Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Базы данных»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**«Интернет-аукцион ценных бумаг EasyBay. Бэкенд»**

Студент Гурик Н.В.

Руководитель Алексеев Ю.И.

Минск 2019

**СОДЕРЖАНИЕ**

Введение 4

1 Анализ предметной области 6

1.1 Постановка задачи 6

1.2 Описание предметной области 6

1.3 Терминология 7

1.3.1 Общие термины и определения 7

1.3.2 Технические термины и определения 7

2 Проектирование задачи 9

2.1 Обоснование выбора технологий 9

3 Разработка приложения 14

3.1 Программная часть 14

3.1.1 Серверная сторона 14

3.1.1.1 Разработка моделей главных сущностей 14

3.1.1.2 Разработка интерфейсов приложения 17

3.1.1.3 AuctionContext 19

3.1.1.4 DBController 20

3.1.1.5 BusinessLogic 22

3.1.1.6 Facade 24

3.1.1.7 API 30

4. Перспективы развития проекта 34

Заключение 35

Список использованных источников 36

Приложение А (обязательное) Текст программы главного модуля

серверной части 37

Приложение B (обязательное) Описание БД 51

**ВВЕДЕНИЕ**

Успех любого бизнеса во многом зависит от того, насколько эффективно компания доводит свою информацию до клиентов и партнеров.

Создание интернет-сайта и размещение его в Интернете – один из альтернативных методов позиционирования компании и информирования целевой аудитории. Не менее важным является создание кроссплатформенного приложения.

Именно в Интернете многие ищут подробную, и свежую информацию, на основе которой можно получить представление об интересующей компании, товарах и услугах.

Сейчас трудно представить крупную компанию без интернет-сайта. Наличие у компании хорошего сетевого представительства не только усиливает положительный образ фирмы, но и говорит в пользу надежности и основательности компании, создает определенный имидж и впечатление.

В настоящее время наиболее популярным видом виртуальноый торговли является интернет-аукцион. Интернет-аукцион обычно содержит наглядный и красочный каталог предоставляемых товаров, с их достаточным описание и указанием цены, что позволяет заинтересовать потенциального покупателя, помочь ему сделать выбор и, в итоге, совершить покупку.

Преимущества интернет-аукциона:

- не требует затрат на аренду площади, найма продавцов и другого персонала;

- доступ к виртуальным витринам аукциона может получить любой покупатель, не зависимо от места проживания;

- интернет-аукцион не имеет ограничений на виртуальную площадь;

- владелец интернет-аукциона может сдавать в аренду свои виртуальные торговые площади так же, как и владелец обычного магазина;

- срок и стоимость создания интернет-аукциона несоизмеримо ниже, чем обычного магазина;

- потенциальный покупатель получает исчерпывающую информацию об интересующем его товаре или услуге без помощи посредников (продавца, менеджера);

- интернет-аукцион, работающий в режиме online, предполагает обратную связь с покупателем (форумы, чаты, гостевые книги), что позволяет оперативно ответить на вопросы пользователей, провести маркетинговые исследования, получить отзывы и пожелания;

- можно разместить сколь угодно много товаров или описать любое количество услуг;

- возможность делать покупки, не выходя из дома, делают интернет-аукционы еще более привлекательными для покупателей.

Делая свой выбор в пользу открытия интернет-аукциона, компания переходит на более высокий уровень развития своего бизнеса и делает свой ассортимент более доступным для широкого круга покупателей.

В ходе выполнения данного курсового проекта необходимо разработать сайт интернет-аукцион по продаже различных лотов. В нем должен быть необходимый перечень товара, а также интуитивно понятный пользователю интерфейс.

**1****АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

**1.1 Постановка задачи**

Создать интернет-аукцион EasyBay. Разработать серверное приложение с поддержкой базы данных, от которого сайт и десктопное WPF приложение будут получать данных.

**1.2 Описание предметной области**

Интернет-аукцион - это привычный каждому пользователю интернета web-сайт, представляющий информацию о товарах в удобном структурированном виде.

Интернет-аукционы создаются с применением систем управления содержимым сайтов, оснащенных необходимыми модулями. Крупные интернет-магазины работают на специально разработанных или адаптированных типовых системах управления.

Система управления содержимым сайта интернет-аукциона может быть коробочным товаром, самостоятельно устанавливаемым на хостинг-площадку, может быть частной разработкой web-студии, ей же обслуживаемой, или может быть программным сервисом, предоставляемым с помесячной оплатой.

С технической точки зрения интернет-аукцион - это совокупность web-витрины и торговой системы - фронт-системы и бэк-офиса. Web-витрина предоставляет интерфейс к базе данных продаваемых товаров (в виде каталога, прайс-листа), работает с виртуальной торговой тележкой, оформляет заказы и регистрирует покупателя, предоставляет помощь покупателю в онлайновом режиме, передает информацию в торговую систему и обеспечивает безопасность личной информации покупателя. Далее торговая система осуществляет автоматическую обработку поступающих заказов - резервирует товар на складе, контролирует оплату и доставку товара.

В общем случае основные функции интернет-аукциона - это информационное обслуживание покупателя, обработка заказов, проведение платежей, а также сбор и анализ различной статистической информации. Как было упомянуто выше, программный комплекс управления интернет-аукционом позволяет формировать и интерфейс с покупателем, и функциональные возможности интернет-аукциона, исходя из потребностей компании.

Целью курсового проекта является разработка интернет-аукциона по продаже различных товаров, который позволит оценить их характеристики, а при необходимости и приобрести. При разработке необходимо чтобы каждый покупатель, независимо от возраста, смог без проблем получить необходимую ему информацию о товаре, а также и при необходимости приобрести его.

**1.3 Терминология**

**1.3.1** Общие термины и определения

ОС – операционная система.

Пользователь — лицо, участвующее в функционировании [автоматизированной системы](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) или использующее результаты её функционирования.

«Клиент-сервер» – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически [клиент](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и [сервер](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) – это [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Обычно эти программы расположены на разных [вычислительных машинах](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) и взаимодействуют между собой через [вычислительную сеть](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) посредством [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB), но они могут быть расположены также и на одной машине.

**1.3.2** Технические термины и определения

WPF – Windows Presentation Foundation.

MVC – Model-View-Controller.

*C#* – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для патформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. C# относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к С++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

*.NET Framework* – программая платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы вляется общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования.

*JSON Web Token (JWT)* — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

*Git* – это [распределенная](https://en.wikipedia.org/wiki/Distributed_version_control) система [контроля](https://en.wikipedia.org/wiki/Distributed_version_control) версий для отслеживания изменений в [исходном коде](https://en.wikipedia.org/wiki/Source_code) во время [разработки программного обеспечения](https://en.wikipedia.org/wiki/Software_development). Он предназначен для координации работы [программистов](https://en.wikipedia.org/wiki/Programmer), но может использоваться для отслеживания изменений в любом наборе [файлов](https://en.wikipedia.org/wiki/Computer_file). Его цели включают в себя скорость, целостность данных, и поддержку распределенных, нелинейных рабочих процессов. Как и в большинстве других распределенных систем контроля версий, и в отличие от большинства систем [клиент-сервер](https://en.wikipedia.org/wiki/Client%E2%80%93server), каждый [каталог](https://en.wikipedia.org/wiki/Directory_(computing)) Git на каждом [компьютере](https://en.wikipedia.org/wiki/Node_(networking)) является полноценным [хранилищем](https://en.wikipedia.org/wiki/Repository_(version_control)) с полной историей и возможностями полного отслеживания версий, независимо от доступа к сети или центрального сервера. Git – это бесплатное программное обеспечение с открытым исходным кодом, распространяемое в соответствии с условиями Стандартной общественной лицензии GNU версии 2.

В системах [управления версиями](https://en.wikipedia.org/wiki/Version_control) *коммит* добавляет последние изменения в [часть] исходного кода в [хранилище](https://en.wikipedia.org/wiki/Repository_(version_control)), делая эти изменения частью основной версии хранилища. В отличие от [коммитов в управлении данными](https://en.wikipedia.org/wiki/Commit_(data_management)), коммиты в системах контроля версий хранятся в хранилище неограниченное время. Таким образом, когда другие пользователи делают update или checkout из репозитория, они получат последнюю зафиксированную версию, если только они не укажут, что хотят получить предыдущую версию исходного кода в репозитории. Системы контроля версий позволяют легко вернуться к предыдущим версиям. В этом контексте фиксация в системе управления версиями защищена, поскольку ее легко откатить даже после применения фиксации.

*Ветка в Git'е* — это просто легковесный подвижный указатель на один из этих коммитов. Ветка по умолчанию в Git'е называется master. Когда вы создаёте коммиты на начальном этапе, вам дана ветка master, указывающая на последний сделанный коммит. При каждом новом коммите она сдвигается вперёд автоматически.

*Кросс-платформенность* или межплатформенность – способность программного обеспечения работать с двумя и более аппаратными платформами и (или) операционными системами. Обеспечивается благодаря использованию высокоуровневых языков программирования, сред разработки и выполнения, поддерживающих условную компиляцию, компоновку и выполнение кода для различных платформ. Типичным примером является программное обеспечение, предназначенное для работы в операционных системах Linux и Windows одновременно.

**2 ПРОЕКТИРОВАНИЕ ЗАДАЧИ**

**2.1 Обоснование выбора технологий**

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 7.0, которая вышла в 7 марта 2017 года вместе с Visual Studio 2017.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, если вы знакомы с одним из этих языков, то овладеть C# будет легче.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (WPF, ASP.NET). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

* Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.
* Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Framework поддерживается на большинстве современных ОС Windows (Windows 10/8.1/8/7/Vista). А благодаря проекту Mono можно создавать приложения, которые будут работать и на других ОС семейства Linux, в том числе на мобильных платформах Android и iOS.
* Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.
* Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

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

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

Разработка над платформой началась еще в 2014 году. Тогда платформа условно называлась ASP.NET vNext. В июне 2016 года вышел первый релиз платформы. А в мае 2018 года вышла версия ASP.NET Core 2.1, которая собственно и охвачена в текущем руководстве.

ASP.NET Core теперь полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на GitHub.

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

Хотя ASP.NET Core преимущественно нацелено на использование .NET Core, но фреймворк также может работать и с полной версией фреймворка .NET.

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

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

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

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

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

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

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

Для обработки запросов теперь используется новый конвейер HTTP, который основан на компонентах Katana и спецификации OWIN. А его модульность позволяет легко добавить свои собственные компоненты.

Если суммировать, то можно выделить следующие ключевые отличия ASP.NET Core от предыдущих версий ASP.NET:

* Новый легковесный и модульный конвейер HTTP-запросов;
* Возможность развертывать приложение как на IIS, так и в рамках своего собственного процесса;
* Использование платформы .NET Core и ее функциональности;
* Распространение пакетов платформы через NuGet;
* Интегрированная поддержка для создания и использования пакетов NuGet;
* Единый стек веб-разработки, сочетающий Web UI и Web API;
* Конфигурация для упрощенного использования в облаке;
* Встроенная поддержка для внедрения зависимостей;
* Расширяемость;
* Кроссплатформенность: возможность разработки и развертывания приложений ASP.NET на Windows, Mac и Linux;
* Развитие как open source, открытость к изменениям;

Эти и другие особенности и возможности стали основой для новой модели программирования.

HTML - HyperText Markup Language - стандартный язык программирования, предназначенный для создания гипертекстовых документов в среде WEB.

HTML-документы могут просматриваться различными типами WEB-браузеров. Когда документ создан с использованием HTML, WEB-браузер может интерпретировать HTML для выделения различных элементов документа и первичной их обработки. Использование HTML позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей.

Большинство документов имеют стандартные элементы, такие, как заголовок, параграфы или списки. Используя тэги HTML можно обозначать данные элементы, обеспечивая WEB-браузеры минимальной информацией для отображения данных элементов, сохраняя в целом общую структуру и информационную полноту документов. Все что необходимо, чтобы прочитать HTML-документ - это WEB-браузер, который интерпретирует тэги HTML и воспроизводит на экране документ.

CSS (Cascading Style Sheeets) - каскадные таблицы стилей.

Стиль - набор параметров, задающий внешнее представление объекта.

Преимущества CSS:

- CSS позволяет значительно сократить размер кода и сделать его читабельным;

- CSS позволяет задавать такие параметры, которые нельзя задать только языком HTML. Например, убрать подчеркивание у ссылок;

- CSS позволяет легко изменять внешний вид страниц. Представьте, вы сделали сайт из 50 страниц, на которых все заголовки синего цвета. Через какое-то время, вы захотели поменять синий цвет на зеленый. Вам придется пройтись по всем 50 страницам и поменять цвет в соответствующем атрибуте. С CSS вам придется сделать это лишь один раз, в таблице стилей;

- с CSS связана так называемая блочная верстка сайта.

JavaScript - это язык написания сценариев, разработанный компанией Netscape Communications, для создания клиентских и серверных интернет-приложений. Netscape Navigator создан для интерпретации сценариев JavaScript, внедренных в веб-страницы.

JavaScript - объектно-ориентированный скриптовый язык программирования. Является диалектом языка ECMAScript.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

**3 РАЗРАБОТКА ПРИЛОЖЕНИЯ**

**3.1 Программная часть**

**3.1.1** Клиентская сторона

**3.1.1.1** Разработка моделей главных сущностей

Основным элементом аукциона, несомненно, является лот. Лот – публикация информации о продаже определенного товара, либо группы товаров, размещённая в системе интернет-аукциона.

Исходя из данной определения, важными элементами лота являются фотография и стоимость.

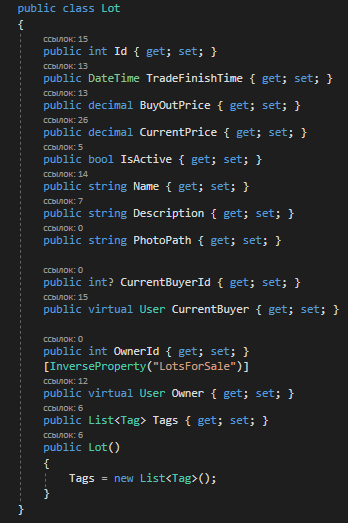


Рисунок 1. Class Lot.

Поясним. Лот обладает следующими полями:

* Id – identification number нужный для поиска в базе данных;
* TradeFinishTime – время окончания аукциона для данного лота;
* BuyOutPrice – цена моментального выкупа;
* IsActive – доступен ли лот для выкупа;
* Name – название лота;
* Description – описание лота;
* PhotoPath – фото лота;
* CurrentBuyerId – id текущего покупателя;
* CurrentBuyer – текущий покупатель (объект);
* OwnerId – id продавца;
* Owner – продавец (объект);
* Tags – тэги, которые соответсвуют данному лоту;

В данном коде можно увидеть элемент Tag, который представляет некоторую логику греппировки лотов.

Его сущность можно отразить всего двумя полями:

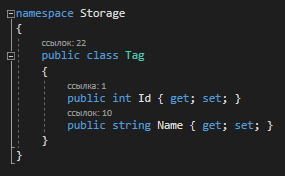


Рисунок 2. Class Tag.

Поле Name отражает название тэга.

И последний, но не менее важный, класс User.

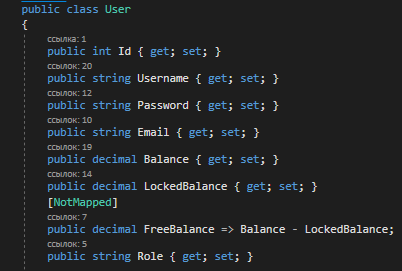


Рисунок 3. Class User.

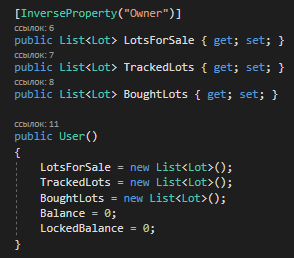


Рисунок 4. Class User (продолжение).

И так же вложенный класс ролей пользователя. Обычный пользователь и админ.

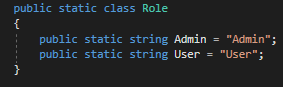


Рисунок 5. Class Role.

Поясним. Пользователь имеет следующие поля:

* Id – identification number нужный для поиска в базе данных;
* Username – логин пользователя;
* Password – пароль нужный для аутентификации пользователя;
* Email – адрес электронной почты пользователя;
* Balance – денежный баланс пользователя;
* LockedBalance – закрытый баланс пользователя (возникает, когда пользователь повышает цену какого-либо лота);
* FreeBalance – свободный баланс поьзователя;
* Role – роль пользователя;
* LotsForSale – продаваемые пользователем лоты;
* TrackedLots – отслеживаемые лоты (возникают, когда пользователь участвует в торгах);
* BoughtLots – купленные пользователем лоты;

**3.1.1.2** Разработка интерфейсов приложения

Интерфейс представляет ссылочный тип, который определяет набор методов и свойств, но не реализует их. Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

В нашем приложении определенно три интерфейса:

* IBusinessLogic – набор некоторых специфических, характерных только для аукциона, функций;

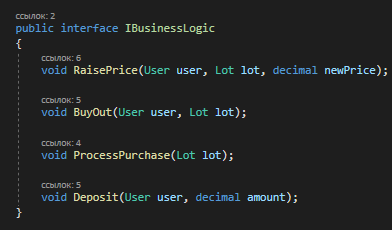


Рисунок 6. Interface IBusinessLogic.

* IDBController – набор функций для работы базой SQLite;

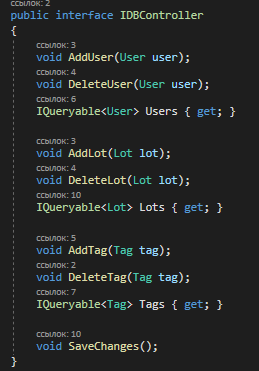
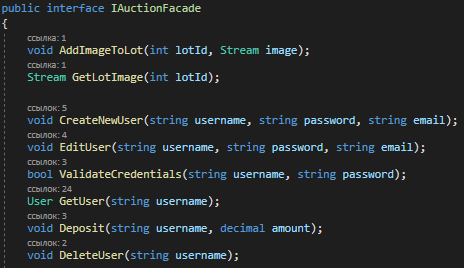
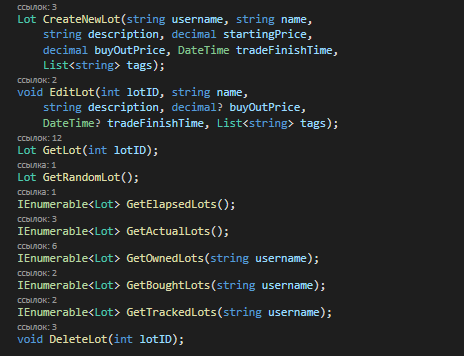


Рисунок 7. Interface IDBController.

* IAuctionFacade – набор функций, который осуществляет как логику приложения, так и работу с базой;





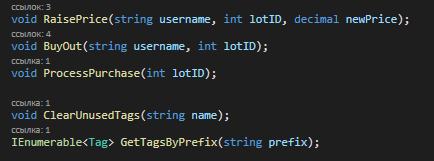


Рисунок 8. Interface IAuctionFacade.

Все пояснения и реализация будут описаны далее.

**3.1.1.3** AuctionContext

ADO.NET Entity Framework – объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft.

Для ее коректной работы создадим класс AuctionContext.

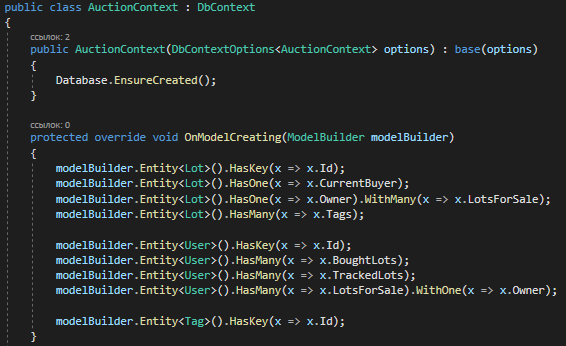


Рисунок 9. Class AuctionContext.

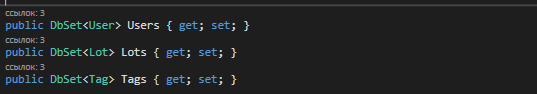


Рисунок 10. Class AuctionContext (продолжение).

Поясним. В базах данных существует понятия PK и FK.

Задание PK:







Рисунок 11. PK.

Задание FK:





Рисунок 12. FK.

И само задание таблиц в EntityFramework:

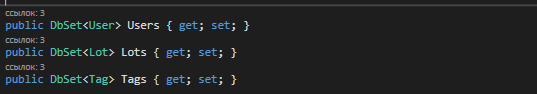


Рисунок 13. Таблицы.

**3.1.1.4** DBController

Этот класс отвечате за все взаимодействие с базой данных. Раннее был указан его интерфейс IDBController.

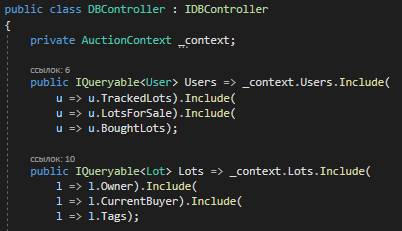




Рисунок 14. DBController.

Переменная \_context отвечает за связь с базой данных. Следующие три метода позволяют получить из базы всех юзеров, лоты и тэги.

Инициализация контекста базы данных:

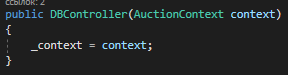


Рисунок 15. DBController конструктор.

Добавление лота, тэга и пользователя в базу:

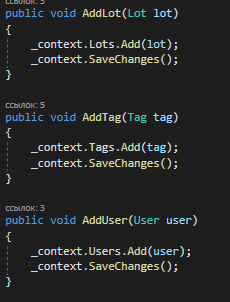


Рисунок 16. DBController методы добавления.

Удаление лота, тэга и пользователя из базы:

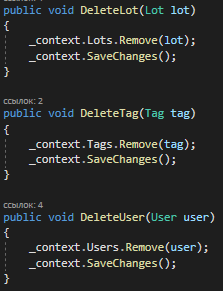


Рисунок 17. DBController методы удаления.

И сохранение данных в базе:

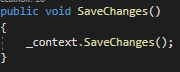


Рисунок 18. DBController метод сохранения.

**3.1.1.5** BusinessLogic

Ранее был описан соответствующий данному классу интерфейс IBusinessLogic.

Класс BucinessLogic отвчает за специфическую логику аукциона:

RaisePrice – поднятие цены и добавление в отслеживаемые лоты, проверка корректности входных данных (происходит, когда пользовать вступает в торги);

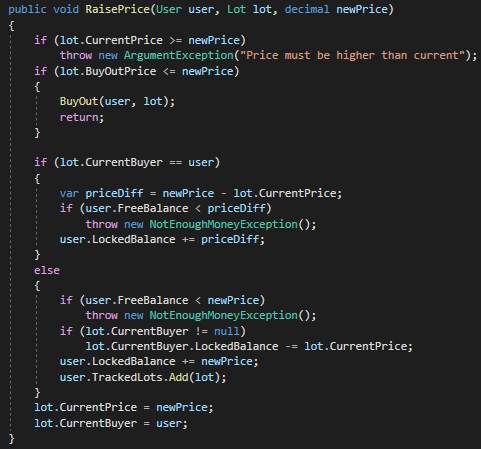


Рисунок 19. RaisePrice.

BuyOut – выкупить лот;

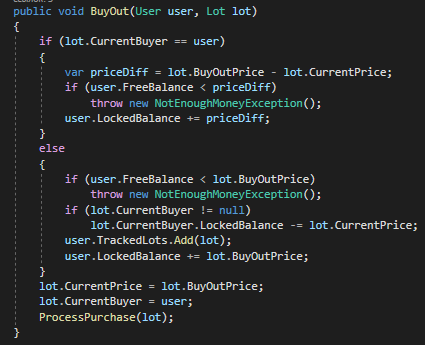


Рисунок 20. BuyOut.

ProcessPurchase – провести покупку, лот добавляется в купленные текущему пользователю, снимает деньги с баланса;

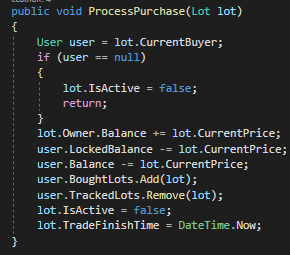


Рисунок 21. ProcessPurchase.

Deposit – добавляет или снимает деньги пользователя;

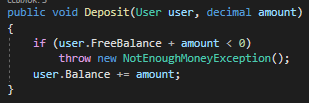


Рисунок 22. Deposit.

**3.1.1.6** Facade

Ранее был описан соответствующий для данного класса интерфес IAuctionFacade.

Этот класс представляет всю работу аукциона. Следующие переменные отвечают за работу базы и логику:



Рисунок 23. Facade.

Метод, позволяющий выкупить лот и зафиксировать изменения в базе:

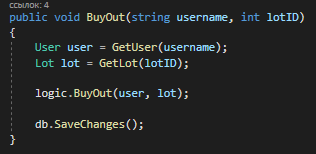


Рисунок 24. Facade BuyOut.

Метод, удаляющий не привязанные тэги и фиксирующий изменения в базе.

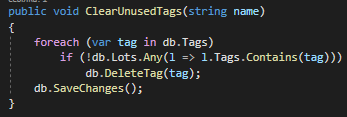
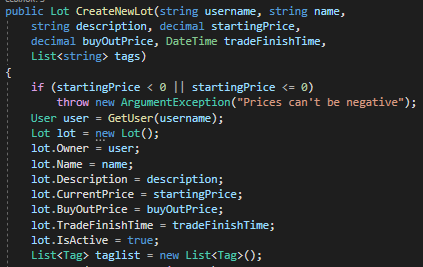


Рисунок 25. Facade ClearTags.

Метод, создающий новый лот в базе:



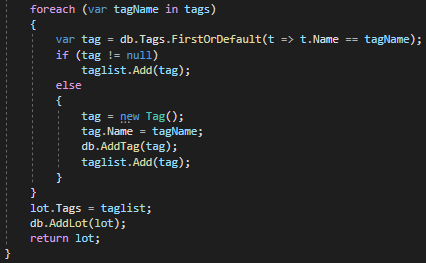


Рисунок 26. Facade CreateNewLot.

Метод, создающий нового пользователя в базе:

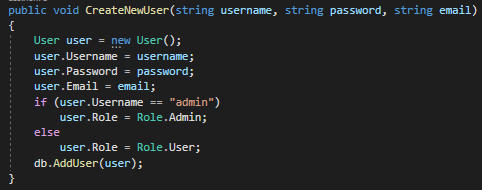


Рисунок 27. Facade CreateNewUser.

Методы, удаляющие лот и пользователя из базы.

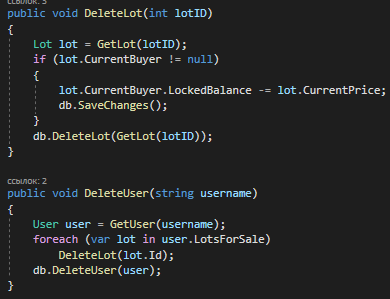


Рисунок 28. Facade Delete.

Метод, добавляющий или удаляющий деньги пользователя:

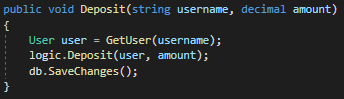


Рисунок 29. Facade Deposit.

Метод, изменяющий данные пользователя:

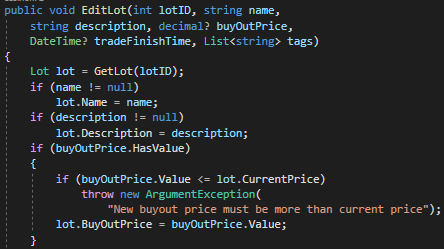


Рисунок 30. Facade EditLot.

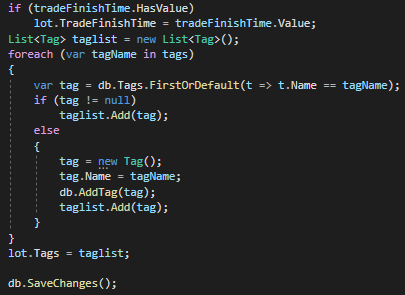


Рисунок 31. Facade EditLot (продолжение).

Метод, изменяющий данные пользователя:

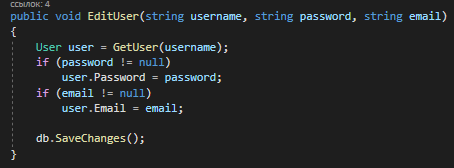
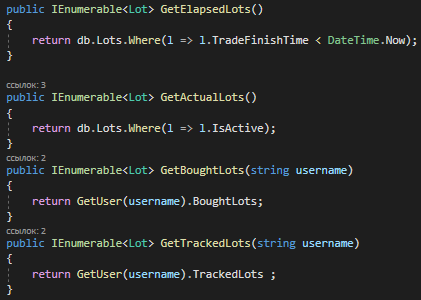


Рисунок 32. Facade EditUser.

Следующая группа метод возвращает актуальные, купленные, отслеживаемые и тд лоты:



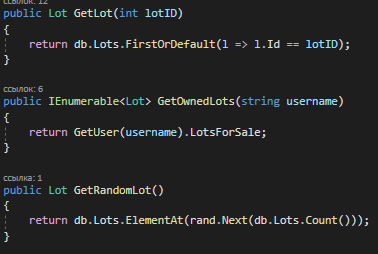
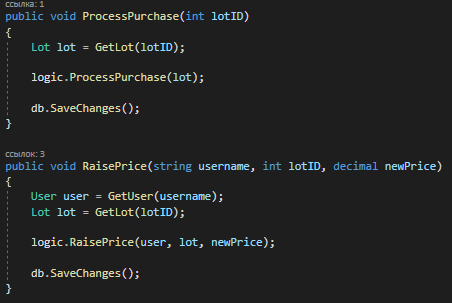


Рисунок 33. Facade Lots.

Следующие методы повторяют логику из BusinessLogic только с фиксацией данных в базе:



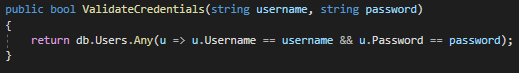


Рисунок 34. Facade Logic.

И методы добавления и получения картинки лота:

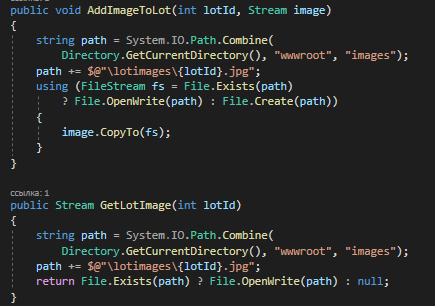


Рисунок 35. Facade Image.

**3.1.1.7** API

За API отвечает группа классов, которая позволяет десктопному WPF приложению связываться с сервером.

Эти классы определяют некоторый набор контроллеров, которые возвращают некоторые данные.

Опишем эти классы.

Класс Auth представляет набор констант для генерации JWT-токенов:

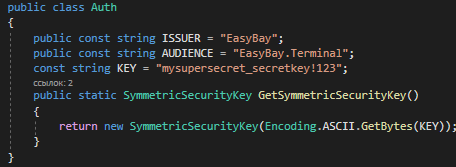
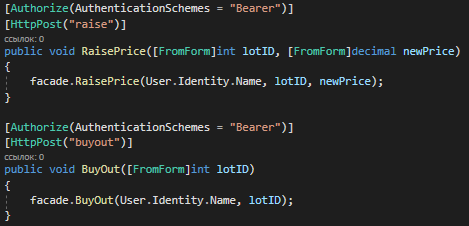
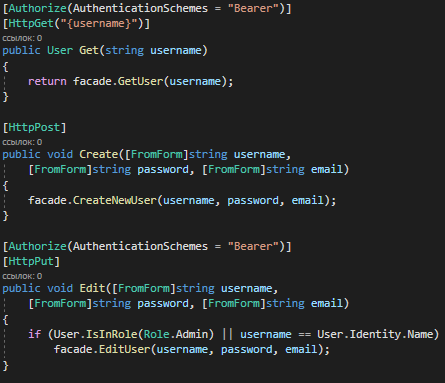
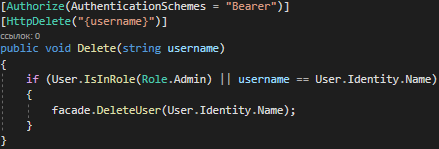


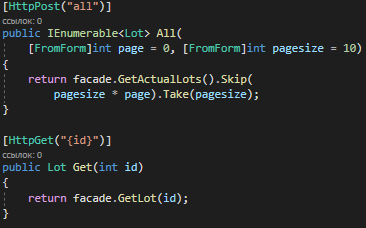
Рисунок 36. Auth.

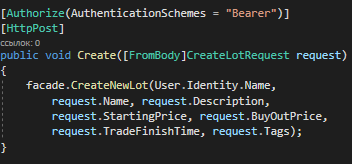
Обёрнутые функции для работы сервера:











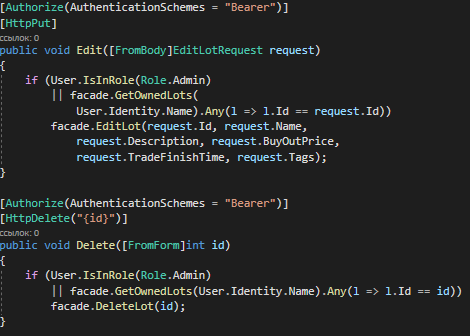


Рисунок 37. API Facade.

Все методы были описаны выше в Facade.

Но объяснения требуют методы GetIdentity и Token:

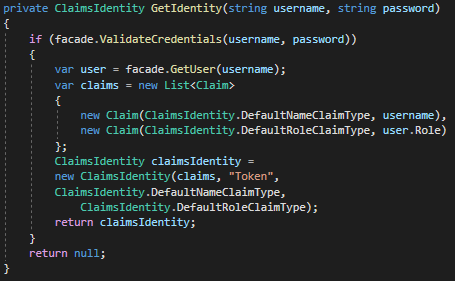


Рисунок 38. ClaimsIdentity.

GetIdentity получает объект claimsIdentity, которые позже позволит нам аутентифицироваться на сервере ASP.

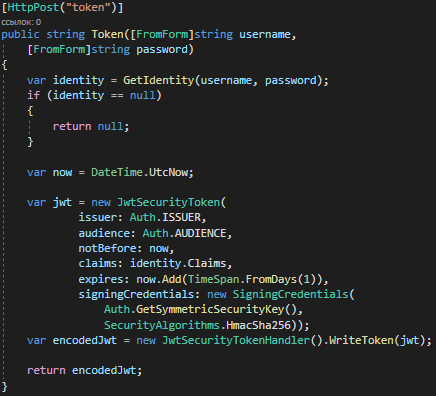


Рисунок 39. Token.

Метод Token так же позволит нам аутентифицироваться на сервере. Данный метод возвращает токен, по которому сервер будет определять пользователя.

**4 ПЕРСПЕКТИВЫ РАЗВИТИЯ ПРОЕКТА**

В дальнейшем планируется улучшение интерфейса интернет-аукциона, увеличение скорости передачи данных на серверную часть за счёт сжатия данных, добавить безопасноть путем использования алгоритмов шифрования, повышение удобства использования. Добавления группировки лотов, возможности оставлять отзывы и комментарии, как лоту, так и продавцу.

**ЗАКЛЮЧЕНИЕ**

На данном этапе развития информационных технологий данная программа может упростить пользователям работу с интернет-аукционом. Однако, она также может повысить интерес к аукциону за счёт возможности легко и удобно покупать разные товары.

При выполнении курсового проекта были пройдены все этапы разработки специализированного прикладного программного обеспечения:

* формализация задачи;
* сбор необходимых исходных данных, используемых в программе;
* ознакомление с литературой;
* более углубленное изучение языка C# и платформы .NET.

В данном курсовом проекте было разработано веб приложение, которое позволяет:

* регистрироваться в системе EasyBay;
* просматривать актуальные лоты в системе;
* повышать ставки на лоты, тем самым добавляя их в отслеживаемые;
* выкупать лоты по указанной продавцом цене;
* добавлять и удалять лоты в систему (возможность администратора).

Для программы был разработан интерфейс, позволяющий неопытному пользователю работать с приложением.

В ходе курсового проекта был более детально изучен языки C#, JS, CSS, HTML, платформа .NET Core, технологии ASP, REST API.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. MSTest Unit [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.automatetheplanet.com/mstest-cheat-sheet/.
2. Создание юнит-тестов. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://metanit.com/sharp/mvc5/18.3.php.
3. ASP.NET Core MVC, Get file from database, and render as image [Электронный ресурс]. – Электронные данные. – Режим доступа: https://stackoverflow.com/questions/37956254/asp-net-core-mvc-get-file-from-database-and-render-as-image.
4. Simple Pagination JS [Электронный ресурс]. – Электронные данные. – Режим доступа: http://flaviusmatis.github.io/simplePagination.js/#page-20.
5. Таймер обратного отсчета на JavaScript [Электронный ресурс]. – Электронные данные. – Режим доступа: https://nikolaus.by/blog/poleznye-fishki-dlya-sayta/taymer-obratnogo-otschyeta-na-javascript/.
6. В ногу со временем: Используем JWT в ASP.NET Core [Электронный ресурс]. – Электронные данные. – Режим доступа: https://habr.com/ru/company/microsoft/blog/311594/.
7. Inspect variables [Electronic resource]. – 2016. – Mode of access: https://docs.microsoft.com/en-us/visualstudio/debugger/autos-and-locals-windows?view=vs-2019.
8. Загрузка файлов на сервер [Electronic resource]. – 2016. – Mode of access: https://metanit.com/sharp/aspnet5/21.3.php.
9. SQLite в WPF. Работа с SQLite [Electronic resource]. – 2015. – Mode of access: https://metanit.com/sharp/aspnet5/1.1.php.
10. ASP.NET Core Введение [Electronic resource]. – 2017. – Mode of access: <https://stackoverflow.com/questions/44403566/add-multiple-extensions-in-one-filetypes-mac-tkinter-filedialog-askopenfilenam>.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программы главного модуля клиентской части**

namespace Storage

{

public class User

{

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public decimal Balance { get; set; }

public decimal LockedBalance { get; set; }

[NotMapped]

public decimal FreeBalance => Balance - LockedBalance;

public string Role { get; set; }

[InverseProperty("Owner")]

public List<Lot> LotsForSale { get; set; }

public List<Lot> TrackedLots { get; set; }

public List<Lot> BoughtLots { get; set; }

public User()

{

LotsForSale = new List<Lot>();

TrackedLots = new List<Lot>();

BoughtLots = new List<Lot>();

Balance = 0;

LockedBalance = 0;

}

}

public static class Role

{

public static string Admin = "Admin";

public static string User = "User";

}

}

namespace Storage

{

public class Tag

{

public int Id { get; set; }

public string Name { get; set; }

}

}

namespace Storage

{

public class Lot

{

public int Id { get; set; }

public DateTime TradeFinishTime { get; set; }

public decimal BuyOutPrice { get; set; }

public decimal CurrentPrice { get; set; }

public bool IsActive { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string PhotoPath { get; set; }

public int? CurrentBuyerId { get; set; }

public virtual User CurrentBuyer { get; set; }

public int OwnerId { get; set; }

[InverseProperty("LotsForSale")]

public virtual User Owner { get; set; }

public List<Tag> Tags { get; set; }

public Lot()

{

Tags = new List<Tag>();

}

}

}

namespace EasyBay.Interfaces

{

public interface IDBController

{

void AddUser(User user);

void DeleteUser(User user);

IQueryable<User> Users { get; }

void AddLot(Lot lot);

void DeleteLot(Lot lot);

IQueryable<Lot> Lots { get; }

void AddTag(Tag tag);

void DeleteTag(Tag tag);

IQueryable<Tag> Tags { get; }

void SaveChanges();

}

}

namespace EasyBay.Interfaces

{

public interface IBusinessLogic

{

void RaisePrice(User user, Lot lot, decimal newPrice);

void BuyOut(User user, Lot lot);

void ProcessPurchase(Lot lot);

void Deposit(User user, decimal amount);

}

}

namespace EasyBay.Interfaces

{

public interface IAuctionFacade

{

void AddImageToLot(int lotId, Stream image);

Stream GetLotImage(int lotId);

void CreateNewUser(string username, string password, string email);

void EditUser(string username, string password, string email);

bool ValidateCredentials(string username, string password);

User GetUser(string username);

void Deposit(string username, decimal amount);

void DeleteUser(string username);

Lot CreateNewLot(string username, string name,

string description, decimal startingPrice,

decimal buyOutPrice, DateTime tradeFinishTime,

List<string> tags);

void EditLot(int lotID, string name,

string description, decimal? buyOutPrice,

DateTime? tradeFinishTime, List<string> tags);

Lot GetLot(int lotID);

Lot GetRandomLot();

IEnumerable<Lot> GetElapsedLots();

IEnumerable<Lot> GetActualLots();

IEnumerable<Lot> GetOwnedLots(string username);

IEnumerable<Lot> GetBoughtLots(string username);

IEnumerable<Lot> GetTrackedLots(string username);

void DeleteLot(int lotID);

void RaisePrice(string username, int lotID, decimal newPrice);

void BuyOut(string username, int lotID);

void ProcessPurchase(int lotID);

void ClearUnusedTags(string name);

IEnumerable<Tag> GetTagsByPrefix(string prefix);

}

}

namespace EasyBay.BusinessLogic

{

public class BusinessLogicInternal : IBusinessLogic

{

public void RaisePrice(User user, Lot lot, decimal newPrice)

{

if (lot.CurrentPrice >= newPrice)

throw new ArgumentException("Price must be higher than current");

if (lot.BuyOutPrice <= newPrice)

{

BuyOut(user, lot);

return;

}

if (lot.CurrentBuyer == user)

{

var priceDiff = newPrice - lot.CurrentPrice;

if (user.FreeBalance < priceDiff)

throw new NotEnoughMoneyException();

user.LockedBalance += priceDiff;

}

else

{

if (user.FreeBalance < newPrice)

throw new NotEnoughMoneyException();

if (lot.CurrentBuyer != null)

lot.CurrentBuyer.LockedBalance -= lot.CurrentPrice;

user.LockedBalance += newPrice;

user.TrackedLots.Add(lot);

}

lot.CurrentPrice = newPrice;

lot.CurrentBuyer = user;

}

public void BuyOut(User user, Lot lot)

{

if (lot.CurrentBuyer == user)

{

var priceDiff = lot.BuyOutPrice - lot.CurrentPrice;

if (user.FreeBalance < priceDiff)

throw new NotEnoughMoneyException();

user.LockedBalance += priceDiff;

}

else

{

if (user.FreeBalance < lot.BuyOutPrice)

throw new NotEnoughMoneyException();

if (lot.CurrentBuyer != null)

lot.CurrentBuyer.LockedBalance -= lot.CurrentPrice;

user.TrackedLots.Add(lot);

user.LockedBalance += lot.BuyOutPrice;

}

lot.CurrentPrice = lot.BuyOutPrice;

lot.CurrentBuyer = user;

ProcessPurchase(lot);

}

public void ProcessPurchase(Lot lot)

{

User user = lot.CurrentBuyer;

if (user == null)

{

lot.IsActive = false;

return;

}

lot.Owner.Balance += lot.CurrentPrice;

user.LockedBalance -= lot.CurrentPrice;

user.Balance -= lot.CurrentPrice;

user.BoughtLots.Add(lot);

user.TrackedLots.Remove(lot);

lot.IsActive = false;

lot.TradeFinishTime = DateTime.Now;

}

public void Deposit(User user, decimal amount)

{

if (user.FreeBalance + amount < 0)

throw new NotEnoughMoneyException();

user.Balance += amount;

}

}

}

namespace EasyBay.BusinessLogic

{

public class AuctionFacade : IAuctionFacade

{

private DBController db;

private BusinessLogicInternal logic;

private Random rand = new Random();

public AuctionFacade(AuctionContext dbContext)

{

db = new DBController(dbContext);

logic = new BusinessLogicInternal();

}

public void BuyOut(string username, int lotID)

{

User user = GetUser(username);

Lot lot = GetLot(lotID);

logic.BuyOut(user, lot);

db.SaveChanges();

}

public void ClearUnusedTags(string name)

{

foreach (var tag in db.Tags)

if (!db.Lots.Any(l => l.Tags.Contains(tag)))

db.DeleteTag(tag);

db.SaveChanges();

}

public Lot CreateNewLot(string username, string name,

string description, decimal startingPrice,

decimal buyOutPrice, DateTime tradeFinishTime,

List<string> tags)

{

if (startingPrice < 0 || startingPrice <= 0)

throw new ArgumentException("Prices can't be negative");

User user = GetUser(username);

Lot lot = new Lot();

lot.Owner = user;

lot.Name = name;

lot.Description = description;

lot.CurrentPrice = startingPrice;

lot.BuyOutPrice = buyOutPrice;

lot.TradeFinishTime = tradeFinishTime;

lot.IsActive = true;

List<Tag> taglist = new List<Tag>();

foreach (var tagName in tags)

{

var tag = db.Tags.FirstOrDefault(t => t.Name == tagName);

if (tag != null)

taglist.Add(tag);

else

{

tag = new Tag();

tag.Name = tagName;

db.AddTag(tag);

taglist.Add(tag);

}

}

lot.Tags = taglist;

db.AddLot(lot);

return lot;

}

public void CreateNewUser(string username, string password, string email)

{

User user = new User();

user.Username = username;

user.Password = password;

user.Email = email;

if (user.Username == "admin")

user.Role = Role.Admin;

else

user.Role = Role.User;

db.AddUser(user);

}

public void DeleteLot(int lotID)

{

Lot lot = GetLot(lotID);

if (lot.CurrentBuyer != null)

{

lot.CurrentBuyer.LockedBalance -= lot.CurrentPrice;

db.SaveChanges();

}

db.DeleteLot(GetLot(lotID));

}

public void DeleteUser(string username)

{

User user = GetUser(username);

foreach (var lot in user.LotsForSale)

DeleteLot(lot.Id);

db.DeleteUser(user);

}

public void Deposit(string username, decimal amount)

{

User user = GetUser(username);

logic.Deposit(user, amount);

db.SaveChanges();

}

public void EditLot(int lotID, string name,

string description, decimal? buyOutPrice,

DateTime? tradeFinishTime, List<string> tags)

{

Lot lot = GetLot(lotID);

if (name != null)

lot.Name = name;

if (description != null)

lot.Description = description;

if (buyOutPrice.HasValue)

{

if (buyOutPrice.Value <= lot.CurrentPrice)

throw new ArgumentException(

"New buyout price must be more than current price");

lot.BuyOutPrice = buyOutPrice.Value;

}

if (tradeFinishTime.HasValue)

lot.TradeFinishTime = tradeFinishTime.Value;

List<Tag> taglist = new List<Tag>();

foreach (var tagName in tags)

{

var tag = db.Tags.FirstOrDefault(t => t.Name == tagName);

if (tag != null)

taglist.Add(tag);

else

{

tag = new Tag();

tag.Name = tagName;

db.AddTag(tag);

taglist.Add(tag);

}

}

lot.Tags = taglist;

db.SaveChanges();

}

public void EditUser(string username, string password, string email)

{

User user = GetUser(username);

if (password != null)

user.Password = password;

if (email != null)

user.Email = email;

db.SaveChanges();

}

public IEnumerable<Lot> GetElapsedLots()

{

return db.Lots.Where(l => l.TradeFinishTime < DateTime.Now);

}

public IEnumerable<Lot> GetActualLots()

{

return db.Lots.Where(l => l.IsActive);

}

public IEnumerable<Lot> GetBoughtLots(string username)

{

return GetUser(username).BoughtLots;

}

public IEnumerable<Lot> GetTrackedLots(string username)

{

return GetUser(username).TrackedLots ;

}

public Lot GetLot(int lotID)

{

return db.Lots.FirstOrDefault(l => l.Id == lotID);

}

public IEnumerable<Lot> GetOwnedLots(string username)

{

return GetUser(username).LotsForSale;

}

public Lot GetRandomLot()

{

return db.Lots.ElementAt(rand.Next(db.Lots.Count()));

}

public IEnumerable<Tag> GetTagsByPrefix(string prefix)

{

return db.Tags.Where(t => t.Name.StartsWith(prefix));

}

public User GetUser(string username)

{

return db.Users.FirstOrDefault(u => u.Username == username);

}

public void ProcessPurchase(int lotID)

{

Lot lot = GetLot(lotID);

logic.ProcessPurchase(lot);

db.SaveChanges();

}

public void RaisePrice(string username, int lotID, decimal newPrice)

{

User user = GetUser(username);

Lot lot = GetLot(lotID);

logic.RaisePrice(user, lot, newPrice);

db.SaveChanges();

}

public bool ValidateCredentials(string username, string password)

{

return db.Users.Any(u => u.Username == username && u.Password == password);

}

public void AddImageToLot(int lotId, Stream image)

{

string path = System.IO.Path.Combine(

Directory.GetCurrentDirectory(), "wwwroot", "images");

path += $@"\lotimages\{lotId}.jpg";

using (FileStream fs = File.Exists(path)

? File.OpenWrite(path) : File.Create(path))

{

image.CopyTo(fs);

}

}

public Stream GetLotImage(int lotId)

{

string path = System.IO.Path.Combine(

Directory.GetCurrentDirectory(), "wwwroot", "images");

path += $@"\lotimages\{lotId}.jpg";

return File.Exists(path) ? File.OpenWrite(path) : null;

}

}

}

namespace EasyBay.DataBase

{

public class AuctionContext : DbContext

{

public AuctionContext(DbContextOptions<AuctionContext> options) : base(options)

{

Database.EnsureCreated();

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Lot>().HasKey(x => x.Id);

modelBuilder.Entity<Lot>().HasOne(x => x.CurrentBuyer);

modelBuilder.Entity<Lot>().HasOne(x => x.Owner).WithMany(x => x.LotsForSale);

modelBuilder.Entity<Lot>().HasMany(x => x.Tags);

modelBuilder.Entity<User>().HasKey(x => x.Id);

modelBuilder.Entity<User>().HasMany(x => x.BoughtLots);

modelBuilder.Entity<User>().HasMany(x => x.TrackedLots);

modelBuilder.Entity<User>().HasMany(x => x.LotsForSale).WithOne(x => x.Owner);

modelBuilder.Entity<Tag>().HasKey(x => x.Id);

}

public DbSet<User> Users { get; set; }

public DbSet<Lot> Lots { get; set; }

public DbSet<Tag> Tags { get; set; }

}

}

namespace EasyBay.DataBase

{

public class DBController : IDBController

{

private AuctionContext \_context;

public IQueryable<User> Users => \_context.Users.Include(

u => u.TrackedLots).Include(

u => u.LotsForSale).Include(

u => u.BoughtLots);

public IQueryable<Lot> Lots => \_context.Lots.Include(

l => l.Owner).Include(

l => l.CurrentBuyer).Include(

l => l.Tags);

public IQueryable<Tag> Tags => \_context.Tags;

public DBController(AuctionContext context)

{

\_context = context;

}

public void AddLot(Lot lot)

{

\_context.Lots.Add(lot);

\_context.SaveChanges();

}

public void AddTag(Tag tag)

{

\_context.Tags.Add(tag);

\_context.SaveChanges();

}

public void AddUser(User user)

{

\_context.Users.Add(user);

\_context.SaveChanges();

}

public void DeleteLot(Lot lot)

{

\_context.Lots.Remove(lot);

\_context.SaveChanges();

}

public void DeleteTag(Tag tag)

{

\_context.Tags.Remove(tag);

\_context.SaveChanges();

}

public void DeleteUser(User user)

{

\_context.Users.Remove(user);

\_context.SaveChanges();

}

public void SaveChanges()

{

\_context.SaveChanges();

}

}

}

namespace EasyBay.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

public class ActionController : ControllerBase

{

private IAuctionFacade facade;

public ActionController(AuctionContext context)

{

facade = new AuctionFacade(context);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpPost("raise")]

public void RaisePrice([FromForm]int lotID, [FromForm]decimal newPrice)

{

facade.RaisePrice(User.Identity.Name, lotID, newPrice);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpPost("buyout")]

public void BuyOut([FromForm]int lotID)

{

facade.BuyOut(User.Identity.Name, lotID);

}

}

}

namespace EasyBay.Controllers.API

{

public class Auth

{

public const string ISSUER = "EasyBay";

public const string AUDIENCE = "EasyBay.Terminal";

const string KEY = "mysupersecret\_secretkey!123";

public static SymmetricSecurityKey GetSymmetricSecurityKey()

{

return new SymmetricSecurityKey(Encoding.ASCII.GetBytes(KEY));

}

}

}

namespace EasyBay.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

public class LotsController : ControllerBase

{

private IAuctionFacade facade;

public LotsController(AuctionContext context)

{

facade = new AuctionFacade(context);

}

[HttpPost("all")]

public IEnumerable<Lot> All(

[FromForm]int page = 0, [FromForm]int pagesize = 10)

{

return facade.GetActualLots().Skip(

pagesize \* page).Take(pagesize);

}

[HttpGet("{id}")]

public Lot Get(int id)

{

return facade.GetLot(id);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpPost]

public void Create([FromBody]CreateLotRequest request)

{

facade.CreateNewLot(User.Identity.Name,

request.Name, request.Description,

request.StartingPrice, request.BuyOutPrice,

request.TradeFinishTime, request.Tags);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpPut]

public void Edit([FromBody]EditLotRequest request)

{

if (User.IsInRole(Role.Admin)

|| facade.GetOwnedLots(

User.Identity.Name).Any(l => l.Id == request.Id))

facade.EditLot(request.Id, request.Name,

request.Description, request.BuyOutPrice,

request.TradeFinishTime, request.Tags);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpDelete("{id}")]

public void Delete([FromForm]int id)

{

if (User.IsInRole(Role.Admin)

|| facade.GetOwnedLots(User.Identity.Name).Any(l => l.Id == id))

facade.DeleteLot(id);

}

}

}

namespace EasyBay.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

public class UserController : ControllerBase

{

private IAuctionFacade facade;

public UserController(AuctionContext context)

{

facade = new AuctionFacade(context);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpGet("{username}")]

public User Get(string username)

{

return facade.GetUser(username);

}

[HttpPost]

public void Create([FromForm]string username,

[FromForm]string password, [FromForm]string email)

{

facade.CreateNewUser(username, password, email);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpPut]

public void Edit([FromForm]string username,

[FromForm]string password, [FromForm]string email)

{

if (User.IsInRole(Role.Admin) || username == User.Identity.Name)

facade.EditUser(username, password, email);

}

[Authorize(AuthenticationSchemes = "Bearer")]

[HttpDelete("{username}")]

public void Delete(string username)

{

if (User.IsInRole(Role.Admin) || username == User.Identity.Name)

{

facade.DeleteUser(User.Identity.Name);

}

}

[HttpPost("token")]

public string Token([FromForm]string username,

[FromForm]string password)

{

var identity = GetIdentity(username, password);

if (identity == null)

{

return null;

}

var now = DateTime.UtcNow;

var jwt = new JwtSecurityToken(

issuer: Auth.ISSUER,

audience: Auth.AUDIENCE,

notBefore: now,

claims: identity.Claims,

expires: now.Add(TimeSpan.FromDays(1)),

signingCredentials: new SigningCredentials(

Auth.GetSymmetricSecurityKey(),

SecurityAlgorithms.HmacSha256));

var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);

return encodedJwt;

}

private ClaimsIdentity GetIdentity(string username, string password)

{

if (facade.ValidateCredentials(username, password))

{

var user = facade.GetUser(username);

var claims = new List<Claim>

{

new Claim(ClaimsIdentity.DefaultNameClaimType, username),

new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role)

};

ClaimsIdentity claimsIdentity =

new ClaimsIdentity(claims, "Token",

ClaimsIdentity.DefaultNameClaimType,

ClaimsIdentity.DefaultRoleClaimType);

return claimsIdentity;

}

return null;

}

}

namespace EasyBay.Terminal.API

{

class EasyBayClient

{

private HttpClient client;

private const string URI = @"http://localhost:51868/api/";

public EasyBayClient(string token)

{

client = new HttpClient();

client.BaseAddress = new Uri(URI);

client.DefaultRequestHeaders.Authorization

= new AuthenticationHeaderValue("Bearer", token);

}

#region User

public static async Task<bool> CreateUser(

string username, string password, string email)

{

var client = new HttpClient();

client.BaseAddress = new Uri(URI);

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("username", username),

new KeyValuePair<string, string>("password", password),

new KeyValuePair<string, string>("email", email)

});

var response = await client.PostAsync("user", content);

return response.IsSuccessStatusCode;

}

public static async Task<string> GetToken(

string username, string password)

{

var client = new HttpClient();

client.BaseAddress = new Uri(URI);

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("username", username),

new KeyValuePair<string, string>("password", password)

});

var response = await client.PostAsync("user/token", content);

return response.IsSuccessStatusCode

? await response.Content.ReadAsStringAsync() : null;

}

public async Task<User> GetUser(string username)

{

var response = await client.GetAsync($"user/{username}");

return response.IsSuccessStatusCode

? JsonConvert.DeserializeObject<User>(

await response.Content.ReadAsStringAsync()) : null;

}

public async Task<bool> EditUser(

string username, string password, string email)

{

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("username", username),

new KeyValuePair<string, string>("password", password),

new KeyValuePair<string, string>("email", email)

});

var response = await client.PutAsync("user", content);

return response.IsSuccessStatusCode;

}

public async Task<bool> DeleteUser(string username)

{

var response = await client.DeleteAsync($"user/{username}");

return response.IsSuccessStatusCode;

}

#endregion

#region Lots

public async Task<bool> CreateLot(CreateLotRequest request)

{

var content = new StringContent(JsonConvert.SerializeObject(request));

var response = await client.PostAsync("user", content);

return response.IsSuccessStatusCode;

}

public async Task<List<Lot>> GetLotPage(int page, int pagesize)

{

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("page", page.ToString()),

new KeyValuePair<string, string>("pagesize", pagesize.ToString())

});

var response = await client.PostAsync("lots/all", content);

return response.IsSuccessStatusCode

? JsonConvert.DeserializeObject<List<Lot>>(

await response.Content.ReadAsStringAsync()) : null;

}

public async Task<Lot> GetLot(int lotId)

{

var response = await client.GetAsync($"lots/{lotId}");

return response.IsSuccessStatusCode

? JsonConvert.DeserializeObject<Lot>(

await response.Content.ReadAsStringAsync()) : null;

}

public async Task<bool> EditLot(EditLotRequest request)

{

var content = new StringContent(JsonConvert.SerializeObject(request));

var response = await client.PutAsync("lots", content);

return response.IsSuccessStatusCode;

}

public async Task<bool> DeleteLot(int lotId)

{

var response = await client.DeleteAsync($"lots/{lotId}");

return response.IsSuccessStatusCode;

}

#endregion

#region Actions

public async Task<bool> RaisePrice(int lotId, decimal newPrice)

{

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("lotId", lotId.ToString()),

new KeyValuePair<string, string>("newPrice", newPrice.ToString())

});

var response = await client.PostAsync("action/raise", content);

return response.IsSuccessStatusCode;

}

public async Task<bool> BuyOut(int lotId)

{

var content = new FormUrlEncodedContent(

new List<KeyValuePair<string, string>>() {

new KeyValuePair<string, string>("lotId", lotId.ToString())

});

var response = await client.PostAsync("action/buyout", content);

return response.IsSuccessStatusCode;

}

**ПРИЛОЖЕНИЕ B**

**(обязательное)**

**Описание БД**

