Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра электронных вычислительных средств

Дисциплина: Операционные системы и управление базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Разработка информационной системы «Аукцион»

БГУИР КП 1-58 01 01 002 ПЗ

Студент: гр. 110901 Русак А.Д.

Руководитель: Д.С. Лихачёв

Минск 2014

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ…………………………………………………………….…………3

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ………………………….………….4

Интерфейс……………………………………………………….………….5Требования к работе программного средства………………….…………5

2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ...6

3 РАЗРАБОТКА БД……………………………………………………………….9

4 РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ…………………………..…………11

4.1 .NET Framework ……………………………………………….……...11

4.2 Технология ADO.NET …………………………………………….…12

4.3 Среда разработки……………………………………………………...12

4.4 ASP.NET MVC……….………………………………………………..13

4.5 Результат разработки клиентской части……………………….……13

5 ПРОВЕРКА РАБОТОСПОСОБНОСТИ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ…………………………………………...23

ЗАКЛЮЧЕНИЕ……………………………………………………………….…25

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………………………26

ПРИЛОЖЕНИЕ А - структура информационной системы………………….27

ПРИЛОЖЕНИЕ Б - структура БД ………………………………………….…..29

ПРИЛОЖЕНИЕ В - листинг программы……………………………………....30

**ВВЕДЕНИЕ**

К настоящему времени человечеством накоплено поистине гигантское количество информации об объектах и явлениях. Но эта информация не лежит мертвым грузом, она хранится в электронном виде и используется в базах данных. Базы данных – это часть информационных систем – программно-аппаратных комплексов, осуществляющих хранение и обработку огромных информационных массивов. Базы данных, как способ хранения больших объемов информации и эффективного манипулирования ею, используются практически во всех областях человеческой деятельности.

База данных была разработана мной с помощью СУБД Microsoft SQL Server, так как Microsoft SQL Server - одна из наиболее мощных систем работы с базами данных. Курсовой проект разработан на платформе .NET Framework с использованием технологии ADO.NET, предоставляющей доступ к данным для приложений, основанных на [Microsoft.NET](https://ru.wikipedia.org/wiki/Microsoft_.NET).

Основой платформы .NET Framework является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования, в том числе и С#. C# - это строго типизированный объектно-ориентированный язык, призванный обеспечить оптимальное сочетание удобства, простоты, выразительности и производительности. C# и .NET в некотором смысле являются "родственными душами": некоторые свойства языка специально разрабатывались, чтобы обеспечить комфортную работу в среде .NET, в то же время и некоторые свойства .NET специально закладывались для поддержки C#. Язык C# разрабатывался "с нуля" и вобрал в себя много полезных свойств таких языков, как C++, Java, Visual Basic, а также Pascal, Delphy и др. При этом необходимость обратной совместимости с предыдущими версиями отсутствует, что позволяет языку C# избежать многих отрицательных сторон своих предшественников.

Данный курсовой проект посвящен информационной системе «Аукцион», которую я создал при помощи вышеуказанных технологий на языке C#.

**1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

Основная цель данного клиентского приложения – предоставить пользователям возможность выставлять свои товары на продажу или искать лоты для покупки. Программный продукт должен позволять осуществлять возможность зарегистрироваться и войти как пользователь или как администратор. Зарегистрированный пользователь может искать лот, добавить лот, добавить описание лота, установить для него цену, срок действия, добавить картинку к лоту. Указать раздел для товара. Также зарегистрированный пользователь может редактировать данные о себе, просматривать и редактировать информацию о лотах и о себе. Может добавлять комментарии к своему лоту и к чужому. Имеет право купить чужой товар. Администратор имеет право редактировать информацию о лотах пользователей, о самом пользователе, создавать свои лоты, редактировать их, производить поиск лотов.

В программе должны быть реализованы следующие компоненты:

1. Хранение всей информации в базе данных.
2. Возможность зарегистрироваться.
3. Возможность входа под пользователем и администратором.
4. Зарегистрированный пользователь может создать лоты и редактировать информацию о них.
5. Пользователь может редактировать информацию о себе.
6. Пользователь может искать лоты.
7. Администратор имеет возможность редактировать лоты пользователя и информацию о пользователе.
8. Администратор может создавать свои лоты и редактировать информацию о них.
9. Администратор может искать лоты.
   1. Интерфейс

Программная оболочка должна иметь интуитивно понятный интерфейс. Пользовательский интерфейс должен быть выполнен в доступном виде и привлекать своим визуальным оформлением.

Программа целиком состоит из форм, каждое из которых отвечает за свой раздел и конкретные функции:

* страница регистрации;
* страница входа в систему;
* стартовая страница, где предоставляется выбор между созданием лота и поиском лота;
* страница создания лота;
* страница поиска лота;
* страница с информацией о лоте;
* страница пользователя;
* страница редактирования информации о пользователе;
* страница с информацией о другом пользователе.

**1.2 Требования к работе программного средства**

Для нашего приложения предъявляются следующие требования:

* разрабатываемое программное средство должно корректно работать;
* высокая скорость работы;
* визуально понятный интерфейс.

**2 ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Информационная система - это совокупность механизмов, обеспечивающих полное осуществление информационного процесса. Информационный процесс - это осуществление всей совокупности следующих элементарных информационных актов: прием или создание информации, ее хранение, передача и использование. Современные ИС де-факто немыслимы без использования баз данных и СУБД, поэтому термин «информационная система» на практике сливается по смыслу с термином «система баз данных». Структура системы - это множество всех возможных отношений между подсистемами и элементами внутри системы. [4]

В моём курсовом проекте я разрабатывал информационную систему, предоставляющую возможность выставления вещей на продажу. Целью данной информационной системы является выставление товаров на продажу или поиск и покупка нужных товаров в удобном виде и частичная автоматизация работы с ней.

Как уже было сказано выше, для хранения информации система использует базу данных под управлением СУБД Microsoft SQL Server. В базе данных хранится вся необходимая информация для работы информационной системы. База данных под управлением СУБД является подсистемой нашей системы. Второй подсистемой системы является клиентская часть. Клиентская часть нужна для удобства работы с базой и получения нужной информации. Она реализована с помощью ASP.NET MVC Framework в виде веб-приложения.

Как правило, компьютеры и программы, входящие в состав информационной системы, не являются равноправными. Некоторые из них владеют ресурсами (файловая система, процессор, принтер, база данных и т.д.), другие имеют возможность обращаться к этим ресурсам. Компьютер (или программу), управляющий ресурсом, называют сервером этого ресурса (файл-сервер, сервер базы данных, вычислительный сервер и т.д.). Клиент и сервер какого-либо ресурса могут находится как в рамках одной вычислительной системы, так и на различных компьютерах, связанных сетью.

Основной принцип технологии "клиент-сервер" заключается в разделении функций приложения на три группы:

* ввод и отображение данных (взаимодействие с пользователем);
* прикладные функции, характерные для данной предметной области;
* функции управления ресурсами (файловой системой, базой данных и т.д.).

Поэтому, в любом приложении выделяются следующие компоненты:

* компонент представления данных;
* прикладной компонент;
* компонент управления ресурсом.

Итак, выделяют две подсистемы:

* СУБД, содержащая в себе базу данных;
* клиентская часть.

Система управления базой данной «Аукцион» предназначена для использования в домашних условиях.

Системой может пользоваться любой пользователь. Тем не менее, их права не равнозначны.

Для системы существует два вида ролей: администратор и пользователь. Администратор имеет права на редактирование информации о лотах любого пользователя и редактирование информации о самом пользователе. Также создавать свои лоты и редактировать их, а также осуществлять поиск лотов. Пользователь имеет возможность производить какие-либо операции над своими лотами, а также имеет возможность найти нужный для себя товар.

Исходя из вышесказанного, следует отметить, что для системы должны быть реализованы следующие функции:

Редактирование информации – операция доступна администратору системы и любому пользователю. Администратор имеет возможность редактировать любые лоты (свои и других пользователей) и личную информацию (свою и других пользователей), в то время как пользователь может редактировать только свою информацию о лотах и о себе в своём профиле.

Добавление информации – операция доступна администратору и пользователю системы. Пользователь может добавлять новые лоты, добавлять комментарии к своим лотам и к чужим, а администратор может добавлять информацию о лотах (свои и других пользователей) и личную информацию (свою и других пользователей). Также может добавлять комментарии к своим и чужим лотам.

Поиск информации – операция доступна администратору и пользователю. Администратор и пользователь могут искать лоты.

Просмотр информации – функция доступны пользователям и администратору. Администратор и любой пользователь может просмотреть информацию о лоте и о другом пользователе.

Также для предотвращения потери или несанкционированного изменения данных программа контролирует следующие аспекты:

* соблюдение типов данных при заполнении полей;
* операции изменения;
* операции удаления;
* операции сохранения.

**3 РАЗРАБОТКА БД**

База данных – это информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств. Программное обеспечение, предназначенное для работы с базами данных, называется система управления базами данных (СУБД). СУБД используется для упорядоченного хранения и обработки больших объёмов информации.

СУБД организует хранение информации таким образом, чтобы её было удобно:

* просматривать,
* пополнять,
* изменять,
* искать нужные сведения,
* делать любые выборки,
* осуществлять сортировку в любом порядке. [3]

В качестве СУБД для моего курсового проекта была выбрана Microsoft SQL Server 2012.

MS SQL Server 2012 представляет собой систему управления базами данных (СУБД). Этот программный продукт, созданный всем известной компанией Microsoft, применяется для организации реляционных баз данных различного масштаба. Использование СУБД MS SQL Server 2012 дает возможность формировать запросы и осуществлять поиск необходимых данных, синхронизировать информацию, а также выполнять аналитическую обработку данных и получать разнообразные отчеты.

Среди основных преимуществ MS SQL Server 2012 можно выделить следующие:

* производится шифрование как всей базы данных, так и всех файлов, в том числе и журналов,
* благодаря сжатию исходящего потока минимизируется нагрузка на сеть и обеспечивается оптимальная производительность,
* управление производится на основе политик, что дает возможность удобно управлять не только одним, но и несколькими экземплярами СУБД,
* предоставление разработчикам баз данных большого количества новых функций.

В результате была создана база данных «Аукцион» со следующими таблицами:

* Lots;
* Categories;
* Comments;
* Currencies;
* Bids;
* Users;
* UserRoles;
* Roles;
* Profiles.

Для каждой таблицы согласно предметной области были созданы поля, в которых хранится какая-то информация.

В базе данных присутствует связь между таблицами «один ко многим». Связь «один ко многим» («многие к одному») определяет ситуацию, когда одной записи родительской таблицы соответствует несколько записей дочерней таблицы.

Созданная база данных находится в третьей нормальной форме, т.к. она приведена ко второй нормальной форме и каждый не ключевой столбец независим друг от друга: если изменить значение в одном столбце, то изменение в другом столбце не потребуется.

Более подробная информация о базе данных и о связях между таблицами представлена на рисунку Б.1 (см. приложение Б).

1. **РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ**

**4.1 .NET Framework**

Платформа .NET Framework — это технология, которая поддерживает создание и выполнение нового поколения приложений и веб-служб XML. При разработке платформы .NET Framework учитывались следующие цели:

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Обеспечение среды выполнения кода, минимизирующей конфликты при развертывании программного обеспечения и управлении версиями.
* Обеспечение среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем.
* Обеспечение среды выполнения кода, исключающей проблемы с производительностью сред выполнения сценариев или интерпретируемого кода.
* Обеспечение единых принципов работы разработчиков для разных типов приложений, таких как приложения Windows и веб-приложения.
* Разработка взаимодействия на основе промышленных стандартов, которое обеспечит интеграцию кода платформы .NET Framework с любым другим кодом. [2]

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET понятный человеку низкоуровневый язык Common Intermediate Language (CIL) (ранее назывался Microsoft Intermediate Language, MSIL). Затем компилятор производит перевод CIL-кода в объектный байт-код (в терминах .NET получается сборка, англ. assembly), а уже байт-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 неё 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 сама заботится о базовой безопасности, управлении памятью и системе исключений, избавляя разработчика от части работы.

* 1. **Технология ADO.NET**

ADO.NET — это набор классов, предоставляющих службы доступа к данным программисту, работающему на платформе .NET Framework. ADO.NET имеет богатый набор компонентов для создания распределенных приложений, совместно использующих данные. Это неотъемлемая часть платформы .NET Framework, которая предоставляет доступ к реляционным данным, XML-данным и данным приложений. ADO.NET удовлетворяет различные потребности разработчиков, включая создание клиентских приложений баз данных, а также бизнес-объектов среднего уровня, используемых приложениями, средствами, языками и браузерам.

ADO.NET предоставляет согласованный доступ к таким источникам данных, как SQL Server и XML, а также к источникам данных, предоставляемым при помощи OLE DB и ODBC. Пользовательские приложения, использующие общие данные, могут использовать ADO.NET для соединения с этими источниками данных и для получения, обработки и обновления имеющихся в них данных. [2]

* 1. **Среда разработки**

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework и Silverlight.

Visual Studio 2010 (кодовое имя Hawaii, для Ultimate — Rosario; внутренняя версия 10.0) — выпущена 12 апреля 2010 года вместе с .NET Framework 4.0. Visual Studio включает поддержку языков C# 4.0 и Visual Basic .NET 10.0, а также языка F#, отсутствовавшего в предыдущих версиях. [1]

* 1. **ASP.NET MVC**

ASP.NET MVC — фреймворк для создания веб-приложений, который реализует шаблон Model-View-Controller. Данный фреймворк добавлен Microsoft в ASP.NET [2]

Так как веб-страницы являются основной частью приложения, важно уделять особое внимание их внешнему виду и функциям. В конечном счете, веб-страница представляет собой пустую доску, которую разработчик оснащает элементами управления, формируя интерфейс пользователя, и кодом для управления данными. Для этого Visual Studio обеспечивает интегрированную среду разработки, способствующую написанию кодов. Дополняя функциональными возможностями этих элементов управления свои коды, пользователь может легко и быстро разработать необходимое приложение. [2]

* 1. **Результат разработки клиентской части**

С помощью технологий, описанных ранее, была создана информационная система «Аукцион», состоящая из следующих страниц:

Страница регистрации необходима для регистрации пользователей. Для регистрации пользователь вводит своё имя, емаил и пароль в соответствующие поля и нажимает кнопку «Register». Все поля проверяются на правильность ввода значений.

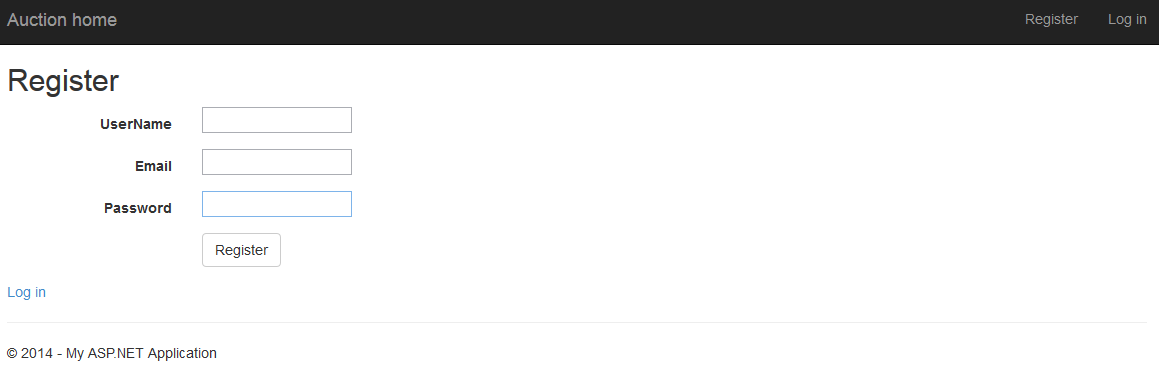


Рисунок 4.1 – Страница регистрации

Страница входа необходима для входа под зарегистрированным пользователем или под администратором. Для входа необходимо ввести емаил и пароль.

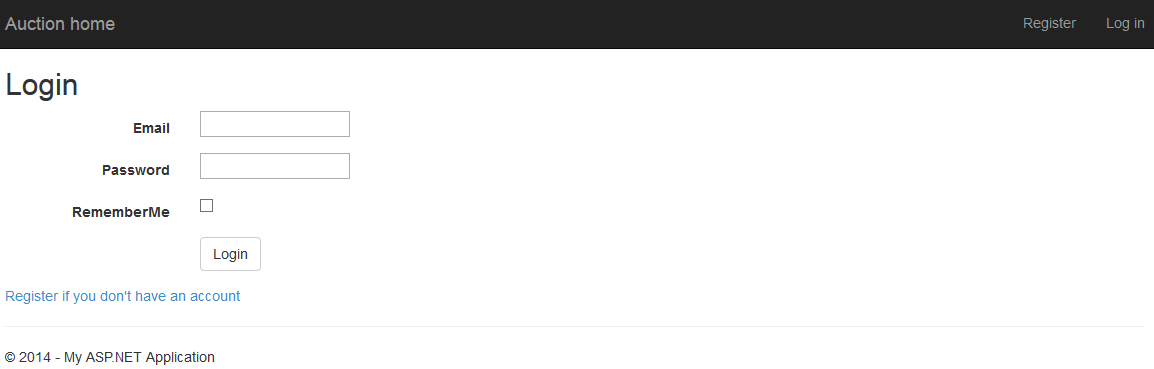


Рисунок 4.2 – Страница входа

Если пользователь войдёт под данными обычного юзера или администратора, то для него будет доступна страница с ссылкой на страницу добавления лота и ссылкой на страницу поиска лотов.

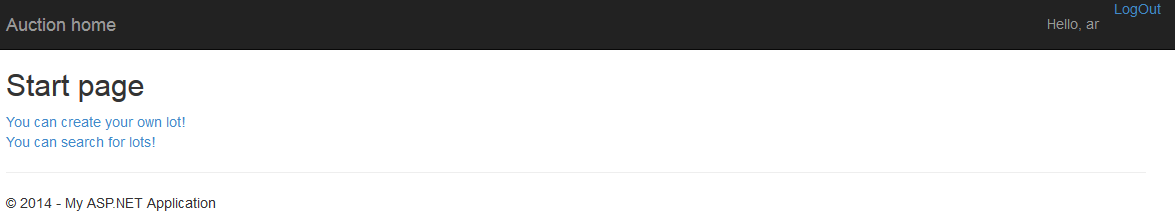


Рисунок 4.3 – Стартовая страница

Если на данной вкладке пользователь или администратор кликнет на «You can create own lot», то перейдёт на страницу создания лота.

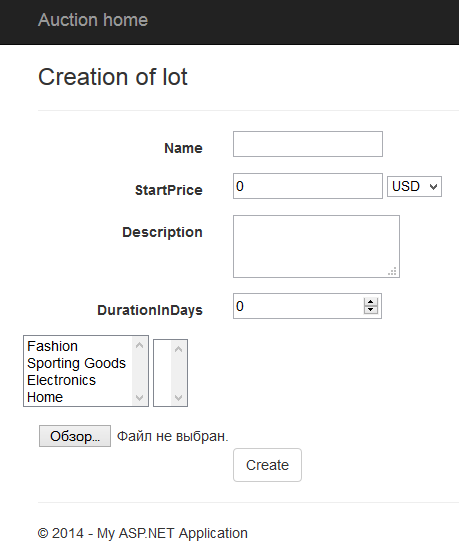


Рисунок 4.4 – Страница создания лота

После нажатия кнопки «Create» пользователь или администратор переходит на личную страницу. На этой странице есть ссылки на редактирование профиля, создание лота и ссылки на созданные лоты.

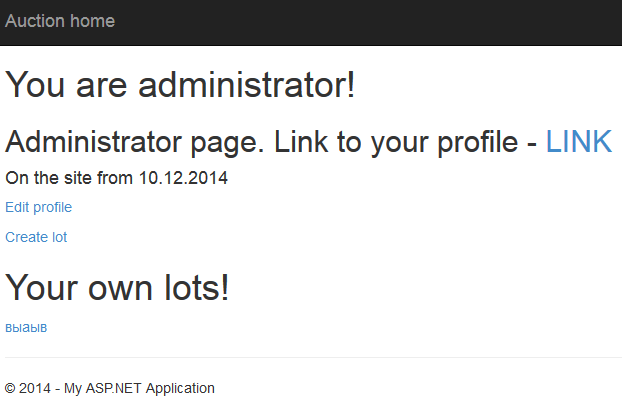


Рисунок 4.5 – Личная страница пользователя

Нажав «Edit profile» пользователь или администратор переходит на страницу редактирования информации о себе.

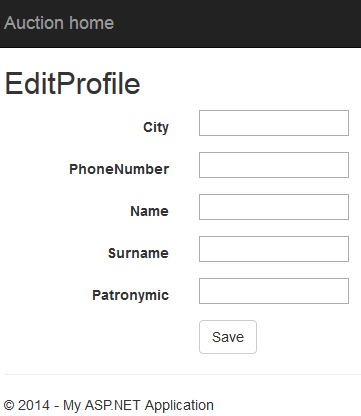


Рисунок 4.6 – Страница редактирования информации о пользователе

Нажав «Create lot» пользователь или администратор переходит на страницу создания лота (рисунок 4.5)

Нажав на название лота администратор или пользователь переходит на страницу информации о лоте и возможность добавить комментарий к лоту.

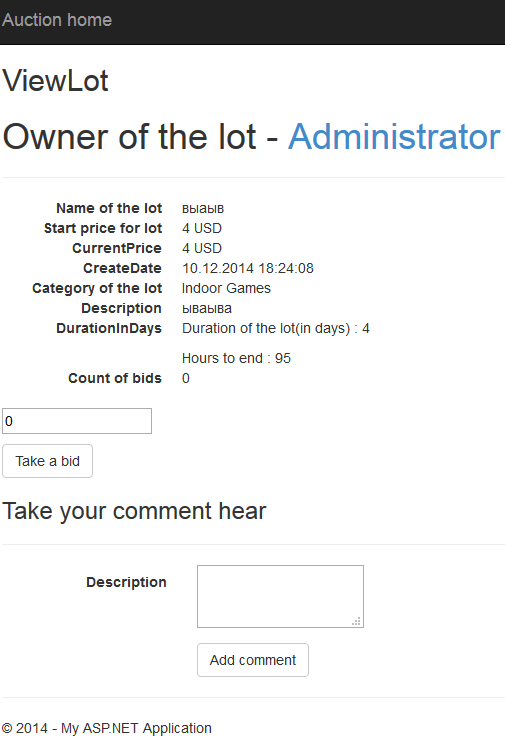


Рисунок 4.7 – Страница с информацией о лоте и добавление комментариев

Нажав на ссылку с именем пользователя происходит переход на страницу профиля с информацией о пользователе, о лотах пользователя и лоты, на которые была сделана ставка.

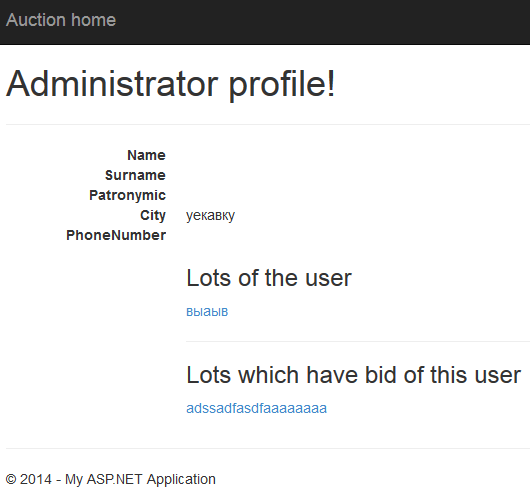


Рисунок 4.8 – Страница профиля

Если пользователь переходит по ссылке «You can search for lots!» на стартовой странице (рисунок 4.4.), то происходит переход на страницу поиска лотов. На странице также сразу предоставлены ссылки на просмотр информации о лотах

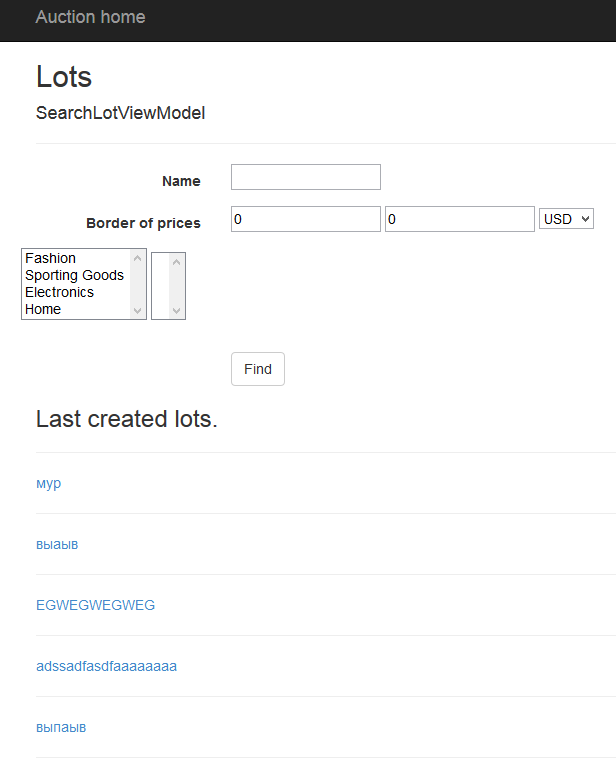


Рисунок 4.9 – Страница поиска для пользователя

Для администратора также предоставлены ссылки для редактирования информации лотов.

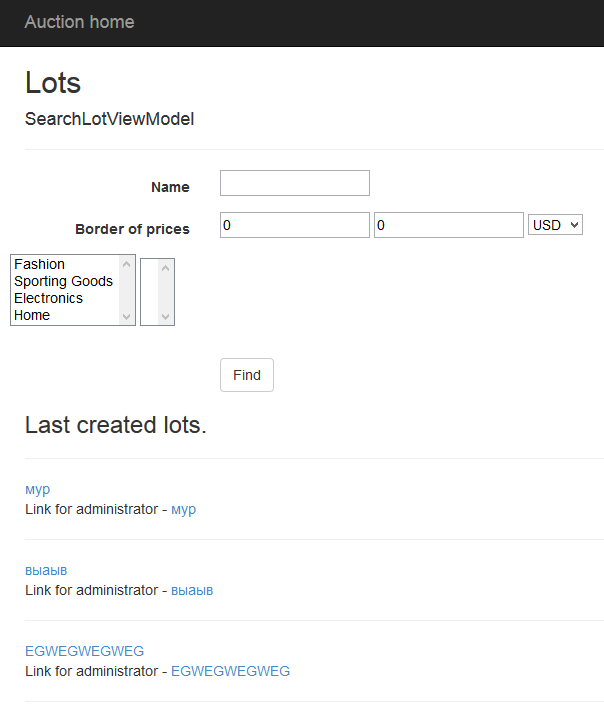


Рисунок 4.10 – Страница поиска для администратора

Если администратор кликнет на ссылку для редактирования лотов других пользователей происходит переход на страницу лота с возможностью его редактирования и с информацией о том, кто и какую заявку оставил на лот.

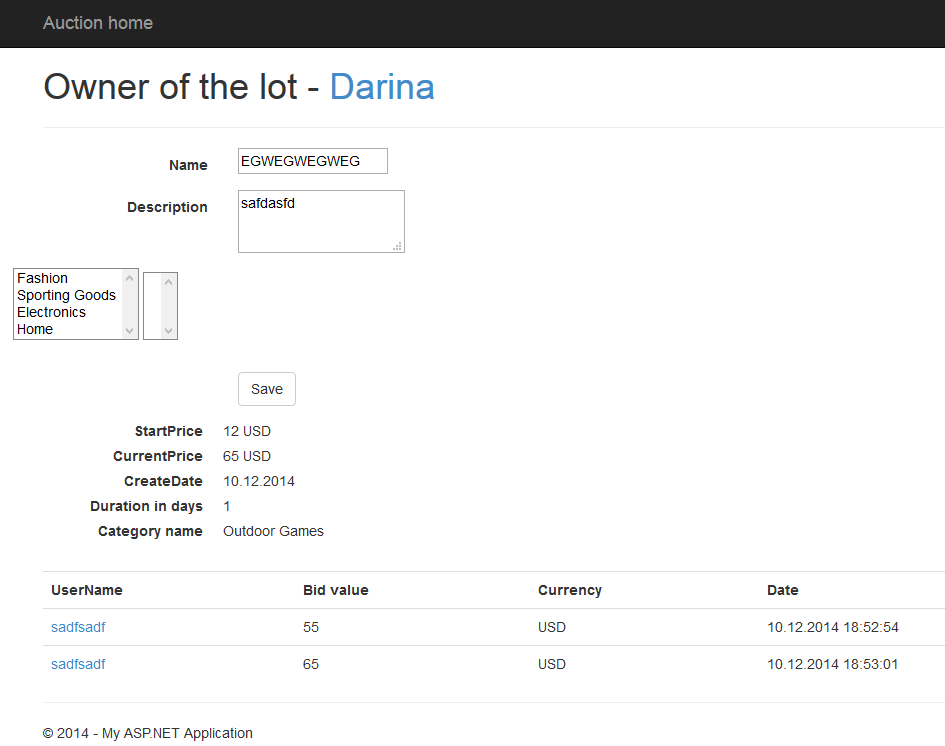


Рисунок 4.11 – Страница редактирования лота

Если администратор нажмёт на ссылку с именем пользователя на странице редактирования лота (рисунок 4.12), то произойдёт переход на страницу редактирования информации пользователя.

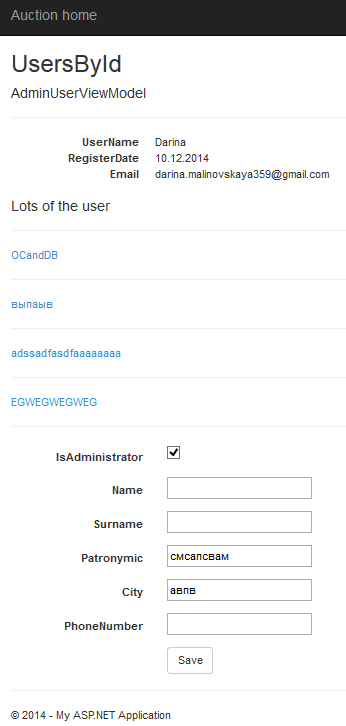


Рисунок 4.12 – Страница редактирования информации о пользователе

# 5 ПРОВЕРКА РАБОТОСПОСОБНОСТИ РАЗРАБОТАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Для проверки работоспособности разработанной системы необходимо выполнить проверку входа в систему, основного функционала, а также реакцию системы на ввод некорректных данных в различные поля. Для проверки функциональных возможностей будет использовано техническое задание (раздел 1 «Анализ технического задания»), описание требований к основному функционалу системы (раздел 2 «Проектирование структуры информационной системы»). Контрольный лист представлен в таблице табл.5.1 Контрольный лист проверки системы.

Таблица 5.1 – Контрольный лист проверки системы

|  |  |  |
| --- | --- | --- |
| Полномочие | Описание | Результат тестирования |
| 1. Регистрация пользователя | Возможность регистрации пользователя в систему. | Выполнено |
| 2. Вход в систему под администратором. | Получение доступа к дополнительным возможностям системы с использованием индивидуальной учетной записи администратора при заполнении полей для входа. | Выполнено |
| 3. Вход в систему под пользователем. | Получение доступа к системе с использованием индивидуальной учетной записи пользователя при заполнении полей для входа. | Выполнено |
| 4. Создание лота. | Создание лота при заполнении всех полей, возможность вставить картинку для лота. | Выполнено |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| Полномочие | Описание | Результат тестирования |
| 5. Редактирование лота | Изменённые данные сохраняются правильно. | Выполнено |
| 6. Редактирование учётной записи пользователя | Изменённые данные сохраняются правильно. | Выполнено |
| 7. Добавление комментария | Возможность разместить комментарий под лотом. | Выполнено |
| 8. Покупка лота | Возможность оповестить пользователя о покупке лота, возможность указать сумму за лот. | Выполнено |

По представленным в данном разделе данным видно, что система соответствует изначально поставленным целям, отвечает всем заявленным функциональным требованиям и готова к использованию.

**Headline**

Аукцион: возникает ошибка на сайте если незарегистрированный пользователь пытается открыть лот

**Description**

1. Запустить приложение
2. Нажать ссылку «[You can search for lots!](http://localhost:51606/search/lots)»
3. Нажать на любой лот из раздела «Last created lots»

**Actual Result**

Ошибка на сайте

**Expected Result**

Переход на страницу входа пользователя

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была создана информационная система «Аукцион», позволяющая продать, купить товар на аукционе. Программа имеет удобный пользовательский интерфейс с необходимыми пунктами меню. Работа с информацией производится в браузере. Информация хранится в базе данных (Microsoft SQL Server).

Данные загружаются из базы данных Microsoft SQL Server 2012. Программный продукт выполнен в среде Visual Studio 2012.

Проведенное тестирование работы приложения не выявило ошибок. Это, однако, не исключает возможности их появления при проведении более глубокого и длительного тестирования.

Программное средство предназначено для работы в любых современных браузерах, что и является главным преимуществом веб-приложения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Википедия [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki.

[2] Microsoft Developer Network [Электронный ресурс]. – Режим доступа : http://msdn.microsoft.com/.

[3] Базы данных [Электронный ресурс]. – Режим доступа : http://www.infosgs.narod.ru/31.htm.

[4] Информационная система [Электронный ресурс]. – Режим доступа : http://www.irkinfo.ru/informatsionnye-protsessy-v-organizatsionno-ekonomicheskoi-sfere.html.

[5] Вывод сообщений в C# [Электронный ресурс]. – Режим доступа : http://vbbook.ru/visual-c.net/vuvod-soobsheniiy-v-c-messagebox/.

[6] Сайт для программистов [Электронный ресурс]. – Режим доступа : http://vbbook.ru/.

[7] КиберФорум [Электронный ресурс]. – Режим доступа : http://www.cyberforum.ru/.

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Структура информационной системы

# Таблица А.1 – Структура информационной системы

|  |  |
| --- | --- |
| Действие | Результат |
| Запуск приложения | Открывается стартовая страница. Возможен переход к регистрации или странице входа |
| Стартовая страница | Возможен переход к созданию лота и поиска лота. При нажатии на ссылки создания лота и поиска происходит автоматический переход на страницу входа в систему |
| Страница регистрации | Для регистрации пользователь вводит своё имя, емаил и пароль в соответствующие поля и нажимает кнопку «Register». |
| Страница входа | Для входа необходимо ввести емаил и пароль. Возможен вход под пользователем или под администратором |
| Вход как пользователь | Переход на стартовую страницу |
| Страница создания лота | Создание лота с указанием нужных характеристик, возможность вставки картинки к лоту. После нажатия «Create» переход на личную страницу пользователя |
| Личная страница | Возможность создания нового лота, редактирования профиля, просмотр своих лотов |
| Страница редактирования профиля | Изменения информации о пользователе |
| Поиск лота | Поиск лота по определённым характеристикам. Нажав на лот происходит переход на страницу информации о лоте |
| Страница информации о лоте | Информация о лоте, возможность оставить заявку на покупку, возможность оставить комментарий |
| Вход как администратор | Переход на стартовую страницу, создание лотов, поиск. |
| Страница создания лотов | Создание лота и переход на личную страницу пользователя |
| Личная страница пользователя | Создание нового лота, редактирование профиля, перечень своих лотов |

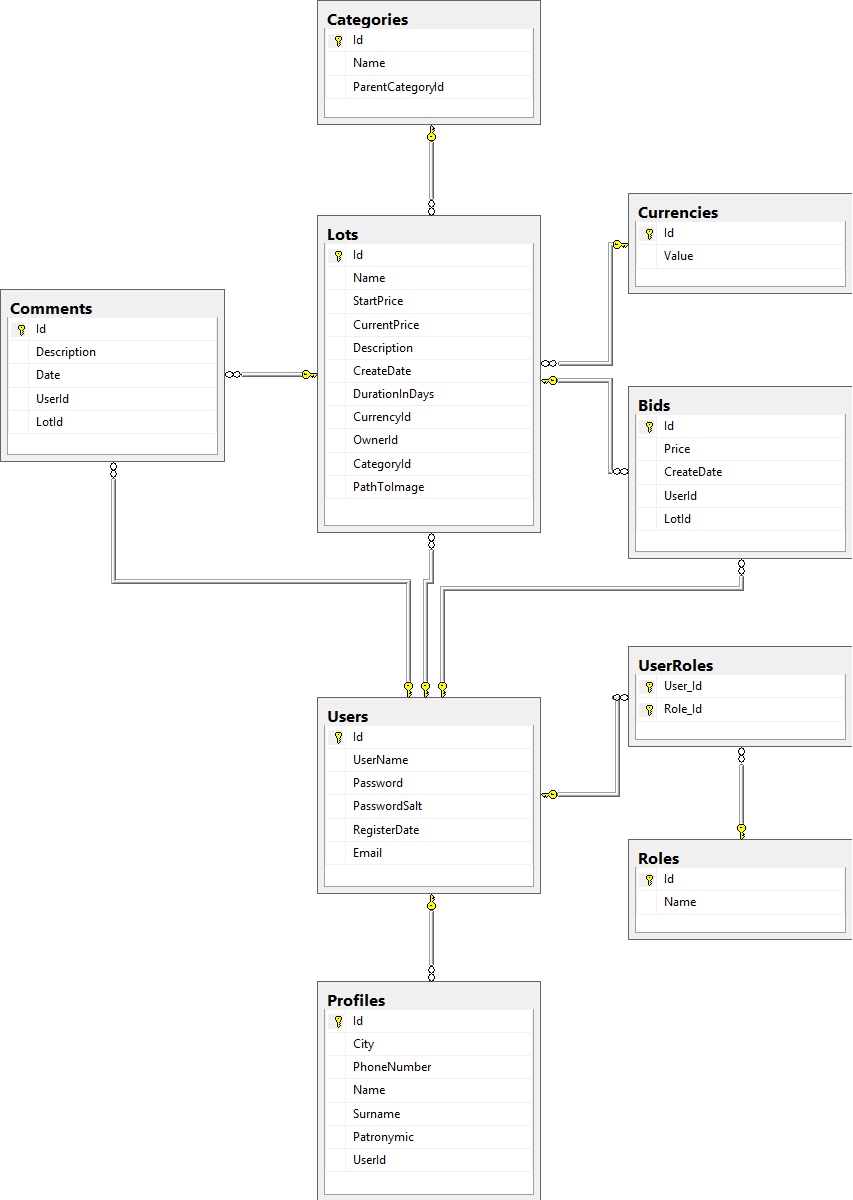
# Продолжение таблицы А.1

|  |  |
| --- | --- |
| Действие | Результат |
| Поиск лотов | Возможна ссылка для администратора, нажав на которую происходит переход на страницу редактирования лота, принадлежащего любому пользователю |
| Страница редактирования лота | Возможность редактирования лота. Просмотр заявок на покупку. Возможность нажать на имя пользователя, что открывает страницу редактирования информации о данном пользователе |

# ПРИЛОЖЕНИЕ Б

# (обязательное)

# Структура БД



# Рисунок Б.1 – Структура БД

# ПРИЛОЖЕНИЕ В

# (обязательное)

# Листинг программы – сервис для работы с пользователями

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using AR.EPAM.Core;

using AR.EPAM.Core.Entities.Membership;

using AR.EPAM.Core.Exceptions;

using AR.EPAM.Infrastructure.Guard;

using AR.EPAM.Services.Exceptions;

namespace AR.EPAM.Services.MembershipServices

{

    public class MembershipService : IService

    {

        private readonly IUnitOfWork \_unitOfWork;

        private readonly IRepositoryFactory \_factoryOfRepositories;

        public MembershipService(IUnitOfWork unitOfWork, IRepositoryFactory factoryOfRepositories)

        {

            Guard.AgainstNullReference(unitOfWork, "unitOfWork");

            Guard.AgainstNullReference(factoryOfRepositories, "factoryOfRepositories");

            \_unitOfWork = unitOfWork;

            \_factoryOfRepositories = factoryOfRepositories;

        }

        public User RegisterUser(string email, string userName, string password)

        {

            //email validation.

            var role = GetRoleByName("Member");

            if (role == null)

            {

                throw new MembershipServiceException("Role doesn't exist");

            }

            var user = GetUserByEmail(email);

            if (user != null)

            {

                throw new MembershipServiceException("User is registered.");

            }

            user = new User { Email = email, UserName = userName, PasswordSalt = DateTime.Now.ToString(), RegisterDate = DateTime.Now };

            Guard.AgainstEmptyStringOrNull(password, "password");

            user.SetPassword(password);

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            userRepository.Create(user);

            user.Roles.Add(role);

            try

            {

                \_unitOfWork.PreSave();

            }

            catch (Exception e)

            {

                throw new ServiceException(e);

            }

            return user;

        }

        public User LoginUser(string email, string password)

        {

            //email validation.

            var user = GetUserByEmail(email);

            if (user == null)

            {

                return null;

            }

            if (user.VerifyPassword(password))

            {

                return user;

            }

            else

            {

                throw new MembershipServiceException("Wrong password.");

            }

        }

        public void ResetPassword(string email, string password, string newPassword)

        {

            var user = GetUserByEmail(email);

            if (user == null)

            {

                throw new MembershipServiceException("User doesn't exist.");

            }

            Guard.AgainstInequalityOfValues(user.Password, (password + user.PasswordSalt).GetHashCode(), "Password is not new.");

            user.PasswordSalt = DateTime.Now.ToString();

            user.SetPassword(newPassword);

            UpdateUser(user);

        }

        public User GetUserById(int id)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                var user = userRepository.GetEntityById(id);

                return user;

            }

            catch (Exception e)

            {

                throw new ServiceException(e.Message);

            }

        }

        public User GetUserByEmail(string email)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                return userRepository.Find(e => e.Email == email);

            }

            catch (ArgumentException ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

            catch (RepositoryException ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

        public User GetUserByUserName(string username)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                return userRepository.Find(e => e.UserName == username);

            }

            catch (ArgumentException ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

            catch (RepositoryException ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

        public Role GetRoleByName(string roleName)

        {

            var roleRepository = \_factoryOfRepositories.GetRoleRepository();

            try

            {

                return roleRepository.Find(e => e.Name == roleName);

            }

            catch (ArgumentException ex)

            {

                throw new ServiceException(ex);

            }

            catch (Exception ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

        public void UpdateUser(User user)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                userRepository.Update(user);

            }

            catch (Exception ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

        public List<User> GetLastRegisteredUsers(int count)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                return userRepository.All().OrderByDescending(e => e.RegisterDate).Take(count).ToList();

            }

            catch (Exception ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

        public List<User> GetUsersByUserNameContaining(string username)

        {

            var userRepository = \_factoryOfRepositories.GetUserRepository();

            try

            {

                return userRepository.Filter(e => e.UserName.Contains(username)).ToList();

            }

            catch (Exception ex)

            {

                throw new MembershipServiceException(ex.Message);

            }

        }

    }

}