ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Доцент департамента
программной инженерии
факультета компьютерных наук
кандидат экономических наук

_____/ Песоцкая Е.Ю.
«__» _____ 2015 г.

УТВЕРЖДАЮ
Академически руководитель
образовательной программы
«Программная инженерия»

_____/ Шилов В.В.
«__» _____ 2015 г.

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

WEB-ПРИЛОЖЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ СТУДЕНТОВ

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.509000-01 81 01-1-ЛУ

Исполнитель
Студент группы 105 ПИ НИУ ВШЭ
_____/ Лазаренко А.В.
«__» _____ 2015 г.

2015

УТВЕРЖДЕНО
RU.17701729.509000-01 81 01-1 ЛУ

WEB-ПРИЛОЖЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ СТУДЕНТОВ

Пояснительная записка

RU.17701729.509000-01 81 01-1

Листов 41

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

2015

АННОТАЦИЯ

В настоящем документе представлена пояснительная записка к Web-приложению для взаимодействия студентов. Настоящий документ содержит следующие разделы: «Введение», «Назначение и область применения программы», «Технические характеристики», «Ожидаемые технико-экономические показатели», «Источники, использованные при разработке», «Схема сущностей базы данных (приложение)», «Список терминов (приложение)», «Таблица преимуществ (приложение)».

В разделе «Введение» представлены сведения о наименовании программы и документа, на основании которого ведется разработка.

В разделе «Назначение и область применения программы» содержится краткое описание назначения и области применения разрабатываемой программы.

В разделе «Технические характеристики» представлена информация о постановке задачи и особенностях функционирования программы. Перед прочтением раздела «Архитектура базы данных» настоятельно рекомендуется ознакомиться с разделом «Схема сущностей базы данных». Более детальное описание API приложения доступно в документе «Руководство программиста».

В разделе «Ожидаемые технико-экономические показатели» описаны предполагаемая потребность и экономические преимущества разработки по сравнению с аналогами. Перед прочтением данного раздела настоятельно рекомендуется ознакомиться с разделом «Таблица преимуществ».

В разделе «Источники, использованные при разработке» приведен список всех информационных источников, информация из которых была использована для разработки.

В разделе «Схема сущности базы данных» представлена схема сущности базы данных, на схеме приведены все связи сущностей и поля сущностей.

В разделе «Список терминов» описаны все термины, которые по мнению автора могут ввести в заблуждение читающего.

Перед ознакомлением рекомендуется прочесть список терминов.

Данное приложение было разработано с использованием следующих языков программирования и языка разметки: C#, JavaScript, CSS, HTML. В разработке интерфейса широко использовались технологии jQuery и Bootstrap. Для реализации real-time взаимодействий использовалась технология SignalR от Microsoft, а так же для реализации базы данных использовалась технология Entity Framework, подход Code-First. Для развертывания приложения, в качестве хостинг-платформы использовался Microsoft Azure, но по экономическим и практическим соображениям эта платформа перестала использоваться через два месяца. В настоящее время приложение развернуто на Amazon EC2, используется виртуальная машина с Windows Server 2014 + SQL. При разработке приложения использовались следующие IDE: Visual Studio 2013 Ultimate (Microsoft), WebStorm (JetBrains). Так как приложение написано с учетом того, что оно развернуто и имеет собственный домен, то некоторые функции, например, отображения фотографий не будут корректно работать на локальной версии. Для тестирования лучше использовать развернутую версию приложения, доступную по адресу http://cocktion.com. Из-за использования виртуальной машины с ограниченными вычислительными ресурсами (ограничения налагаемые бесплатностью версии) развернутое приложение может откликаться немножко медленнее, чем среднестатистический сайт. Весь код приложения доступен в private репозитории на GitHub. Если необходимо получить доступ к коду на GitHub - писать по адресу: lazarenko.ale@gmail.com.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1 Наименование программы

Наименование программы - «Web-приложение для взаимодействия студентов»
Краткое наименование - «W-app»

1.2 Документы, на основании которых ведется разработка

Разработка ведется на основании приказа Национального исследовательского университета «Высшая школа экономики» No 6.18.1-02/1912-10 от 19.12.2014

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ

2.1. Назначение программы

2.1.1. Функциональное назначение

Функциональным назначением приложения является предоставление пользователю возможности многоцелевого взаимодействия друг с другом через организацию обмена вещами в реальном времени и обсуждения друг с другом своих вещей (в форме аукциона), возможность оценивать и обсуждать разных пользователей.

2.1.2. Эксплуатационное назначение

Программа помогает пользователю организовать наиболее эффективно взаимодействие с другими пользователями для обмена вещами.

2.2. Краткая характеристика области применения

Программа будет применяться студентами ВУЗов РФ в повседневной жизни, в свободное время.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку программы.

Разрабатываемая программа должна:
1) Иметь систему регистрации и авторизации пользователей
2) Иметь WEB интерфейс

3) Позволять пользователям создавать аукционы для обмена вещами и добавлять свои вещи на существующие аукционы

4) Записывать аукционы и вещи в базу данных, а так же считывать их от туда

5) Иметь механизм передачи личных сообщений от пользователя к пользователю

3.2. Описание функционирования программы

3.2.1. Архитектура решения

В данном разделе будет дано описание архитектуры решения.

При взаимодействии с приложением возможно два сценария: пользователь использует браузер или стороннее приложение (используется API приложения).

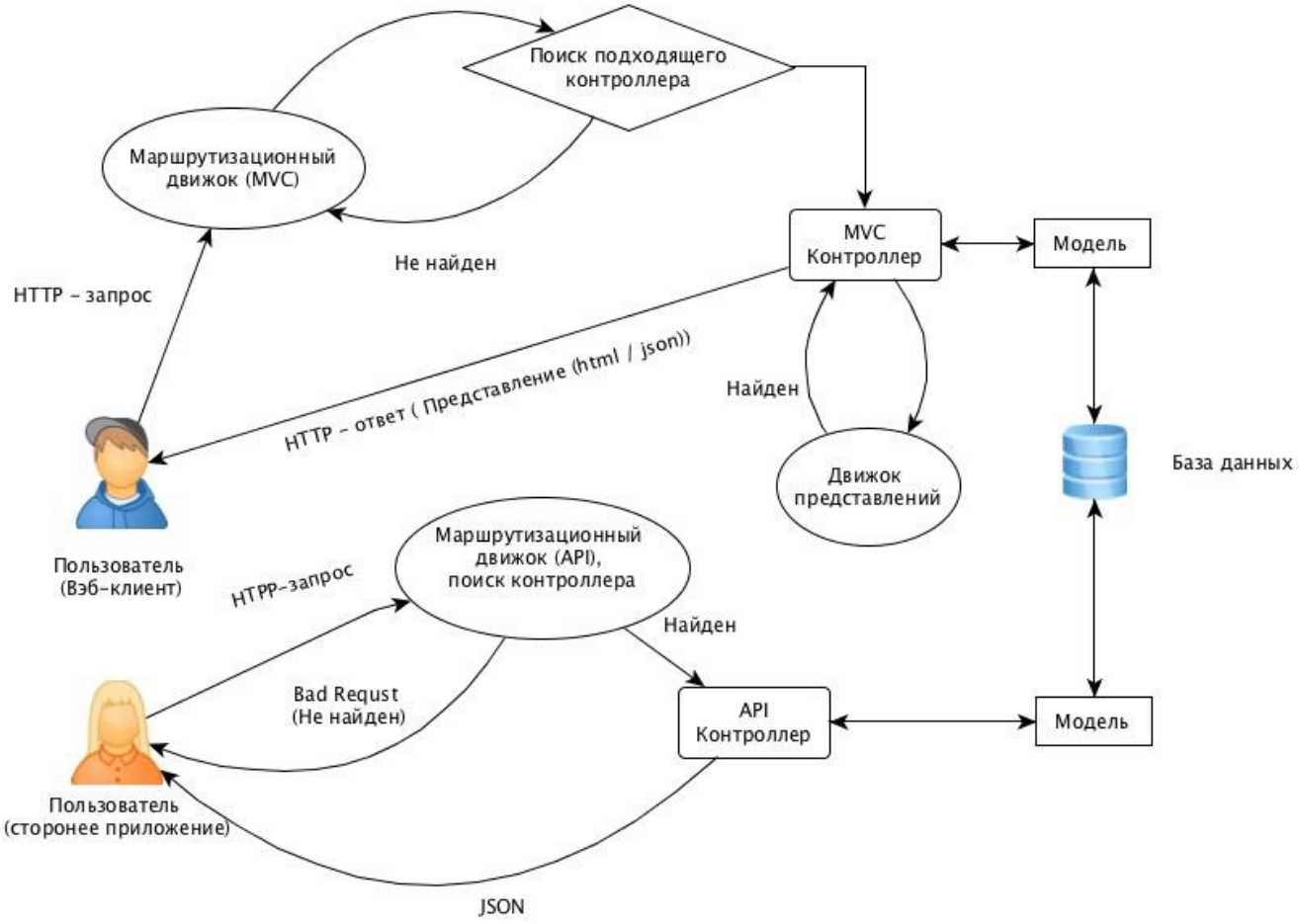


Рисунок 1.

На рисунке 1 представлена общая схема работы приложения. Рассмотрим более подробно первый сценарий взаимодействия с системой. Реализуются следующие шаги:

1) Запрос проходит через стек ASP.NET и передается в маршрутизационный движок.
2) Используя настройки маршрутизации, маршрутизационный движок ищет подходящий контроллер. Если контроллер найден, то он вызывается, а в противном случае возвращается

«controller not found» и пользователь получает сообщение о том, что запрашиваемая страница не найдена, цикл прекращается.

3) Если необходимо, контроллер взаимодействует со слоем «Модель», а «Модель» при необходимости взаимодействует с базой данных, записывая в нее значения или получая. После этого, управление возвращается обратно к контроллеру.
4) Контроллер делает запрос для представления с (или без) данных из модели. Ищется необходимое представление, а затем происходит его рендеризация.
5) Затем управление передается к Движку представления, который возвращает результат контроллеру.
6) Контроллер посылает результат обратно, как часть HTTP ответа.

Рассмотрим второй вариант сценария: пользователь использует для взаимодействия с платформой стороннее приложение, которое в свою очередь использует API.

1) Посылается HTTP запрос по конкретному адресу, управление переходит в маршрутизационный движок (API), производится поиск нужного контроллера, если контроллер не найден - посылается обратно bad request.
2) Управление переходит к API контроллеру, который передает управление модели, а модель взаимодействует с базой данных, записывая в нее данные или считывая. Управление возвращается (вместе с данными или результатом выполнения операции) обратно к контроллеру.
3) Контроллер выполняет JSON сериализацию объекта и посылает его обратно, вызывающему приложению.

3.2.2. Архитектура базы данных

Приложение использует реляционную базу данных. Взаимодействие с базой данных производится с использованием технологии Entity Framework Code First.
Далее будут приведены описания сущностей, общую же диаграмму базы смотреть в приложении 1. Подробное описание полей классов сущностей смотреть в приложении 1.

3.2.2.1.Сущность AspNetRole

Данная сущность необходима для реализации механизма ролей в приложении. Автогенерированная. При разработке не использовалась.

3.2.2.2. Сущность AspNetUser

Данная сущность моделирует пользователя платформы.

3.2.2.3. Сущность AspNetUserClaim

Автосгенерированная сущность, при разработке не использовалась.

3.2.2.4. Сущность AspNetUserLogin

3.2.2.5. Сущность C_MigrationHistory

Автосгенерированная сущность, используется для отслеживания миграций.

3.2.2.6. Сущность Auction

Моделирует аукцион.

3.2.2.7. Сущность BidCluster

Моделирует кластер ставок на аукционе. Используется для того, чтобы связывать ставки каждого пользователя друг с другов в случае, если он решил поставить что-нибудь еще на одном и том же аукционе.

3.2.2.8. Сущность Product

Моделирует вещь, выставленную на аукцион.

3.2.2.9. Сущность Interest

Моделирует интерес пользователя.

3.2.2.10. Сущность Picture

Моделирует фотографию, которой обладает товар или человек.

3.2.2.11. Сущность PrivateMessage

Моделирует личное сообщение пользователя в чате.

3.2.2.12. Сущность UsersFeedback

Моделирует отзыв о пользователе, оставленный другим пользователем.

3.2.2.13. Сущность ForumPost

Моделирует сообщение на форуме

3.2.2.14. Сущность House

Моделирует локацию университета.

3.2.2.15. Сущность HouseHolder

Моделирует университет.

3.2.2.16. Сущность ToteBoard

Моделирует тотализатор

3.2.2.17. Сущность ToteEntity

Моделирует сущность ставки на тотализаторе.

3.2.2.18. Сущность ToteResult

Моделирует результат проведения тотализатора.

3.2.2.19. Сущность Device

Моделирует мобильное устройство пользователя. Необходимо для push-уведомлений, посылаемых с сервера.

3.2.3. Описание функциональных элементов приложения

3.2.3.1. Описание MVC контроллеров.

3.2.3.1.1. Описание контроллера AuctionController

Метод Index() отвечает за вывод главной страницы с аукционами. Происходит подключение к базе данных, из нее выбираются все аукционы, которые активны в данный момент (активным считается такой аукцион, у которого время окончания больше текущего и поле isActive имеет значение true). Входных параметров не имеет, возвращает представление Index, с коллекцией сущностей типа Auction (см. рис. 2).

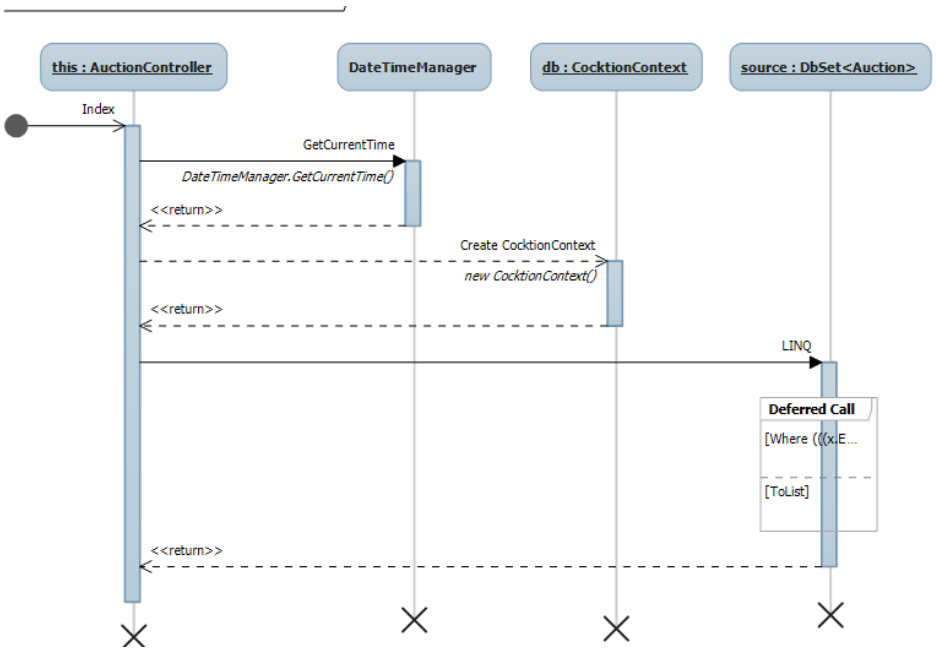


Рисунок 2.

Метод Create() отвечает за вывод страницы, на которой пользователь создает аукцион. Возвращает представление Create.

Метод ShowMyAuctions() отвечает за вывод всех аукционов, которые создал пользователь. Выбирает из базы данных все такие аукционы, формирует коллекцию типа Auction и возвращает представление MyAuctions с этой коллекцией (см. рис. 3)

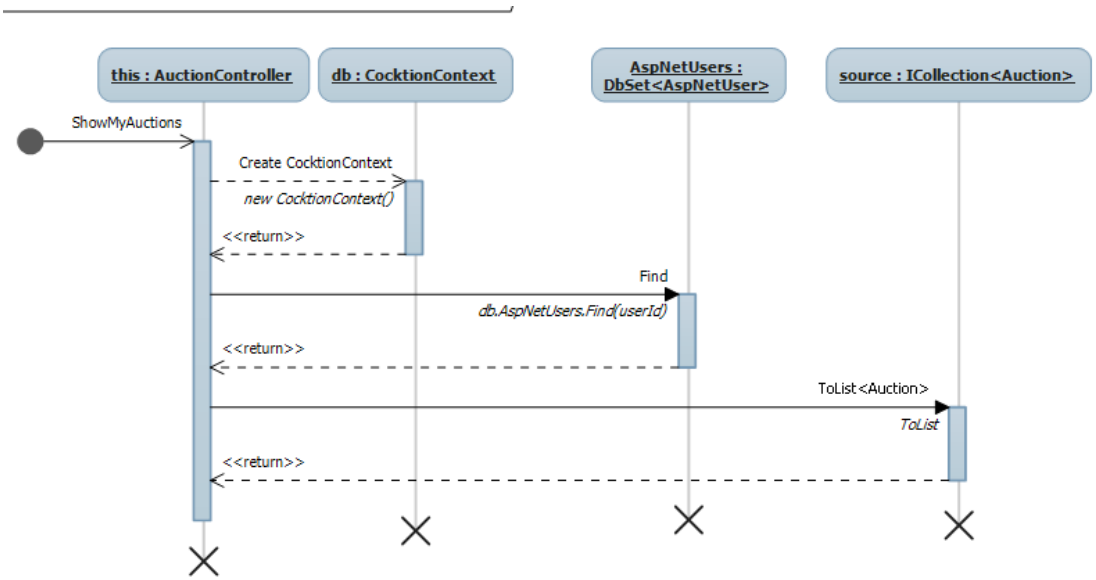


Рисунок 3

Метод CurrentAuction(int id) отвечает за вывод какого-то конкретного аукциона. Во входном параметре используется номер аукциона, который надо посмотреть в базе данных. Возвращается представление CurrentAuction, если аукцион с таким номером существует или PageNotFound, в противном случае (см. рис. 4).

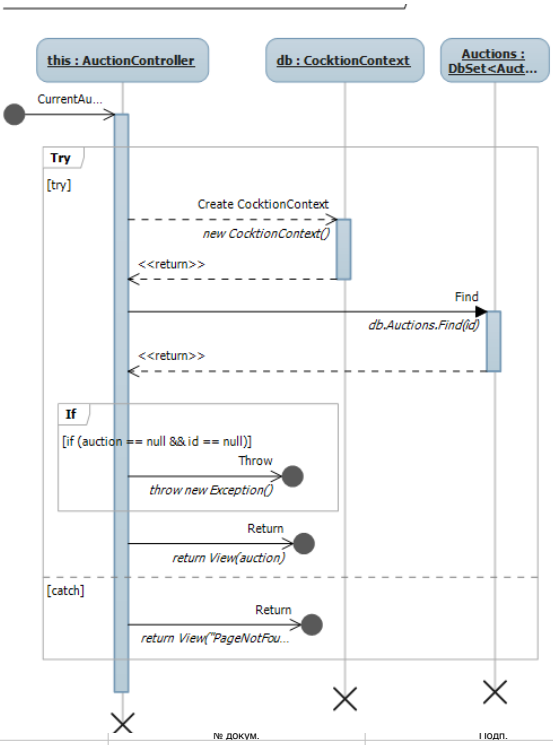


Рисунок 4.

3.2.3.1.2. Описание контроллера AuctionRealTimeController

Метод AddToteRate() отвечает за ставку на тотализаторе во внутренней валюте. С клиента посылается POST-запрос, который перехватывается данным методом, на вход в запросе к нему посылаются поля с информацией о номере аукциона, номере товара и количестве яиц, которое необходимо поставить. Возвращается JSON со статусом добавления и количеством оставшихся у пользователя яиц (см. рис. 5).

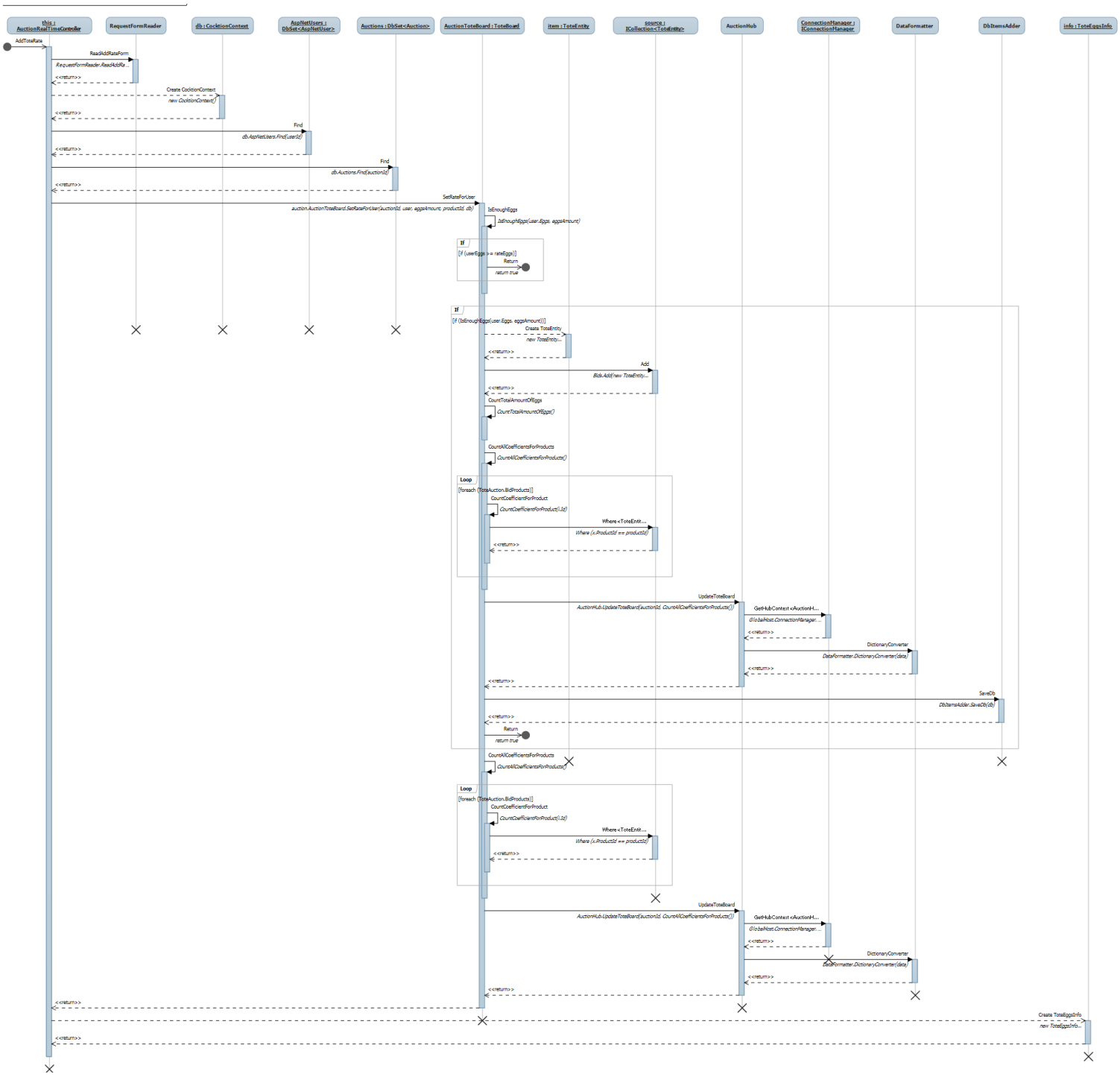


Рисунок 5.

Метод AddLider() отвечает за добавление лидера аукциона. С клиента поступает запрос, в котором посылается номер аукциона и номер вещи, которую необходимо выбрать лидером. Затем, если аукцион найден, у него устанавливается лидер и индикатор выбранности лидера принимает значение true. Данные сохраняются в базу, вызывается метод AuctionHub.SetLider, который рассказывает остальным открытым клиентам, что лидер выбран. Возвращается JSON, в котором содержится статус добавления и название товара - лидера (см. рис. 6).

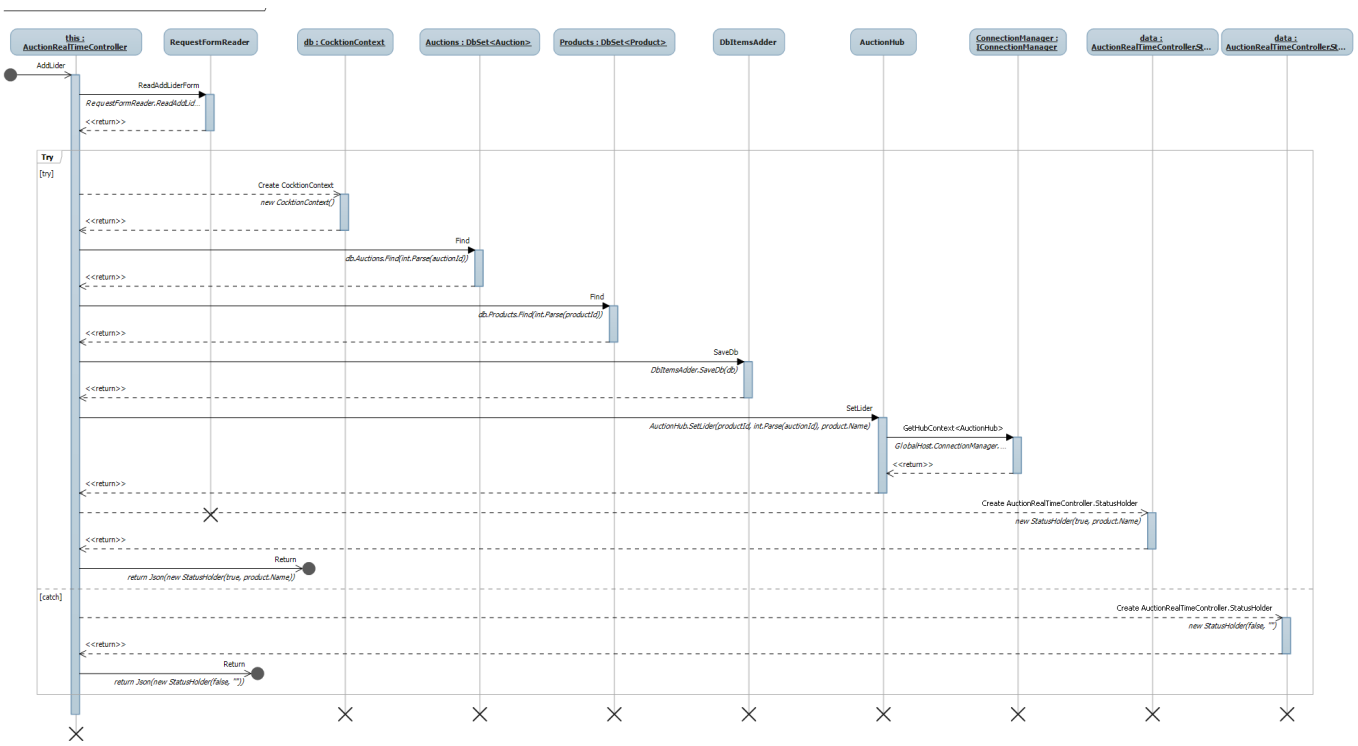


Рисунок 6.

Метод CheckIfUserBidded() служит для првоерки того, ставил ли пользователь ставку на конкретном аукционе. В запросе посылается номер аукциона в базе данных, затем происходит проверка ставил ли пользователь и, затем, возвращается JSON с информацией о том, ставил или нет (см. рис. 7).

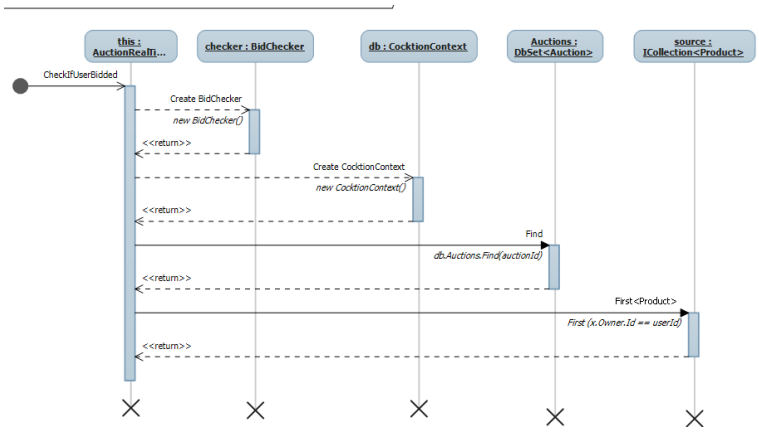


Рисунок 7.

Метод SendInfoAboutProduct() отвечает за посылку информации о товаре на клиент. Из запроса достается номер вещи, затем в базе данных находится вещь с нужным номером и собирается информация о ее названии, описании, названии файла с фоткой и категории. Обратно посылается JSON с этой информацией (см. рис. 8).

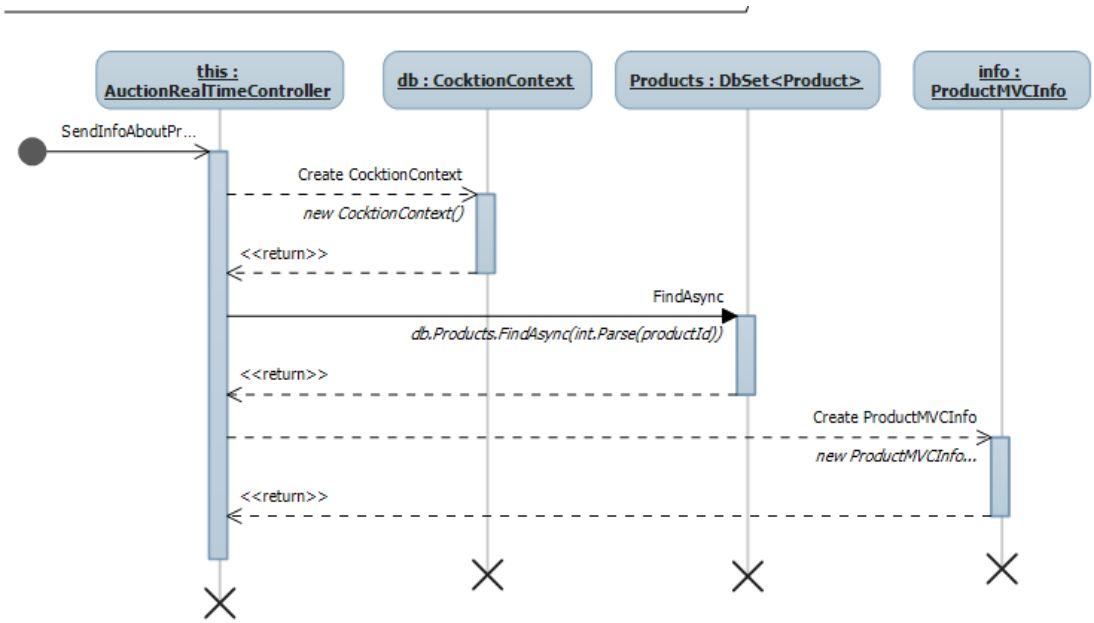


Рисунок 8.

Метод SendAuctionResults() отвечает за рассылку информации об окончании аукциона. Из запроса достается айдишник аукциона, который закончился. В зависимости от типа пользователя возвращается JSON с результатами аукциона (см. рис. 9).

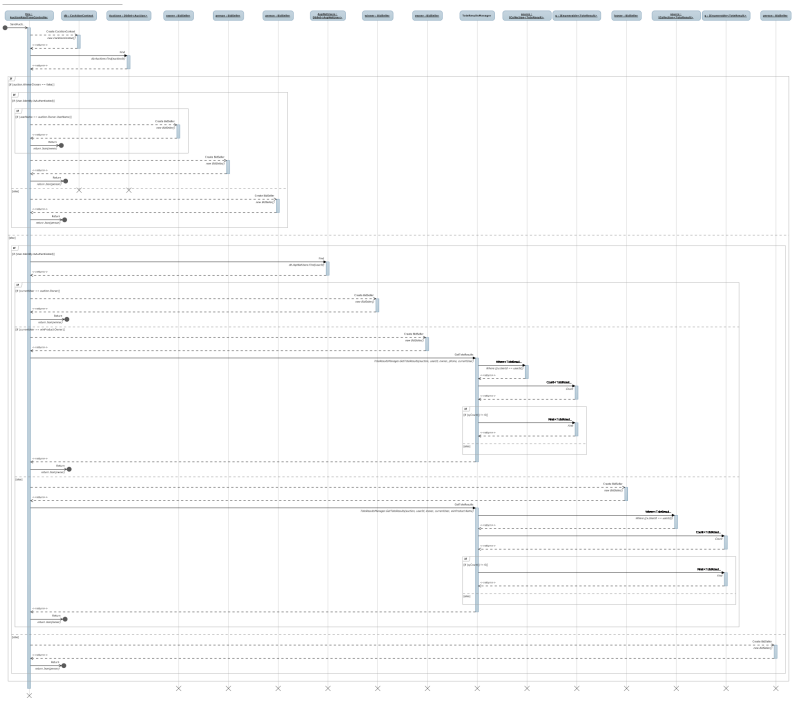


Рисунок 9.

3.2.3.1.3. Описание контроллера AuctionShowerController

Метод ShowOldAuctions(int id) служит для вывода всех закончившихся аукционов на конкретной локации. Из базы данных извлекаются все законченные аукционы, возвращается представление ShowOldAuctions с коллекцией завершившихся аукционов (см. рис.10).

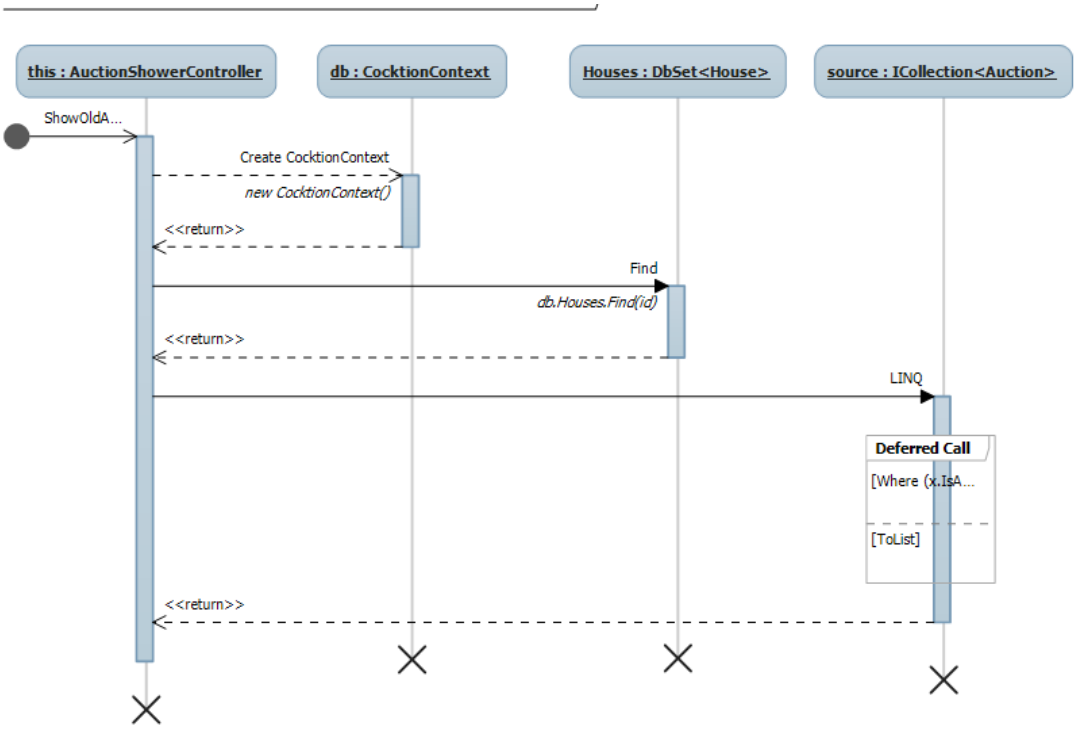


Рисунок 10.

Метод ShowActiveAuctions(int id) служит для вывода всех активных аукционов на конкретной локации. Из базы данных извлекаются все активные аукционы, возвращается представление ShowActiveAuctions с коллекцией активных аукционов (см. рис. 11)

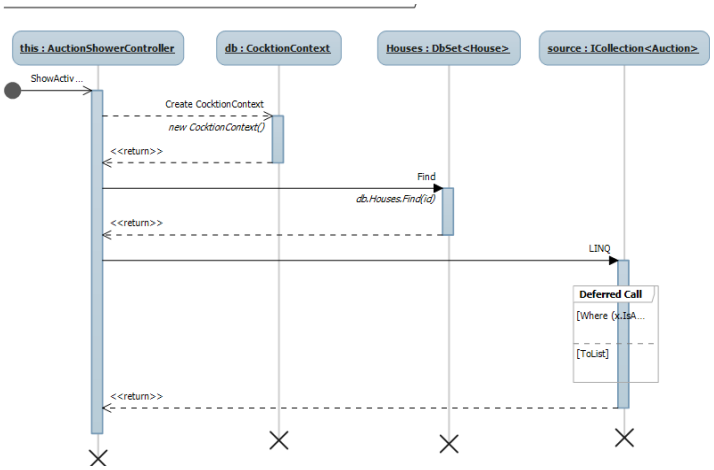


Рисунок 11

3.2.3.1.4. Описание контроллера SearchController

Метод SearchUniversity() ищет в базе данных университет по имени, в запросе посылается поисковая строка, в ответ возвращается JSON с коллекцией имен найденных университетов по заданной строке (см. рис. 12).

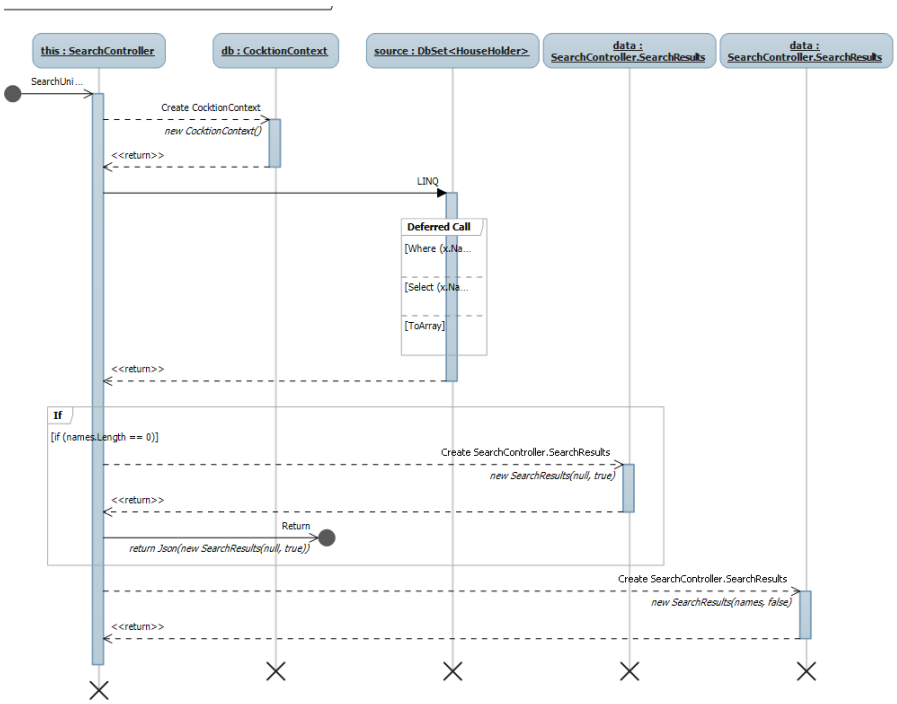


Рисунок 12.

Метод GetFacultyList() возвращает JSON со списком факультетов университета. В запросе посылается имя университета, по нему выбираются все подходящие факультеты (см. рис. 13).

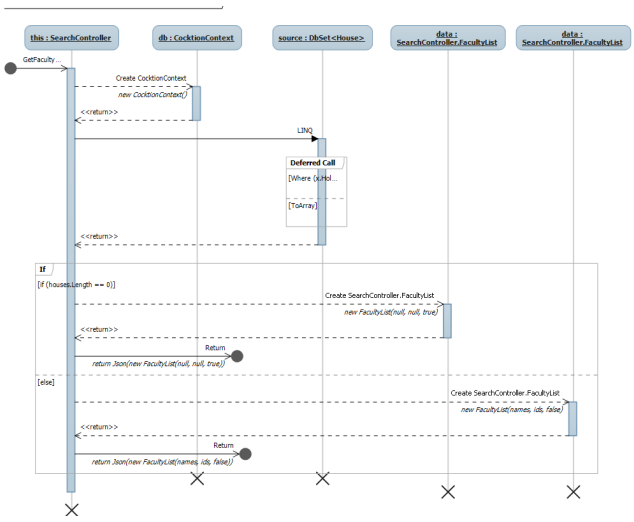


Рисунок 13.

Метод Index() отвечает за вывод поисковой страницы. Возвращает представление поисковой страницы.

Метод SearchEverywhere() ищет по всем аукционам. В запросе посылается поисковая строка, из базы данных извлекаются все аукционы, у которых название товара содержит поисковую строку. Возвращается JSON со всеми подходящими аукционами (см. рис. 14).

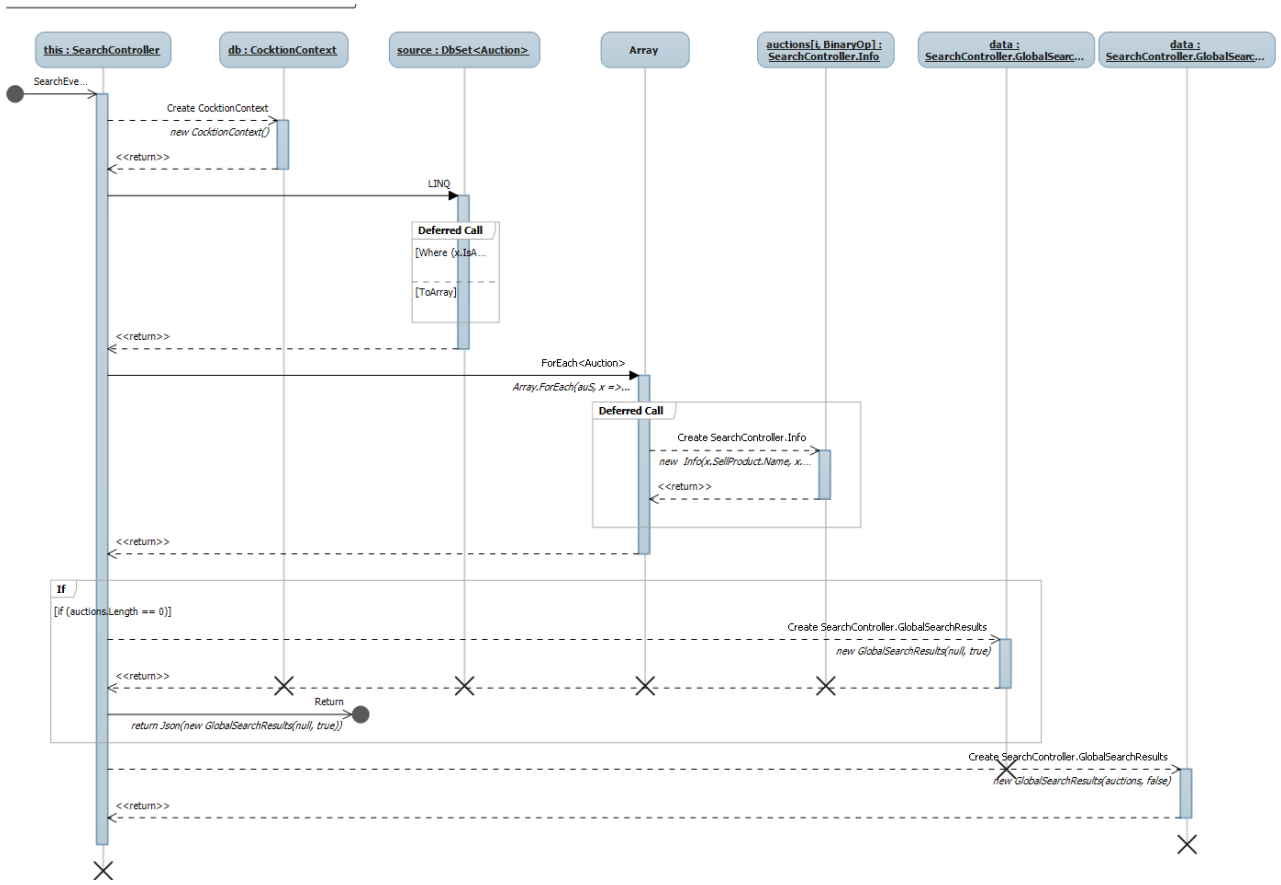


Рисунок 14.

3.2.3.1.5. Описание контроллера UsersController

Метод Index() отвечает за вывод страницы со всеми пользователями платформы. Собирает в коллекцию всех пользователей платформы. Возвращается представление типа Index (Users/) с этой коллекцией (см. рис. 15).

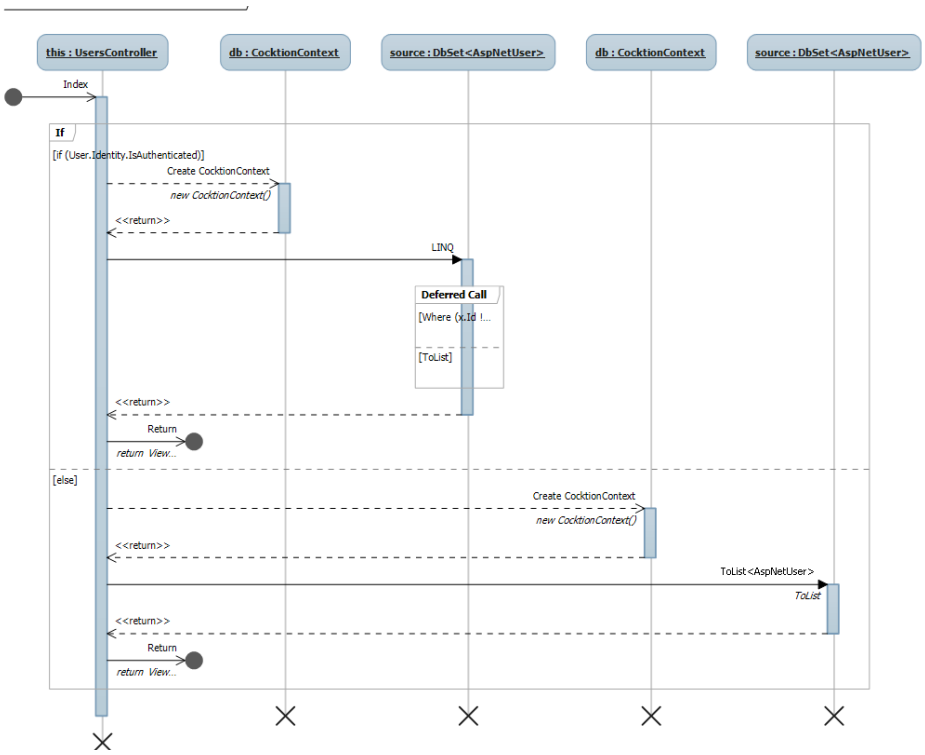


Рисунок 15.

Метод GetUser(int id) отвечает за вывод страницы конкретного пользователя. По айди из базы данных извлекается пользователь, возвращается представление типа GetUser с этим пользователем (см. рис. 16)

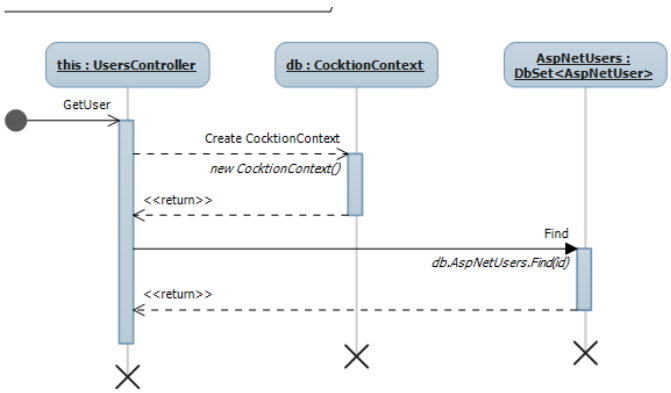


Рисунок 16.

3.2.3.1.6. Описание контроллера BidAuctionCreatorController

Метод CreateAuction() отвечает за создание нового аукциона. В запросе поступают следующие данные: название, описание, категория, временные рамки, локации, в которых будут проводиться аукционы, фотография товара, если есть. Создается новый аукцион, записывается в базу данных, обновляется список аукционов на остальных открытых страницах, возвращается JSON с айдишником созданного аукциона (см. рис. 17).

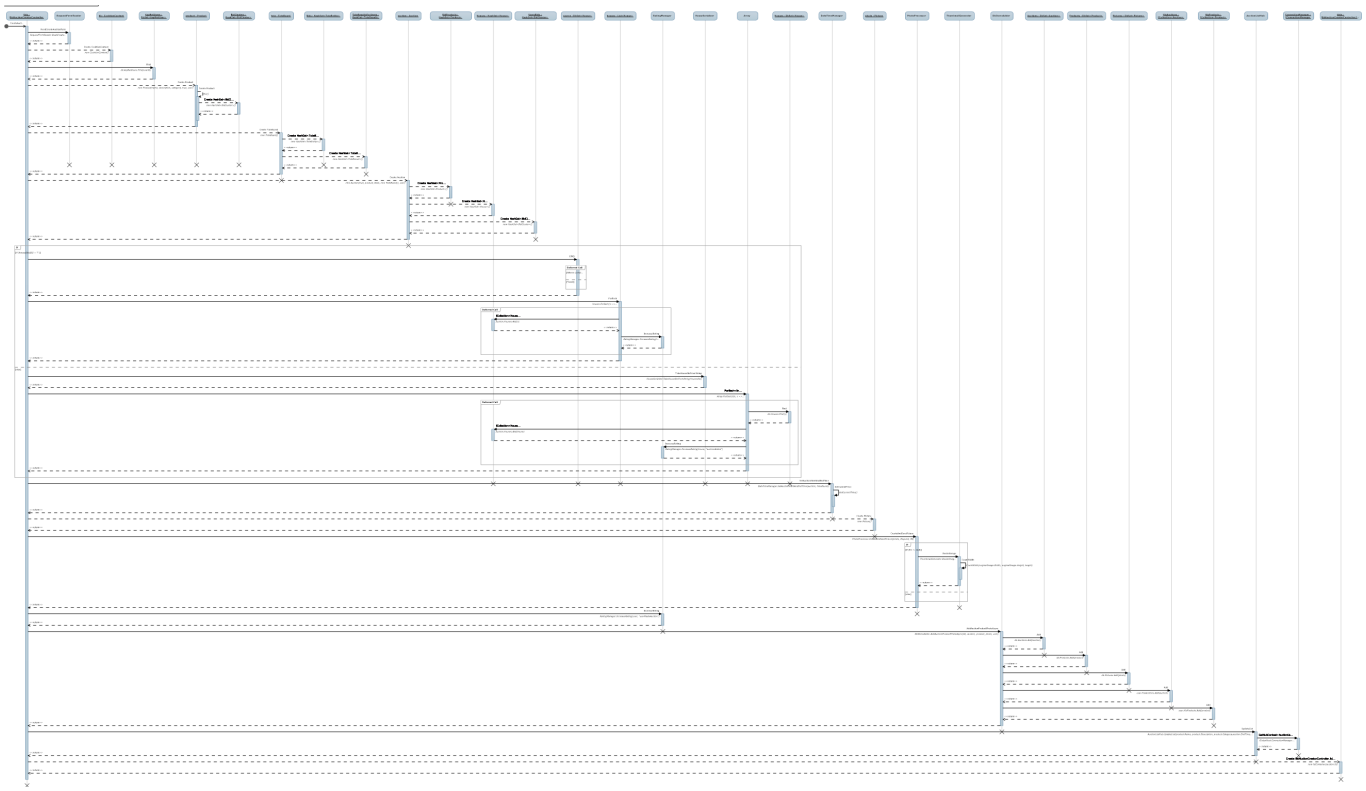


Рисунок 17.

Метод AddProductBet() отвечает за добавление ставки на аукцион. В запросе получает следующие данные: название товара, айдишник аукциона, категорию, описание, фотографию. Обновляется аукцион, на него добавляется новый товар (см. рис. 18).

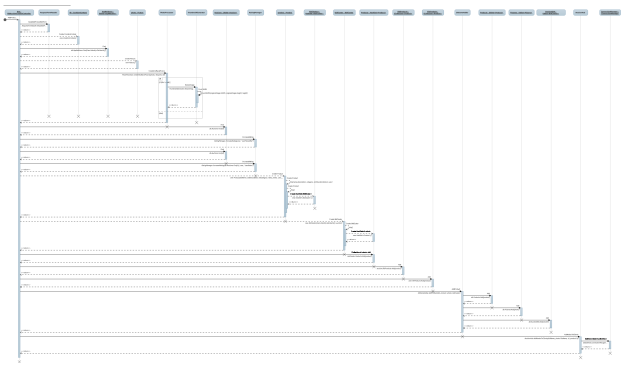


Рисунок 18.

3.2.3.1.7. Описание контроллера SubscriptionController

Метод UnsubscribeFromHouse() отвечает за отписку пользователей от локации. В запросе посылается номер локации в базе, в ответ возвращается JSON с результатом отписки (см. рис. 19).

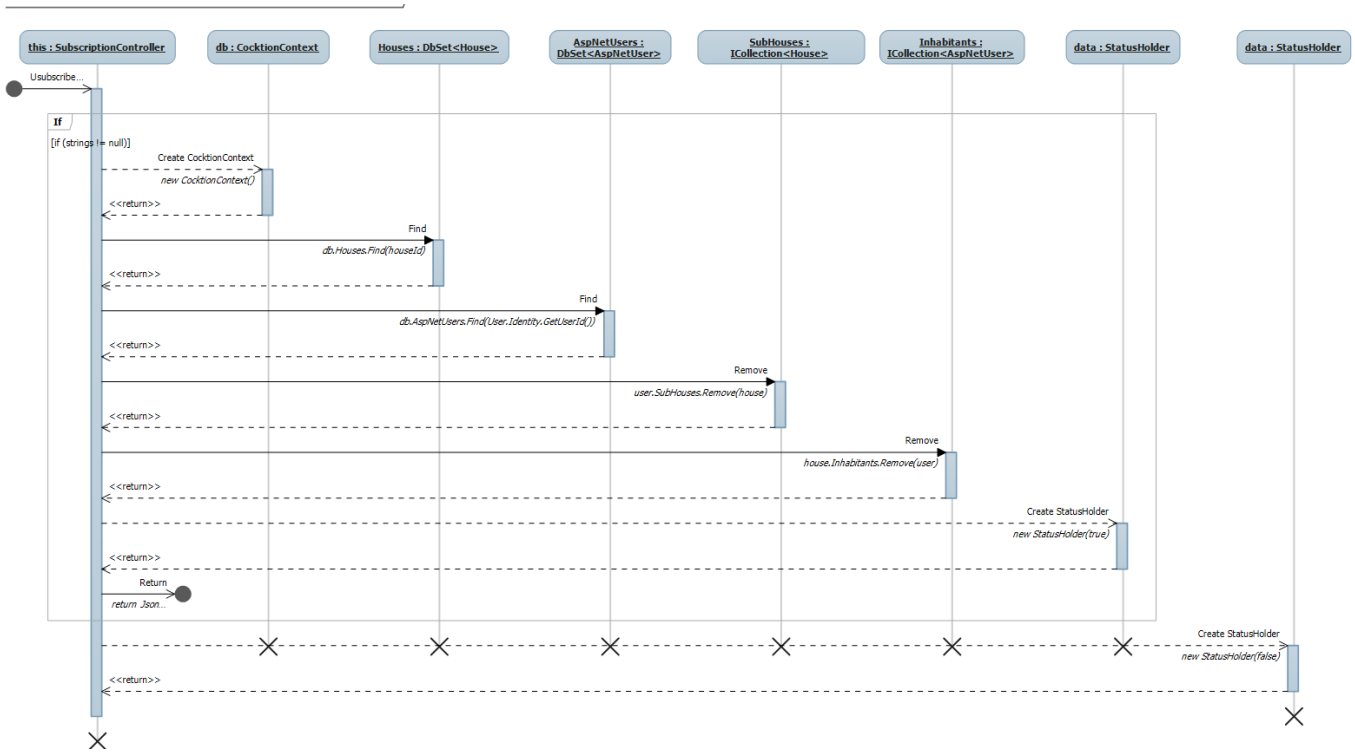


Рисунок 19

Метод SubscribeOnHouse() отвечает за подписку пользователя на локацию. В запросе посылается номер локации, на которую нужно подписаться, в ответ возвращается JSON с результатом подписки (см. рис. 20).

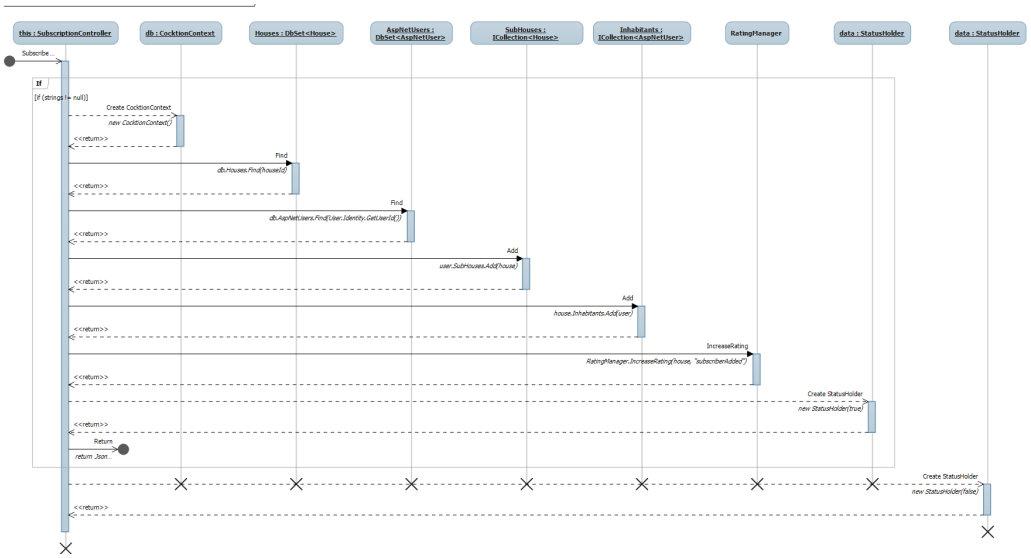


Рисунок 20.

Метод CheckHouseSubscription() позволяет проверить, подписан ли пользователь на локацию. В запросе посылается номер локации, в ответ возвращается JSON со статусом подписки (см. рис. 21).

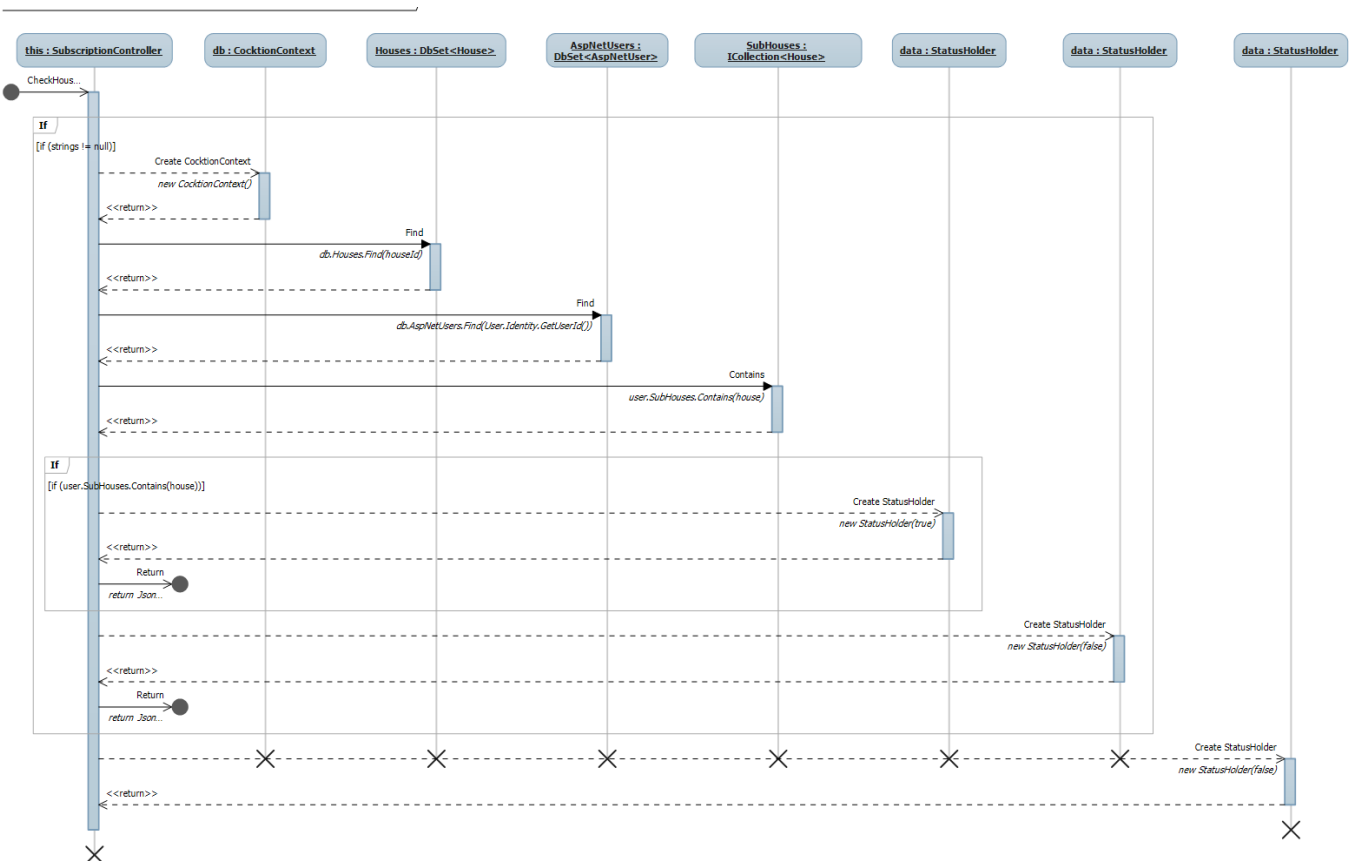
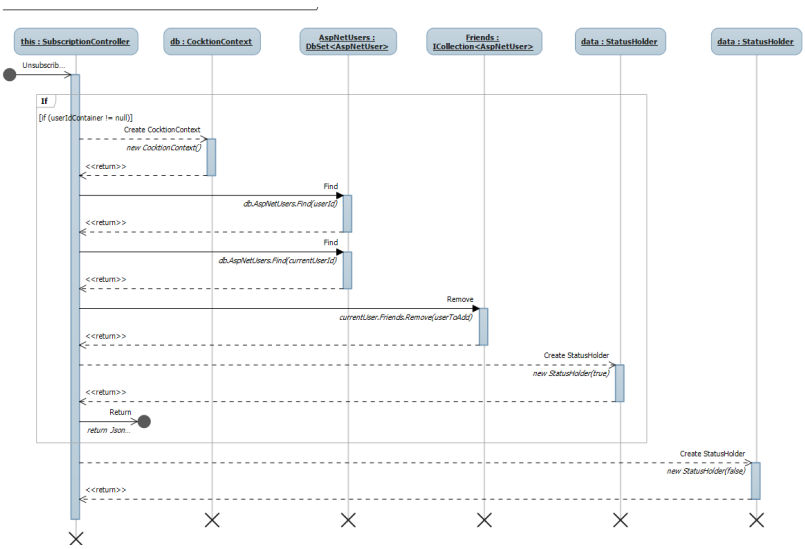


Рисунок 21.

Метод UnsubscribeFromUser() позволяет отписаться от пользователя. В запросе посылается



айдишник пользователя, в ответ присылается JSON с результатом отписки (см. рис. 22).

Рисунок 22.

Метод SubscribeOnUser() позволяет подписаться на пользователя. В запросе посылается айдишник пользователя, в ответ посылается JSON с результатом подписки (см. рис. 23).

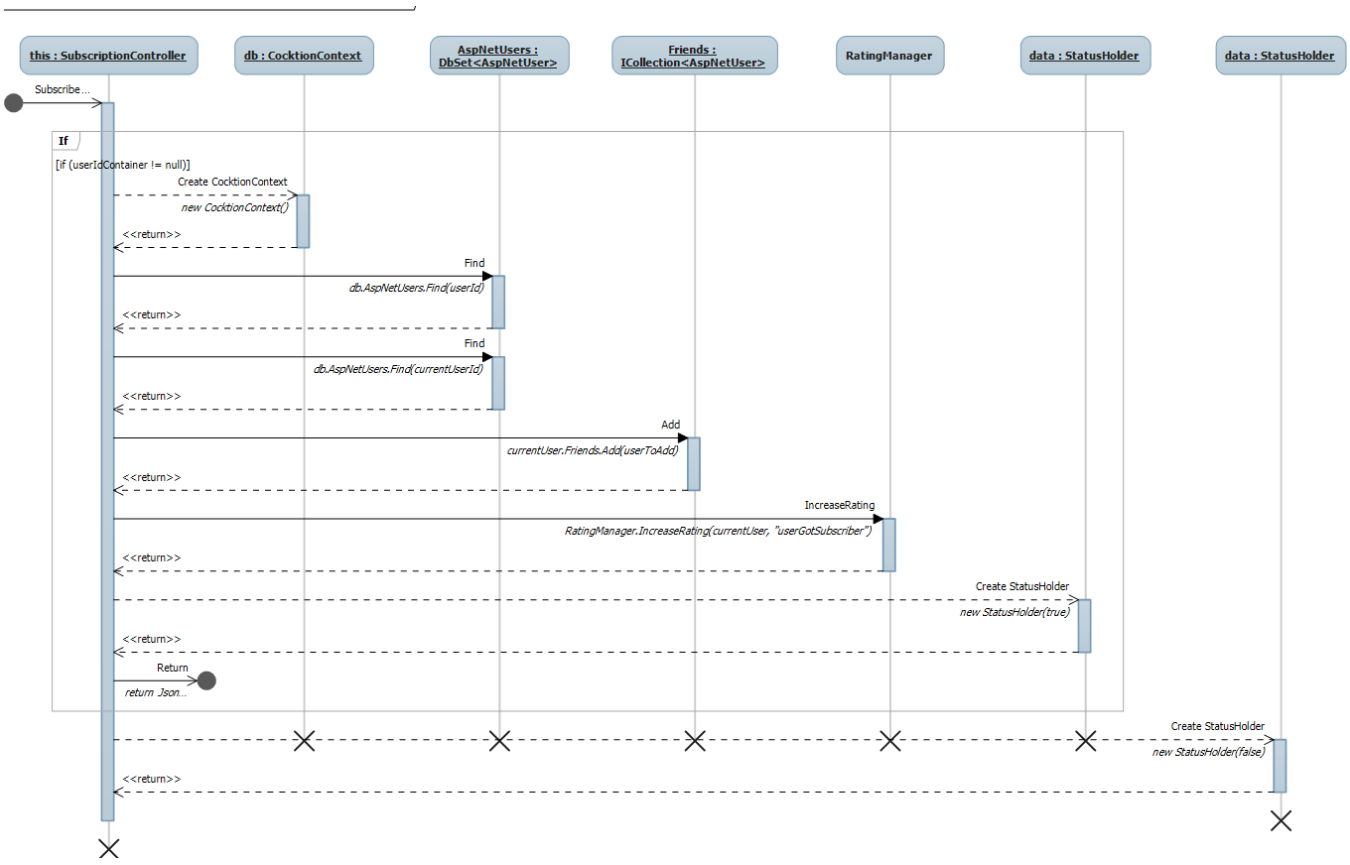


Рисунок 23.

Метод CheckUsersSubscription() позволяет проверить подписан ли данный пользователь на какого-то конкретного пользователя. В запросе посылается айдишник пользователя, в ответ присылается JSON со статусом подписки (см. рис. 24).

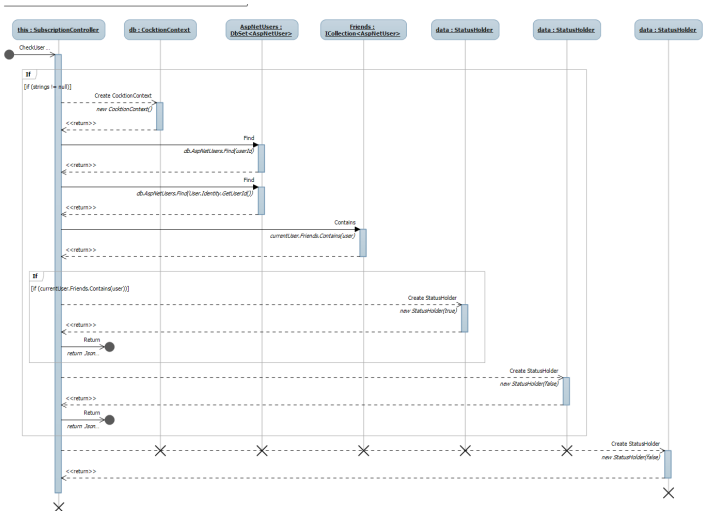


Рисунок 24.

3.2.3.1.8. Описание контроллера AuctionHouseController

Метод Index() отвечает за вывод страницы со всеми университетами. Из базы данных он берет коллекцию университетов, возвращает представление Index с этими университетами.

Метод GetUniversityHouses(int id) показывает пользователю страничку со всеми домами конкретного университета. id - номер университета в базе данных. В методе выбираются все дома, которые принадлежат данному университету, возвращает представление типа GetUniversityHouses с коллекцией домов (рис. 25).

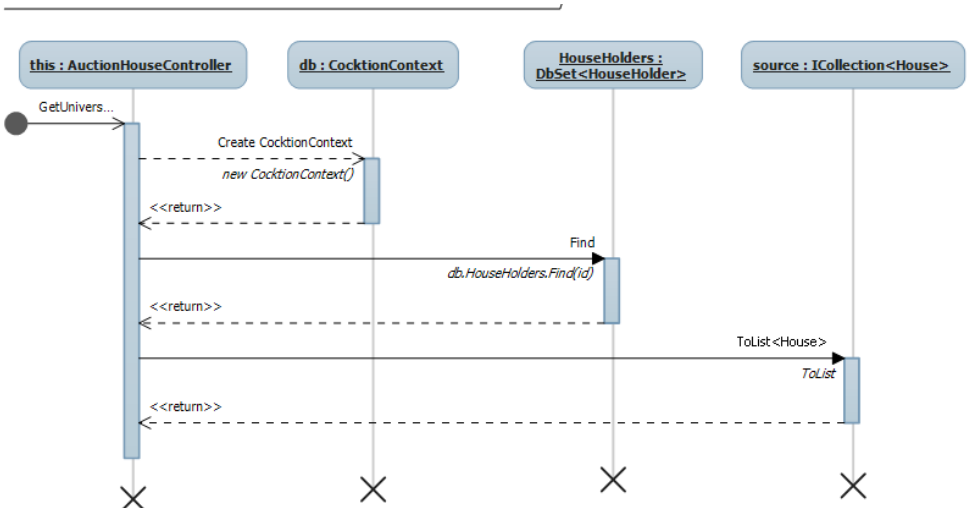


Рисунок 25.

Метод GetCurrentAuctionHouse(int id) отвечает за вывод страницы о конкретном доме. В базе данных ищется дом, с таким айди, который указали в параметре, возвращается представление типа GetCurrentAuctionHouse с этим домом (см. рис. 26).

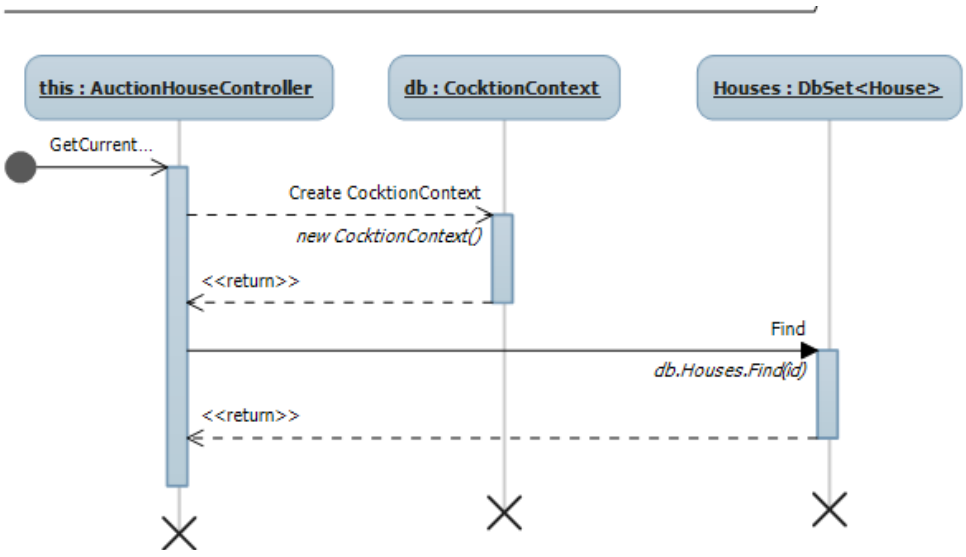


Рисунок 26.

Метод AddComment() позволяет добавлять сообщение на форуме конкретного дома. В запросе посылаются сообщение и номер дома, на форуме которого надо это сообщение оставить. В ответ возвращается JSON с статусом отправки сообщения на форум (см. рис. 27).

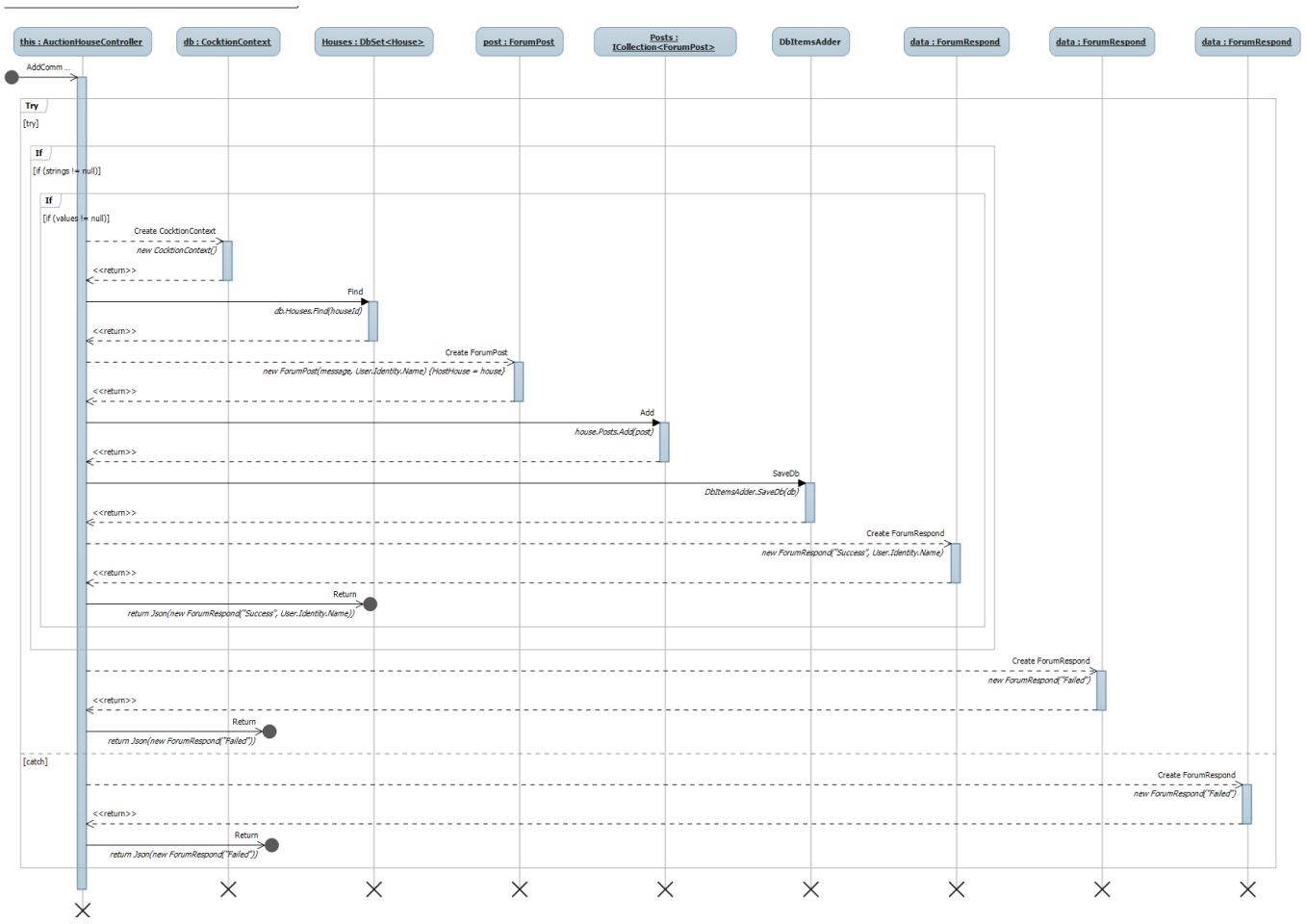


Рисунок 27.

3.2.3.1.9. Описание контроллера ProfileController

Метод Index() отвечает за вывод личного кабинета пользователя пользователю. Сначала проверяется, авторизован ли пользователь, если он не авторизован, возвращается представление HowItCouldBe, в противном случае, возвращается представление типа Profile с данными о пользователе (см. рис. 28).

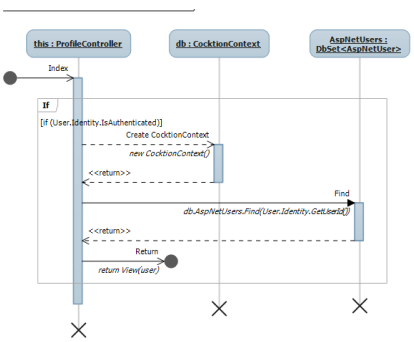


Рисунок 28.

Метод AddUsersFeedback() позволяет добавить отзыв о пользователе. Посылается запрос с сообщением и айдишником пользователя, о котором надо оставить отзыв. В запросе посылается сообщение и айдишник, в ответ присылается JSON со статусом добавления отзыва (см. рис. 29).

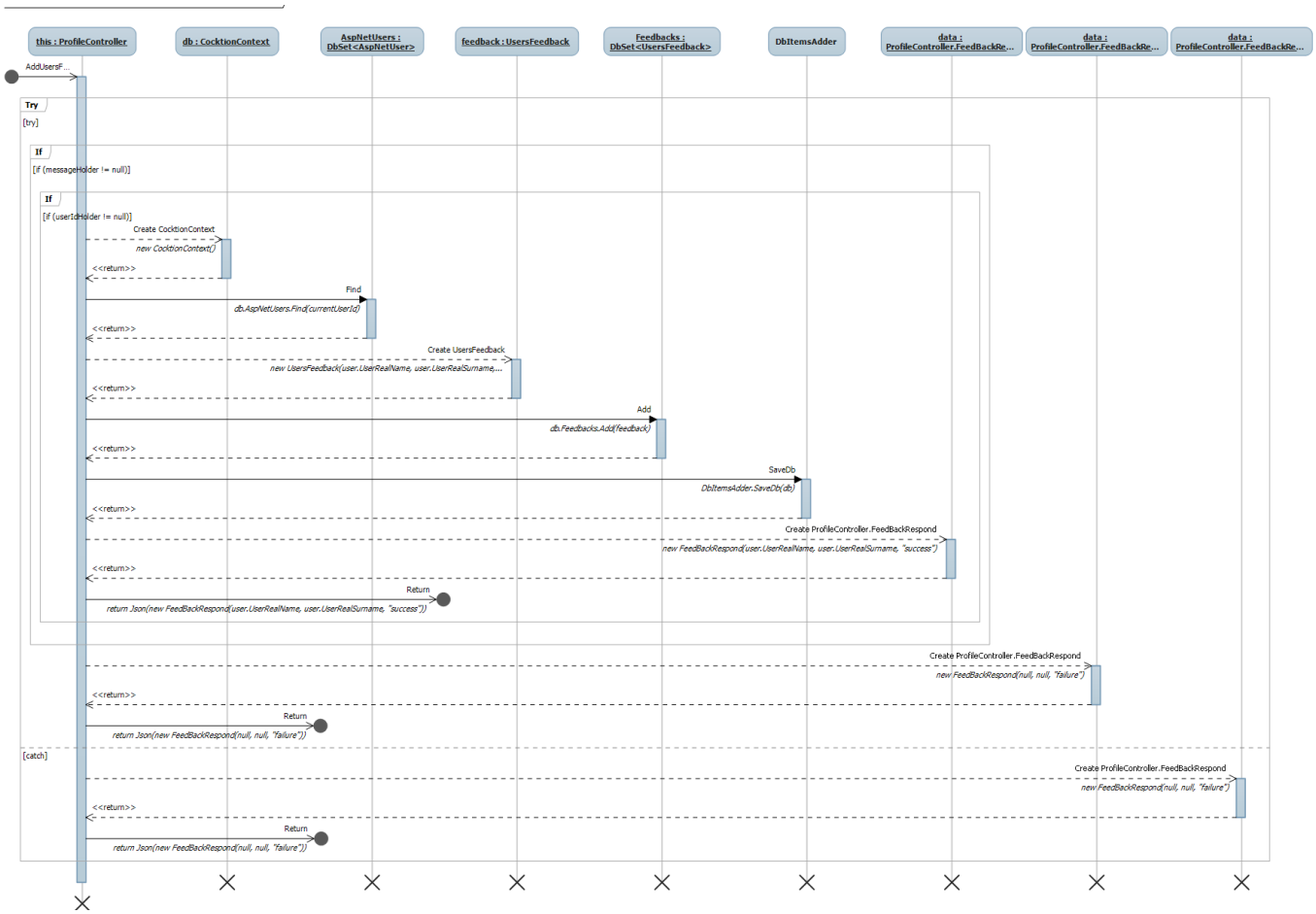


Рисунок 29.

Метод AddPhotoToUser() добавляет пользователю аватарку. В запросе посылается фотография, в ответ присылается JSON со статусом результата добавления (см. рис. 30)

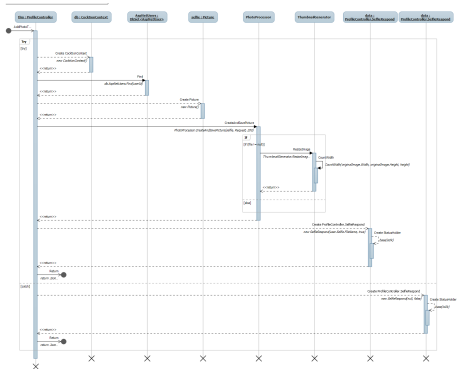


Рисунок 30.

Метод HowItCouldBe() возвращает представление HowItCouldBe, которое показывает пользователю какой могла бы быть его страница, если бы он завел аккаунт.

Метод AddInterests() добавляет пользователю интересы. Возвращает статус добавления интеерса (см. рис. 31).

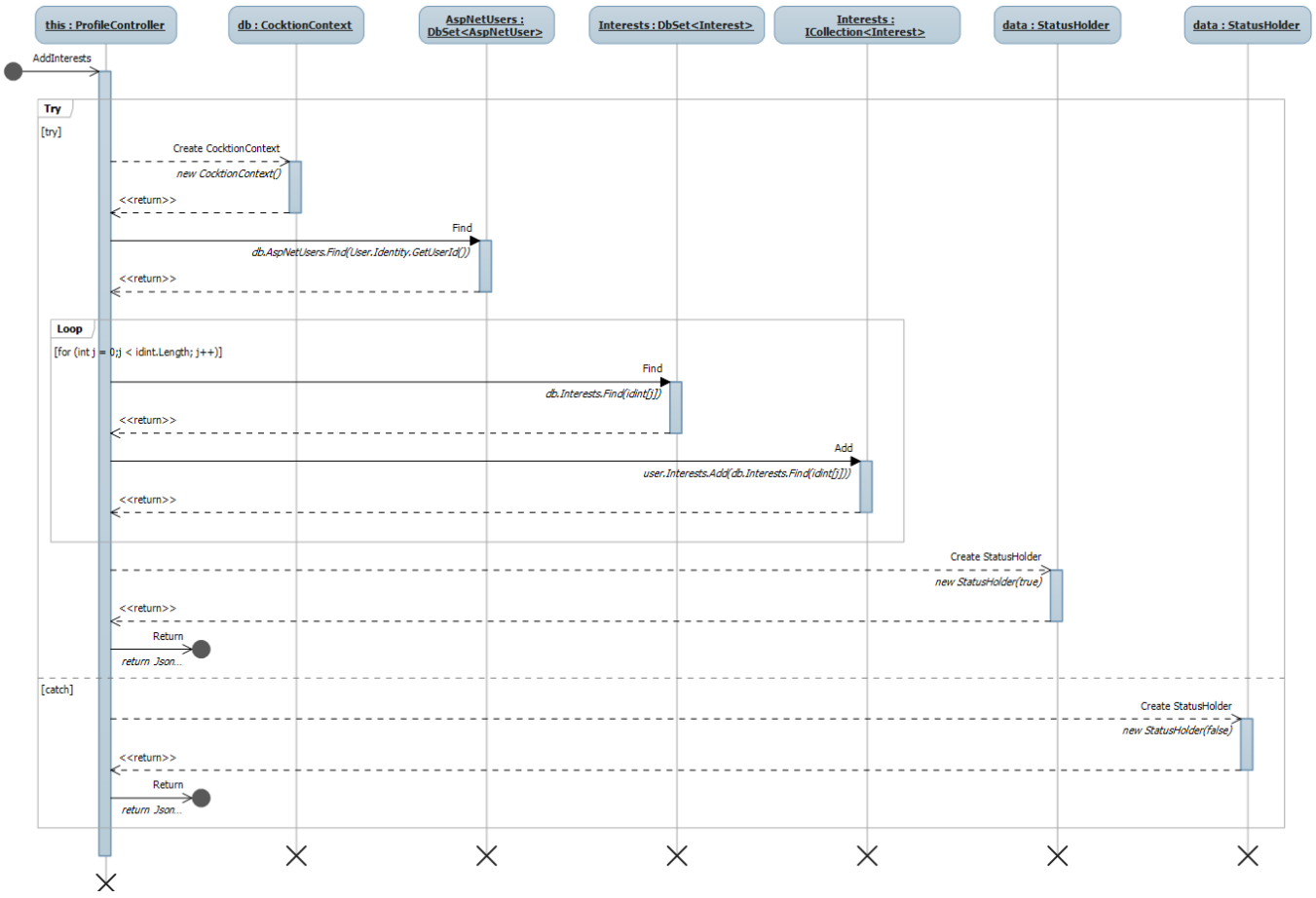


Рисунок 31.

Метод EditProfileInformation() позволяет изменять информацию пользователя (имя, фамилию, университет) для этого эти поля посылаются в запросе, в ответ присылается джейсон со статусом изменения (см. рис. 32).

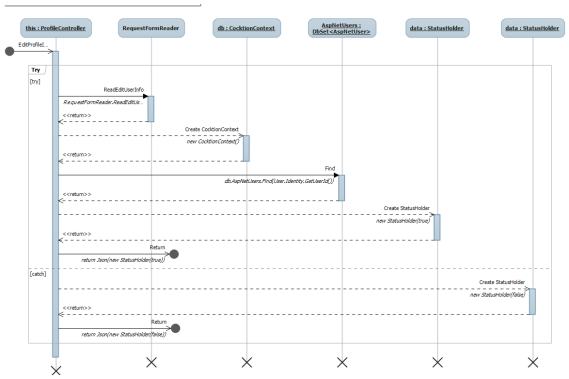


Рисунок 32.

3.2.3.1.10. Описание различных классов и функций, не входящих в контроллеры.

Класс AuctionChecker содержит методы для проверки статусов окончания аукционов. Метод StartChecking запускает таймер, который раз в час проверяет аукционы и завершает те, у которых вышел срок годности. Метод CheckAuctions (см. рис. 33) завершает просроченные аукционы и высылает людям, которые в нем участвовали результаты его окончания. (см. рис. 33)

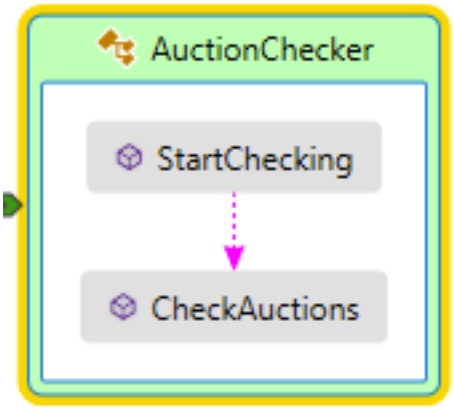


Рисунок 33.

Класс DataFormatter содержит методы для работы с данными, обработкой словарей и парсингом строк (см. рис. 34).

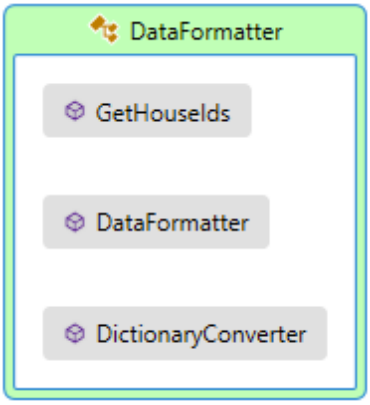


Рисунок 34.

Класс DateTimeManager содержит методы для работы со временем. Метод GetCurrentTime() конвертирует время из UTC в московское. Метод SetAuctionStartAndEndTime устанавливает время начала и завершения аукциона (см. рис. 35)

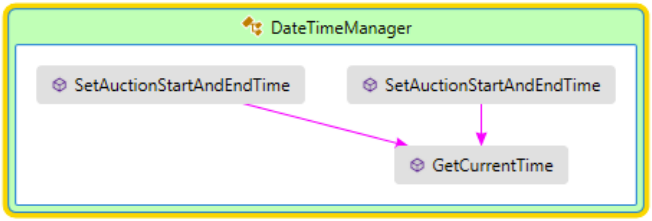


Рисунок 35.

Класс DbItemsAdder содержит методы, отвечающие за добавление объектов в базу данных. Метод AddProduct добавляет товар в базу данных. Метод SaveDb асинхронно сохраняет изменения

в базе данных. Метод AddAuctionProductPhoto добавляет в базу аукцион, товар и фотографию (см. рис. 36).

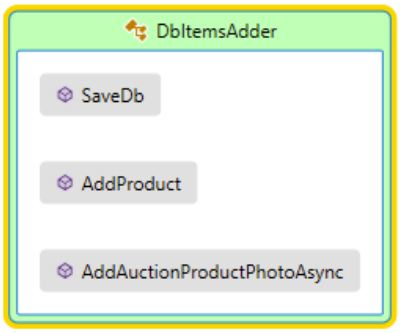


Рисунок 36,

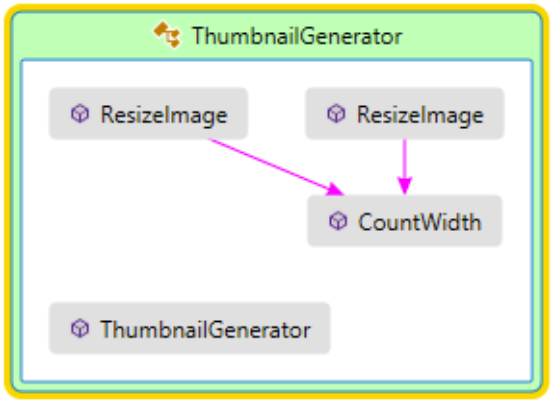
Класс HouseSerializer содержит метод TakeHouseIdsFromString из строки достает айдишники домов.

Класс PhotoProcessor является контейнером для методов, работающих с фотографиями. Метод CreateAndSavePicture из запроса достает фотографию и сохраняет ее в базе данных.

Класс RequestFormReader содержит методы для чтения полей из HTTP запросов. Метод ReadAddRateForm читает данные из запроса для ставки на тотализаторе аукциона. Метод ReadEditUserInfo читает данные из запроса для изменения информации о пользователе. Метод ReadAddLiderForm читает данные из запроса о добавлении лидера аукциона. Метод ReadCreateAuctionForm читает данные из запроса о добавлении аукциона. Метод ReadCreateAuctionFormMobile читает данные из запроса о добавлении аукциона с мобильника. Метод ReadAddProductBetForm читает данные из запроса о добавлении ставки на аукцион. Метод ReadAddHouseForm читает данные из запроса о добавлении дома (см. рис. 37).

Рисунок 37.

Класс ThumbnailGenerator содержит методы для ресайза фотографий. Метод ResizeImage из



запроса достает фотографию, и изменяет ее размер до необходимого. Метод CountWidth считает оптимальную ширину по заданной высоте (см. рис. 38).

Рисунок 38.

Класс FinishAuctionManager содержит методы для завершения аукционов. Метод AddAndSendMsgToOwnerAndWinner рассылает сообщения владельцу и победителю аукциона. Метод FalseAuctionStatus деактивирует аукцион и асинхронно выполняет сохранения нового статуса аукциона в базу данных (см. рис. 40).

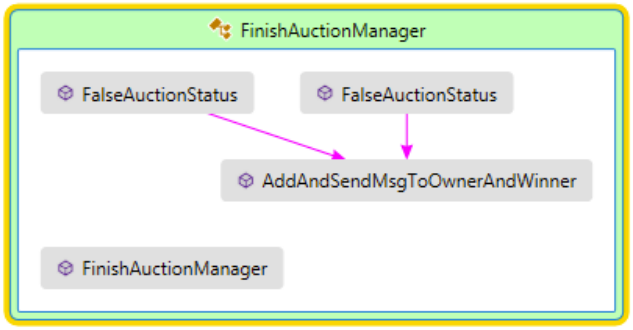


Рисунок 40.

Класс RatingManager содержит методы для работы с рейтингом. Метод IncreaseRating увеличивает рейтинг у того объекта, который помещается в параметр. (см. рис. 41)

Рисунок 41

Класс EmailSender содержит метод и поля для отправки email сообщений. Метод SendEmail отправляет сообщение с необходимым текстом по указанному адресу. (см. рис. 42)

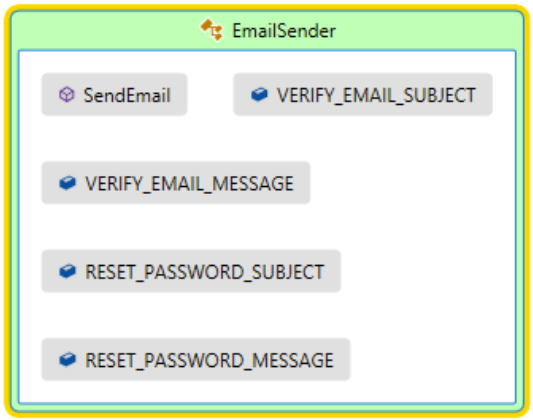


Рисунок 42.

Класс Notificator содержит методы для работы с push-уведомлениями. Метод RegisterAppleService регистрирует apple-службу для нотификации. Метод SendNotification посылает Push-уведомление на устройство (см. рис. 43).

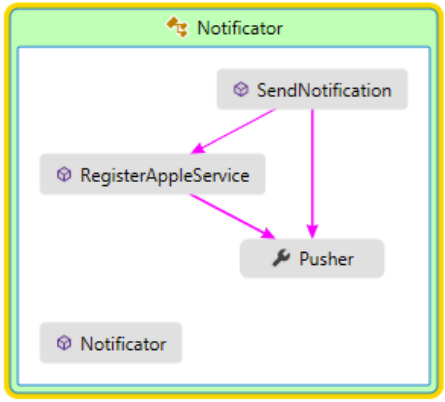


Рисунок 43.

Класс AuctionHub используется для real-time взаимодействий, возникающих на странице CurrentAuction. Метод Send посылает сообщения всем клиентам в данной группе. Метод AddNewRoom создает на сервере новую группу для чата. Метод AddNodesToClients добавляет товары в диаграмму на всех клиентах, где открыта страничка. Метод AddExtraNodeToClients добавляет довесок на всех клиентах, где открыта страничка с аукционом. Метод SetLider посылает информацию о лидере на все аукционы. Метод JoinRoom добавляет клиента в группу. Метод FinishAuction заканчивает аукцион и рассылает всем информацию о его окончании. Метод FinishAuctionMobile аналогичен по функционалу FinishAuction, но работает для мобильного устройства (см. рис. 44).

Класс AuctionListHub используется для real-time обновления панели с аукционами. Метод UpdateList Метод обновляет страничку с аукционами при добавлении нового аукциона на страницу (см. рис. 45).

Класс MessageHub используется для организации механизма чатов и личных сообщений между пользователями платформы. Метод Send посылает сообщение от одного пользователя другому. Метод AddNewRoom добавляет новую чат комнату на сервере. Метод GetListOfReceivers получает список всех людей, которые когда-либо общались с данным пользователем. Метод

GetMessages посылает сообщения чата, в котором участвует данный пользователь из базы данных (см. рис. 46).

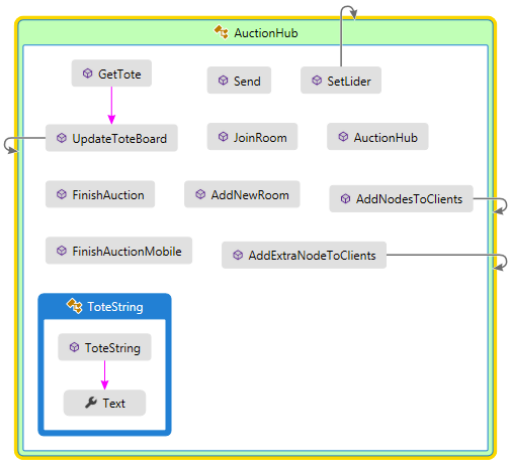


Рисунок 44.

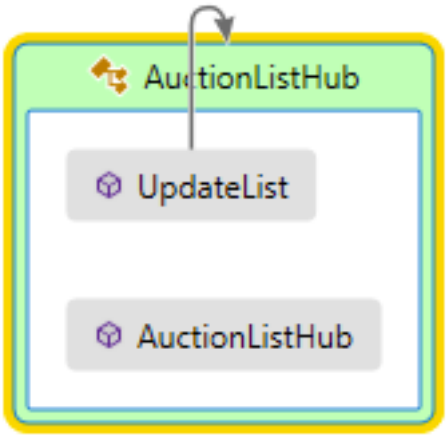


Рисунок 45.

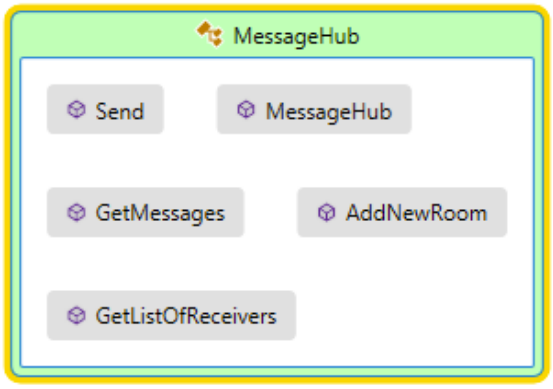


Рисунок 46

3.2.3.2. Описание функционирования API.

Общую схему функционирования API можно посмотреть на рис. 1. Если выражаться более детально, то происходит следующее: устройство проходит авторизацию, получает свой токен, который затем помещает в тело всех http-запросов, поскольку API не допускает неавторизованного доступа к функциям приложения. Далее, программа стороннего разработчика вольна делать все, что ей заблагорассудится.

3.2.3.2.1. Описание контроллеров API.

Контроллер AccountController. Содержит методы для работы с аккаунтом пользователя на платформе. Метод Registrate регистрирует пользователя с устройства. Метод Authenticate производит аутентификацию пользователя через логин и пароль, возвращает обратно токен. Метод AddDevice добавляет устройство пользователя в базу данных (см. рис. 47).

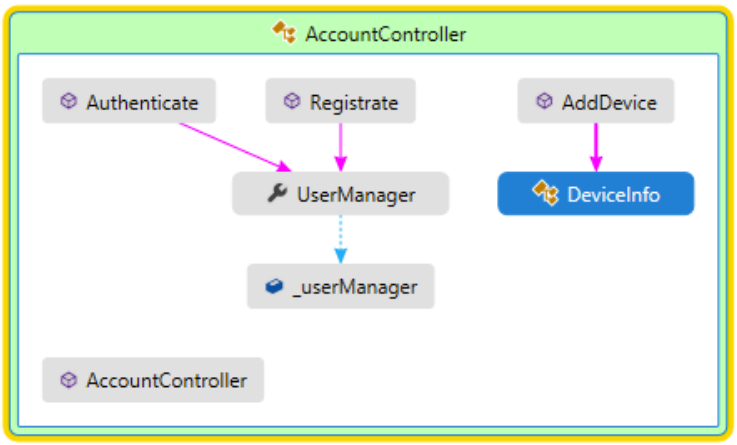


Рисунок 47.

Контроллер AuctionController. Метод GetActiveAuctions достает из базы все активные аукционы и посылает их на мобильное устройство. Метод GetAuctionBids отправляет информацию о всех товарах, находящихся на данном аукционе. Метод CreateAuction создает аукцион. Метод ChooseLeader позволяет выбирать лидера на аукционе. EndAuction - позволяет завершать аукцион с мобильника. Метод AddBid добавляет ставку на аукцион (см. рис. 48).

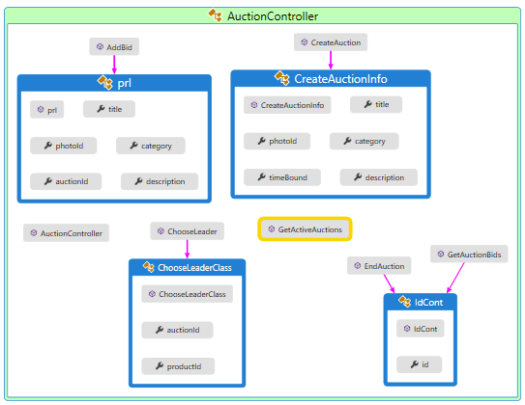


Рисунок 48.

Контроллер DataController. Содержит методы для извлечения разнообразных данных, имеющихся на платформе. Метод GetHouses посылает все имеющиеся дома. Метод FindAuctions позволяет найти любой аукцион по названию товара. Метод UploadPhoto добавляет фотографию на сервер (см. рис. 49).

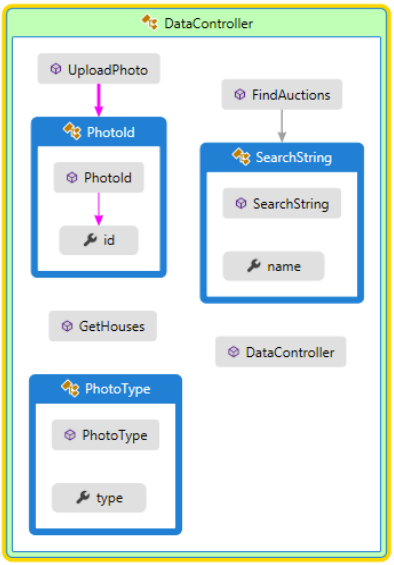


Рисунок 49.

Контроллер HouseController. Содержит методы для работы с домами. Метод GetGuilds позволяет получить список всех университетов, доступных на платформе. Метод GetGuildsHouses посылает все локации университета. Метод GetHouse посылает информацию о конкретном доме. Метод GetHouseForumPosts посылает все посты с форума дома. Метод UnsubscribeFromHouse позволяет отписываться от дома. Метод SubscribeOnHouse позволяет подписаться на дом. Метод SendPost посылает пост на форум конкретного дома (см. рис. 50).

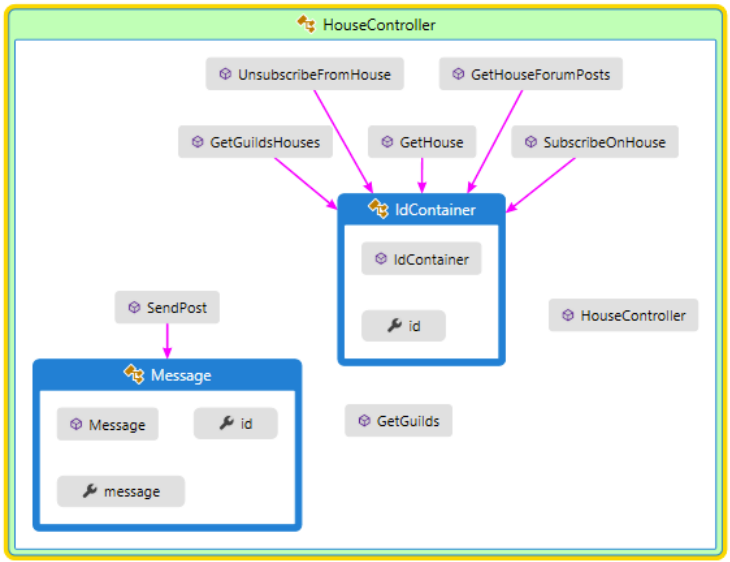


Рисунок 50.

Контроллер ProfileController. Содержит методы для работы с профилем пользователя. Метод GetInfo посылает информацию о профиле пользователя. Метод GetMyHouses позволяет получить список всех домов, на которые подписан пользователь. Метод GetMyAuctions позволяет получить различные аукционы, связанные каким-либо образом с пользователем. Метод

UploadProfilePhoto позволяет сменить аватарку пользователя. Метод EditProfile позволяет пользователю редактировать информацию о профиле. Метод GetMyInformators позволяет получить список всех людей, на которых подписался данный пользователь (см. рис. 51).

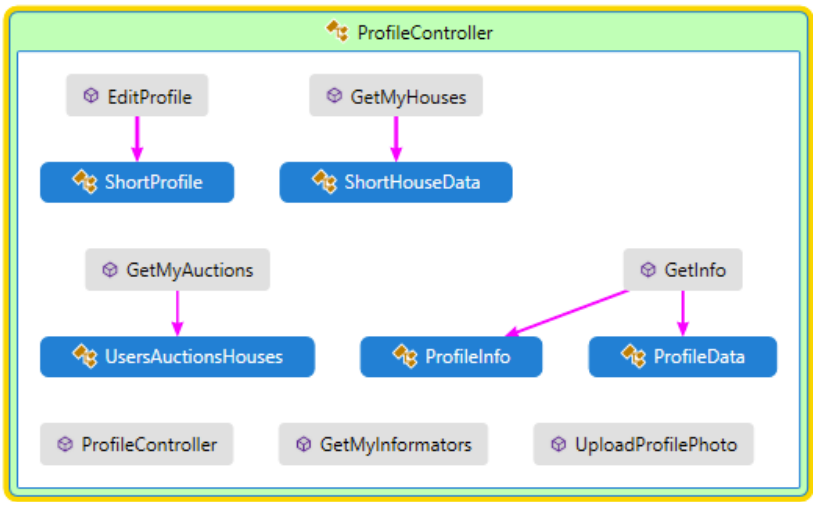


Рисунок 51.

Контроллер UsersController содержит функции для обработки информации о пользователях. Метод SubscribeOnUser позволяет подписаться на конкретного пользователя. Метод UnsubscribeFromUser позволяет отписаться от пользователя. Метод GetAllUsers посылает список всех пользователей платформы (см. рис. 52).

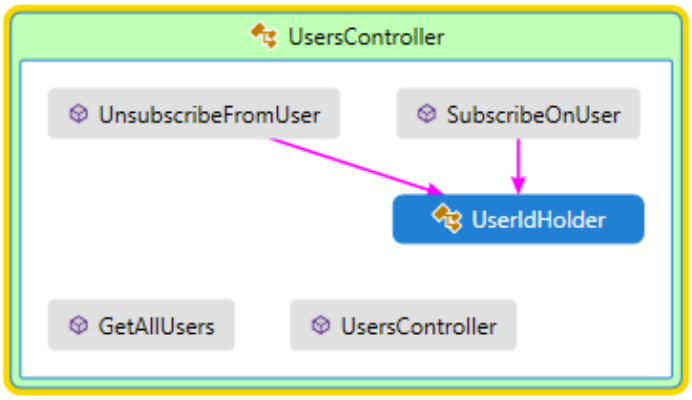


Рисунок 52.

3.2.3.2.2. Примеры для входов и выходов API.

Все обращения к API производятся через JSON’ы. Детальное описание API смотреть в руководстве программиста.

Обращения к методом контроллера AccountController.

Метод Registrate. Запрос по адресу: http://cocktion.com/api/Account/Registrate
Пример входных данных: {"email":"zz@zz.ru", «password":"moscow1"}
Пример выходных данных: {«Message":null,"Status":"Success"}

Метод Authenticate. Запрос по адресу: http://cocktion.com/api/Account/Authenticate
Пример входных данных: {"email":"zz@zz.ru", «password":"moscow1"}

Пример выходных данных: {«Token»:»HDh094ZWJdbZAoMhuhTsbLeaiUJkCjl5-0LdaccZ-ETXFPeXIKSu8nDT5N6BiBvyi-o0XZHm8P1VPx0s6gCz2w3IRuswzOJieR8NazeQ8Huvb61octMmriDAj9L6q9NX3CbumHtqq4_9ffdL4NA6wlrllQul1IoNrcwMoW_1XVIC4Og6peQG3RmLAdkf-_OwLw66-nudSJkzGvY4wLXRbXFLIdgVkl0Ff3KtCmNH94VQo9K4tZNk3AvUplLDs7zSepTwxoa140sIok0vskCpIGkDBYPEXWo0n5WCpEJ839Pxr980DJactEsXeLDSrDGSkiWUCUDTF3HkbYVzOax75w"}

Метод AddDevice требует токен устройства, не обозревается.

Обращения к методам контроллера AuctionController

Метод GetActiveAuctions. Запрос по адресу http://cocktion.com/api/Auction/GetActiveAuctions.

Входные параметры не требуются.

Пример выходных данных :

[{"description":"Стандартное описание #как_обычно»,"title":"мышка","photoPath":"http://cocktion.com/Images/Thumbnails/4e58f9ab-69d1-4a94-a6b6-7632edb4a5b2.png","endTime":3762,"auctionId":261,"leaderId":-1,"category":"Товар","isActive":false,"owner":null}]

Метод GetAuctionBids. Запрос по адресу http://cocktion.com/api/Auction/GetAuctionBids.

Пример входных данных: {«id":261}

Пример выходных данных: [{«title":"Елка","description":";lk;l","photoPath":"http://cocktion.com/Images/Thumbnails/99703688-2010-4b8d-89c6-84b66385a365.jpg","id":468,"category":"Вещь"},{"title":"шщиелки","description":"вап","photoPath":"http://cocktion.com/Images/Thumbnails/2c8c938e-8069-4d63-ac01-c17162b5b4f5.jpg","id":469,"category":"Вещь"}]

Метод CreateAuction. Запрос по адресу http://cocktion.com/api/Auction/CreateAuction

Пример входных данных: {"title":"часы", "description":"быстрые и новые", "category":"товар", "photoId":-1, "timeBound":"4hoursTime"}

Пример выходных данных: {«Status»:"Success","PhotoPath":"http://cocktion.com/Images/Thumbnails/placeholder.jpg"}

Метод ChooseLeader. Запрос по адресу http://cocktion.com/api/Auction/ChooseLeader

Пример входных данных: {"auctionId":261, "productId": 469}

Пример выходных данных: {«Status":"Failure"}

Метод AddBid. Запрос по адресу http://cocktion.com/api/Auction/AddBid

Пример входных данных: {"title":"кружка", "description":"стеклянная", "category":"вещь", "auctionId":261, «photoId":-1}

Пример выходных данных: {«Status":"Success"}

Метод EndAuction. Запрос по адресу

Пример входных данных: {«id»:261}

Пример выходных данных: {«Status":"Failure"}

Обращение к методам контроллера DataController.

Метод FindAuctions. Запрос по адресу http://cocktion.com/api/Data/FindAuctions

Пример входных данных: {«name":"вода"}

Пример выходных данных: [{"description":"Стандартное описание #как_обычно ","title":"вода холодная","photoPath":"http://cocktion.com/Images/Thumbnails/01ccf11b-c86e-4c51-a8a6-6e44a26a4d0b.png","endTime":-37059,"auctionId":214,"leaderId":-1,"category":"Услуга","isActive":false,"owner":{"id":"2a13260c-5c18-4c8c-9b21-ffc7cda189e7","title":"anton@anton.ru","photoPath":"http://cocktion.com/Images/Thumbnails/432c992d-29c7-44b4-bc86-78c04bd34e4e.png","isInformator":false}},{"description":"Стандартное описание #как_обычно\n ","title":"вода","photoPath":"http://cocktion.com/Images/Thumbnails/

placeholder.jpg","endTime":-37408,"auctionId":
221,"leaderId":-1,"category":"Услуга","isActive":false,"owner":{"id":"2a13260c-5c18-4c8c-9b21-ffc7cda189e7","title":"anton@anton.ru","photoPath":"http://cocktion.com/Images/Thumbnails/432c992d-29c7-44b4-bc86-78c04bd34e4e.png","isInformator":false}}]

Метод UploadPhoto используется для загрузки фотографии. Запрос по адресу http://cocktion.com/api/Data/UploadPhoto. Пример возвращаемых данных {«id»:200}

Обращение к методам контроллера HouseController

Метод GetGuilds. Запрос по адресу http://cocktion.com/api/House/GetGuilds входные параметры отсутствуют. Возвращаемые данные: [{«title":"МГУ","photoPath":"http://cocktion.com/Images/Thumbnails/11aacb39-6d10-4749-a0e2-2620531c4f19.jpg","id":2},
{"title":"ВШЭ","photoPath":"http://cocktion.com/Images/Thumbnails/d8b35c5c-8113-4761-994c-e2f48f2cb4c7.jpg","id":25}]

Метод GetGuildsHouses. Запрос по адресу http://cocktion.com/api/House/GetGuildsHouses
Пример входных данных: {«id":25}
Пример выходных данных: [{"title":"Компьютерных наук","id":32,"adress":"Кочновский, 3","photoPath":"http://cocktion.com/Images/Thumbnails/658d75ff-53b2-449c-8db7-d5bbedfb810d.jpg","isSubscribed":false},{"title":"Бизнеса и менеджмента","id":33,"adress":"Кирпичная, 33»,»photoPath»:"http://cocktion.com/Images/Thumbnails/7f9cb2e8-7dd8-4e37-8f1d-ad860898f1bb.jpg","isSubscribed":false}]

Метод GetHouse. Запрос по адресу http://cocktion.com/api/House/GetHouse
Пример входных данных: {«id»:32}
Пример выходных данных: {"photoPath":"http://cocktion.com/Images/Thumbnails658d75ff-53b2-449c-8db7-d5bbedfb810d.jpg","likes":0,"dislikes":0,"rating":672,"peopleAmount":4,"auctionsAmount":17,"description":"Самые четкие ребята из этого дома","isSubscribed":false,"title":"Компьютерных наук"}

Метод GetHouseForumPosts. Запрос по адресу http://cocktion.com/api/House/GetHouseForumPosts
Пример входных данных: {«id":32}
Пример выходных данных: [{«authorName":"lazarenko.ale@gmail.com","message":"Куку","likes":0},
{"authorName":"lazarenko.ale@gmail.com","message":"Кур-ру-рук","likes":0},
{"authorName":"lazarenko.ale@gmail.com","message":"ку","likes":0}]

Метод SubscribeOnHouse. Запрос по адресу http://cocktion.com/api/House/SubscribeOnHouse.
Пример входных данных: {«id":32}
Пример выходных данных: {«Status":"Success"}

Метод UnsubscribeFromHouse. Запрос по адресу http://cocktion.com/api/House/UnsubscribeFromHouse.
Пример входных данных: {«id":32}
Пример выходных данных: {«Status":"Success"}

Метод SendPost. Запрос по адресу http://cocktion.com/api/House/SendPost
Пример входных данных: {"message":"Крутотеньский дом", "id":32}
Пример выходных данных: {«Status»:»Success"}

Обращение к методам контроллера ProfileController.

Метод GetInfo. Запрос по адресу http://cocktion.com/api/Profile/GetInfo
Пример входных данных: {«id":"7f7241df-222b-4ccc-bf4a-386eaae17c58"}
Пример выходных данных:
{"title":"Альберт","surname":"Эйнштейн","login":"test@cocktion.com","rating":1016,"eggs":1000,"auctionsAmount":1,"productsAmount":1,"city":"none","society":"НИУ

ВШЭ»,"photoPath":"http://cocktion.com/Images/Thumbnails/f9480dac-a36e-4ee5-8d64-1ab074915ab1.jpg"}

Метод GetMyHouses. Запрос по адресу http://cocktion.com/api/Profile/GetMyHouses
Входные данные отсутствуют. Пример выходных данных: [{"id":32,"holderName":"ВШЭ","houseName":"Компьютерных наук»,"photoPath":"http://cocktion.com/Images/Thumbnails/658d75ff-53b2-449c-8db7-d5bbedfb810d.jpg"}]

Метод GetMyAuctions. Запрос по адресу http://cocktion.com/api/Profile/GetMyAuctions. Входные параметры не нужны. Пример выходных данных: {«active":[{"description":"быстрые и новые
","title":"часы","photoPath":"http://cocktion.com/Images/Thumbnails/placeholder.jpg","endTime":200,"auctionId":263,"leaderId":-1,"category":"товар","isActive":true,"owner":{"id":"d152160d-f8df-4573-b571-368fc6ed2720","title":"zz@zz.ru","photoPath":"http://cocktion.com/Images/Thumbnails/anonPhoto.jpg","isInformator":false}}],"finished":[],"houses":[],"products":[{"description":"Стандартное описание #как_обычно
","title":"мышка","photoPath":"http://cocktion.com/Images/Thumbnails/4e58f9ab-69d1-4a94-a6b6-7632edb4a5b2.png","endTime":3717,"auctionId":261,"leaderId":-1,"category":"Товар","isActive":true,"owner":{"id":"0000537c-a19c-4bcd-8c85-fe0b9a319f53","title":"sanalazar@yandex.ru","photoPath":"http://cocktion.com/Images/Thumbnails/95dc5572-e41f-4043-90d5-8c6d152a3056.jpg","isInformator":false}}]}

Метод UploadPhoto. Запрос по адресу http://cocktion.com/api/Profile/UploadProfilePhoto. Во входных параметрах посылать фотографию. Пример ответа: {«Status»:»Success»}

Метод EditProfile. Запрос по адресу http://cocktion.com/api/Profile/EditProfile.
Пример входных данных: {"title":"Сергей", "surname":"Бобровский", "society":"МГУ", «city":"Москва"}
Пример выходных данных: {«Status":"Success"}

Метод GetMyInformators. Запрос по адресу http://cocktion.com/api/Profile/GetMyInformators. Входные данные отсутствуют. Пример выходных данных: [{«id":"9e830116-d043-4a08-a48d-d46405f6c37c","title":"Диман","photoPath":"http://cocktion.com/Images/Thumbnails/0d8119ae-20aa-4dba-9a45-e61e78e6f0bd.png","isInformator":true}]

Обращение к методам контроллера UsersController.

Метод UnsubscribeFromUser. Запрос по адресу http://cocktion.com/api/Users/UnsubscribeFromUser
Пример входных данных: {"id":"9e830116-d043-4a08-a48d-d46405f6c37c"}
Пример выходных данных: {«Status":"Success"}

Метод SubscribeOnUser. Запрос по адресу http://cocktion.com/api/Users/SubscribeOnUser
Пример входных данных: {"id":"9e830116-d043-4a08-a48d-d46405f6c37c"}
Пример выходных данных: {«Status»:»Success}

Метод GetAllUsers. Запрос по адресу http://cocktion.com/api/Users/GetAllUsers. Входные параметры не нужны, выходные данные: [{«id»:"0000537c-a19c-4bcd-8c85-fe0b9a319f53","title":"sanalazar@yandex.ru","photoPath":"http://cocktion.com/Images/Thumbnails/95dc5572-e41f-4043-90d5-8c6d152a3056.jpg","isInformator":false},{"id":"1b772d06-1d2c-4444-83d2-d991ade79706","title":"vk@vk.com","photoPath":"http://cocktion.com/Images/Thumbnails/dff57724-b31a-4d59-8cdd-158708486fbd.png","isInformator":false}]

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПАКАЗАТЕЛИ

4.1 Предполагаемая потребность

У студентов скапливается дома (или в общежитии) масса ненужных вещей, которые просто доживают свой век на полках, занимая полезное место и накапливая пыль. С другой стороны, потраченные деньги за них уже вряд ли кто-то пойдет возвращать и у студентов есть потребность в обмене ненужных вещей на более нужные. Программа представляет собой очень удобный и приятный к использованию инструмент, позволяющий максимально ускорить процесс обмена (в форме аукциона), обсуждения и ускорения переговоров между людьми, регулярно посещающих одни и те же места.

4.2 Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

4.2.1. Сравнение с платформой eBuy

Платформа представляет собой Интернет-аукционы вещь за деньги. Люди выкладывают вещи, а другие люди кидают денежные ставки. Отличительные черты и преимущества: Совершенно другой принцип, ориентация на другие потребности пользователей. Наиболее не похож на разрабатываемую программу.

4.2.2. Сравнение с платформой Avito

Платформа представляет собой доску объявлений для продажи вещей. Люди просто-напросто выкладывают объявления. Отличительные черты и преимущества: Авито является по сути доской объявлений, а разрабатываемое приложение - интерактивной платформой для взаимодействия пользователей через аукционы и общение.

4.2.3. Сравнение с платформой VseVObmen

Платформа представляет собой сервис для обмена вещей. Наибольшая степень аналогичности с разрабатываемым. Люди выкладывают вещь для обмена, а в качестве предложений пишут текстовые сообщения. Отличительные черты и преимущества: Данный сервис наиболее похож на разрабатываемое приложение. В первую очередь, разрабатываемое приложение ориентировано на студентов, поэтому потребность реализовать обмен наиболее быстро и удобно реализована напрямую, через подписки на пользователей и дома, а так же на четую привязку аукциона к геолокациям, что делает процесс обмена более понятным и ясным. На платформе «VseVObmen» можно размещать товары без четкой привязки. Так же, процесс обмена интерактивен в разрабатываемом приложении, что дает возможность пользователям просматривать предложения других людей в виде конкретных предметов, а не текстовых полей, обсуждать в реальном времени ход торга. На «VseVObmen» предложить свой товар можно только в виде текстового поля, так же отсутствует возможность обсуждать ход торгов в реальном времени, нет подписок на локации и нельзя посмотреть предложения по местоположению.

4.2.4. Сравнение с платформой SwapTreasures

Платформа представляет собой сервис, похожий на аукцион. Люди выкладывают вещи и за внутреннюю валюту можно ставить ставки. Валюта докупается на реальные деньги. Отличительные черты и преимущества: преимущества аналогичны с преимуществами относительно VseVObmen, плюс ко всему для использования разрабатываемого приложения не требуются реальные деньги. Разрабатываемое приложение имеет интерфейс на русском языке.

4.2.5. Сравнение с группой в Вконтакте «Обменник»

Представляет собой группу, где люди могут просто выкладывать свои вещи и дальше с ними что-то делать. Отличительные черты и преимущества: преимущества аналогичны с перечисленными выше, кроме того, разрабатываемое приложение является самостоятельным, автономным сервисом, разработанным под необходимый функционал.

5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. Макконелл С. Совершенный код. Мастер-класс / Пер. с англ. - М. : Издательство «Русская редакция», 2010. - 896 стр. : ил.
12. Munro J. 20 Recipes for Programming MVC 3. -U.S.A.: O’Reilly Media, 2011.
13. Galloway J. Professional ASP.NET MVC 5. -U.S.A.: John Wiley & Sons, 2014.
14. Rolando Guay Paz J. Beginning ASP.NET MVC 4. - U.S.A.: Apress, 2012.
15. Палермо Д. ASP.NET MVC 4 в действии. Издательство Manning, 2012.
16. Сандерсон С. ASP.NET Framework с примерами на C# для профессионалов. : Пер. с англ. - М: ООО «И.Д. Вильямс», 2010. - 560 с. : ил. - Парал. тит. англ.
17. Chadwick J. Programming ASP.NET MVC 4. -U.S.A.: O’Reilly Media, 2012.

18. Guidance to ASP.NET MVC [Электронный ресурс]// URL: http://www.asp.net/mvc (Дата обращения 15.12.2014, режим доступа: свободный).
19. Guidance to ASP.NET Web API [Электронный ресурс] // URL: http://www.asp.net/web-api (Дата обращения 5.03.2015, режим доступа: свободный).
20. Guidance to ASP.NET SignalR [Электронный ресурс] // URL: http://www.asp.net/signalr (Дата обращения 15.12.2014, режим доступа: свободный).
21. Guidance to Entity Framework [Электронный ресурс] // URL.: http://www.asp.net/entity-framework (Дата обращения 15.12.2014, режим доступа: свободный).
22. Powers S. JavaScript Cookbook. -U.S.A: O'Reilly Media, 2010.
23. Solis D. Illustrated C# 2012. -U.S.A.: Apress, 2012.
24. Шилдт Г. C# 4.0: Полное руководство. :Пер. с англ. - М. :ООО «И.Д. Вильямс», 2011. - 1056 с. : ил. - Парал. тит. англ.
25. Подбельский В. Язык C# Базовый курс: учеб. пособие/ -М.: Издательство «Финансы и статистика», 2010. - с.: ил.
26. Sharp J. Microsoft Visual C# 2012 Step by Step. -U.S.A.: O'Reilly Media, 2012.
27. jQuery API Documentation [Электронный ресурс] // URL.: http://api.jquery.com (Дата обращения 25.12.2014, режим доступа: свободный).
28. Справочник HTML [Электронный ресурс] // URL.: http://htmlbook.ru/blog/spravochnik-html (Дата обращения 25.12.2014, режим доступа: свободный).
29. Справочник CSS [Электронный ресурс] //URL.: http://htmlbook.ru/css (Дата обращения 25.12.2014, режим доступа: свободный).
30. vis.js documentation [Электронный ресурс] // URL.: http://visjs.org/docs/index.html (Дата обращения 25.12.2014, режим доступа: свободный).
31. Bootstrap [Электронный ресурс] // URL.: http://getbootstrap.com/components/ (Дата обращения 25.12.2014, режим доступа: свободный).
32. jQuery.countdown Documentation [Электронный ресурс] //URL.: http://hilios.github.io/jQuery.countdown/documentation.html (Дата обращения 25.12.2014, режим доступа: свободный).
33. Claims And Token Based Authentication (ASP.NET Web API) [Электронный ресурс] // URL.: http://www.codeproject.com/Tips/821772/Claims-And-Token-Based-Authentication-ASP-NET-Web (Дата обращения 20.03.2015, режим доступа: свободный).
34. PushSharp Documentation [Электронный ресурс] // URL.: https://github.com/Redth/PushSharp/wiki (Дата обращения 04.04.2015, режим доступа: свободный).
35. Документация Microsoft Azure [Электронный ресурс] // URL.: http://azure.microsoft.com/ru-ru/ (Дата обращения 14.01.2015, режим доступа: свободный).
36. Документация Amazon EC2 [Электронный ресурс] //URL.: http://aws.amazon.com/ru/documentation/ec2/ (Дата обращения 14.03.2015, режим доступа: свободный).
37. reCAPTCHA for .NET [Электронный ресурс] //URL.: http://recaptcha-net.mtd226.com (Дата обращения 20.03.2015, режим доступа: свободный)
38. Aegis Implicit Mail [Электронный ресурс] //URL.: http://sourceforge.net/projects/netimplicitssl/ (Дата обращения 24.03.2015, режим доступа: свободный)
39. DNS-хостинг Яндекс [Электронный ресурс] //URL.: https://help.yandex.ru/pdd/hosting.xml (Дата обращения 12.02.2015, режим доступа: свободный)
40. The MVC Pattern and ASP.NET MVC - Back to Basics //URL.: http://www.dotnetcurry.com/showarticle.aspx?ID=922 (Дата обращения 12.05.2015, режим доступа: свободный)
41. Веб-приложение [Электронный ресурс] //URL.: https://ru.wikipedia.org/wiki/Веб-приложение (Дата обращения 19.05.2015, режим доступа: свободный)

42. API [Электронный ресурс] //URL.: https://ru.wikipedia.org/wiki/API (Дата обращения 19.05.2015, режим доступа: свободный)
43. JSON [Электронный ресурс] //URL.: https://ru.wikipedia.org/wiki/JSON (Дата обращения 19.05.2015, режим доступа: свободный)

ПРИЛОЖЕНИЕ 1

СХЕМА СУЩНОСТЕЙ БАЗЫ ДАННЫХ

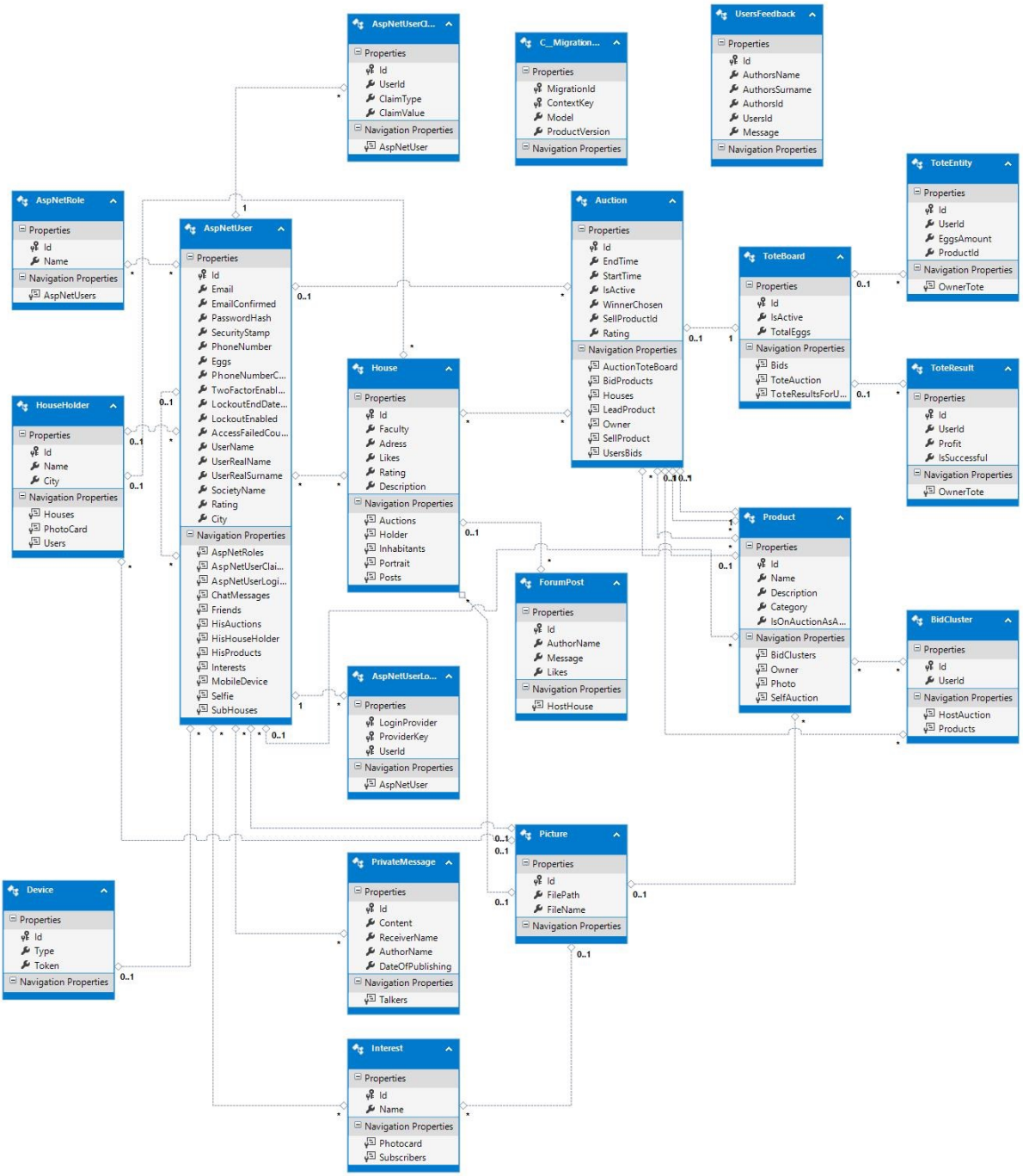


Рисунок 1

ПРИЛОЖЕНИЕ 2

СПИСОК ТЕРМИНОВ

Web-приложение - клиент-серверное приложение, в котором клиентом выступает браузер, а сервером - веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. [41]

Сущность - любой единственный, идентифицируемый раздельный объект.

API - (англ. application programming interface) - набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах. [42]

JSON - (англ. JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. [43]

Аукцион - формат взаимодействия студентов, при котором они в режиме реального времени имеют возможность обмениваться вещами и обсуждать обмен.

Локация - факультет или общежитие, относящееся к конкретному университету.

Подписка - добавление пользователя или локации в список, на основании которого будет производиться отбор аукционов для отображения пользователю.

Отписка - изъятие пользователя или локации из списка, на основании которого будет производиться отбор аукционов для отображения пользователю.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	Замененных	новых	аннулированных					