможет посмотреть полную информацию. Возле каждой записи на странице результатов есть кнопка add to favorites, при нажатии на которую, товар добавляется пользователь в избранное. Если пользователь не залогинен во время нажатия кнопки, то откроется форма логина, и после авторизации выбранный товар все-равно будет добавлен. Страница «Избранное» показана на рисунке 5.12.

Тут пользователь может кликнуть на изображение для просмотра более подробной информации и перехода на сайт. Для сравнения двух товаров пользователю необходимо кликнуть на них add to compare. После первого нажатия кнопка становится неактивной, и после выбора второго товара происходит перенаправление на страницу сравнения, на которой пользователь может сравнить различные параметры товаров.

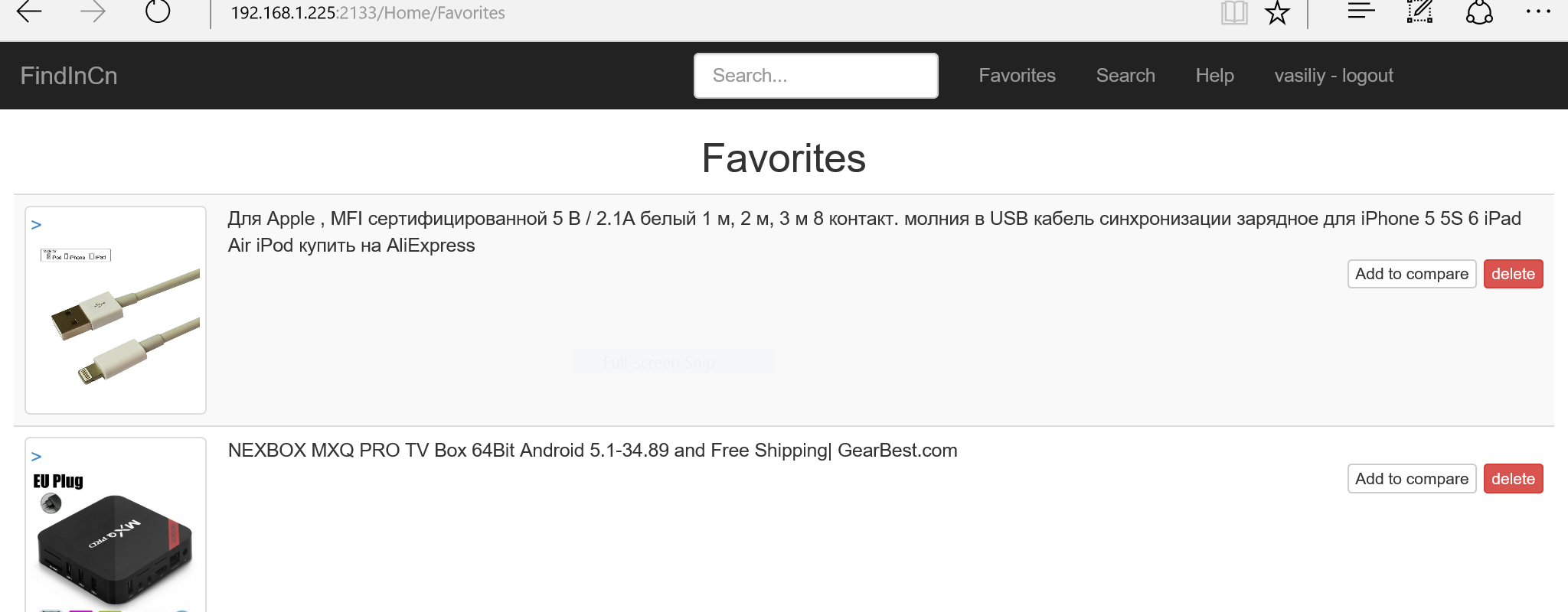


Рисунок 5.12 – Страница избранного.

При сравнении множества свойств сравниваемых товаров может не совпадать когда у одного товара их больше чем у другого, это разные классы товаров, либо продавцы разных магазинов указали слишком различное описание. В этом случае система вначале показывает общие параметры, а затем уникальные для каждого товара. На рисунке 5.13 показана страница сравнения на которой в начале списка идут общие параметры.

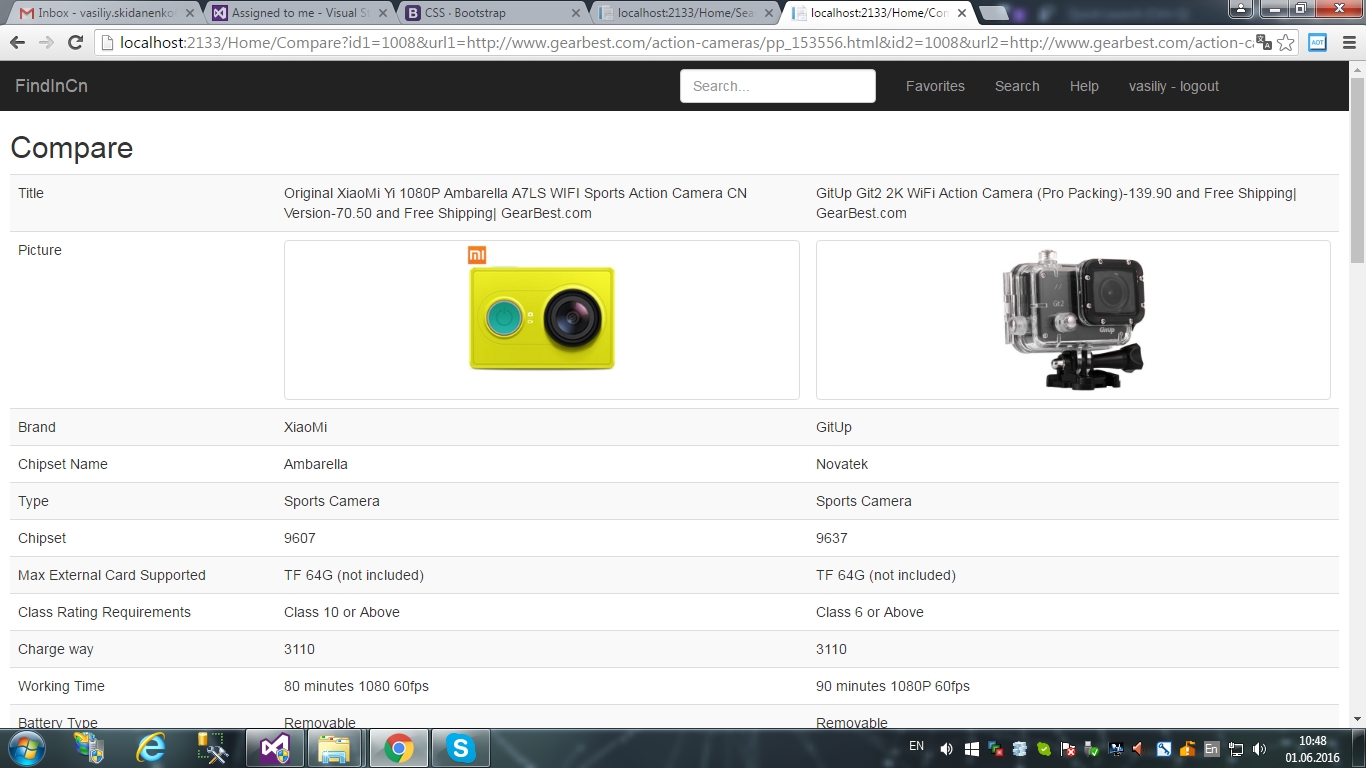


Рисунок 5.13 – Страница сравнения товаров.

Если пользователь выберет совсем несовместимые классы товаров, например молоток и флеш накопитель, то свойств которые можно сравнить будет совсем немного, и они будут неинформативными, так как пользователь и так знает, что масса размеры и бренд отличаются, а более конкретные свойства будут уже заполнены только для одного из двух столбцов таблицы.

Кроме динамических параметров, которые могут зависеть от сайта, продавца, и возможно других факторов, в таблице для всех видов товаров всегда можно будет сравнить название, фотографию и цену, т.к. они есть у всех товаров.

ВЫВОДЫ

В ходе разработки дипломного проекта была исследована предметная область, выполнена постановка задачи, введены ограничения, выбраны средства реализации и создана модель системы для улучшения качества обслуживания клиентов.

Была разработана схема базы данных и запросы получения и разбора данных с внешних серверов магазинов. Была освоена работа с технологиями разработки веб приложений с использованием ASP.Net MVC и получены навыки применения шаблонов проектирования MVС.

Был произведен концептуальный анализ предметной области, который использовался

далее для создания системы.

Во время разработки структуры базы данных был проведен сравнительный анализ и изучены основные преимущества современных СУБД для серверных платформ, и выбран Microsoft SQL Server.

Во время разработки дипломного проекта было изучено межсерверное взаимодействие, использование REST сервисов, способы разбора XML данных

Таким образом, в рамках разработки дипломного проекта задание выполнено в полном объеме. Был собран и систематизирован материал для прохождения аттестационной работы, выполнено построение инфологической структуры системы.

ПЕРЕЧЕНЬ ССЫЛОК

1. Фримен, А. ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов [Текст]/ Адам Фримен, 5-е издание. – Изд-во: Вильямс, 2014. – 736 с.

2. Пугачев, С. Разработка приложений для Windows 8 на языке C# [Текст]/ С. Пугачев, А. Шериев, К. Кичинский. – Изд-во: БХВ-Петербург, 2013. – 414 с.

3. Гарсия-Молина, Г. Системы баз данных. Полный курс. [Текст]: Пер. с англ./ Г.Гарсия-Молина, Дж. Ульман, Дж. Уидом. – Изд-во: Вильямс, 2003. – 1088 с.

4. Троелсен Э. C# и платформа .NET. Библиотека программиста [Текст]/ Э. Троелсен. – Изд-во: Питер, 2005. – 796 с.

5. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# [Текст]/ Дж. Рихтер. – Изд-во: Питер, 2013. – 896 с.

6. Мартин, Роберт К. Чистый код: создание, анализ и рефакторинг. Библиотека программиста [Текст]/ Роберт К. Мартин. – Изд-во: Вильямс, 2015. – 464 с.

7. Вільна енциклопедія Википедия [Електронний ресурс] / Мережева енциклопедія Wikipedia. 2000. - Forefront TMG 2010: Режим доступу: http://ru.wikipedia.org/ . - Загл. с екрана.

8. Metanit.сom: Сайт о программировании, про создание сайтов и IT-технологии [Електронний ресурс] / metanit.com, 2012-2014. Режим доступу: http://metanit.com/index.php. - Загл. с екрана.

9. Буч, Г. Язык UML. Руководство пользователя [Текст] / Г. Буч – М.: ДМК Пресс, 2-е издание, 2006. – 248 с.

10. Кренке, Г. Теорія й практика побудови баз даних [Текст] / Г. Кренке. - Спб.: Питер, 2001. - 858с.