Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

**Факультет комп'ютерних наук**

**Кафедра Програмної інженерії**

Курсова робота

з дисципліни «Технологіі компонентного программного забезпечення»

на тему «Інформаційна система «Аукціон»»

Виконав

студент гр. ПІз-11-1 Нікулін А.В.

Перевірив:

К.т.н., доцент Лещинський В. О.

Харків, 2014 р.

\_\_\_\_\_\_\_\_\_Харківський національний університет радіоелектроніки\_\_\_\_\_\_\_\_\_

Факультет комп’ютерних наук. Кафедра програмної інженерії \_\_\_\_\_

Спеціальність Програмна інженерія \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Курс \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Семестр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Начальна дисципліна Розробка бібліотек програмних компонентів\_\_\_\_\_\_\_\_\_\_\_

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ СТУДЕНТОВI

***Никулин Антон Володимирович***

1. Тема роботи: Программне забезпечення системи «Аукціон»»

2. Термін узгодження завдання курсової роботи «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

3. Термін здачі студентом закінченої роботи «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р.

4. Вихідні дані до проекту (роботи): *Використовувати ОС Windows, MS SQL, ORM CodeFirst,ASP.NET MVC 4 середовище об'єктно-орієнтованого проектування Visual Studio 2012*

*5*. Зміст пояснювальної записки *(перелік питань, що належить розробити)*

*мета роботи, аналіз проблемної області і постановка задачі, опис*

*об'єктних моделей, використовувані методи та алгоритми, структура бази*

*даних, опис розробленої програмної системи, захист інформації (за необхідністю), аналіз можливих застосувань.*

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

*Мета завдання, обґрунтування доцільності розроблення, постановка задачі, об'єктна модель системи, базові моделі, методи й алгоритми, структура бази даних, структурно-логічна схема взаємодії даних, план захисту інформації (за необхідністю), інтерфейс програмної системи, результати тестування програмної системи.*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| *№* | *Назва етапів курсової роботи* | *Термін виконання етапів проекту (роботи)* | *Примітка* |
| 1 | Функціональна специфікація програми |  | виконано |
| 2 | Проектування програми |  | виконано |
| 3 | Кодування програми |  | виконано |
| 4 | Тестування програми |  | виконано |
| 5 | Оформлення пояснювальної записки |  | виконано |
| 6 | Захист курсової роботи |  | виконано |

Дата видачі завдання «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ р.

Керівник к.т.н., доцент *Лещинський В. О.*

Завдання прийняв до виконання

студент гр. ПIз-11-1 *\_\_Нiкулiн А.В\_\_\_\_\_*РЕФЕРАТ / ABSTRACT

Пояснювальна записка до курсової роботи: 19 с., 6 рис., 3

додатки, 27 джерел.

Метою роботи э розробка системи електронного аукціону для реалізації будь яких товарів через мережу інтернет за допомогою системи аукціонів.

Методи розробки базуються на технології ASP.NET MVC 4, сервер бази данних, SQLSEVER CE, ORM Entity Framework та веб сервер IIS

У результаті роботи здійснено програмну реалізацію системи "аукціон". З можливістю реєстрації, та участі зареєстрованих користувачів в аукціонах та можливістю створювати свої.

АУКЦІОН, WEB-САЙТ, CODEFIRST, IIS, MVC, ASP.NET,

The purpose of the system engineering is electronic auction for the sale of goods via the Internet through a system of auctions.

Development based on technologies ASP.NET MVC 4, the database server, SQLSEVER CE, ORM Entity Framework and IIS web server  
As a result of work performed the software implementation of the "auction". With the ability to register and participate in auctions of registered users and the ability to create their own.  
AUCTION, WEB-SITE, CODEFIRST, IIS, MVC, ASP.NET,

ЗМІСТ

[ВВЕДЕНИЕ 6](#_Toc403352151)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc403352152)

[2 АРХИТЕКТУРА ПРОГРАММНОГО ПРОДУКТА 9](#_Toc403352153)

[2.1 Общие сведения 9](#_Toc403352154)

[2.2 Модель 9](#_Toc403352155)

[2.3 Сведения о веб сервере 11](#_Toc403352156)

[2.4 Применение технологии AJAX 11](#_Toc403352157)

[2.5 Архитектура базы данных 13](#_Toc403352158)

[2.6 Примеры экранных форм 14](#_Toc403352159)

[ВЫВОД 18](#_Toc403352160)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 19](#_Toc403352161)

ВВЕДЕНИЕ

Одним из наиболее распространенных современных методов программирования является компонентный подход, суть которого заключается в построении программных систем путем интеграции отдельных самостоятельных программных елементов - программных компонентов [1].

Под компонентом понимают независимый модуль для повторного использования и расширения. Применение компонентного программирования призвано обеспечить более простую, быструю и прямолинейную процедуру первой инсталляции прикладного программного обеспечения, а также увеличить процент повторного использования кода, т.е. усилить основные преимущества объектно-ориентированного программирования.

Понятие самостоятельности компонента означает, что он в общем случае создается без ориентации на конкретное применение, то есть предполагается, что компонент будет применяться в различных компонентных системах при условии выполнения требований по его функциональных возможностей и свойств в компонентном среде.

Компонентное программирование - следующий эволюционный шаг на пути развития передовых технологий. Оно представляет собой логическое продолжение структурного и объектно-ориентированного программирования. Компонентное программирование при своем появлении принесло очень важные технологические элементы: единую оболочку для функционирования объектов, унификации способов взаимодействия и доступа к возможностям объектов. Оно позволяет строить программное обеспечение по принципу конструктора - из независимых готовых компонентов, гораздо эффективнее, чем создавать с нуля. Для разработки каждого такого «строительного блока» программист может использовать любой язык программирования. Но самое главное, что обеспечивается прозрачный доступ к удаленным объектам.

Обеспечение условий широкого использования компонента является одной из главных целей на этапе его разработки. Кроме унификации и стандартизации архитектурных, структурных, технологических характеристик компонента, важную роль играет выбор его функциональности и методов доступа. Это, в свою очередь, требует более детального подхода к определению и спецификации компонента в определенной предметной области, конкретных функций обработки данных и тому подобное. Сложность этого подхода заключается в том, что для компонента не существует предопределенных функциональных требований, как в случае спецификации компонентов для конкретной целевой системы. Поэтому необходимым условием разработки компонентов есть условие существования определенных объективных концепций, принципов, критериев по выбору и спецификации их функциональных свойств.

Концепция компонентного программирования подразумевает полное отделение внутренних функций компонента от функций доступа к нему извне. Есть обращаясь к компоненту совсем не обязательно знать его внутреннее строение, для этого достаточно знать лишь то, как вызывать эту функцию. Другими словами, необходимо знать, как взаимодействовать с компонентом, который его интерфейс.

В курсовом проекте решается задачи построение торговой интернет платформы типа аукцион, где каждый зарегистрированный пользователь может создать аукцион для продажи товара.

Программа реализована на языке С# с применением Visual Studio 2012 ASP.NET MVC 4, базой данных, SQLSEVER CE, ORM Entity Framework та веб сервер IIS

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Целью курсового проекта является разработка системы для продажи товаров "Аукцион". Которая призвана облегчить и в значительной степени автоматизировать процесс покупки/продажи различных видов товаров пользователями.

Таким образом предметной областью данного курсового проекта является платформа для продаж типа аукцион.

Требования для программы перечислены ниже:

1. Регистрация новых пользователей
2. Авторизация существующих пользователей
3. Возможность для зарегистрированных пользователей создавать аукционы
4. Зарегистрированные пользователи могут повышать ставки на существующих аукционах
5. Не зарегистрированные пользователи могут только просматривать созданные аукционы.

2 АРХИТЕКТУРА ПРОГРАММНОГО ПРОДУКТА

## 2.1 Общие сведения

Данный программный продукт построен на основе шаблона проектирования MVC. Model-view-controller «модель-представление-контроллер) — схема использования нескольких [шаблонов проектирования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), с помощью которых [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) приложения, [пользовательский интерфейс](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Мы используем эту схему проектирования для построения [архитектурного каркаса](https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29) нашего веб приложения. Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

Модель ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Model). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать. Реализована на основе EntityFramework CodeFirst.

Представление, вид ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) View). Отвечает за отображение информации (визуализацию HTML контента.

Контроллер ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Controller). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

## 2.2 Модель

Для сохранения обьектов в базу данных используется ADO.NET Entity Framework и основанная на ней технология CodeFirst.

ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является [object-relational mapping](https://ru.wikipedia.org/wiki/ORM) (ORM) решением для [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Предоставляет возможность взаимодействия с объектами как посредством [LINQ](https://ru.wikipedia.org/wiki/LINQ) в виде LINQ to Entities, так и с использованием Entity SQL.

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

Entity Framework позволяет работать с данными в форме специфических для домена объектов и свойств, таких как клиенты и их адреса, без необходимости обращаться к базовым таблицам и столбцам базы данных, где хранятся эти данные. Entity Framework дает разработчикам возможность работать с данными на более высоком уровне абстракции; создавать и сопровождать приложения, ориентированные на данные, используя меньше кода, чем в традиционных приложениях. Поскольку Entity Framework является компонентом .NET Framework, приложения Entity Framework могут работать на любом компьютере, где установлена платформа .NET Framework, начиная с версии 3.5 с пакетом обновления 1 (SP1).

Модель хранения и сопоставления при необходимости могут быть изменены без изменения концептуальной модели, классов данных и кода приложения. Модели хранения зависят от поставщика, поэтому можно работать с согласованной концептуальной моделью через различные источники данных.

В Entity Framework файлы модели и сопоставления служат для выполнения операций создания, чтения, обновления и удаления, выполняемых над сущностями и связями концептуальной модели, в эквивалентные операции в источнике данных. Entity Framework поддерживает даже сопоставление сущностей в концептуальной модели с хранимыми процедурами в источнике данных.

Code First позволяет описать модель с помощью классов C# или Visual Basic .NET. Базовая форма модели обнаруживается с помощью соглашений. Соглашения — это наборы правил, используемых для автоматической настройки концептуальной модели на основе определений класса при работе с шаблоном Code First.

При использовании Code First процесс разработки обычно начинается с создания классов .NET Framework, которые определяют концептуальную модель (домен). Кроме определения классов также необходимо сообщить классу **DbContext**, какие типы следует включить в модель. Для этого определяется класс контекста, производный от **DbContext** и предоставляющий свойства **DbSet** для типов, которые должны быть частью модели. Code First включит эти типы, а также запросит все ссылочные типы, даже если последние определены в другой сборке.

Если типы участвуют в иерархии наследования, достаточно задать **DbSet** для базового класса, и производные типы будут включены автоматически, если они находятся в той же сборке, что и базовый класс.

## 2.3 Сведения о веб сервере

Для написания веб серверной части используется ASP.NET MVC. 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.

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. Основой синтаксиса Razor является знак @, после которого осуществляется переход к коду на языках C#. Также возможно и использование сторонних движков. Файлы представлений не являются стандарными статическими страницами с кодом html, а в процессе генерации контроллером ответа с использованием представлений компилируются в классы, из которых затем генерируется страница html.

## 2.4 Применение технологии AJAX

Для отправки сообщение на сервер, в нашем случае о повышении ставки на лот, используется POST запрос который отправляет данные на веб с помощью формы а потом сервер производит перенаправление клиента на страницу и отправкой ее. Поэтапно это выглядит так:

- пользователь заходит на веб-страницу и нажимает на какой-нибудь её элемент.

- [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) формирует и отправляет [запрос](https://ru.wikipedia.org/wiki/HTTP) [серверу](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80).

- в ответ сервер генерирует совершенно новую веб-страницу и отправляет её браузеру и т. д. После чего браузер полностью перегружает всю страницу.

Это создает пересылку множества не нужной информации. Мы отправляем всю страницу хотя требуется лишь небольшая ее часть. Для решения это проблемы мы можем, с помощью технологии AJAX отправлять лишь ту часть которая необходима. AJAX, Ajax (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Asynchronous Javascript and XML — «асинхронный [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) и [XML](https://ru.wikipedia.org/wiki/XML)») — подход к построению интерактивных [пользовательских интерфейсов](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), заключающийся в «фоновом» [обмене данными](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D0%B5%D0%BD_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8) [браузера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) с [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80). В результате, при обновлении данных [веб-страница](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) не перезагружается полностью, и веб-приложения становятся быстрее и удобнее. AJAX — не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

- использование технологии динамического обращения к [серверу](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_%28%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5%29) «на лету», без перезагрузки всей страницы полностью, например с использованием [XMLHttpRequest](https://ru.wikipedia.org/wiki/XMLHttpRequest) (основной объект);

- через динамическое создание дочерних фреймов[[1]](https://ru.wikipedia.org/wiki/AJAX#cite_note-1);

- через динамическое создание тега <script>[[2]](https://ru.wikipedia.org/wiki/AJAX#cite_note-JsHttpRequest-2).

- через динамическое создание тега <img>, как это реализовано в google analytics.

- использование [DHTML](https://ru.wikipedia.org/wiki/DHTML) для динамического изменения содержания страницы;

Действия с интерфейсом преобразуются в операции с элементами [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) 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.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) 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.

## 2.5 Архитектура базы данных

Структура модели БД показана на рисунке 2.1. Где представлены 2 модели. Непосредственно аукциона и модель Bid которая является ничем иным как ставкой пользователя на интересующем его товаре. Содержит такие поля:

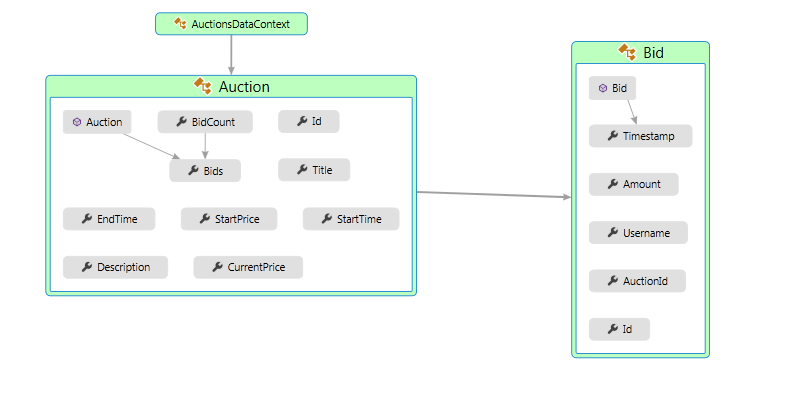


Рисунок 2.1 – Модель аукциона и ставки

**TimeStamp** – время когда была сделана ставка

**Amount** – количество поставленных денег

**Username** – пользователь которой сделал ставку

**AuctionId** – уникальный идентификатор аукциона на который была сделана ставка

**Id** – уникальный идентификатор ставки

Модель Аукциона описывает аукцион созданный пользователем, содержит поля:

**BidCount** – отображает количество ставок в текущем лоте

**Id** – уникальный идентификатор аукциона служит для связи аукциона и ставок

**Bids** – ставки сделаные на этот лот

**Title** – заглавное название аукциона

**EndTime** – время окончания аукциона

**StartTime** – время когда аукцион был начат

**StartPrice** – сума за которую лот был выставлен изначально

**Description** – описание лота

**CurrentPrice** – текущая цена на этом лоте.

## 2.6 Примеры экранных форм

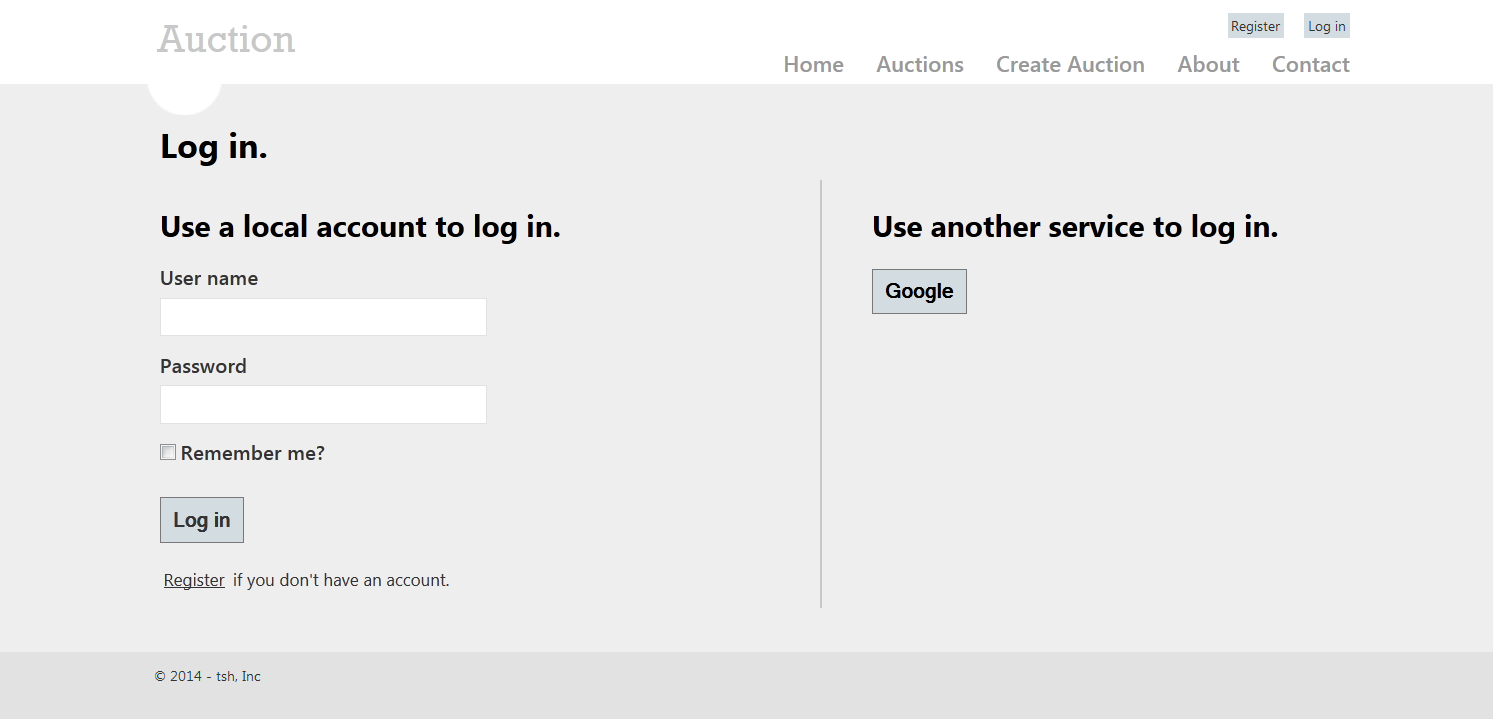


Рисунок 2.2 – Вход в систему

Страница для входа в систему изображена на рисунке 2.2

Здесь пользователю предлагается для входа в систему ввести свои имя и пароль, или войти используя учетную запись google.

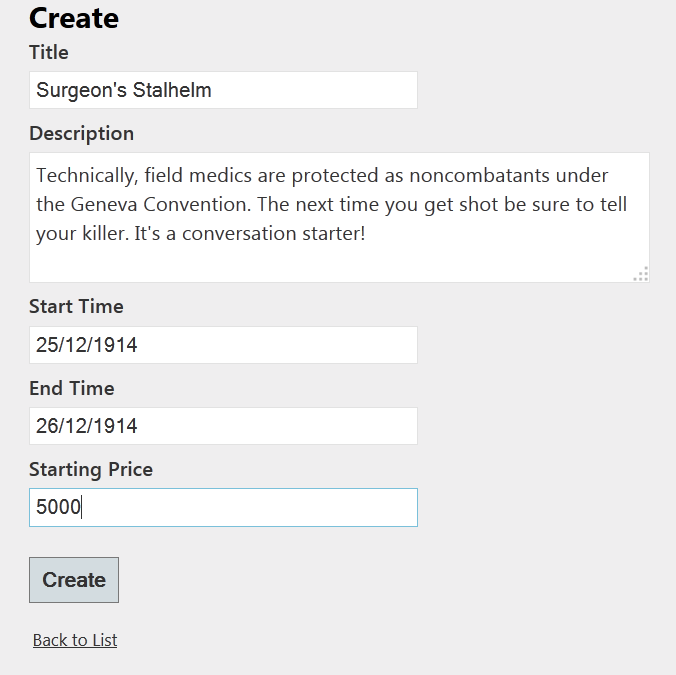


Рисунок 2.3 – Создания лота

На рисунке 2.3 изображен пример создания лота, с заполненными полями. В случае ввода некорректных данных или если поля которые необходимы являются пустыми то системы укажет на ошибку высветив тип ошибки красным цветом рядом с ошибочным полем.

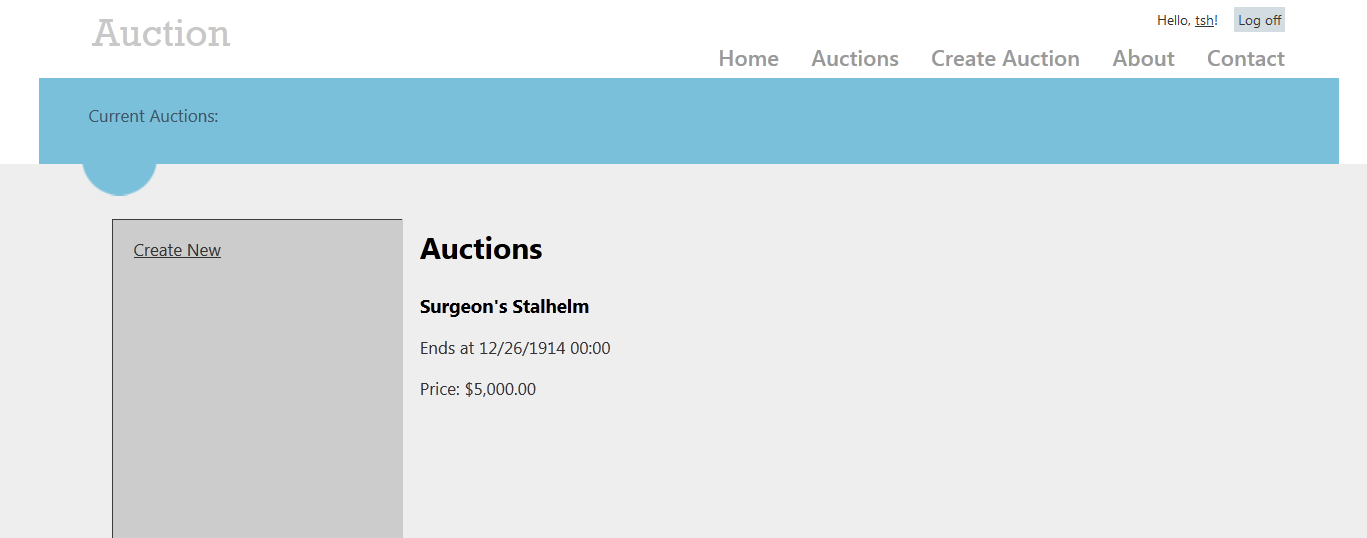


Рисунок 2.4 – Созданные аукционы

На рисунке 2.4 изображены уже созданные аукционы и так же предложение создать новый лот для продажи.

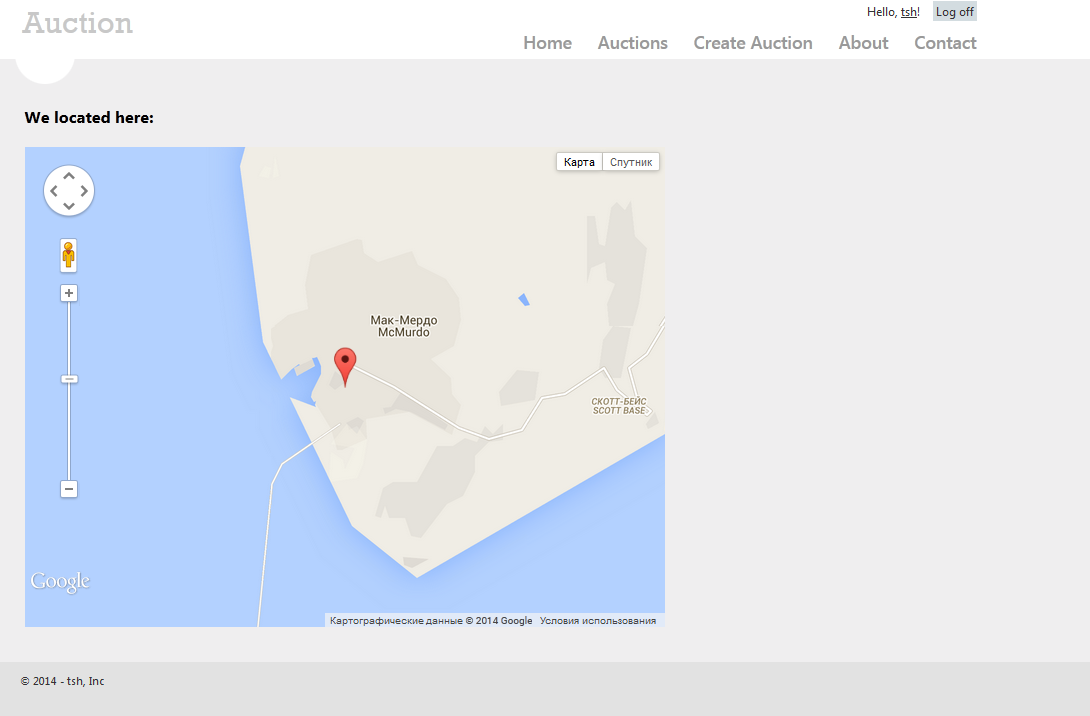


Рисунок 2.5 – Контакты

На рисунке 2.5 изображена страница с контактами о компании, на которой с помощью JavaScript и google map api указано место положение штаб квартиры компании.

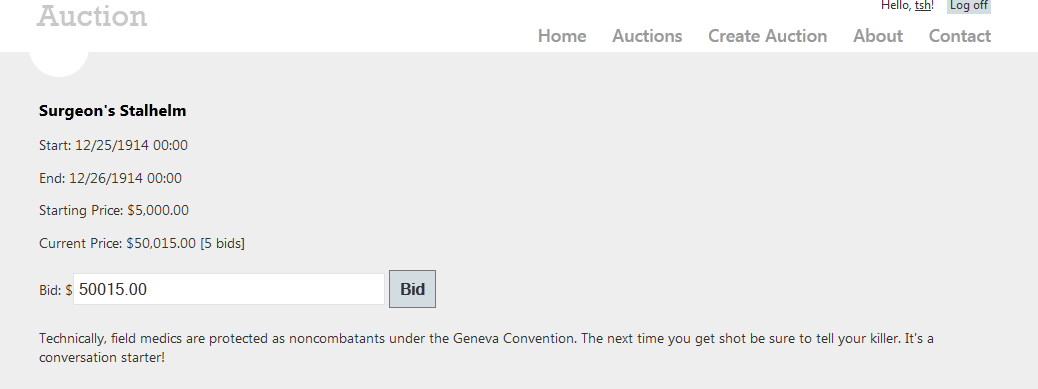


Рисунок 2.6 Ставка на лот

На рисунке 2.6 изображено окно для размещение ставки на выбраном лоте. В поле Current Price указана текущая сумма лота, и количество ставок на нем.В поле Bid пользователь указывает сумму которую он желает разместить, сервер проверяет что бы желаемая сумма была выше текущей. Обновление сумм осуществляется без перегрузки странницы. Если сумма от пользователя принята обновленние странницы будет осуществленно с помощью AJAX и js скрипта.

ВЫВОД

На практике был реализован программный продукт – торговая платформа типа аукцион на языке С# с применением Visual Studio 2012 ASP.NET MVC 4, базой данных, SQLSEVER CE, ORM Entity Framework и веб сервером IIS . Получены навыки проектирования компонентного по, использование шаблонов многоразового использования библиотек .NET. Самостоятельно изучены принципы написания чистого кода. Получены навыки, основ рефакторинга и реинжиниринга кода.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Грищенко В.Н., Лаврищева Е.М. Методы и средства компонентного

программирова­ния // Кибернетика и системный анализ.– 2003.–

2. Сведения о Entity Framework – Режим доступа: http://msdn.microsoft.com/en-

us/data/ee712907

3. Архитектура ASP.NET MVC - Режим доступа: http://www.asp.net/mvc