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
Академический 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

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.509000-01 12 01-1-ЛУ

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

2015

УТВЕРЖДЕНО
RU.17701729.509000-01 12 01-1 ЛУ

WEB-ПРИЛОЖЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ СТУДЕНТОВ

Текст программы

RU.17701729.509000-01 12 01-1

Листов 266

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

2015

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ

1.1. MVC контроллеры

1.1.1. Класс EditHousesController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using System.Web.WebPages;
using CocktionMVC.Functions;
using CocktionMVC.Functions.DataProcessing;
using CocktionMVC.Models;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using Microsoft.Owin.Security.Twitter.Messages;

namespace CocktionMVC.Controllers.AdminControllers
{
    /// <summary>
    /// Служит для редактирования и добавления новых аукционных домов в базу данных
    /// </summary>
    public class EditHousesController : Controller
    {
        /// <summary>
        /// Возвращает страничку со списком аукционных домов
        /// либо открывает доступ к редактированию их для
        /// пользователей - администраторов
        /// </summary>
        [Authorize]
        public ActionResult Index()
        {
            //Возвращаем разные вьюшки с правами, в зав-ти от пользователя
            if (User.Identity.Name == "darya-coo@cocktion.com" || User.Identity.Name ==
"lazarenko.ale@gmail.com")
            {
                CocktionContext db = new CocktionContext();
                var houses = db.Houses.ToList();
                var holders = db.HouseHolders.ToList();
                Tuple<List<House>, List<HouseHolder>> tuple
                    = new Tuple<List<House>, List<HouseHolder>>(houses, holders);
                return View(tuple);
            }
            return View("PageNotFound");
        }
```

```
/// <summary>
/// Добавляет дом в базу данных
/// </summary>
[HttpPost]
[Authorize]
public JsonResult AddHouse()
{
    //Чтение из запроса с клиента данных о доме.
    string holderId;
    string faculty;
    string adress;
    string description;
    RequestFormReader.ReadAddHouseForm(Request, out faculty, out adress, out holderId);

    CocktionContext db = new CocktionContext();
    var holder = db.HouseHolders.Find(int.Parse(holderId));

    //обработать фотку
    Picture photo = new Picture();
    PhotoProcessor.CreateAndSavePicture(photo, Request, 200);

    //создать дом
    House house = new House(adress, faculty, holder, photo);
    description = Request.Form.GetValues("description")[0].Trim();
    house.Description = description;

    //добавить дом в холдер
    holder.Houses.Add(house);

    //добавить дом в базу
    db.Houses.Add(house);
    db.Pictures.Add(photo);
    db.SaveChanges();
    //вернуть статус

    return Json(new StatusHolder(true));
}

/// <summary>
/// Удаляет дом, указанный в айдишнике
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
public JsonResult DeleteHouse()
{
    string houseId = Request.Form.GetValues("houseId")[0].Trim();
```

```
    int id = int.Parse(houseId);
    CocktionContext db = new CocktionContext();
    db.Houses.Remove(db.Houses.Find(id));
    db.SaveChanges();
    return Json(new StatusHolder(true));
}

/// <summary>
/// Удаляет указанный в айдишнике холдер
/// </summary>
/// <returns>стандартный ответ</returns>
[HttpPost]
public JsonResult DeleteHolder()
{
    string holderId = Request.Form.GetValues("holderId")[0].Trim();
    int id = int.Parse(holderId);
    CocktionContext db = new CocktionContext();
    db.HouseHolders.Remove(db.HouseHolders.Find(id));
    db.SaveChanges();
    return Json(new StatusHolder(true));
}

public class Q
{

    public string Status { get; set; }

}

/// <summary>
/// Позволяет редактировать холдер,
/// указанный в айдишнике
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult EditHolder()
{
    Picture selfie = new Picture();
    if (Request.Files.Count != 0)
    {
        PhotoProcessor.CreateAndSavePicture(selfie, Request, 200);
    }

    CocktionContext db = new CocktionContext();
    string holderId = Request.Form.GetValues("holderId")[0].Trim();
    string holderName = Request.Form.GetValues("holderName")[0].Trim();
```

```
    string holderCity = Request.Form.GetValues("holderCity")[0].Trim();

    var holder = db.HouseHolders.Find(int.Parse(holderId));
    if (!selfie.FileName.IsEmpty())
    {
        holder.PhotoCard = selfie;
    }

    if (!holderName.IsEmpty())
    {
        holder.Name = holderName;
    }

    if (!holderCity.IsEmpty())
    {
        holder.City = holderCity;
    }

    db.SaveChanges();
    return Json(new StatusHolder(true));
}

/// <summary>
/// Позволяет редактировать нужный дом
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult EditHouse()
{
    Picture selfie = new Picture();
    if (Request.Files.Count != 0)
    {
        PhotoProcessor.CreateAndSavePicture(selfie, Request, 200);
    }

    CocktionContext db = new CocktionContext();
    string houseId = Request.Form.GetValues("houseId")[0].Trim();
    string houseName = Request.Form.GetValues("houseName")[0].Trim();
    string houseDescription = Request.Form.GetValues("houseDescription")[0].Trim();
    string houseAdress = Request.Form.GetValues("houseAdress")[0].Trim();

    var house = db.Houses.Find(int.Parse(houseId));
    if (!selfie.FileName.IsEmpty())
    {
        house.Portrait = selfie;
    }
```

```
    if (!houseName.IsEmpty())
    {
        house.Faculty = houseName;
    }

    if (!houseDescription.IsEmpty())
    {
        house.Description = houseDescription;
    }

    if (!houseAdress.IsEmpty())
    {
        house.Adress = houseAdress;
    }

    db.SaveChanges();
    return Json(new StatusHolder(true));
}

/// <summary>
/// Добавляет холдера
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult AddHolder()
{
    try
    {
        string holderName = Request.Form.GetValues("holderName")[0].Trim();
        string holderCity = Request.Form.GetValues("holderCity")[0].Trim();
        Picture photo = new Picture();

        PhotoProcessor.CreateAndSavePicture(photo, Request, 200);

        CocktionContext db = new CocktionContext();
        db.Pictures.Add(photo);
        db.HouseHolders.Add(new HouseHolder(holderName,holderCity, photo));

        db.SaveChanges();
        return Json(new StatusHolder(true));
    }
    catch (Exception q)
    {
        return Json(new Q {Status = q.InnerException.InnerException.Message});
    }
```

```
        }
    }
}
```

1.1.2. Класс EditInterestsController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using CocktionMVC.Functions.DataProcessing;
using CocktionMVC.Models;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;

namespace CocktionMVC.Controllers.AdminControllers
{
    public class EditInterestsController : Controller
    {
        // GET: EditInterests
        [Authorize]
        public ActionResult Index()
        {
            if (User.Identity.Name == "darya-coo@cocktion.com" || User.Identity.Name ==
"lazarenko.ale@gmail.com")
            {
                CocktionContext db = new CocktionContext();
                return View(db.Interests.ToList());
            }
            return View("PageNotFound");
        }

        [HttpPost]
        [Authorize]
        public JsonResult AddInterest()
        {
            try
            {
                string name = Request.Form.GetValues("name")[0].Trim();

                Picture photo = new Picture();

                PhotoProcessor.CreateAndSavePicture(photo, Request, 200);

                CocktionContext db = new CocktionContext();
```

```
            db.Pictures.Add(photo);
            db.Interests.Add(new Interest(name, photo));

            db.SaveChanges();
            return Json(new StatusHolder(true));
        }
        catch (Exception q)
        {
            return Json(new EditHousesController.Q { Status =
q.InnerException.InnerException.Message });
        }

    }
  }
}
```

1.2.3. Класс AuctionController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.Identity;

namespace CocktionMVC.Controllers
{
  /// <summary>
  /// Данный контроллер содержит в себе только те методы,
  /// которые открывают непосредственно вэб страницы.
  ///
  /// Методы, которые используются для обработки RealTime
  /// находятся в контроллере AuctionRealTime
  /// </summary>
  public class AuctionController : Controller
  {
    /// <summary>
    /// Главная страничка, на которой отображаются
    /// все активные в данный момент аукционы.
    /// </summary>
    /// <returns></returns>
    [AllowAnonymous]
    public ActionResult Index()
    {
      //TODO Сделать страничку с бесконечной прокруткой
      //Конвертим время
      var controlTime = DateTimeManager.GetCurrentTime();
```

```
    CocktionContext db = new CocktionContext();

    //Получаем все активные в данный момент аукционы
    var auctions = (from x in db.Auctions
        where ((x.EndTime > controlTime) && (x.IsActive))
        select x).ToList<Auction>();

    return View(auctions);
}

/// <summary>
/// Выводит страничку, где нужно создать
/// аукцион
/// </summary>
/// <returns></returns>
[Authorize]
[HttpGet]
public ActionResult Create() //метод для создания находится в контроллере FileSaver
{
    return View();
} //end of create

/// <summary>
/// Выводит страничку, которая показывает
/// все созданные пользователем аукционы
/// </summary>
/// <returns></returns>
[Authorize]
public ActionResult ShowMyAuctions()
{
    var db = new CocktionContext();
    string userId = User.Identity.GetUserId();
    var user = db.AspNetUsers.Find(userId);

    List<Auction> auctions = user.HisAuctions.ToList();

    return View("MyAuctions", auctions);
}

/// <summary>
/// Выводит страничку, где показан
/// сам торг
/// </summary>
/// <param name="id">Айди аукциона, который надо показать</param>
[HttpGet]
public ActionResult CurrentAuction(int? id)
{
```

```
            try
            {
                var db = new CocktionContext();
                Auction auction = db.Auctions.Find(id);
                //TODO надо добавить рейтинг за вход на него пользователя

                //Если не удалось найти аукцион - кидаем эксепшн
                if (auction == null && id == null) throw new Exception();
                return View(auction);
            }
            catch
            {
                return View("PageNotFound");
            }
        }
    }
}
```

1.1.4. Класс AuctionRealTimeController

```
using System.Linq;
using System.Threading.Tasks;
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Models;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.ToteRelatedModels;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace CocktionMVC.Controllers
{
    /// <summary>
    /// Контроллер используется для отсылки клиентам
    /// данных, связанных с аукционом в реальном времени
    /// </summary>
    public class AuctionRealTimeController : Controller
    {
        /// <summary>
        /// Метод для добавления ставки в тотализаторе
        /// </summary>
        /// <returns></returns>
        [HttpPost]
        [Authorize]
        public async Task<JsonResult> AddToteRate()
        {
            //Получаем инморфмацию из формы о добавленной ставке
```

```
        int auctionId;
        int productId;
        int eggsAmount;
        RequestFormReader.ReadAddRateForm(Request, out auctionId, out productId, out eggsAmount);

        //Идентифицируем пользователя
        string userId = User.Identity.GetUserId();

        //Подключемся к базе данных
        CocktionContext db = new CocktionContext();

        var user = db.AspNetUsers.Find(userId);

        //Находим в базе этот аукцион
        Auction auction = db.Auctions.Find(auctionId);

        //Активируем тотализатор
        auction.AuctionToteBoard.IsActive = true;

        //TODO ДОБАВИТЬ ПРОВЕРКУ НА ВЫБОР ПОЛЬЗОВАТЕЛЕМ ТОВАРА-ПРОДАВЦА

        //Добавляем ставку пользователя
        //эта же штука и в сайгнал все посылает
        bool x = await auction.AuctionToteBoard.SetRateForUser(auctionId, user, eggsAmount,
productId, db);




        //Создаем результат добавления ставки
        ToteEggsInfo info = new ToteEggsInfo
        {
            Status = x,
            UsersAmountOfEggs = user.Eggs
        };
        return Json(info);
    }


    /// <summary>
    /// Записыват лидера в аукцион
    /// Добавляет эти данные в бд.
    /// </summary>
    /// <returns>Строку со статусом добавления</returns>
    [HttpPost]
```

```
public async Task<JsonResult> AddLider()
{
   //TODO: сделать эту хрень асинхронной
   string auctionId;
   string productId;
   RequestFormReader.ReadAddLiderForm(Request, out auctionId, out productId);
   try
   {
      var db = new CocktionContext();
      Auction auction = db.Auctions.Find(int.Parse(auctionId));
      Product product = db.Products.Find(int.Parse(productId));
      auction.LeadProduct = product;
      auction.WinnerChosen = true;
      await DbItemsAdder.SaveDb(db);
      AuctionHub.SetLider(productId, int.Parse(auctionId), product.Name);

      return Json(new StatusHolder(true, product.Name));
   }
   catch
   {
      return Json(new StatusHolder(false, ""));
   }
}

/// <summary>
/// Универсальный контейнер статуса
/// </summary>
class StatusHolder
{
   //Используется сразу строка для удобства


public StatusHolder(bool truthKeeper, string liderName)
      {
         Status = truthKeeper.ToString();
         LiderName = liderName;
      }
      public string Status { get; set; }
      public string LiderName { get; set; }
}

/// <summary>
/// Проверяет ставил ли пользователь свою ставку на этом
/// аукционе
/// </summary>
/// <returns>Джейсончик со статусом</returns>
[HttpPost]
public JsonResult CheckIfUserBidded()
```

```
        {
            BidChecker checker = new BidChecker();
            CocktionContext db = new CocktionContext();
            var userId = User.Identity.GetUserId();
            int auctionId = int.Parse(Request.Form.GetValues("auctionId")[0]);

            //Ищем товар с владельцем с данным айдишником.
            checker.HaveBid = db.Auctions.Find(auctionId).BidProducts.First(x => x.Owner.Id == userId) !=
null;
            return Json(checker);
        }

        /// <summary>
        /// Получает с клиента айдишник, по нему находит
        /// искомый продукт и возвращает информацию о нем.
        /// </summary>
        /// <returns>Объект, содержащий информацию о продукте.</returns>
        [HttpPost]
        public async Task<JsonResult> SendInfoAboutProduct()
        {
            string productId = Request.Form.GetValues("Id")[0];
            var db = new CocktionContext();
            Product product = await db.Products.FindAsync(int.Parse(productId));
            ProductMVCInfo info = new ProductMVCInfo
            {
                Name = product.Name,
                Description = product.Description,
                FileName = product.Photo.FileName,
                Category = product.Category
            };
            return Json(info);
        }

        /// <summary>
        /// Посылает результаты аукциона на клиент
        /// </summary>
        /// <returns>Объект JSON, который содержит
        /// в себе всю необходимую информацию.</returns>
        [HttpPost]
        public JsonResult SendAuctionResults()
        {
            //Получаем информацию с клиента
            int auctionId = int.Parse(Request.Form.GetValues("auctionId")[0]);

            //ищем аукцион, который завершился в базе
            var db = new CocktionContext();
            Auction auction = db.Auctions.Find(auctionId);
```

```
//раздаем результаты
if (auction.WinnerChosen == false)
{//если человек не выбрал победителя
   //TODO рандомно выбираем победителя
   if (User.Identity.IsAuthenticated)
   {//Если пользователь авторизован
      string userName = User.Identity.Name;
      if (userName == auction.Owner.UserName)
      {//если пользователь является создателем
         BidSeller owner = new BidSeller();
         owner.Name = userName;
         owner.Type = "Owner_undfnd";
         owner.Message = "Необходимо выбрать лидера!!!";
         return Json(owner);
      }
      //если любой другой
      BidSeller person = new BidSeller();
      person.Type = "Info";
      person.Message = "Создатель аукциона совсем не смог выбрать :(";
      return Json(person);
   }
   else
   {//если пользователь не авторизован
      BidSeller person = new BidSeller();
      person.Type = "Info";
      person.Message = "Создатель аукциона совсем не смог выбрать :(";

      return Json(person);
   }
}
else
{//если победитель выбран
   //Ищем продукт - победиель
   Product winProduct = auction.LeadProduct;
   if (User.Identity.IsAuthenticated)
   {//если пользователь авторизован
      string userName = User.Identity.Name;
      string userId = User.Identity.GetUserId();
      var currentUser = db.AspNetUsers.Find(userId);
      if (currentUser == auction.Owner)

      {//если пользователь - владелец
         BidSeller winner = new BidSeller();

         winner.Id = winProduct.Owner.Id;
         winner.Name = winProduct.Owner.UserName;
         string phone = winProduct.Owner.PhoneNumber == null
```

```
                    ? "Телефона нет ;("
                    : winProduct.Owner.PhoneNumber == "" ? "Телефона нет" :
winProduct.Owner.PhoneNumber;
                winner.Type = "Winner";
                winner.Message = "Аукцион закончен, вам необходимо связаться с победителем! " +
phone;
                return Json(winner);
            }
            else if (currentUser == winProduct.Owner)
            {//если пользователь - победитель
                BidSeller owner = new BidSeller();
                owner.Id = auction.Owner.Id;
                owner.Name = auction.Owner.UserName;
                string phone = auction.Owner.PhoneNumber;
                owner.Type = "Owner";

                //Получаем результаты тотализатора
                ToteResultsManager.GetToteResults(auction, userId, owner, phone, currentUser);

                return Json(owner);
            }
            else
            {//если пользователь - любой другой пользователь
                BidSeller looser = new BidSeller();
                looser.Name = userName;
                looser.Type = "Looser";

                //получаем результаты тотализатора
                ToteResultsManager.GetToteResults(auction, userId, looser, currentUser,
winProduct.Name);
                return Json(looser);
            }
        }
        else
        {//если пользователь неавторизован
            BidSeller person = new BidSeller();
            person.Type = "Info";
            person.Message = "Аукцион закончился, выйграл товар " + winProduct.Name;

            return Json(person);
        }
      }//end of else
    }//end of SendAuctionResults
  }//end of AuctionRealTime контроллера
}//конец неймспейса
```

1.1.5. Класс AuctionShowerController

```
using System.Linq;
using System.Web.Mvc;
using CocktionMVC.Models.DAL;

namespace CocktionMVC.Controllers
{
   public class AuctionShowerController : Controller
   {
      /// <summary>
      /// Показывает аукционы, прошедшие и закончившиеся
      /// в данном доме
      /// </summary>
      /// <param name="id">Айдишник дома</param>
      /// <returns>Страничку, показывающую все завершившиеся аукционы</returns>
      public ActionResult ShowOldAuctions(int? id)
      {
         CocktionContext db = new CocktionContext();

         //получаем все закончившиеся аукционы в доме
         var auctions = db.Houses.Find(id).Auctions.Where(x => x.IsActive == false).ToList();
         return View(auctions);
      }

      /// <summary>
      /// Показывает аукционы, проходящие в данный момент
      /// в данном аукционном доме
      /// </summary>
      /// <param name="id">Айдишник дома</param>
      /// <returns>Страничку, показывающую все активные аукционы</returns>
      public ActionResult ShowActiveAuctions(int? id)
      {
         CocktionContext db = new CocktionContext();

         //получаем все активные аукционы в доме
         var auctions = db.Houses.Find(id).Auctions.Where(x => x.IsActive).ToList();
         return View(auctions);
      }
   }
}
```

1.1.6. Класс SearchController

```
using System;
using System.Linq;
```

```
using System.Web.Mvc;
using CocktionMVC.Models.DAL;

namespace CocktionMVC.Controllers.DataControllers
{
   /// <summary>
   /// Содержит методы, используемые для поиска
   /// университетов, аукционов и т.д.
   /// </summary>
   public class SearchController : Controller
   {
      /// <summary>
      /// Контейнер для информации, которую поиск
      /// хочет рассказать пользователю.
      /// </summary>
      class SearchResults
      {
          public string[] Names { get; set; }
          public bool IsSearchEmpty { get; set; }

         public SearchResults(string[] names, bool isSearchEmpty)
         {
            Names = names;
            IsSearchEmpty = isSearchEmpty;
         }
      }

      /// <summary>
      /// Позволяет искать университет
      /// </summary>
      /// <returns>Ответ с информацией о том, нашел или нет.</returns>
      [HttpPost]
      [Authorize]
      public JsonResult SearchUniversity()
      {
         string searchLine = Request.Form.GetValues("university")[0];
         CocktionContext db = new CocktionContext();
         string[] names = (from x in db.HouseHolders
            where x.Name.Contains(searchLine)
            select x.Name).ToArray();
         if (names.Length == 0)
         {
            return Json(new SearchResults(null, true));
         }
         return Json(new SearchResults(names, false));
      }

      /// <summary>
```

```
/// Контейнер для выдачи поиском информации о
/// факультете.
/// </summary>
class FacultyList
{
    public string[] Names { get; set; }
    public int[] Ids { get; set; }

    public bool IsEmpty { get; set; }
    public FacultyList(string[] names, int[] ids, bool isEmpty)
    {
        Names = names;
        Ids = ids;
        IsEmpty = isEmpty;
    }

}

/// <summary>
/// Позволяет получить список факультетов, которые находятся
/// в данном вузе
/// </summary>
/// <returns>Джейсон с факультетами</returns>
[HttpPost]
[Authorize]
public JsonResult GetFacultyList()
{
    string university = Request.Form.GetValues("university")[0];
    CocktionContext db = new CocktionContext();
    House[] houses = (from x in db.Houses
        where x.Holder.Name == university
        select x).ToArray();
    if (houses.Length == 0)
    {
        return Json(new FacultyList(null, null, true));
    }
    else
    {
        string[] names = new string[houses.Length];
        int[] ids = new int[houses.Length];
        for (int i = 0; i < houses.Length; i++)
        {
            names[i] = houses[i].Faculty;
            ids[i] = houses[i].Id;
        }
        return Json(new FacultyList(names, ids, false));
    }
}
```

```
/// <summary>
/// Главная поисковая страница
/// </summary>
public ActionResult Index()
{
    return View();
}

/// <summary>
/// Ищет по всем-всем аукционам
/// </summary>
/// <returns>Аукционы, найденные по строке поиска</returns>
public JsonResult SearchEverywhere()
{
    string searchString = Request.Form.GetValues("searchString")[0];

    CocktionContext db = new CocktionContext();
    var auS = (from x in db.Auctions
        where x.IsActive & x.SellProduct.Name.Contains(searchString)
        select x).ToArray();
    Info[] auctions = new Info[auS.Length];
    int i = 0;
    Array.ForEach(auS, x =>
    {
        auctions[i++] = new Info(x.SellProduct.Name, x.Id, x.SellProduct.Photo.FileName,
            x.SellProduct.Description);
    });
    if (auctions.Length == 0)
    {
        return Json(new GlobalSearchResults(null, true));
    }
    return Json(new GlobalSearchResults(auctions, false));
}

/// <summary>
/// Контейнер для информации об аукционе,
/// который находится поиском
/// </summary>
public class Info
{
    public Info(string name, int id, string photo, string description)
    {
        Name = name;
        Id = id;
        Photo = photo;
        Description = description;
    }
```

```
        public string Name { get; set; }
        public int Id { get; set; }
        public string Photo { get; set; }
        public string Description { get; set; }
    }

    /// <summary>
    /// Контейнер для результатов поиска по всем аукционам
    /// </summary>
    public class GlobalSearchResults
    {
        public GlobalSearchResults(Info[] auctions, bool isEmpty)
        {
            Auctions = auctions;
            IsEmpty = isEmpty;
        }
        public Info[] Auctions { get; set; }
        public bool IsEmpty { get; set; }
    }
  }
}
```

1.1.7. Класс UsersController

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.ViewModels;
using Microsoft.AspNet.Identity;

// ReSharper disable once CheckNamespace
namespace CocktionMVC.Controllers
{
  public class UsersController : Controller
  {
    /// <summary>
    /// Показывает страницу со всеми пользвателями
    /// на кокшне
    /// </summary>
    /// <returns></returns>
    public ActionResult Index()
    {
      if (User.Identity.IsAuthenticated)
      {
        CocktionContext db = new CocktionContext();
        string userId = User.Identity.GetUserId();
        List<AspNetUser> users = (from x in db.AspNetUsers
```

```
                where x.Id != userId
                select x).ToList();
            return View(users);
        }
        else
        {
            CocktionContext db = new CocktionContext();
            List<AspNetUser> users = db.AspNetUsers.ToList();
            return View(users);
        }
    }


    /// <summary>
    /// Показывает профиль конкретного пользователя
    /// </summary>
    /// <param name="id">Айдишник пользователя</param>
    public ActionResult GetUser(string id)
    {
        CocktionContext db = new CocktionContext();
        //находим пользователя
        var user = db.AspNetUsers.Find(id);

        return View(user);
    }
  }
}
```

1.1.8. Класс BidAuctionCreatorController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Mvc;
using System.Web.Mvc.Html;
using CocktionMVC.Functions;
using CocktionMVC.Functions.DataProcessing;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.Identity;
using CocktionMVC.Models.Hubs;

namespace CocktionMVC.Controllers
{
  public class BidAuctionCreatorController : Controller
  {
    /// <summary>
```

```
/// Создает аукцион и записывает его в базу данных
/// </summary>
[Authorize]
[HttpPost]
public async Task<JsonResult> CreateAuction()
{
    //Получаем информацию из формы
    string name, description, category, timeBound, housesIds;
    RequestFormReader.ReadCreateAuctionForm(Request, out name, out description,
        out category, out timeBound, out housesIds);

    //получаем информацию о пользователе
    string userId = User.Identity.GetUserId();

    //подключаемся к базе данных
    CocktionContext db = new CocktionContext();
    var user = db.AspNetUsers.Find(userId);

    //создаем продуктик, ха
    Product product = new Product(name, description, category, true, user);

    //инициализация аукциона
    Auction auction = new Auction(true, product, false, new ToteBoard(), user);

    //TODO сделать динамический выбор дома
    if (housesIds[0] != '?')
    {
        var houses = (from x in db.Houses
            where x.Holder.Name == housesIds
            select x).ToList();
        houses.ForEach(x =>
        {
            auction.Houses.Add(x);
            RatingManager.IncreaseRating(x, "auctionAdded");
        });
    }
    else
    {
        int[] ids = HouseSerializer.TakeHouseIdsFromString(housesIds);
        Array.ForEach(ids, x =>
        {
            var house = db.Houses.Find(x);
            auction.Houses.Add(house);
            RatingManager.IncreaseRating(house, "auctionAdded");
        });
    }

    //задаем время окончания и начала аукциона
```

```
        DateTimeManager.SetAuctionStartAndEndTime(auction, timeBound);

        //добавление фотографии для товара
        Picture photo = new Picture();
        PhotoProcessor.CreateAndSavePicture(photo, Request, 90);
        product.Photo = photo;


        auction.Rating = (int)(user.Rating * 0.4);

        //добавляем пользователю немного рейтинга
        RatingManager.IncreaseRating(user, "userMadeAuction");


        await DbItemsAdder.AddAuctionProductPhotoAsync(db, auction, product, photo, user);

        AuctionListHub.UpdateList(product.Name, product.Description,
product.Category,auction.EndTime,
            photo.FileName, auction.Id);

        //возвращаем статус
        return Json(new IdContainer(auction.Id));
    }



    class IdContainer
    {
        public IdContainer(int id)
        {
            Id = id;
        }
        public int Id { get; set; }
    }

    /// <summary>
    /// Добавляет довесок к товару
    /// </summary>
    [HttpPost]
    public async Task AddExtraBid()
    {
        //Adding new product to the DB
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        //Инициализируем и добавляем фоточку
        Picture photo = new Picture();
        PhotoProcessor.CreateAndSavePicture(photo, Request, 60);
```

```
    //обработка данных из формы
    string bidName, bidDescription, bidCategory, auctionId;
    RequestFormReader.ReadAddProductBetForm(Request, out bidName, out auctionId,
        out bidCategory, out bidDescription);

    int id = int.Parse(auctionId);

    //Создаем товар для базы данных
    Product product = new Product(bidName, bidDescription, bidCategory, false, user);

    product.Photo = photo;

    var auction = db.Auctions.Find(id);
    auction.BidProducts.Add(product);

    //Выбираем все кластер, где пользователь == данный
    //TODO эта херня работать не будет
    BidCluster cluster = (from x in auction.UsersBids
                          where x.UserId == user.Id
                          select x).First();
    cluster.Products.Add(product);

    //Добавляем продукт в базу данных
    await DbItemsAdder.AddProduct(db, product, photo, cluster);

    //Найдем вхождние товара в кластере первое
    int firstProductId = cluster.Products.First().Id;
    foreach (var bidProduct in cluster.Products)
    {
        int bidProductId = bidProduct.Id;
        if (bidProductId != firstProductId)
        {
            //Добавляем на всех клиентах довесочек
            AuctionHub.AddExtraNodeToClients(bidProduct.Name, bidProduct.Photo.FileName,
                                id, firstProductId, bidProductId);
        }
    }
}//AddExtraBid

/// <summary>
/// Метод добавления ставки на аукцион (товара)
/// </summary>
[HttpPost]
[Authorize]
public async Task AddProductBet()
{
    //обработка данных из формы
```

```
        string bidName, auctionId, bidCategory, bidDescription;
        RequestFormReader.ReadAddProductBetForm(Request, out bidName, out auctionId,
            out bidCategory, out bidDescription);

        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        //Инициализируем фоточку
        Picture photo = new Picture();
        PhotoProcessor.CreateAndSavePicture(photo, Request, 60);

        //добавляем продукт
        //Коннектимся к базе
        int id = int.Parse(auctionId);
        Auction auction = db.Auctions.Find(id);

        //добавляем рейтинг пользователю и аукциону
        RatingManager.IncreaseRating(user, "userPlacedBet");
        RatingManager.IncreaseRating(db.Auctions.Find(id), user, "userBeted");

        //Создаем товар для базы данных
        Product product = new Product(bidName, bidDescription, bidCategory, false, photo, user,
            auction);

        //находи кластер
        BidCluster bidCluster = new BidCluster(User.Identity.GetUserId(), auction);
        bidCluster.Products.Add(product);
        auction.BidProducts.Add(product);
        user.HisProducts.Add(product);

        //сохраняем все в базу
        await DbItemsAdder.AddProduct(db, product, photo, bidCluster);

        //добавляем нодики на клиенты
        AuctionHub.AddNodesToClients(bidName, photo.FileName, id, product.Id);
    }//end of AddProductBet
  }
}
```

1.1.9. Класс SubscriptionController

```
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using Microsoft.AspNet.Identity;
```

```
namespace CocktionMVC.Controllers.FunctionalControllers
{
    public class SubscriptionController : Controller
    {
        /// <summary>
        /// Позволяет отписаться от дома
        /// </summary>
        /// <returns>Стандартный ответ</returns>
        [HttpPost]
        [Authorize]
        public JsonResult UsubscribeFromHouse()
        {
            var strings = Request.Form.GetValues("houseId");
            if (strings != null)
            {
                int houseId = int.Parse(strings[0]);

                CocktionContext db = new CocktionContext();

                var house = db.Houses.Find(houseId);
                var user = db.AspNetUsers.Find(User.Identity.GetUserId());

                user.SubHouses.Remove(house);
                house.Inhabitants.Remove(user);

                db.SaveChanges();

                return Json(new StatusHolder(true));
            }
            return Json(new StatusHolder(false));
        }

        /// <summary>
        /// Позволяет подписаться на дом
        /// </summary>
        /// <returns>Стандартный ответ</returns>
        [HttpPost]
        [Authorize]
        public JsonResult SubscribeOnHouse()
        {
            var strings = Request.Form.GetValues("houseId");
            if (strings != null)
            {
                int houseId = int.Parse(strings[0]);
                CocktionContext db = new CocktionContext();

                var house = db.Houses.Find(houseId);
```

```
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        user.SubHouses.Add(house);
        house.Inhabitants.Add(user);

        if (user.SocietyName == null)
            user.SocietyName = house.Holder.Name;

        if (user.City == null)
            user.City = house.Holder.City;

        //добавляем дому немного рейтинга
        RatingManager.IncreaseRating(house, "subscriberAdded");

        db.SaveChanges();

        return Json(new StatusHolder(true));
    }
    return Json(new StatusHolder(false));
}

/// <summary>
/// Позволяет проверить, подписан ли пользователь
/// на дом
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult CheckHouseSubscription()
{
    var strings = Request.Form.GetValues("modelId");
    if (strings != null)
    {
        int houseId = int.Parse(strings[0]);

        CocktionContext db = new CocktionContext();

        var house = db.Houses.Find(houseId);
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        if (user.SubHouses.Contains(house))
        {
            return Json(new StatusHolder(true));
        }
        return Json(new StatusHolder(false));
    }
    return Json(new StatusHolder(false));
}
```

```
/// <summary>
/// Пощзволяет отписаться от пользователя
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult UnsubscribeFromUser()
{

    var userIdContainer = Request.Form.GetValues("userId");
    if (userIdContainer != null)
    {
        string userId = userIdContainer[0];
        var currentUserId = User.Identity.GetUserId();

        CocktionContext db = new CocktionContext();

        var userToAdd = db.AspNetUsers.Find(userId);
        var currentUser = db.AspNetUsers.Find(currentUserId);

        currentUser.Friends.Remove(userToAdd);

        db.SaveChanges();

        return Json(new StatusHolder(true));
    }
    return Json(new StatusHolder(false));
}

/// <summary>
/// Позволяет подписаться на пользователя
/// </summary>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public JsonResult SubscribeOnUser()
{
    var userIdContainer = Request.Form.GetValues("userId");
    if (userIdContainer != null)
    {
        string userId = userIdContainer[0];
        var currentUserId = User.Identity.GetUserId();

        CocktionContext db = new CocktionContext();

        var userToAdd = db.AspNetUsers.Find(userId);
        var currentUser = db.AspNetUsers.Find(currentUserId);
```

```
            currentUser.Friends.Add(userToAdd);

            //добавляем рейтиг пользовтелю
            RatingManager.IncreaseRating(currentUser, "userGotSubscriber");

            db.SaveChanges();
            return Json(new StatusHolder(true));
        }
        return Json(new StatusHolder(false));
    }

    /// <summary>
    /// Позволяет провертить, подписан ли данный пользователь на того,
    /// который прислан в поле юсерАйди
    /// </summary>
    /// <returns>Стандартный ответ</returns>
    [HttpPost]
    [Authorize]
    public JsonResult CheckUsersSubscription()
    {
        var strings = Request.Form.GetValues("userId");
        if (strings != null)
        {
            string userId = strings[0];

            CocktionContext db = new CocktionContext();

            var user = db.AspNetUsers.Find(userId);
            var currentUser = db.AspNetUsers.Find(User.Identity.GetUserId());

            if (currentUser.Friends.Contains(user))
            {
                return Json(new StatusHolder(true));
            }
            return Json(new StatusHolder(false));
        }
        return Json(new StatusHolder(false));
    }
  }
}
```

1.1.10. Класс AuctionHouseController

```
using System.Collections.Generic;
using System.Data.Entity.Infrastructure.Interception;
```

```
using System.Linq;
using System.Threading.Tasks;
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;

// ReSharper disable once CheckNamespace
namespace CocktionMVC.Controllers
{
    /// <summary>
    /// Отвечает за взаимодействие пользователя с домами
    /// </summary>
    public class AuctionHouseController : Controller
    {
        /// <summary>
        /// Выводит пользователя на страницу со списком всех холдеров
        /// </summary>
        /// <returns>страницу со списком</returns>
        public ActionResult Index()
        {
            //Мб добавим потом тут какой-то код для связи с БД
            CocktionContext db = new CocktionContext();
            var holders = db.HouseHolders.ToList();
            return View(holders);
        }

        /// <summary>
        /// Показывает пользователю страничку со всеми домами конкретного
        /// университета
        /// </summary>
        /// <param name="id">Университет, который надо добавить</param>
        /// <returns>страницу со списком</returns>
        public ActionResult GetUniversityHouses(int id)
        {
            //подключаемся к базе
            CocktionContext db = new CocktionContext();

            //Выбираем все дома, относящиеся к данному ВУЗу
            var holder = db.HouseHolders.Find(id);
            var houses = holder.Houses.ToList();

            return View(houses);
        }

        /// <summary>
        /// Показывает даныйй дом
        /// </summary>
```

```
        /// <param name="id">айдишник дома</param>
        /// <returns>страницу с домом, соотв. данному айдишнику</returns>
        public ActionResult GetCurrentAuctionHouse(int id)
        {
            CocktionContext db = new CocktionContext();
            var house = db.Houses.Find(id);
            return View(house);
        }

        /// <summary>
        /// Добавляет комментарий на форум
        /// </summary>
        /// <returns>Джейсончик со статусом и именем автора</returns>
        public async Task<JsonResult> AddComment()
        {
            try
            {
                var strings = Request.Form.GetValues("message"); //получаем значения поля message в
форме
                if (strings != null)
                {//если значение в порядке
                    string message = strings[0].Trim();
                    var values = Request.Form.GetValues("houseId"); //получаем значение поля houseId
                    if (values != null)
                    {//если все ок
                        int houseId = int.Parse(values[0].Trim());

                        //Подключаемся к базе данных
                        CocktionContext db = new CocktionContext();

                        //находим дом
                        var house = db.Houses.Find(houseId);

                        //Создаем пост и добавляем в дом
                        ForumPost post = new ForumPost(message, User.Identity.Name) {HostHouse = house};
                        house.Posts.Add(post);

                        //сохранение значений в базу
                        await DbItemsAdder.SaveDb(db);

                        //возвращаем успех
                        return Json(new ForumRespond("Success", User.Identity.Name));
                    }
                }
                //если в какой-то из форм не было значения - ничего не добавляем
                return Json(new ForumRespond("Failed"));
            }
            catch
```

```
            {//если произошла какая-то ошибка
                return Json(new ForumRespond("Failed"));
            }
        }

    }
}
```

1.1.11. Класс AccountController

```
using System;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using Recaptcha.Web;
using Recaptcha.Web.Mvc;

namespace CocktionMVC.Controllers
{
    [Authorize]
    public class AccountController : Controller
    {
        private ApplicationUserManager _userManager;



        public AccountController()
        {
        }

        public AccountController(ApplicationUserManager userManager, ApplicationSignInManager
signInManager )
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public ApplicationUserManager UserManager
        {
            get
            {
```

```
            return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
        }
        private set
        {
            _userManager = value;
        }
    }

    //
    // GET: /Account/Login
    [AllowAnonymous]
    public ActionResult Login(string returnUrl)
    {
        ViewBag.ReturnUrl = returnUrl;
        return View();
    }

    private ApplicationSignInManager _signInManager;

    public ApplicationSignInManager SignInManager
    {
        get
        {
            return _signInManager ?? HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
        }
        private set { _signInManager = value; }
    }

    //
    // POST: /Account/Login
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }

        // This doesn't count login failures towards account lockout
        // To enable password failures to trigger account lockout, change to shouldLockout: true
        var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
        switch (result)
        {
            case SignInStatus.Success:
```

```
                return RedirectToLocal(returnUrl);
            case SignInStatus.LockedOut:
                return View("Lockout");
            case SignInStatus.RequiresVerification:
                return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe =
model.RememberMe });
            case SignInStatus.Failure:
            default:
                ModelState.AddModelError("", "Пользователь с таким именем еще не
зарегистрировался.");
                return View(model);
        }
    }

    //
    // GET: /Account/VerifyCode
    [AllowAnonymous]
    public async Task<ActionResult> VerifyCode(string provider, string returnUrl, bool rememberMe)
    {
        // Require that the user has already logged in via username/password or external login
        if (!await SignInManager.HasBeenVerifiedAsync())
        {
            return View("Error");
        }
        var user = await UserManager.FindByIdAsync(await SignInManager.GetVerifiedUserIdAsync());
        if (user != null)
        {
            var code = await UserManager.GenerateTwoFactorTokenAsync(user.Id, provider);
        }
        return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl,
RememberMe = rememberMe });
    }

    //
    // POST: /Account/VerifyCode
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }

        // The following code protects for brute force attacks against the two factor codes.
        // If a user enters incorrect codes for a specified amount of time then the user account
        // will be locked out for a specified amount of time.
```

```
        // You can configure the account lockout settings in IdentityConfig
        var result = await SignInManager.TwoFactorSignInAsync(model.Provider, model.Code,
isPersistent:  model.RememberMe, rememberBrowser: model.RememberBrowser);
        switch (result)
        {
            case SignInStatus.Success:
                return RedirectToLocal(model.ReturnUrl);
            case SignInStatus.LockedOut:
                return View("Lockout");
            case SignInStatus.Failure:
            default:
                ModelState.AddModelError("", "Invalid code.");
                return View(model);
        }
    }

    //
    // GET: /Account/Register
    [AllowAnonymous]

    public ActionResult Register()
    {
        return View();
    }

    //
    // POST: /Account/Register
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]

    public async Task<ActionResult> Register(RegisterViewModel model)
    {
        RecaptchaVerificationHelper recaptchaHelper = this.GetRecaptchaVerificationHelper();

        if (String.IsNullOrEmpty(recaptchaHelper.Response))
        {
            ModelState.AddModelError("", "В капчу надо хоть что-то ввести...");
            return View(model);
        }

        RecaptchaVerificationResult recaptchaResult = await
recaptchaHelper.VerifyRecaptchaResponseTaskAsync();
        if (recaptchaResult != RecaptchaVerificationResult.Success)
        {
            ModelState.AddModelError("", "Не угадали... Может быть вы робот?");
            return View(model);
        }
```

```
        else
        {
            if (ModelState.IsValid)
            {
                var user = new ApplicationUser
                {
                    UserName = model.Email,
                    Email = model.Email,
                    Eggs = 1000,
                    Rating = 1000
                };
                var result = await UserManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    //await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

                    // For more information on how to enable account confirmation and password reset please visit http://go.microsoft.com/fwlink/?LinkID=320771
                    // Send an email with this link
                    string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
                    var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);
                    //await UserManager.SendEmailAsync(user.Id, "Confirm your account", "Please confirm your account by clicking <a href=\"" + callbackUrl + "\">here</a>");

                    string message = EmailSender.VERIFY_EMAIL_MESSAGE+"\n" + "<a href=\"" + callbackUrl + "\">here</a>";
                    EmailSender.SendEmail(message, EmailSender.VERIFY_EMAIL_SUBJECT, user.UserName);
                    return RedirectToAction("VerifyEmail", "Home");
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);

        }
    }

    //
    // GET: /Account/ConfirmEmail
    [AllowAnonymous]
    public async Task<ActionResult> ConfirmEmail(string userId, string code)
    {
        if (userId == null || code == null)
        {
            return View("Error");
```

```
        }
        var result = await UserManager.ConfirmEmailAsync(userId, code);
        return View(result.Succeeded ? "ConfirmEmail" : "Error");
    }

    //
    // GET: /Account/ForgotPassword
    [AllowAnonymous]
    public ActionResult ForgotPassword()
    {
        return View();
    }

    //
    // POST: /Account/ForgotPassword
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
    {
        if (ModelState.IsValid)
        {
            var user = await UserManager.FindByNameAsync(model.Email);
            if (user == null || !(await UserManager.IsEmailConfirmedAsync(user.Id)))
            {
                // Don't reveal that the user does not exist or is not confirmed
                string whatsWithUser;
                whatsWithUser = user == null ? "Проверьте, пожалуйста, свой адрес :)" : "Вы не
подтвердили свою электро-почту ;(";
                return View("ForgotPasswordConfirmation", "_Layout", whatsWithUser);
            }

            // For more information on how to enable account confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkID=320771
            // Send an email with this link
            string code = await UserManager.GeneratePasswordResetTokenAsync(user.Id);
            var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code =
code }, protocol: Request.Url.Scheme);


            // await UserManager.SendEmailAsync(user.Id, "Reset Password", "Please reset your password
by clicking <a href=\"" + callbackUrl + "\">here</a>");
            string message = EmailSender.RESET_PASSWORD_MESSAGE + " " + " <a href=\"" +
callbackUrl + "\">here</a>";
            EmailSender.SendEmail(message, EmailSender.VERIFY_EMAIL_SUBJECT, model.Email);
            return RedirectToAction("ForgotPasswordConfirmation", "Account");
        }
```

```
            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Account/ForgotPasswordConfirmation
        [AllowAnonymous]
        public ActionResult ForgotPasswordConfirmation()
        {
            return View("ForgotPasswordConfirmation", "_Layout",  "Необходимо пройти по ссылке в
вашем сообщении :)");
        }

        //
        // GET: /Account/ResetPassword
        [AllowAnonymous]
        public ActionResult ResetPassword(string code)
        {
            return code == null ? View("Error") : View();
        }

        //
        // POST: /Account/ResetPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var user = await UserManager.FindByNameAsync(model.Email);
            if (user == null)
            {
                // Don't reveal that the user does not exist
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            var result = await UserManager.ResetPasswordAsync(user.Id, model.Code, model.Password);
            if (result.Succeeded)
            {
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            AddErrors(result);
            return View();
        }

        //
```

```
        // GET: /Account/ResetPasswordConfirmation
        [AllowAnonymous]
        public ActionResult ResetPasswordConfirmation()
        {
            return View();
        }

        //
        // POST: /Account/ExternalLogin
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            // Request a redirect to the external login provider
            return new ChallengeResult(provider, Url.Action("ExternalLoginCallback", "Account", new
{ ReturnUrl = returnUrl }));
        }

        //
        // GET: /Account/SendCode
        [AllowAnonymous]
        public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)
        {
            var userId = await SignInManager.GetVerifiedUserIdAsync();
            if (userId == null)
            {
                return View("Error");
            }
            var userFactors = await UserManager.GetValidTwoFactorProvidersAsync(userId);
            var factorOptions = userFactors.Select(purpose => new SelectListItem { Text = purpose, Value =
purpose }).ToList();
            return View(new SendCodeViewModel { Providers = factorOptions, ReturnUrl = returnUrl,
RememberMe = rememberMe });
        }

        //
        // POST: /Account/SendCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SendCode(SendCodeViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View();
            }
```

```
            // Generate the token and send it
            if (!await SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider))
            {
                return View("Error");
            }
            return RedirectToAction("VerifyCode", new { Provider = model.SelectedProvider, ReturnUrl =
model.ReturnUrl, RememberMe = model.RememberMe });
        }

        //
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null)
            {
                return RedirectToAction("Login");
            }

            // Sign in the user with this external login provider if the user already has a login
            var result = await SignInManager.ExternalSignInAsync(loginInfo, isPersistent: false);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe =
false });
                case SignInStatus.Failure:
                default:
                    // If the user does not have an account, then prompt the user to create an account
                    ViewBag.ReturnUrl = returnUrl;
                    ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                    return View("ExternalLoginConfirmation", new ExternalLoginConfirmationViewModel
{ Email = loginInfo.Email });
            }
        }

        //
        // POST: /Account/ExternalLoginConfirmation
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string returnUrl)
```

```
{
    if (User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Index", "Manage");
    }

    if (ModelState.IsValid)
    {
        // Get the information about the user from the external login provider
        var info = await AuthenticationManager.GetExternalLoginInfoAsync();
        if (info == null)
        {
            return View("ExternalLoginFailure");
        }
        var user = new ApplicationUser
        {
            UserName = model.Email,
            Email = model.Email,
            PhoneNumber = model.PhoneNumber,
            UserRealName = model.UserRealName,
            UserRealSurname = model.UserRealSurname
        };
        var result = await UserManager.CreateAsync(user);
        if (result.Succeeded)
        {
            result = await UserManager.AddLoginAsync(user.Id, info.Login);
            if (result.Succeeded)
            {
                await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                return RedirectToLocal(returnUrl);
            }
        }
        AddErrors(result);
    }

    ViewBag.ReturnUrl = returnUrl;
    return View(model);
}

//
// POST: /Account/LogOff
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult LogOff()
{
    AuthenticationManager.SignOut();
    return RedirectToAction("Index", "Home");
}
```

```
//
// GET: /Account/ExternalLoginFailure
[AllowAnonymous]
public ActionResult ExternalLoginFailure()
{
    return View();
}

#region Helpers
// Used for XSRF protection when adding external logins
private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager
{
    get
    {
        return HttpContext.GetOwinContext().Authentication;
    }
}

private void AddErrors(IdentityResult result)
{
    foreach (var error in result.Errors)
    {
        ModelState.AddModelError("", error);
    }
}

private ActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    return RedirectToAction("Index", "Home");
}

internal class ChallengeResult : HttpUnauthorizedResult
{
    public ChallengeResult(string provider, string redirectUri)
        : this(provider, redirectUri, null)
    {
    }

    public ChallengeResult(string provider, string redirectUri, string userId)
    {
        LoginProvider = provider;
```

```
            RedirectUri = redirectUri;
            UserId = userId;
        }

        public string LoginProvider { get; set; }
        public string RedirectUri { get; set; }
        public string UserId { get; set; }

        public override void ExecuteResult(ControllerContext context)
        {
            var properties = new AuthenticationProperties { RedirectUri = RedirectUri };
            if (UserId != null)
            {
                properties.Dictionary[XsrfKey] = UserId;
            }
            context.HttpContext.GetOwinContext().Authentication.Challenge(properties, LoginProvider);
        }
    }
    #endregion
  }
}
```

1.1.12. Класс HelpController

```
using System.Web.Mvc;

namespace CocktionMVC.Controllers
{
  public class HelpController : Controller
  {
    /// <summary>
    /// Метод для отображения страницы с ошибкой
    /// Если произошло что-то очень плохое.
    /// </summary>
    /// <returns></returns>
    public ActionResult Index()
    {
      return View();
    }
  }
}
```

1.1.13. Класс HomeController

```
using System.Web.Mvc;
```

```
// ReSharper disable once CheckNamespace
namespace CocktionMVC.Controllers
{
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         if (User.Identity.IsAuthenticated)
         {
            return View();
         }
         return View("Index2");

      }

      public ActionResult About()
      {
         ViewBag.Message = "Аукционы, в которых вы еще не участвовали!";

         return View();
      }

      public ActionResult Contact()
      {
         ViewBag.Message = "Наши контакты.";

         return View();
      }

      public ActionResult VerifyEmail()
      {
         return View();
      }
   }
}
```

1.1.14. Класс ManageController

```
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using CocktionMVC.Models;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
```

```
namespace CocktionMVC.Controllers
{
   [Authorize]
   public class ManageController : Controller
   {
      public ManageController()
      {
      }

      public ManageController(ApplicationUserManager userManager)
      {
         UserManager = userManager;
      }

      private ApplicationUserManager _userManager;
      public ApplicationUserManager UserManager
      {
         get
         {
            return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
         }
         private set
         {
            _userManager = value;
         }
      }

      //
      // GET: /Manage/Index
      public async Task<ActionResult> Index(ManageMessageId? message)
      {
         ViewBag.StatusMessage =
            message == ManageMessageId.ChangePasswordSuccess ? "Your password has been changed."
            : message == ManageMessageId.SetPasswordSuccess ? "Your password has been set."
            : message == ManageMessageId.SetTwoFactorSuccess ? "Your two-factor authentication
provider has been set."
            : message == ManageMessageId.Error ? "An error has occurred."
            : message == ManageMessageId.AddPhoneSuccess ? "Your phone number was added."
            : message == ManageMessageId.RemovePhoneSuccess ? "Your phone number was removed."
            : "";

         var model = new IndexViewModel
         {
            HasPassword = HasPassword(),
            PhoneNumber = await UserManager.GetPhoneNumberAsync(User.Identity.GetUserId()),
            TwoFactor = await UserManager.GetTwoFactorEnabledAsync(User.Identity.GetUserId()),
```

```
            Logins = await UserManager.GetLoginsAsync(User.Identity.GetUserId()),
            BrowserRemembered = await
AuthenticationManager.TwoFactorBrowserRememberedAsync(User.Identity.GetUserId())
        };
        return View(model);
    }

    //
    // GET: /Manage/RemoveLogin
    public ActionResult RemoveLogin()
    {
        var linkedAccounts = UserManager.GetLogins(User.Identity.GetUserId());
        ViewBag.ShowRemoveButton = HasPassword() || linkedAccounts.Count > 1;
        return View(linkedAccounts);
    }

    //
    // POST: /Manage/RemoveLogin
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)
    {
        ManageMessageId? message;
        var result = await UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new
UserLoginInfo(loginProvider, providerKey));
        if (result.Succeeded)
        {
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInAsync(user, isPersistent: false);
            }
            message = ManageMessageId.RemoveLoginSuccess;
        }
        else
        {
            message = ManageMessageId.Error;
        }
        return RedirectToAction("ManageLogins", new { Message = message });
    }

    //
    // GET: /Manage/AddPhoneNumber
    public ActionResult AddPhoneNumber()
    {
        return View();
    }
```

```
        //
        // POST: /Manage/AddPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            // Generate the token and send it
            var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), model.Number);
            if (UserManager.SmsService != null)
            {
                var message = new IdentityMessage
                {
                    Destination = model.Number,
                    Body = "Your security code is: " + code
                };
                await UserManager.SmsService.SendAsync(message);
            }
            return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber = model.Number });
        }

        //
        // POST: /Manage/EnableTwoFactorAuthentication
        [HttpPost]
        public async Task<ActionResult> EnableTwoFactorAuthentication()
        {
            await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInAsync(user, isPersistent: false);
            }
            return RedirectToAction("Index", "Manage");
        }

        //
        // POST: /Manage/DisableTwoFactorAuthentication
        [HttpPost]
        public async Task<ActionResult> DisableTwoFactorAuthentication()
        {
            await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
```

```
            await SignInAsync(user, isPersistent: false);
        }
        return RedirectToAction("Index", "Manage");
    }

    //
    // GET: /Manage/VerifyPhoneNumber
    public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)
    {
        var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(), phoneNumber);
        // Send an SMS through the SMS provider to verify the phone number
        return phoneNumber == null ? View("Error") : View(new VerifyPhoneNumberViewModel
{ PhoneNumber = phoneNumber });
    }

    //
    // POST: /Manage/VerifyPhoneNumber
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel model)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }
        var result = await UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(),
model.PhoneNumber, model.Code);
        if (result.Succeeded)
        {
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInAsync(user, isPersistent: false);
            }
            return RedirectToAction("Index", new { Message = ManageMessageId.AddPhoneSuccess });
        }
        // If we got this far, something failed, redisplay form
        ModelState.AddModelError("", "Failed to verify phone");
        return View(model);
    }

    //
    // GET: /Manage/RemovePhoneNumber
    public async Task<ActionResult> RemovePhoneNumber()
    {
        var result = await UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);
        if (!result.Succeeded)
```

```
        {
            return RedirectToAction("Index", new { Message = ManageMessageId.Error });
        }
        var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
        if (user != null)
        {
            await SignInAsync(user, isPersistent: false);
        }
        return RedirectToAction("Index", new { Message = ManageMessageId.RemovePhoneSuccess });
    }

    //
    // GET: /Manage/ChangePassword
    public ActionResult ChangePassword()
    {
        return View();
    }

    //
    // POST: /Manage/ChangePassword
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)
    {
        if (!ModelState.IsValid)
        {
            return View(model);
        }
        var result = await UserManager.ChangePasswordAsync(User.Identity.GetUserId(),
model.OldPassword, model.NewPassword);
        if (result.Succeeded)
        {
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInAsync(user, isPersistent: false);
            }
            return RedirectToAction("Index", new { Message =
ManageMessageId.ChangePasswordSuccess });
        }
        AddErrors(result);
        return View(model);
    }

    //
    // GET: /Manage/SetPassword
    public ActionResult SetPassword()
    {
```

```
            return View();
        }

        //
        // POST: /Manage/SetPassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SetPassword(SetPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var result = await UserManager.AddPasswordAsync(User.Identity.GetUserId(),
model.NewPassword);
                if (result.Succeeded)
                {
                    var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                    if (user != null)
                    {
                        await SignInAsync(user, isPersistent: false);
                    }
                    return RedirectToAction("Index", new { Message =
ManageMessageId.SetPasswordSuccess });
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Manage/ManageLogins
        public async Task<ActionResult> ManageLogins(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."
                : message == ManageMessageId.Error ? "An error has occurred."
                : "";
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user == null)
            {
                return View("Error");
            }
            var userLogins = await UserManager.GetLoginsAsync(User.Identity.GetUserId());
            var otherLogins = AuthenticationManager.GetExternalAuthenticationTypes().Where(auth =>
userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();
            ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;
            return View(new ManageLoginsViewModel
```

```
            {
                CurrentLogins = userLogins,
                OtherLogins = otherLogins
            });
        }

        //
        // POST: /Manage/LinkLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LinkLogin(string provider)
        {
            // Request a redirect to the external login provider to link a login for the current user
            return new AccountController.ChallengeResult(provider, Url.Action("LinkLoginCallback",
"Manage"), User.Identity.GetUserId());
        }

        //
        // GET: /Manage/LinkLoginCallback
        public async Task<ActionResult> LinkLoginCallback()
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
            if (loginInfo == null)
            {
                return RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
            }
            var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
            return result.Succeeded ? RedirectToAction("ManageLogins") :
RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
        }

#region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private async Task SignInAsync(ApplicationUser user, bool isPersistent)
        {
            AuthenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie,
DefaultAuthenticationTypes.TwoFactorCookie);
```

```
            AuthenticationManager.SignIn(new AuthenticationProperties { IsPersistent = isPersistent }, await
user.GenerateUserIdentityAsync(UserManager));
        }

        private void AddErrors(IdentityResult result)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
        }

        private bool HasPassword()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PasswordHash != null;
            }
            return false;
        }

        private bool HasPhoneNumber()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PhoneNumber != null;
            }
            return false;
        }

        public enum ManageMessageId
        {
            AddPhoneSuccess,
            ChangePasswordSuccess,
            SetTwoFactorSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
            RemovePhoneSuccess,
            Error
        }

#endregion
    }
}
```

1.1.15. Класс ProfileController

```
using System;
using System.Threading.Tasks;
using System.Web.Mvc;
using CocktionMVC.Functions;
using CocktionMVC.Functions.DataProcessing;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using Microsoft.AspNet.Identity;

namespace CocktionMVC.Controllers.StandartControllers
{
   public class ProfileController : Controller
   {
      /// <summary>
      /// Выводит пользователя на страничку с информацией о его аккаунте
      /// </summary>
      public ActionResult Index()
      {
         if (User.Identity.IsAuthenticated)
         {//если пользватель авторизован
            CocktionContext db = new CocktionContext();

            var user = db.AspNetUsers.Find(User.Identity.GetUserId());

            return View(user);
         }

         //если пользователь не авторизован
         return RedirectToAction("HowItCouldBe");
      }

      /// <summary>
      /// Добавляет отзыв о пользователе в базу
      /// </summary>
      public async Task<JsonResult> AddUsersFeedback()
      {
         try
         {
            //Получаем текст сообщения и айдишник пользователя, который будет тут задействован
            var messageHolder = Request.Form.GetValues("message");
            if (messageHolder != null)
            {
               string message = messageHolder[0].Trim();
               var userIdHolder = Request.Form.GetValues("userId");
               if (userIdHolder != null)
               {
```

```
                string userId = userIdHolder[0].Trim();
                CocktionContext db = new CocktionContext();

                string currentUserId = User.Identity.GetUserId();

                //находим пользователя (который оставляет комментарий)
                var user = db.AspNetUsers.Find(currentUserId);

                //инициализация фидбека
                UsersFeedback feedback = new UsersFeedback(user.UserRealName,
user.UserRealSurname,
                    currentUserId, userId, message);
                db.Feedbacks.Add(feedback);

                //сохранение значений в базу
                await DbItemsAdder.SaveDb(db);

                return Json(new FeedBackRespond(user.UserRealName, user.UserRealSurname,
"success"));
                }
            }
            return Json(new FeedBackRespond(null, null, "failure"));
        }
        catch
        {
            return Json(new FeedBackRespond(null, null, "failure"));
        }
    }

    public class SelfieRespond : StatusHolder
    {
        public SelfieRespond(string fileName, bool isOk) : base(isOk)
        {
            FileName = fileName;
        }
        public string FileName { get; set; }
    }

    /// <summary>
    /// Добавляет пользователю аватарку
    /// </summary>
    /// <returns>Ответ с путем к аватарке и инфой получилось или нет</returns>
    public JsonResult AddPhotoToUser()
    {
        try
        {
            var userId = User.Identity.GetUserId();
            CocktionContext db = new CocktionContext();
```

```
            var user = db.AspNetUsers.Find(userId);

            Picture selfie = new Picture();
            PhotoProcessor.CreateAndSavePicture(selfie, Request, 200);

            user.Selfie = selfie;
            db.SaveChanges();

            return Json(new SelfieRespond(user.Selfie.FileName, true));
        }
        catch
        {
            return Json(new SelfieRespond(null, false));
        }
    }

    class FeedBackRespond
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Status { get; set; }

        public FeedBackRespond(string name, string surname, string status)
        {
            Name = name;
            Surname = surname;
            Status = status;
        }
    }

    /// <summary>
    /// Показывает пользователю то, какой могла бы быть его страничка
    /// </summary>
    public ActionResult HowItCouldBe()
    {
        //TODO сделать эту страниу
        return View();
    }

    /// <summary>
    /// Добавляет интересы пользователя
    /// </summary>
    /// <returns>Стандартный ответ сервака</returns>
    [HttpPost]
    [Authorize]
    public JsonResult AddInterests()
    {
```

```
    try
    {
        string ids = Request.Form.GetValues("ids")[0];
        string[] sep = {"int_"};
        string[] idcont = ids.Split(sep, StringSplitOptions.RemoveEmptyEntries);

        int i = 0;
        int[] idint = new int[idcont.Length];
        Array.ForEach(idcont, x => idint[i++] = int.Parse(x));
        CocktionContext db = new CocktionContext();

        var user = db.AspNetUsers.Find(User.Identity.GetUserId());
        for (int j = 0; j < idint.Length; j++)
        {
            user.Interests.Add(db.Interests.Find(idint[j]));
        }
        db.SaveChanges();
        return Json(new StatusHolder(true));
    }
    catch
    {
        return Json(new StatusHolder(false));
    }
}

/// <summary>
/// Позволяет редактировать информацию пользователя
/// </summary>
/// <returns></returns>
public JsonResult EditProfileInformation()
{
    try
    {
        string name;
        string surname;
        string school;

        RequestFormReader.ReadEditUserInfo(Request, out name, out surname, out school);

        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        if (name.Length != 0)
            user.UserRealName = name;

        if (surname.Length != 0)
            user.UserRealSurname = surname;
```

```
            if (school.Length != 0)
                user.SocietyName = school;

            db.SaveChanges();
            return Json(new StatusHolder(true));
        }
        catch
        {
            return Json(new StatusHolder(false));
        }
    }

  }
}
```

1.2. API контроллеры

1.2.1. Класс AccountController

```
using System;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using CocktionMVC.Models;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.MobileClientModels;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Infrastructure;
using Microsoft.Owin.Security;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.Identity;
namespace CocktionMVC.Controllers.ApiControllers
{
  public class AccountController : ApiController
  {

    private ApplicationUserManager _userManager;
    public ApplicationUserManager UserManager
    {
      get
      {
        return _userManager ??
HttpContext.Current.Request.GetOwinContext().GetUserManager<ApplicationUserManager>();
      }
    }
```

```
/// <summary>
/// Регистрирует пользователя с мобильного клиента
/// </summary>
/// <param name="data">Данные для регистрации</param>
/// <returns>Результат регистрации</returns>
[HttpPost]
public async Task<StatusHolder> Registrate(LoginCredentials data)
{
    var user = new ApplicationUser
    {
        UserName = data.email,
        Email = data.email,
        Rating = 1000,
        Eggs = 100
    };
    var resultOfRegistration = await UserManager.CreateAsync(user, data.password);
    if (resultOfRegistration.Succeeded)
    {
        return new StatusHolderWithError(true, null);
    }
    return new StatusHolderWithError(false, "Такой пользователь уже зарегистрирован");
}

/// <summary>
/// Производит аутентификацию пользователя через логин пароль
/// и возвращает мобильнику его токен
/// </summary>
/// <param name="model">Модель с логином и паролем</param>
/// <returns>Токен для авторизации последующей</returns>
[HttpPost]
public TokenResponse Authenticate(LoginCredentials model)
{
    //полчуаем информацию о пользоватлее
    string user = model.email, password = model.password;

    //Если чего-то нет (логин или парль) авторизация невозможна
    if (string.IsNullOrEmpty(user) || string.IsNullOrEmpty(password))
        return new TokenResponse { Token = "Failure" };

    //Ищем пользователя с таким паролем и логином
    var userIdentity = UserManager.FindAsync(user, password).Result;

    if (userIdentity != null)
    { //если пользователь нашелся
        //проверяем все и возвращаем токен
        var identity = new ClaimsIdentity(Startup.OAuthBearerOptions.AuthenticationType);
```

```
            identity.AddClaim(new Claim(ClaimTypes.Name, user));
            identity.AddClaim(new Claim(ClaimTypes.NameIdentifier, userIdentity.Id));

            AuthenticationTicket ticket = new AuthenticationTicket(identity, new
AuthenticationProperties());
            var currentUtc = new SystemClock().UtcNow;
            ticket.Properties.IssuedUtc = currentUtc;
            ticket.Properties.ExpiresUtc = currentUtc.Add(TimeSpan.FromDays(100));

            string accessToken = Startup.OAuthBearerOptions.AccessTokenFormat.Protect(ticket);
            return new TokenResponse { Token = accessToken };
        }

        //если не нашелся - возвращаем ошибку
        return new TokenResponse { Token = "Failure" };
    }

    public class DeviceInfo
    {
        public string type { get; set; }
        public string token { get; set; }
    }

    [HttpPost]
    [Authorize]
    public StatusHolder AddDevice(DeviceInfo device)
    {
        try
        {
            CocktionContext db = new CocktionContext();
            var user = db.AspNetUsers.Find(User.Identity.GetUserId());
            switch(device.type)
            {
                case "ios":
                    user.MobileDevice = new Device(device.type, device.token);
                    break;
                case "android":
                    //do smth here
                    break;
                case "wp":
                    //do smth here
                    break;
            }
            db.SaveChanges();
            return new StatusHolder(true);
        }
        catch
        {
```

```
            return new StatusHolder(false);
        }
    }
}
}
```

1.2.2. Класс AuctionController

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.Hubs;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.MobileClientModels;
using Microsoft.AspNet.Identity;
namespace CocktionMVC.Controllers.ApiControllers
{
   public class AuctionController : ApiController
   {
      /// <summary>
      /// Метод, достающий из базы данных
      /// все активные аукционы и обрабатывающий их
      /// для нужд мобильного приложения
      /// </summary>
      /// <returns>Джейсоном лист из аукционов</returns>
      [Authorize]
      [HttpPost]
      public List<AuctionInfo> GetActiveAuctions()
      {
         CocktionContext db = new CocktionContext();

         //Конвертация времени из ютс в местное
         DateTime controlTime = DateTimeManager.GetCurrentTime();

         //Выборка активных в данный момент аукционов
         var auctions = (from x in db.Auctions
                         where ((x.EndTime > controlTime) && (x.IsActive))
                         select x).ToList<Auction>();

         List<AuctionInfo> auctiInfo = new List<AuctionInfo>();
```

```
        //формируем список в том формате, который нужен для отображения в приложении
        foreach (var auction in auctions)
        {
            //для мобилки нужно количество минут до конца
            int minutes = (int)auction.EndTime.Subtract(controlTime).TotalMinutes;
            auctiInfo.Add(
                  new AuctionInfo
                  {
                     description = auction.SellProduct.Description.Trim(),
                     endTime = minutes,
                     photoPath = @"http://cocktion.com/Images/Thumbnails/" +
auction.SellProduct.Photo.FileName,
                     title = auction.SellProduct.Name.Trim(),
                     auctionId = auction.Id,
                     leaderId = auction.LeadProduct == null ? -1 : auction.LeadProduct.Id,
                     category = auction.SellProduct.Category.Trim()
                  }
                  );
        }//end of foreach
        return auctiInfo;
    }//end of GetActiveAuctions()

    //В АПИ КОНТРОЛЛЕРАХ РУТИНГ ДРУГОЙ!!!

    /// <summary>
    /// Метод отправляет на мобильный клиент информацию о всех товарах,
    /// которые поставлены на данном аукционе
    /// </summary>
    /// <param name="id">Айди аукциона</param>
    /// <returns>Лист с информацией о продуктах</returns>
    [Authorize]
    [HttpPost]
    public List<ProductInfo> GetAuctionBids(IdCont aId)
    {
        CocktionContext db = new CocktionContext();

        //находим аукцион по айди
        Auction auction = db.Auctions.Find(aId.id);
        List<ProductInfo> bidProducts = new List<ProductInfo>();

        //добавляем все в коллекцию
        foreach (var bid in auction.BidProducts)
            bidProducts.Add(new ProductInfo
                {
                    description = bid.Description.Trim(),
                    photoPath = @"http://cocktion.com/Images/Thumbnails/" + bid.Photo.FileName,
                    title = bid.Name.Trim(),
```

```
            category = bid.Category.Trim(),
            id = bid.Id,
        });
    return bidProducts;
}

public class CreateAuctionInfo
{
    public string title { get; set; }

    public string description { get; set; }
    public string category { get; set; }
    public int photoId { get; set; }
    public string timeBound { get; set; }
}

/// <summary>
/// Создание аукциона с мобильного клиента
/// </summary>
[Authorize]
[HttpPost]
public async Task<StatusAndPhotoPath> CreateAuction(CreateAuctionInfo auctionInfo)
{
    //Заводим возвращалку статуса
    StatusAndPhotoPath info =
        new StatusAndPhotoPath();
    try
    {
        //получаем информацию о пользователе
        string userId = User.Identity.GetUserId();

        //подключаемся к базе данных
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(userId);

        //создаем продукт и сохраняем информацию о нем.
        Product product = new Product(auctionInfo.title, auctionInfo.description,
            auctionInfo.category, true, user);

        //инициализация аукциона
        Auction auction = new Auction(true, product, false, new ToteBoard(), user);

        auction.Rating = (int)(user.Rating * 0.4);
        auction.Owner = user;
        //Совершаем манипуляции со временем
        DateTimeManager.SetAuctionStartAndEndTime(auction, auctionInfo.timeBound);

        //увеличиваем рейтинг пользователя
```

```
            RatingManager.IncreaseRating(user, "userMadeAuction");
            Picture photo;
            //забиваем данные о фотке
            if (auctionInfo.photoId != -1)
            {
               photo = db.Pictures.Find(auctionInfo.photoId);
            }
            else
            {
               photo = new Picture();
               const string placeHolderName = "placeholder.jpg";
               photo.FileName = placeHolderName;
               string path = Path.Combine(
                  HttpContext.Current.Request.MapPath("~/Content/SiteImages"), placeHolderName);
               photo.FilePath = path;
            }

            product.Photo = photo;
            //все в базу добавляем
            await DbItemsAdder.AddAuctionProductPhotoAsync(db, auction, product, photo, user);

            //обновляем список аукционов
            AuctionListHub.UpdateList(product.Name, product.Description, product.Category,
auction.EndTime,
            photo.FileName, auction.Id);


            //шлем ссылочку на фотку
            info.PhotoPath = "http://cocktion.com/Images/Thumbnails/" + photo.FileName;
            info.Status = "Success";

         }




         catch
         {
            info.PhotoPath = null;
            info.Status = "Failure";
         }
         return info;
      }

      public class ChooseLeaderClass
      {
         public int auctionId { get; set; }
         public int productId { get; set; }
```

```
}

/// <summary>
/// Позволяет выбирать лидера с мобильного клиента
/// </summary>
/// <returns>Стандартный статус (удалось или нет)</returns>
[HttpPost]
[Authorize]
public StatusHolder ChooseLeader(ChooseLeaderClass leader)
{
   CocktionContext db = new CocktionContext();
   string userId = User.Identity.GetUserId();
   try
   {
      var auction = db.Auctions.Find(leader.auctionId);
      var product = db.Products.Find(leader.productId);

      if (userId != auction.Owner.Id)
         throw new Exception();
      auction.LeadProduct = product;

      auction.WinnerChosen = true;

      db.SaveChanges();
      AuctionHub.SetLider(leader.productId.ToString(), leader.auctionId, product.Name);
      return new StatusHolder(true);
   }
   catch
   {
      return new StatusHolder(false);
   }
}

public class IdCont
{
   public int id { get; set; }
}

/// <summary>
/// Завершает аукцион с мобильника
/// </summary>
/// <returns>Стандартный статус</returns>
[HttpPost]
[Authorize]
public async Task<StatusHolder> EndAuction(IdCont aId)
{
   //TODO решить проблему взаимодействия с клиентами. Они не могут узнать, что аукцион завершен с мобильника
```

```
    CocktionContext db = new CocktionContext();
    string userId = User.Identity.GetUserId();
    try
    {
        //находим аукцион
        var auction = db.Auctions.Find(aId.id);

        //Проверяем, можно ли его закончить

        //является ли пользователь владельцем?
        bool isUserOwner = (userId == auction.Owner.Id);
        bool canEnd = auction.IsActive & auction.WinnerChosen;
        if ((!isUserOwner) & canEnd)
            throw new Exception();

        await AuctionHub.FinishAuctionMobile(auction.Id);

        return new StatusHolder(true);
    }
    catch
    {
        return new StatusHolder(false);
    }

}

public class prI
{
    public string title { get; set; }

    public string description { get; set; }

    public string category { get; set; }

    public int auctionId { get; set; }

    public int photoId { get; set; }
}

[HttpPost]
[Authorize]
public async Task<StatusHolder> AddBid(prI info)
{
    try
    {
        //получаем информацию о пользователе
        string userId = User.Identity.GetUserId();
```

```
//подключаемся к базе данных
CocktionContext db = new CocktionContext();

var user = db.AspNetUsers.Find(userId);

//создаем продукт и сохраняем информацию о нем.
Product product = new Product(info.title, info.description, info.category, false, user);

Picture photo;
//забиваем данные о фотке
if (info.photoId != -1)
{
   photo = db.Pictures.Find(info.photoId);
}
else
{
   photo = new Picture();
   const string placeHolderName = "placeholder.jpg";
   photo.FileName = placeHolderName;
   string path = Path.Combine(
      HttpContext.Current.Request.MapPath("~/Content/SiteImages"), placeHolderName);
   photo.FilePath = path;
}
//добавляем продукт
//Коннектимся к базе
var auction = db.Auctions.Find(info.auctionId);
auction.BidProducts.Add(product);
product.SelfAuction = auction;

product.Photo = photo;

//находи кластер
BidCluster bidCluster = new BidCluster();
bidCluster.UserId = userId;
bidCluster.HostAuction = auction;
bidCluster.Products.Add(product);
product.SelfAuction = auction;

user.HisProducts.Add(product);

//обновляем рейтинг аукциона
RatingManager.IncreaseRating(auction, user, "userBeted");

//обновляем рейтинг пользователя
RatingManager.IncreaseRating(user, "userPlacedBet");

//сохраняем все в базу
await DbItemsAdder.AddProduct(db, product, photo, bidCluster);
```

```
            //добавляем нодики на клиенты
            AuctionHub.AddNodesToClients(info.title, photo.FileName, info.auctionId, product.Id);

            return new StatusHolder(true);
        }
        catch
        {
            return new StatusHolder(false);
        }

    }
  }
}
```

1.2.3. Класс DataController

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.MobileClientModels;
using Microsoft.AspNet.Identity;
using PushSharp.Apple;
using PushSharp;
namespace CocktionMVC.Controllers.ApiControllers
{
  public class DataController : ApiController
  {
    /// <summary>
    /// Посылает все дома на кокшне
    /// человеку
    /// </summary>
    /// <returns></returns>
    [Authorize]
    [HttpPost]
    public List<HouseInfo> GetHouses()
    {
      CocktionContext db = new CocktionContext();
      List<HouseInfo> houses = new List<HouseInfo>();
```

```
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    foreach (var house in db.Houses)
    {
        houses.Add(new HouseInfo()
        {
            adress = house.Adress,
            title = house.Faculty,
            likes = house.Likes,
            rating = house.Rating,
            isSubscribed = user.SubHouses.Contains(house)
        });
    }
    return houses;
}


public class SearchString
{
    public string name { get; set; }
}

/// <summary>
/// Ищет аукционы по указанной строке
/// </summary>
/// <param name="searchString">че ищем</param>
/// <returns>Коллекцию с аукционами, удовлетворяющими критериям поиска</returns>
[Authorize]
[HttpPost]
public List<AuctionInfo> FindAuctions(SearchString searchString)
{
    CocktionContext db = new CocktionContext();
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    DateTime controlTime = DateTimeManager.GetCurrentTime();
    List<Auction> aciveAuctions;
    aciveAuctions = (from x in db.Auctions
                     where (x.SellProduct.Name.Contains(searchString.name) ||
                     x.SellProduct.Description.Contains(searchString.name) ||
                     x.SellProduct.Category.Contains(searchString.name))
                     select x).ToList();
    List<AuctionInfo> auctis = new List<AuctionInfo>();
    foreach (var x in aciveAuctions)
    {
        if (x.Owner != null)
        {
            auctis.Add(new AuctionInfo(x.SellProduct.Description,
                    (int)x.EndTime.Subtract(controlTime).TotalMinutes,
                    @"http://cocktion.com/Images/Thumbnails/" + x.SellProduct.Photo.FileName,
                    x.SellProduct.Name.Trim(), x.Id, x.LeadProduct == null ? -1 : x.LeadProduct.Id,
```

```
                    x.SellProduct.Category.Trim(), new UserInfo(x.Owner.Id, x.Owner.UserName,
x.Owner.Selfie.FileName,
                        user.Friends.Contains(x.Owner)),
                    (x.IsActive && (controlTime < x.EndTime))));
            }
        }
        return auctis;
    }


    public class PhotoId
    {
        public int id { get; set; }

        public PhotoId(int _id)
        {
            id = _id;
        }
    }

    public class PhotoType
    {
        public string type { get; set; }
    }


    /// <summary>
    /// Добавляет фотографию на сервак
    /// (только для товара или аукциона)
    /// </summary>
    /// <returns>Возвращает айдишник фотографии в базе данных</returns>
    [HttpPost]
    [Authorize]
    public async Task<PhotoId> UploadPhoto()
    {
        try
        {
            //получаем файлик
            string type = HttpContext.Current.Request.Form.GetValues("type")[0].Trim();
            string filePath = "";
            string fileName = "";
            HttpPostedFile postedFile = null;
            var httpRequest = HttpContext.Current.Request;
            if (httpRequest.Files.Count > 0)
            {
                foreach (string file in httpRequest.Files)
                {
                    postedFile = httpRequest.Files[file];
```

```
                string extension = Path.GetExtension(postedFile.FileName);
                fileName = Guid.NewGuid() + extension; //генерируем новое имя для фотки
                filePath = HttpContext.Current.Server.MapPath("~/Images/Photos/" + fileName);
                postedFile.SaveAs(filePath);
            }
        }

        //добавляем фоточку и фабмнейл
        string thumbNailPath = HttpContext.Current.Server.MapPath("~/Images/Thumbnails/"); //путь на сервере для сохранения
        string thumbNailFilePath = thumbNailPath + fileName;

        if (type == "auction")
        {
            ThumbnailGenerator.ResizeImage(postedFile, thumbNailFilePath, 90);
        }
        else if (type == "product")
        {
            ThumbnailGenerator.ResizeImage(postedFile, thumbNailFilePath, 60);
        }

        CocktionContext db = new CocktionContext();

        Picture picture = new Picture();
        picture.FileName = fileName;
        picture.FilePath = thumbNailPath;
        db.Pictures.Add(picture);

        await DbItemsAdder.SaveDb(db);

        return new PhotoId(picture.Id);
      }
      catch
      {
        return new PhotoId(-1);
      }
    }

  }
}
```

1.2.4. Класс HouseController

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
```

```
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.HouseRelatedModels;
using Microsoft.AspNet.Identity;

namespace CocktionMVC.Controllers.ApiControllers
{
    /// <summary>
    /// Контейнер для методов, посылающих дома и все, что с ними связано мобильным клиентам
    /// </summary>
    public class HouseController : ApiController
    {
        /// <summary>
        /// Отправляет на мобильный клиент список всех доступных
        /// хаус-холдеров
        /// </summary>
        /// <returns>Коллекцию хаусхолдеров</returns>
        [HttpPost]
        [Authorize]
        public List<Guild> GetGuilds()
        {
            List<Guild> guilds = new List<Guild>();

            CocktionContext db = new CocktionContext();

            var holders = db.HouseHolders.ToList();
            int amountOfHolders = holders.Count;

            for (int i = 0; i < amountOfHolders; i++)
            {
                guilds.Add(new Guild(holders[i].Name, "http://cocktion.com/Images/Thumbnails/" +
holders[i].PhotoCard.FileName,
                    holders[i].Id));
            }

            return guilds;
        }


        public class IdContainer
        {
            public int id { get; set; }
        }

        /// <summary>
        /// Посылает на мобилку коллекцию с домами хаусхолдера
        /// </summary>
        /// <returns>Коллекцию</returns>
```

```
[HttpPost]
[Authorize]
public List<GuildsHouse> GetGuildsHouses(IdContainer gn)
{
   List<GuildsHouse> houses = new List<GuildsHouse>();
   CocktionContext db = new CocktionContext();
   var user = db.AspNetUsers.Find(User.Identity.GetUserId());
   var holders = db.HouseHolders.Find(gn.id).Houses.ToList();
   foreach (var holder in holders)
   {
      houses.Add(new GuildsHouse(holder.Faculty, holder.Id, holder.Adress,
         "http://cocktion.com/Images/Thumbnails/" + holder.Portrait.FileName,
         user.SubHouses.Contains(holder)));
   }
   return houses;
}


/// <summary>
/// Посылает на мобилку информаию о доме
/// </summary>
/// <returns>Класс с инфой о доме</returns>
[HttpPost]
[Authorize]
public HouseMobile GetHouse(IdContainer hsNum)
{
   //TODO добавить везде проверки
   CocktionContext db = new CocktionContext();
   var user = db.AspNetUsers.Find(User.Identity.GetUserId());

   //обработка guildId из формы
   House house = db.Houses.Find(hsNum.id);
   HouseMobile mobileHouse = new HouseMobile(house.Faculty,@"http://cocktion.com/Images/
Thumbnails" + house.Portrait.FileName, house.Likes,
      0, house.Rating, house.Inhabitants.Count, house.Auctions.Count, house.Description,
      user.SubHouses.Contains(house));

   return mobileHouse;
}

/// <summary>
/// Шлет все посты с форума на мобилку
/// </summary>
/// <returns>Коллекцию с постами для мобилки</returns>
[HttpPost]
[Authorize]
public List<ForumPostMobile> GetHouseForumPosts(IdContainer idContainer)
{
```

```
        CocktionContext db = new CocktionContext();
        var forumPosts = db.Houses.Find(idContainer.id).Posts;
        List<ForumPostMobile> posts = new List<ForumPostMobile>(forumPosts.Count);

        posts.AddRange(forumPosts.Select(post => new ForumPostMobile(post.AuthorName,
post.Message, post.Likes)));

        return posts;
    }

    public class Message
    {
        public string message { get; set; } //само сообщение
        public int id { get; set; } //айдишник дома
    }




    /// <summary>
    /// Позволяет отписываться от дома.
    /// </summary>
    /// <param name="id">Айдишник дома, от которого надо отписаться</param>
    /// <returns>Стандартный ответ</returns>
    [HttpPost]
    [Authorize]
    public StatusHolder UnsubscribeFromHouse(IdContainer id)
    {
        try
        {
            CocktionContext db = new CocktionContext();
            var user = db.AspNetUsers.Find(User.Identity.GetUserId());
            var house = db.Houses.Find(id.id);
            if (user.SubHouses.Contains(house))
            {
                user.SubHouses.Remove(house);
                db.SaveChanges();
            }
            return new StatusHolder(true);
        }
        catch
        {
            return new StatusHolder(false);
        }
    }
```

```
/// <summary>
/// Позволяет подписаться на дом
/// </summary>
/// <param name="id">Айдишник дома, на который надо подписаться</param>
/// <returns>Стандартный ответ</returns>
[HttpPost]
[Authorize]
public StatusHolder SubscribeOnHouse(IdContainer id)
{
    try
    {
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());
        var house = db.Houses.Find(id.id);
        if (!user.SubHouses.Contains(house))
        {
            user.SubHouses.Add(house);
            db.SaveChanges();
        }
        return new StatusHolder(true);
    }
    catch
    {
        return new StatusHolder(false);
    }

}

/// <summary>
/// Добавляет пост на форум конкретного дома
/// </summary>
/// <returns>Статус удалось или нет</returns>
[HttpPost]
[Authorize]
public StatusHolder SendPost(Message msg)
{
    try
    {
        CocktionContext db = new CocktionContext();

        //Получаем информацию для постав
        string authorName = User.Identity.Name;

        //Находим дом
        var house = db.Houses.Find(msg.id);

        //добавляем пост
```

```
            house.Posts.Add(new ForumPost(msg.message, authorName));
            db.SaveChanges();

            //возвращаем статус
            return new StatusHolder(true);
        }
        catch
        {
            return new StatusHolder(false);
        }
    }
}
```

1.2.5. Класс ProfileController

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.JsonModels.MobileClientModels;
using Microsoft.AspNet.Identity;

namespace CocktionMVC.Controllers.ApiControllers
{
    public class ProfileController : ApiController
    {
        /// <summary>
        /// Контейнер для профайлайди поля
        /// </summary>
        public class ProfileInfo
        {
            public string id { get; set; }
        }

        /// <summary>
        /// Посылает мобильному клиенту информацию о профиле пользователя
        /// </summary>
        /// <param name="profInf">Контейнер с айдишником</param>
        /// <returns>Информацию о пользователе</returns>
```

```
[HttpPost]
[Authorize]
public ProfileData GetInfo(ProfileInfo profInf)
{
   CocktionContext db = new CocktionContext();
   AspNetUser user;
   if (profInf.id == "self")
   {
      string userId = User.Identity.GetUserId();
      user = db.AspNetUsers.Find(userId);
   }
   else
   {
      user = db.AspNetUsers.Find(profInf.id);
   }
   string name = user.UserRealName ?? "none";
   string surname = user.UserRealSurname ?? "none";
   string city = String.IsNullOrEmpty(user.City) ? "none" : user.City;
   string society = String.IsNullOrEmpty(user.SocietyName) ? "none" : user.SocietyName;
   ProfileData data;
   if (user.Selfie == null)
   {
      data = new ProfileData(name, surname, user.UserName,
      user.Rating, user.Eggs, user.HisAuctions.Count, user.HisProducts.Count, city,
      society, @"http://cocktion.com/Content/SiteImages/" + "anonPhoto.jpg");
   }
   else
   {
       data = new ProfileData(name, surname, user.UserName,
       user.Rating, user.Eggs, user.HisAuctions.Count, user.HisProducts.Count, city,
       society, @"http://cocktion.com/Images/Thumbnails/" + user.Selfie.FileName);
   }

   return data;

}

/// <summary>
/// Контейнер для информации о доме.
/// </summary>
public class ShortHouseData
{
   public ShortHouseData(int id, string holderName, string houseName, string photoPath)
   {
      this.id = id;
      this.holderName = holderName;
      this.houseName = houseName;
      this.photoPath = @"http://cocktion.com/Images/Thumbnails/" + photoPath;
```

```
    }
    public int id { get; set; }
    public string holderName { get; set; }
    public string houseName { get; set; }

    public string photoPath {get; set;}
}

/// <summary>
/// Контейнер для информации о профиле пользователя.
/// </summary>
public class ProfileData
{
    public string title { get; set; }
    public string surname { get; set; }
    public string login { get; set; }
    public int? rating { get; set; }
    public int eggs { get; set; }
    public int auctionsAmount { get; set; }
    public int productsAmount { get; set; }
    public string city { get; set; }
    public string society { get; set; }
    public string photoPath { get; set; }
    public ProfileData(string name, string surname, string login, int? rating,
        int eggs, int auctionsAmount, int productsAmount, string city, string society, string photoPath)
    {
        this.title = name;
        this.surname = surname;
        this.login = login;
        this.rating = rating;
        this.eggs = eggs;
        this.auctionsAmount = auctionsAmount;
        this.productsAmount = productsAmount;
        this.city = city;
        this.society = society;
        this.photoPath = photoPath;
    }
}

/// <summary>
/// Позволяет получить список всех домов,
/// на которые подписан пользователь
/// </summary>
/// <returns>Коллекцию с информацией о домах</returns>
[HttpPost]
[Authorize]
public List<ShortHouseData> GetMyHouses()
{
```

```
    CocktionContext db = new CocktionContext();
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    List<ShortHouseData> data = (from x in user.SubHouses
                                 select new ShortHouseData(x.Id, x.Holder.Name,
                                    x.Faculty, x.Portrait.FileName)).ToList();
    return data;
}

/// <summary>
/// Контейнер для информации об аукционах, которые пользователь хочет
/// получить
/// </summary>
public class UsersAuctionsHouses
{
    public List<AuctionInfo> active { get; set; }

    public List<AuctionInfo> finished { get; set; }

    public List<AuctionInfo> houses { get; set; }

    public List<AuctionInfo> products { get; set; }

    public UsersAuctionsHouses(List<AuctionInfo> active, List<AuctionInfo> finished,
        List<AuctionInfo> inHouse, List<AuctionInfo> inProducts)
    {
        this.active = active;
        this.finished = finished;
        houses = inHouse;
        products = inProducts;
    }
}

/// <summary>
/// Позволяет получать информацию о различных аукционов для данного пользователя
/// 1)Активные аукционы пользователя
/// 2)Завершенные аукционы пользователя
/// 3)Аукционы в колхозе пользователя
/// 4)Аукцион
/// </summary>
/// <returns></returns>
[HttpPost]
[Authorize]
public UsersAuctionsHouses GetMyAuctions()
{
    CocktionContext db = new CocktionContext();
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    //Конвертация времени из ютс в местное
    DateTime controlTime = DateTimeManager.GetCurrentTime();
```

```
//Выборка активных в данный момент аукционов,
//которые принадлежат пользователю
List<AuctionInfo> myActive;
try
{
   if (user.Selfie == null)
   {
      myActive = (from x in user.HisAuctions
                  where (x.IsActive && x.EndTime > controlTime)
                  select new AuctionInfo(x.SellProduct.Description,
                     (int)x.EndTime.Subtract(controlTime).TotalMinutes,
                     @"http://cocktion.com/Images/Thumbnails/" + x.SellProduct.Photo.FileName,
                     x.SellProduct.Name.Trim(), x.Id, x.LeadProduct == null ? -1 : x.LeadProduct.Id,
                     x.SellProduct.Category.Trim(),
                     new UserInfo(x.Owner.Id, x.Owner.UserName, "anonPhoto.jpg",
                        user.Friends.Contains(x.Owner)),
                     isActive: true)).ToList();
   }
   else
   {
      myActive = (from x in user.HisAuctions
                  where (x.IsActive && x.EndTime > controlTime)
                  select new AuctionInfo(x.SellProduct.Description,
                     (int)x.EndTime.Subtract(controlTime).TotalMinutes,
                     @"http://cocktion.com/Images/Thumbnails/" + x.SellProduct.Photo.FileName,
                     x.SellProduct.Name.Trim(), x.Id, x.LeadProduct == null ? -1 : x.LeadProduct.Id,
                     x.SellProduct.Category.Trim(),
                     new UserInfo(x.Owner.Id, x.Owner.UserName, x.Owner.Selfie.FileName,
                        user.Friends.Contains(x.Owner)),
                     isActive: true)).ToList();
   }

}
catch
{
   myActive = null;
}
 //Выборка всех оконченных аукционов пользователя
List<AuctionInfo> myFinished;
try
{
   myFinished = (from x in user.HisAuctions
                 where (x.IsActive == false || x.EndTime < controlTime)
                 select new AuctionInfo(x.SellProduct.Description,
                    (int)x.EndTime.Subtract(controlTime).TotalMinutes,
                    @"http://cocktion.com/Images/Thumbnails/" + x.SellProduct.Photo.FileName,
                    x.SellProduct.Name.Trim(), x.Id, x.LeadProduct == null ? -1 : x.LeadProduct.Id,
```

```
                    x.SellProduct.Category.Trim(),
                    new UserInfo(x.Owner.Id, x.Owner.UserName, x.Owner.Selfie.FileName,
                       user.Friends.Contains(x.Owner)),
                    isActive: false)).ToList();
        }
        catch
        {
           myFinished = null;
        }


        //Выборка всех в его домах
        List<AuctionInfo> inHisHouses = new List<AuctionInfo>();
        foreach (var house in user.SubHouses)
        {
           var housesInfo = (from x in house.Auctions
                    where (x.IsActive && x.EndTime > DateTime.Now)
                    select x).ToList();

           foreach (var hI in housesInfo)
           {
              AspNetUser thisOwner;
              try
              {
                 thisOwner = hI.Owner;
                 var auctInfo = new AuctionInfo(hI.SellProduct.Description,
                          (int)hI.EndTime.Subtract(controlTime).TotalMinutes,
                          @"http://cocktion.com/Images/Thumbnails/" + hI.SellProduct.Photo.FileName,
                          hI.SellProduct.Name.Trim(), hI.Id, hI.LeadProduct == null ? -1 :
hI.LeadProduct.Id,
                          hI.SellProduct.Category.Trim(),
                         new UserInfo(thisOwner.Id, thisOwner.UserName, thisOwner.Selfie.FileName,
                            user.Friends.Contains(thisOwner)),
                         isActive: true);
                 if (auctInfo != null)
                 {
                    if (!inHisHouses.Any(x => x.auctionId == auctInfo.auctionId))
                       inHisHouses.Add(auctInfo);
                 }

              }
              catch
              {
                 var auctInfo = new AuctionInfo(hI.SellProduct.Description,
                          (int)hI.EndTime.Subtract(controlTime).TotalMinutes,
                          @"http://cocktion.com/Images/Thumbnails/" + hI.SellProduct.Photo.FileName,
                          hI.SellProduct.Name.Trim(), hI.Id, hI.LeadProduct == null ? -1 :
hI.LeadProduct.Id,
```

```
                        hI.SellProduct.Category.Trim(),
                        null, isActive: true);
                if (auctInfo != null)
                {
                    if (!inHisHouses.Any(x => x.auctionId == auctInfo.auctionId))
                        inHisHouses.Add(auctInfo);
                }
            }

        }
    }

    //Выборка аукционов по всем его ставкам
    List<AuctionInfo> myBetsAuctions = new List<AuctionInfo>();
    try
    {
        foreach (var bet in user.HisProducts)
        {
            if (bet.IsOnAuctionAsALot == false)
            {
                var auction = bet.SelfAuction;
                if (auction != null)
                {
                    bool isActive = (auction.IsActive && auction.EndTime > controlTime);
                    try
                    {
                        AuctionInfo aI = new AuctionInfo(auction.SellProduct.Description,
                        (int)auction.EndTime.Subtract(controlTime).TotalMinutes,
                        @"http://cocktion.com/Images/Thumbnails/" + auction.SellProduct.Photo.FileName,
                         auction.SellProduct.Name.Trim(), auction.Id,
                        auction.LeadProduct == null ? -1 : auction.LeadProduct.Id,
                        auction.SellProduct.Category.Trim(),
                        new UserInfo(auction.Owner.Id, auction.Owner.UserName,
auction.Owner.Selfie.FileName,
                            user.Friends.Contains(auction.Owner)),
                        isActive);
                        if (!myBetsAuctions.Any(x => x.auctionId == aI.auctionId))
                            myBetsAuctions.Add(aI);
                    }
                    catch
                    {
                        AuctionInfo aI = new AuctionInfo(auction.SellProduct.Description,
                        (int)auction.EndTime.Subtract(controlTime).TotalMinutes,
                        @"http://cocktion.com/Images/Thumbnails/" + auction.SellProduct.Photo.FileName,
                        auction.SellProduct.Name.Trim(), auction.Id,
                        auction.LeadProduct == null ? -1 : auction.LeadProduct.Id,
                        auction.SellProduct.Category.Trim(),
                        null, isActive);
```

```
                    if (!myBetsAuctions.Any(x => x.auctionId == aI.auctionId))
                        myBetsAuctions.Add(aI);
                }

            }
        }
    }
    catch
    {
    }

    return new UsersAuctionsHouses(myActive, myFinished, inHisHouses, myBetsAuctions);
}


/// <summary>
/// Меняет пользовательскую аватарку
/// </summary>
/// <returns>Стандартный ответ сервера</returns>
[HttpPost]
[Authorize]
public async Task<StatusHolder> UploadProfilePhoto()
{
    try
    {
        //получаем файлик
        string fileName = "";
        HttpPostedFile postedFile = null;
        var httpRequest = HttpContext.Current.Request;
        if (httpRequest.Files.Count > 0)
        {
            foreach (string file in httpRequest.Files)
            {
                postedFile = httpRequest.Files[file];
                string extension = Path.GetExtension(postedFile.FileName);
                fileName = Guid.NewGuid() + extension; //генерируем новое имя для фотки
            }
        }

        //добавляем фоточку и фабмнейл
        string thumbNailPath = HttpContext.Current.Server.MapPath("~/Images/Thumbnails/"); //путь
на сервере для сохранения
        ThumbnailGenerator.ResizeImage(postedFile, thumbNailPath + fileName, 90);

        string userId = User.Identity.GetUserId();
        CocktionContext db = new CocktionContext();
```

```
        var user = db.AspNetUsers.Find(userId);
        Picture picture = new Picture();
        picture.FileName = fileName;
        picture.FilePath = thumbNailPath;
        db.Pictures.Add(picture);
        user.Selfie = picture;
        await DbItemsAdder.SaveDb(db);
        return new StatusHolder(true);
    }
    catch
    {
        return new StatusHolder(false);
    }

}

/// <summary>
/// Служит в качестве контейнера для полей, котоыре
/// пользователь хочет поменять из мобильного клиента
/// </summary>
public class ShortProfile
{
    public string title { get; set; }
    public string surname { get; set; }
    public string society { get; set; }
    public string city { get; set; }
}

/// <summary>
/// Позволяет пользователю редактировать информацию о профиле
/// Можно менять:
/// 1) Имя
/// 2) Фамилию
/// 3) Город
/// 4) Общество
/// </summary>
/// <param name="editInfo">Параметры для изменения</param>
/// <returns>Стандартный ответ сервера</returns>
[HttpPost]
[Authorize]
public async Task<StatusHolder> EditProfile(ShortProfile editInfo)
{
    try
    {
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());

        if (!String.IsNullOrEmpty(editInfo.title))
```

```
            user.UserRealName = editInfo.title;

        if (!String.IsNullOrEmpty(editInfo.surname))
            user.UserRealSurname = editInfo.surname;

        if (!String.IsNullOrEmpty(editInfo.society))
            user.SocietyName = editInfo.society;

        if (!String.IsNullOrEmpty(editInfo.city))
            user.City = editInfo.city;

        await DbItemsAdder.SaveDb(db);
        return new StatusHolder(true);
    }
    catch
    {
        return new StatusHolder(false);
    }

}

/// <summary>
/// Позволяет получить список всех людей, на которых подписался
/// данный пользователь
/// </summary>
/// <returns>Коллекцию информаторов</returns>
[HttpPost]
[Authorize]
public List<UserInfo> GetMyInformators()
{
    CocktionContext db = new CocktionContext();
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    List<UserInfo> informators = new List<UserInfo>();
    foreach (var i in user.Friends)
    {
        try
        {
            if (!String.IsNullOrEmpty(i.UserRealName))
                informators.Add(new UserInfo(i.Id, i.UserRealName, i.Selfie.FileName, true));
            else
                informators.Add(new UserInfo(i.Id, i.UserName,  i.Selfie.FileName, true));
        }
        catch
        {

        }
    }
    return informators;
```

```
        }
    }
}
```

1.2.6. Класс UsersController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Microsoft.AspNet.Identity;
using CocktionMVC.Models.JsonModels;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels.MobileClientModels;
namespace CocktionMVC.Controllers.ApiControllers
{
    public class UsersController : ApiController
    {
        public class UserIdHolder
        {
            public string id { get; set; }
        }

        /// <summary>
        /// Позволяет подписаться на пользователя с мобильного клиента
        /// </summary>
        /// <param name="id">Айдишник пользователя, на которого надо подписаться</param>
        /// <returns>Стандартный ответ сервера</returns>
        [HttpPost]
        [Authorize]
        public StatusHolder SubscribeOnUser(UserIdHolder id)
        {
            try
            {
                CocktionContext db = new CocktionContext();
                var user = db.AspNetUsers.Find(User.Identity.GetUserId());
                var userToSubscribeOn = db.AspNetUsers.Find(id.id);
                if (!user.Friends.Contains(userToSubscribeOn))
                {
                    user.Friends.Add(userToSubscribeOn);
                    db.SaveChanges();
                }
                return new StatusHolder(true);
            }
```

```
    catch
    {
        return new StatusHolder(false);
    }
}

/// <summary>
/// Позволяет отписаться от пользователя
/// </summary>
/// <param name="id">Айдишник пользователя, от которого надо отписаться</param>
/// <returns>Стандартный ответ сервера</returns>
[HttpPost]
[Authorize]
public StatusHolder UnsubscribeFromUser(UserIdHolder id)
{
    try
    {
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(User.Identity.GetUserId());
        var userToUnsubscribeOn = db.AspNetUsers.Find(id.id);
        if (user.Friends.Contains(userToUnsubscribeOn))
        {
            user.Friends.Remove(userToUnsubscribeOn);
            db.SaveChanges();
        }
        return new StatusHolder(true);
    }
    catch
    {
        return new StatusHolder(false);
    }
}

/// <summary>
/// Посылает в ответ на запрос список всех пользователей кокшна
/// </summary>
/// <returns>Коллекцию пользователей</returns>
[HttpPost]
[Authorize]
public List<UserInfo> GetAllUsers()
{
    CocktionContext db = new CocktionContext();
    List<UserInfo> users = new List<UserInfo>();
    var user = db.AspNetUsers.Find(User.Identity.GetUserId());
    foreach (var i in db.AspNetUsers.ToList())
    {
        try
```

```
            {
                users.Add(new UserInfo(i.Id, i.UserName, i.Selfie.FileName, user.Friends.Contains(i)));
            }
            catch
            {

            }
        }
        return users;
    }
  }
}
```

1.3. Код вспомогательных функций

1.3.1. Класс AuctionChecker

```
using CocktionMVC.Functions.Managers;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.Hubs;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Timers;
using System.Web;

namespace CocktionMVC.Functions
{
  public static class AuctionChecker
  {
    /// <summary>
    /// Запускает таймер для проверок аукйионов
    /// </summary>
    public static void StartChecking()
    {
      Timer aTimer = new Timer(3600000);
      aTimer.Elapsed += CheckAuctions;
      aTimer.AutoReset = true;
      aTimer.Enabled = true;
      //1 сек = 1000 млс
      //1 min = 60 000
      //1 hour = 3 600 000
    }

    /// <summary>
```

```
        /// Проверяет аукционы на то, закончились они или нет
        /// </summary>
        /// <param name="obj"></param>
        public static void CheckAuctions(Object source, ElapsedEventArgs e)
        {
            CocktionContext db = new CocktionContext();
            DateTime controllTime = DateTimeManager.GetCurrentTime();
            var acol = db.Auctions.ToList();
            foreach(Auction auction in acol)
            {
                //Если он активен, а время закончилось - завершаем аукцион, посылаем хозяину
имейликкк
                if (auction.IsActive && controllTime >= auction.EndTime)
                {
                    if (!auction.WinnerChosen)
                    {
                        //MessageHub.Send("Ваш аукцион закончен! Необходимо выбрать лидера!", "Кокшн",
                          //  auction.Owner.UserName, null, auction.Owner.Id);
                        PrivateMessage message = new PrivateMessage("Ваш аукцион закончен и необходимо
выбрать лидера",
                            "Коки", auction.Owner.UserName, DateTimeManager.GetCurrentTime());
                        auction.Owner.ChatMessages.Add(message);
                    }
                    else
                    {
                        FinishAuctionManager.FalseAuctionStatus(db, auction);
                    }
                }
            }
            db.SaveChanges();
        }
    }
}
```

1.3.2. Класс DataFormatter

```
using System;
using System.Collections.Generic;
using System.Linq;

// ReSharper disable once CheckNamespace
namespace CocktionMVC.Functions
{
    public class DataFormatter
    {
        /// <summary>
```

```
/// Метод для формирвания данных из словаря
/// для последующей отправки клиентам
/// Данный метод формирует строку с информацией о коэффициентах
/// для всех продуктов.
/// </summary>
/// <param name="data">Словарь, который необходимо отформатировать</param>
/// <returns>Массив типа string с данными</returns>
public static string DictionaryConverter(Dictionary<string, double> data)
{
   string result ="";
   int dictLength = data.Count;
   if (dictLength == 0)
   {
      result = "Еще никто из пользователей не поставил ставки на аукционе";
   }




   else
   {
      //Достаем все ключи и значения из словаря
      string[] key = data.Select(x => x.Key).ToArray();
      double[] value = data.Select(x => x.Value).ToArray();

      //Формируем результирующий массив данных
      for (int i = 0; i < dictLength; i++)
      {
         result += (String.Format("{0} => {1:f2} \n", key[i], value[i]));
      }

   }

   //возвращаем результат
   return result;
}

/// <summary>
/// Из строки в формате 1!2!34!48
/// достает айдишники аукционных домов
/// </summary>
/// <param name="housesId">Строка с зашифрованными айдишниками домов</param>
/// <returns>Интовый массив с номерами</returns>
public static int[] GetHouseIds(string housesId)
{
   string[] locIds = housesId.Split('!');
   int[] locIdsInt = Array.ConvertAll(locIds, x => int.Parse(x));
```

```
            return locIdsInt;
        }
    }
}
```

1.3.3. Класс DateTimeManager

```
using System;
using CocktionMVC.Models.DAL;

// ReSharper disable once CheckNamespace
namespace CocktionMVC.Functions
{
    /// <summary>
    /// Контейнер для функции, отвечающих за работу со временем
    /// </summary>
    public static class DateTimeManager
    {
        /// <summary>
        /// Конвертирует время в UTC во время в текущем часовом поясе
        /// (Пока что только Москва)
        /// </summary>
        /// <returns>Время в данный момент, в часовом поясе Москвы</returns>
        public static DateTime GetCurrentTime()
        {
            DateTime controlTime = DateTime.Now;
            TimeZoneInfo tst = TimeZoneInfo.FindSystemTimeZoneById("Russian Standard Time");
            controlTime = TimeZoneInfo.ConvertTime(controlTime, TimeZoneInfo.Local, tst);
            return controlTime;
        }

        /// <summary>
        /// Устанавливает время окончания аукциона и время
        /// его начала
        /// </summary>
        /// <param name="auction">Аукцион, у которого надо все установить</param>
        /// <param name="hoursString">Длительность в часах</param>
        /// <param name="minutesString">Длительность в минутах</param>
        public static void SetAuctionStartAndEndTime(Auction auction, string hoursString, string
minutesString)
        {
            //получаем текущее время и устанавливаем время старта
            DateTime auctionsEndTime = GetCurrentTime();
            DateTime auctionStartTime = auctionsEndTime;

            //преобразование времени из строк в числа
            int minutes = int.Parse(minutesString);
```

```
    int hours = int.Parse(hoursString);

    //вычисляем время окончания аукциона
    auctionsEndTime = auctionsEndTime.AddHours(hours);
    auctionsEndTime = auctionsEndTime.AddMinutes(minutes);

    //устанавливаем время окончания и начала аукциона
    auction.EndTime = auctionsEndTime;
    auction.StartTime = auctionStartTime;
}

/// <summary>
/// Устанавливает время начала и окончания аукциона.
/// </summary>
/// <param name="auction">Аукцион, у которого надо установить временные границы</param>
/// <param name="timeBound">
/// На вход подается строка, которую сервер принимает с клиента
/// Возможны 3 варианта:
/// 1) 4hourTime - аукцион нужно завершить через 4 часа
/// 2) 1dayTime - аукцион нужно завершить через 1 день
/// 3) 1weekTime - аукцион нужно завершить через 1 неделю
/// </param>
public static void SetAuctionStartAndEndTime(Auction auction, string timeBound)
{
    //получаем текущее время и устанавливаем время старта
    DateTime auctionsEndTime = GetCurrentTime();
    DateTime auctionStartTime = auctionsEndTime;




    //Вычисляем время окончания аукциона
    switch (timeBound)
    {
        case "4hoursTime":
            auctionsEndTime = auctionsEndTime.AddHours(4);
            break;
        case "12hoursTime":
            auctionsEndTime = auctionsEndTime.AddHours(12);
            break;
        case "1dayTime":
            auctionsEndTime = auctionsEndTime.AddDays(1);
            break;
        case "1weekTime":
            auctionsEndTime = auctionsEndTime.AddDays(7);
            break;
        case "3minutesTime":
```

```
                auctionsEndTime = auctionsEndTime.AddMinutes(3);
                break;
            }

            //устанавливаем время окончания и начала аукциона
            auction.EndTime = auctionsEndTime;
            auction.StartTime = auctionStartTime;
        }
    }
}
```

1.3.4. Класс DbItemsAdder

```
using System.Threading.Tasks;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.Hubs;

namespace CocktionMVC.Functions
{
    /// <summary>
    /// Методные контейнер, для хранения солдат,
    /// работающих с базой данных в асинхроне
    /// </summary>
    public static class DbItemsAdder
    {
        /// <summary>
        /// Добавляет product и photo в базу данных и сохраняет
        /// изменения
        /// </summary>
        /// <param name="db">База, в которую надо добавить изменения</param>
        /// <param name="product">Сущность продукта, которую надо добавить</param>
        /// <param name="photo">Фото, которое надо добваить</param>
        /// <param name ="cluster">Хранилище ставок, в которое надо вставить данный продукт</
param>
        /// <returns></returns>
        public static async Task AddProduct(CocktionContext db, Product product, Picture photo,
BidCluster cluster)
        {
            db.Products.Add(product);
            db.Pictures.Add(photo);
            db.AuctionBids.Add(cluster);
            await Task.Run(() => db.SaveChangesAsync());
        }

        /// <summary>
        /// Просто асинхронно сохраняет изменения в базе данных
```

```
        /// </summary>
        /// <param name="db">база, в которой нужно все поменять</param>
        /// <returns>г</returns>
        public static async Task SaveDb(CocktionContext db)
        {
            await Task.Run(() => db.SaveChangesAsync());
        }

        /// <summary>
        /// Асинхронно добавляет аукцион,
        /// товар, фотку в базу
        /// </summary>
        /// <param name="db">База, в которую необходимо добавить информацию</param>
        /// <param name="auction">Аукцион, который необходимо добавить</param>
        /// <param name="product">Товар, который необходимо добавить</param>
        /// <param name="photo">фотка, которую надо добавить</param>
        /// <param name="user"></param>
        public static async Task AddAuctionProductPhotoAsync(CocktionContext db, Auction auction,
            Product product, Picture photo, AspNetUser user)
        {
            db.Auctions.Add(auction);
            db.Products.Add(product);
            db.Pictures.Add(photo);
            user.HisAuctions.Add(auction);
            user.HisProducts.Add(product);
            await Task.Run(() => db.SaveChangesAsync());
        }

    }
}
```

1.3.5. Класс HouseSerializer

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace CocktionMVC.Functions.DataProcessing
{
    public class HouseSerializer
    {
        public static int[] TakeHouseIdsFromString(string housesString)
        {
            string[] idString = housesString.TrimEnd('!').TrimStart('?').Split('!');
```

```
            int[] ids = new int[idString.Length];
            for (int i = 0; i < idString.Length; i++)
            {
                ids[i] = int.Parse(idString[i]);
            }
            return ids;
        }
    }
}
```

1.3.6. Класс PhotoProcessor

```
using System;
using System.IO;
using System.Web;
using CocktionMVC.Models.DAL;

namespace CocktionMVC.Functions.DataProcessing
{
    /// <summary>
    /// Контейнер для методов, которые работают с фоточками
    /// </summary>
    public class PhotoProcessor
    {
        public static void CreateAndSavePicture(Picture picture, HttpRequestBase requestBase, int height)
        {
            HttpPostedFileBase file = null;
            if (requestBase.Files.Count != 0)
            {
                file = requestBase.Files[0];
            }

            if (file != null)
            {
                //Получаем информацию о том, откуда ж взялся файлик
                //string fileName = Path.GetFileName(file.FileName); //имя файла
                string extension = Path.GetExtension(file.FileName);
                string fileName = Guid.NewGuid() + extension;

                //создаем мини-картинку
                string thumbNailPath = requestBase.MapPath("~/Images/Thumbnails/"); //путь на сервере для
сохранения
                picture.FileName = fileName;
                picture.FilePath = thumbNailPath + fileName;
                ThumbnailGenerator.ResizeImage(file, picture.FilePath, height);
            }
            else
```

```
            {
                const string placeHolderName = "placeholder.jpg";
                picture.FileName = placeHolderName;
                string path = Path.Combine(
                    requestBase.MapPath("~/Content/SiteImages"), placeHolderName);
                picture.FilePath = path;
            }
        }
    }
}
```

1.3.7. Класс RequestFormReader

```
using System.Web;

namespace CocktionMVC.Functions
{
    /// <summary>
    /// Контейнер для методов, считывающих значения с форм на клиентах
    /// либо получающих инфу о каких-то полях из запроса
    /// </summary>
    public class RequestFormReader
    {
        /// <summary>
        /// Читает данные из формы с добавлением ставки на тотализаторе
        /// </summary>
        /// <param name="request">Запрос, который содержит форму с информацией о ставке на тотализаторе</param>
        /// <param name="auctionId">Айдишник аукциона, к которому добавляют ставку на тотализатор</param>
        /// <param name="productId">Айдишника товара, на который ставят яйца</param>
        /// <param name="eggsAmount">Количество яиц, которые хотят поставить</param>
        public static void ReadAddRateForm(HttpRequestBase request, out int auctionId,
            out int productId, out int eggsAmount)
        {
            //получаем информацию из формы
            string auctionIdStr = request.Form.GetValues("auctionId")[0];
            string productIdStr = request.Form.GetValues("productId")[0];
            string eggsAmountStr = request.Form.GetValues("eggsAmount")[0];

            //записываем значения
            auctionId = int.Parse(auctionIdStr);
            productId = int.Parse(productIdStr);
            eggsAmount = int.Parse(eggsAmountStr);
        }

        /// <summary>
```

```
/// Читает данные из формы с изменением информации профиля пользователя
/// </summary>
/// <param name="request">Запрос с формой</param>
/// <param name="name">Имя пользователя</param>
/// <param name="surname">Фамилия пользователя</param>
/// <param name="school">Универ/школа пользователя</param>
public static void ReadEditUserInfo(HttpRequestBase request, out string name,
   out string surname, out string school)
{
   name = request.Form.GetValues("name")[0];
   surname = request.Form.GetValues("surname")[0];
   school = request.Form.GetValues("school")[0];
}


/// <summary>
/// Читает данные из формы с добавлением лидера аукциона
/// </summary>
/// <param name="request">Запрос, который содержит форму с информацией о лидере</param>
/// <param name="auctionId">Айдишник аукциона, на котором все происходит</param>
/// <param name="productId">Айдишник товара, на котором все происходит</param>
public static void ReadAddLiderForm(HttpRequestBase request, out string auctionId,
   out string productId)
{
   auctionId = request.Form.GetValues("auctionId")[0];
   productId = request.Form.GetValues("productId")[0];
}

/// <summary>
/// Читает данные из формы с созданием аукциона.
/// </summary>
/// <param name="request">Запрос, содержащий форму с информацией об аукционе</param>
/// <param name="name">Наименование товара</param>
/// <param name="description">Описание товара</param>
/// <param name="category">Категория, в которую товар плавненько попадает</param>
/// <param name="timeBound">Временной промежуток, который вводит пользователь</param>
public static void ReadCreateAuctionForm(HttpRequestBase request, out string name,
   out string description, out string category, out string timeBound, out string housesIds)
{
   name = request.Form.GetValues("name")[0].Trim();
   description = request.Form.GetValues("description")[0].Trim();
   //если вдруг пользователь решил обойтись без описания
   if (description == "")
   {
      description = "Слишком круто, чтобы описывать :)";
   }
   category = request.Form.GetValues("category")[0].Trim();
```

```
    timeBound = request.Form.GetValues("timeBound")[0].Trim();
    housesIds = request.Form.GetValues("housesIds")[0].Trim();
}

/// <summary>
/// Читает данные из формы с созданием аукциона. (Мобильный клиент)
/// </summary>
/// <param name="request">Запрос, содержащий форму с информацией об аукционе</param>
/// <param name="name">Наименование товара</param>
/// <param name="description">Описание товара</param>
/// <param name="category">Категория, в которую товар плавненько попадает</param>
/// <param name="housesId">Айдишники домов, в которых торгуется данный товар</param>
/// <param name="minutes">Количество минут, которые будет длиться аукцион</param>
/// <param name="hours">Количество часов продолжительности аукциона</param>
public static void ReadCreateAuctionFormMobile(HttpRequest request, out string name,
    out string description, out string category, out string minutes,
    out string hours)
{
    name = request.Form.GetValues("name")[0].Trim();
    description = request.Form.GetValues("description")[0].Trim();
    category = request.Form.GetValues("category")[0].Trim();
    minutes = request.Form.GetValues("minutes")[0].Trim();
    hours = request.Form.GetValues("hours")[0].Trim();
}

/// <summary>
/// Читает данные из формы с добавлением ставки на аукцион
/// </summary>
/// <param name="requestBase">Запрос, содержащий форму с информацией о товаре</param>
/// <param name="name">Наименование товара</param>
/// <param name="auctionId">Айдишник аукциона</param>
/// <param name="category">Категория, к которой относится товар</param>
/// <param name="description">Описание товара</param>
public static void ReadAddProductBetForm(HttpRequestBase requestBase, out string name,
    out string auctionId, out string category, out string description)
{
    name = requestBase.Form.GetValues("name")[0].Trim();
    description = requestBase.Form.GetValues("description")[0].Trim();
    if (description == "")
        description = "Слишком круто, чтобы описывать!";
    category = requestBase.Form.GetValues("category")[0].Trim();
    auctionId = requestBase.Form.GetValues("auctionId")[0].Trim();
}

/// <summary>
/// Читает данные из формы с добавлением ставки на аукцион (для мобильника)
/// </summary>
```

```
        /// <param name="requestBase">Запрос, содержащий форму с информацией о товаре</param>
        /// <param name="name">Наименование товара</param>
        /// <param name="auctionId">Айдишник аукциона</param>
        /// <param name="category">Категория, к которой относится товар</param>
        /// <param name="description">Описание товара</param>
        public static void ReadAddProductBetForm(HttpRequest requestBase, out string name,
            out string auctionId, out string category, out string description)
        {
            name = requestBase.Form.GetValues("name")[0].Trim();
            description = requestBase.Form.GetValues("description")[0].Trim();
            category = requestBase.Form.GetValues("category")[0].Trim();
            auctionId = requestBase.Form.GetValues("auctionId")[0].Trim();
        }

        /// <summary>
        /// Считывает значения из формы, которая позволяет добавлять дома в базу данных
        /// </summary>
        /// <param name="requestBase">Запрос, в котором все поступает</param>
        /// <param name="university"></param>
        /// <param name="faculty"></param>
        /// <param name="adress"></param>
        public static void ReadAddHouseForm(HttpRequestBase requestBase,
            out string faculty, out string adress, out string holderId)
        {
            holderId = requestBase.Form.GetValues("holderId")[0].Trim();
            faculty = requestBase.Form.GetValues("faculty")[0].Trim();
            adress = requestBase.Form.GetValues("adress")[0].Trim();
        }
    }
}
```

1.3.8. Класс ThumbnailGenerator

```
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Web;

namespace CocktionMVC.Functions
{
    public class ThumbnailGenerator
    {
        /// <summary>
        /// Метод, который на выходе дает картинку
        /// любого, наперед заданного размера!!!
        /// </summary>
```

```
/// <param name="file">Файл, который надо поменять</param>
/// <param name="fileFullPath">Полный путь и имя файла</param>
/// <param name="width">Необходимая ширина картинки</param>
/// <param name="height">Необходимая высота картинки</param>
public static void ResizeImage(HttpPostedFileBase file, string fileFullPath, int height)
{
    //Стрим для сохранения файла после ресайза
    FileStream stream = new FileStream(fileFullPath, FileMode.OpenOrCreate);

    //Конвертация файла в картинку
    Image originalImage = Image.FromStream(file.InputStream);

    int width = CountWidth(originalImage.Width, originalImage.Height, height);

    //Создание нового битмапа с размером картинки
    Bitmap bitMapTempImg = new Bitmap(width, height);

    //Создание новой картинки, содержащей настройки качества
    Graphics newImage = Graphics.FromImage(bitMapTempImg);
    newImage.CompositingQuality = CompositingQuality.HighQuality;
    newImage.SmoothingMode = SmoothingMode.HighQuality;
    newImage.InterpolationMode = InterpolationMode.HighQualityBicubic;

    //Создаем прямоугольник и рисуем картинку
    Rectangle imageRectangle = new Rectangle(0, 0, width, height);
    newImage.DrawImage(originalImage, imageRectangle);

    //Сохраняем получившееся изображение
    bitMapTempImg.Save(stream, originalImage.RawFormat);

    //Зачистка ресурсов
    newImage.Dispose();
    bitMapTempImg.Dispose();
    originalImage.Dispose();
    stream.Close();
    stream.Dispose();

}

/// <summary>
/// Создает уменьшенную копию фотки заданной ширины и высоты
///
/// ВЕРСИЯ ДЛЯ МОБИЛЬНОГО КЛИЕНТА
/// </summary>
/// <param name="file">Файл с фоткой</param>
/// <param name="serverFullPath">Полный путь к новой фотке на сервере</param>
/// <param name="width">Ширина (которую надо сделать)</param>
```

```
/// <param name="height">Высота (которую хотим получить)</param>
public static void ResizeImage(HttpPostedFile file, string serverFullPath, int height)
{

    //Стрим для сохранения файла после ресайза
    FileStream stream = new FileStream(serverFullPath, FileMode.OpenOrCreate);

    //Конвертация файла в картинку
    Image originalImage = Image.FromStream(file.InputStream);

    int width = CountWidth(originalImage.Width, originalImage.Height, height);

    //Создание нового битмапа с размером картинки
    Bitmap bitMapTempImg = new Bitmap(width, height);

    //Создание новой картинки, содержащей настройки качества
    Graphics newImage = Graphics.FromImage(bitMapTempImg);
    newImage.CompositingQuality = CompositingQuality.HighQuality;
    newImage.SmoothingMode = SmoothingMode.HighQuality;
    newImage.InterpolationMode = InterpolationMode.HighQualityBicubic;

    //Создаем прямоугольник и рисуем картинку
    Rectangle imageRectangle = new Rectangle(0, 0, width, height);
    newImage.DrawImage(originalImage, imageRectangle);

    //Сохраняем получившееся изображение
    bitMapTempImg.Save(stream, originalImage.RawFormat);

    //Зачистка ресурсов
    newImage.Dispose();
    bitMapTempImg.Dispose();
    originalImage.Dispose();
    stream.Close();
    stream.Dispose();

}

/// <summary>
/// Считает оптимальную ширину по заданой высоте
/// </summary>
/// <param name="sourceWidth">Исходная ширина</param>
/// <param name="sourceHeight">Исходная высота</param>
/// <param name="newHeight">Нужная высота</param>
/// <returns>Значение нужной ширины</returns>
private static int CountWidth(int sourceWidth, int sourceHeight, int newHeight)
{
    return (newHeight*sourceWidth)/sourceHeight;
```

```
        }
    }
}
```

1.3.9. Класс FinishAuctionManager

```
using CocktionMVC.Models.DAL;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;

namespace CocktionMVC.Functions.Managers
{
    public class FinishAuctionManager
    {
        /// <summary>
        /// Рассылает сообщения владельцу и победителю аукциона
        /// </summary>
        /// <param name="auction"></param>
        private static void AddAndSendMsgToOwnerAndWinner(Auction auction)
        {
            //сообщение владельцу
            string messageToOwner = "Привет, я - тот, кого ты выбрал!;) (мгновенная связь с
победителем)";
            PrivateMessage toOwner = new PrivateMessage(messageToOwner,
                auction.LeadProduct.Owner.UserName, auction.Owner.UserName,
DateTimeManager.GetCurrentTime());
            auction.LeadProduct.Owner.ChatMessages.Add(toOwner);
            auction.Owner.ChatMessages.Add(toOwner);
            if (auction.Owner.MobileDevice != null)
                Notificator.SendNotification(auction.Owner.MobileDevice, messageToOwner, 5);


            //сообщение победителю
            string messageToWinner = "Привет! Я - владелец! (мгновенная связь с владельцем)";
            PrivateMessage toWinner = new PrivateMessage("Привет! Я - владелец!",
                auction.Owner.UserName, auction.LeadProduct.Owner.UserName,
DateTimeManager.GetCurrentTime());
            auction.LeadProduct.Owner.ChatMessages.Add(toWinner);
            auction.Owner.ChatMessages.Add(toWinner);
            if (auction.LeadProduct.Owner.MobileDevice != null)
                Notificator.SendNotification(auction.LeadProduct.Owner.MobileDevice, messageToWinner, 5);
        }
```

```
        /// <summary>
        /// Деактивирует аукцион и асинхронно
        /// выполняет сохранение данных в базу
        /// </summary>
        /// <param name="db">База, в которую надо сохранить изменения</param>
        /// <param name="auction">Аукцион, который необходимо завершить</param>
        public static void FalseAuctionStatus(CocktionContext db, Auction auction)
        {

            //заканчиваем аукцион
            auction.IsActive = false;

            //вырубаем тотализатор
            auction.AuctionToteBoard.FinishTote(auction.LeadProduct.Id, db);

            //отправляем сообщение владельцу и победителю
            AddAndSendMsgToOwnerAndWinner(auction);
        }

        /// <summary>
        /// Деактивирует аукцион и асинхронно
        /// выполняет сохранение данных в базу
        /// </summary>
        /// <param name="db">База, в которую надо сохранить изменения</param>
        /// <param name="auctionId">Айди аукциона</param>
        /// <returns></returns>
        public static async Task FalseAuctionStatus(CocktionContext db, int auctionId)
        {
            //заканчиваем аукцион
            Auction auction = db.Auctions.Find(auctionId);
            auction.IsActive = false;

            //отправляем сообщения
            AddAndSendMsgToOwnerAndWinner(auction);

            //вырубаем тотализатор
            auction.AuctionToteBoard.FinishTote(auction.LeadProduct.Id, db);

            //сохраняем изменения в базу данных
            await Task.Run(() => db.SaveChangesAsync());
        }

    }
}
```

1.3.10. Класс RatingManager

```
using System.Threading.Tasks;
using System.Web.Mvc;
using CocktionMVC.Models.DAL;
using WebGrease;
namespace CocktionMVC.Functions
{
    /// <summary>
    /// Содержит методы для работы с рейтингом у
    /// 1) Аукционов
    /// 2) Домов
    /// 3) Пользователей
    /// </summary>
    public class RatingManager
    {
        /// <summary>
        /// Увеличивает рейтинг аукциона
        ///
        /// В строке reasonOfInc передается параметр, в
        /// зав-ти от которого увеличивается рейтинг аукциона.
        /// Есть следующие варианты:
        /// </summary>
        /// <param name="auction">Аукцион, который надо увеличить</param>
        /// <param name="user">Инстанс пользователя</param>
        /// <param name="reasonOfInc">
        /// Параметр для увеличения
        /// 1) userBeted - если пользователь поставил ставку/создал аукцион -
        /// увеличиваем рейтинг аукциона на 0.4 * рейтинг пользователя
        ///
        /// 2) newVisit - кто-то посмотрел аукцион - добавляем 20 очков
        /// </param>
        [Authorize]
        public static void IncreaseRating(Auction auction,  AspNetUser user, string reasonOfInc)
        {
            switch (reasonOfInc)
            {
                case "userBeted":
                    auction.Rating += (int) (user.Rating*0.4);
                    break;
                case "newVisit":
                    auction.Rating += 20;
                    break;
            }
        }
```

```
/// <summary>
/// Увеличивает рейтинг пользователя
///
/// В строке reasonOfInc подается причина увеличения рейтинга
/// всего может быть три варианта:
/// </summary>
/// <param name="user">Пользователь, которому надо увеличить рейтинг</param>
/// <param name="reasonOfInc">
/// Строка, содержащая причину
/// 1) userMadeAuction => добавляем 16 очков
/// 2) userGotSubscriber => добавляем 16 очков
/// 3) userPlacedBet => добаляем 8 очков
/// </param>
[Authorize]
public static void IncreaseRating(AspNetUser user,  string reasonOfInc)
{
   switch (reasonOfInc)
   {
      case "userMadeAuction":
         user.Rating += 16;
         break;
      case "userGotSubscriber":
         user.Rating += 16;
         break;
      case "userPlacedBet":
         user.Rating += 8;
         break;
   }
}

/// <summary>
/// Увеличивает рейтинг дома
/// </summary>
/// <param name="house">Дом, у которого надо увеличить рейтинг</param>
/// <param name="reasonOfInc">Причина увеличения рейтинга
/// 1) auctionAdded => + 16
/// 2) subscriberAdded => + 16
/// 3) likeAdded => +8</param>
[Authorize]
public static void IncreaseRating(House house, string reasonOfInc)
{
   switch (reasonOfInc)
   {
      case "auctionAdded":
         house.Rating += 16;
         break;
      case "subscriberAdded":
```

```
                house.Rating += 16;
                break;
            case "likeAdded":
                house.Rating += 8;
                break;
        }
    }
  }
}
```

1.3.11. Класс ToteResultsManager

```
using System;
using System.Linq;
using CocktionMVC.Models;
using CocktionMVC.Models.DAL;
using CocktionMVC.Models.JsonModels;

namespace CocktionMVC.Functions
{
  /// <summary>
  /// Контейнер для методов, связанных с окончанием тотализатора и
  /// получением его результатов
  /// </summary>
  public class ToteResultsManager
  {
    /// <summary>
    /// Получает результаты тотализатора.
    /// Инициализирует необходимые поля в классе,
    /// который потом вернется джейсоном клиентам
    /// (Версия для проигравших аукцион)
    /// </summary>
    /// <param name="auction">Аукцион, на котором был тотализатор</param>
    /// <param name="userId">Айди пользователя</param>
    /// <param name="currentBidseller">Существующий объект BidSeller'a</param>
    /// <param name="currentUser">Текущий пользователь</param>
    /// <param name="winProductName">Название продукта - победителя аукциона</param>
    public static void GetToteResults(Auction auction, string userId, BidSeller currentBidseller,
      AspNetUser currentUser, string winProductName)
    {
      //Выбираем результаты тотализатора этого пользователя
      var q = (from x in auction.AuctionToteBoard.ToteResultsForUsers
               where (x.UserId == userId)
               select x);

      if (q.Count() != 0)
```

```
        {//если он ставил яйца
            //считаем прибыль
            currentBidseller.ProfitFromTote = q.First().Profit;
            currentBidseller.Message = String.Format("Дорогой, аукцион закончен и победил товар
{0}", winProductName) +
                        String.Format("Вы срубили бабла {0}", currentBidseller.ProfitFromTote) +
                        String.Format("У вас бабла тепрь {0}", currentUser.Eggs);
        }
        else
        {//если он не ставил яйца
            currentBidseller.Message = String.Format("Дорогой, аукцион закончен и победил товар
{0}", winProductName);
        }
    }

    /// <summary>
    /// Получает результаты тотализатора.
    /// Инициализирует необходимые поля в классе,
    /// который потом вернется джейсоном клиентам
    /// (Версия для победителя аукциона)
    /// </summary>
    /// <param name="auction">Аукцион, на котором был тотализатор</param>
    /// <param name="userId">Айдишник пользователя, который победил</param>
    /// <param name="owner">объект BidSeller'a</param>
    /// <param name="phone">Телефон владельца аукциона</param>
    /// <param name="currentUser">Текущий пользователь</param>



public static void GetToteResults(Auction auction, string userId, BidSeller owner, string phone,
        AspNetUser currentUser)
    {
        //Выбираем результаты тотализатора этого пользователя
        var q = (from x in auction.AuctionToteBoard.ToteResultsForUsers
                 where (x.UserId == userId)
                 select x);

        if (q.Count() != 0)
        {//если он что-нибудь ставил
            //считаем прибыль
            owner.ProfitFromTote = q.First().Profit;
            owner.Message = "Аукцион закончен, вам необходимо связаться с продавцом! " + phone +
"/n" +
                        String.Format("Вы срубили бабла {0} ", owner.ProfitFromTote) +
                        String.Format("У вас бабла тепрь {0}", currentUser.Eggs);
        }
```

```
            else
            {//если он ничего не ставил
                owner.Message = "Аукцион закончен, вам необходимо связаться с продавцом! " + phone;
            }
        }
    }
}
```

1.3.12. Класс EmailSender

```
using AegisImplicitMail;

namespace CocktionMVC.Functions
{
    /// <summary>
    /// Занимается отправкой писем
    /// </summary>
    public static class EmailSender
    {
        /// <summary>
        /// Содержит поле с текстом для подтверждения имейла
        /// </summary>
        public const string VERIFY_EMAIL_MESSAGE = "Please, click this link to verify your email :)";

        public const string VERIFY_EMAIL_SUBJECT = "Email validation";

        public const string RESET_PASSWORD_MESSAGE = "Please, help us to reset your password!";

        public const string RESET_PASSWORD_SUBJECT = "Some problems with password...";

        /// <summary>
        /// Отправляет имейл на указанный адрес с указанными данными.
        /// </summary>
        /// <param name="message">Текст сообщения</param>
        /// <param name="subject">Тема сообщения</param>
        /// <param name="adressToMail">Адрес, на который нужно отправить сообщение</param>
        public static void SendEmail(string message, string subject, string adressToMail)
        {
            const string MAIL = "cocky@cocktion.com";
            const string HOST = "smtp.yandex.ru";
            const int PORT = 465;
            const string USER = "cocky@cocktion.com";
            const string PASSWORD = "IloveCocktion290996";

            //Generate Message
            var mymessage = new MimeMailMessage();
```

```
            mymessage.From = new MimeMailAddress(MAIL);
            mymessage.To.Add(adressToMail);
            mymessage.Subject = subject;
            mymessage.Body = message;

            //Create Smtp Client
            var mailer = new MimeMailer(HOST, PORT);
            mailer.User = USER;
            mailer.Password = PASSWORD;
            mailer.SslType = SslMode.Ssl;
            mailer.AuthenticationMode = AuthenticationType.Base64;
            mailer.EnableImplicitSsl = true;
            mailer.SendMail(mymessage);
        }
    }
}
```

1.3.13. Класс Notificator

```
using CocktionMVC.Models.DAL;
using PushSharp;
using PushSharp.Apple;
using PushSharp.Core;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
namespace CocktionMVC.Functions
{
    public class Notificator
    {
        /// <summary>
        /// Брокер для служб пуш-нотификаций
        /// </summary>
        public static PushBroker Pusher { get; set;}


        /// <summary>
        /// Регистрирует службу для нотификаций эппл
        /// </summary>
        public static void RegisterAppleService()
        {
            try
            {
```

```
            Pusher = new PushBroker();
            WebClient client = new WebClient();
            var appleCert = client.DownloadData(@"http://cocktion.com/Content/
Cocktion.Push.Development.p12");
            client.Dispose();
            Pusher.RegisterAppleService(new ApplePushChannelSettings(false, appleCert,
"hateMicrosoftlove$"));
         }
         catch (Exception e)
         {
            EmailSender.SendEmail(e.Message, "auctionInfo", "lazarenko.ale@gmail.com");
         }

      }


      /// <summary>
      /// Посылает пуш на устройство
      /// </summary>
      /// <param name="device">Устройство, на которое надо послать</param>
      /// <param name="message">Сообщение в пуше, которое надо отобразить</param>
      /// <param name="badge">Бэдж, который надо показать</param>
      public static void SendNotification(Device device, string message, int badge)
      {
         if (Pusher == null)
            RegisterAppleService();

         switch(device.Type)
         {
            case "ios":
               Pusher.QueueNotification(new AppleNotification()
            .ForDeviceToken(device.Token)
            .WithAlert(message)
            .WithBadge(badge)
            .WithSound("sound.caf"));
               break;
            case "wp":
               //do smth here
               break;
            case "android":
               //do smth here
               break;
         }
      }
   }
}
```

1.4. Код моделей

1.4.1. Класс AspNetUser

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace CocktionMVC.Models.DAL
{
   public class AspNetUser
   {
      public AspNetUser()
      {
         AspNetUserClaims = new HashSet<AspNetUserClaim>();
         AspNetUserLogins = new HashSet<AspNetUserLogin>();
         AspNetRoles = new HashSet<AspNetRole>();
         HisAuctions = new HashSet<Auction>();
         HisProducts = new HashSet<Product>();
         SubHouses = new HashSet<House>();
         Friends = new HashSet<AspNetUser>();
         ChatMessages = new HashSet<PrivateMessage>();
         //Selfie = new Picture("anonPhoto.jpg");
      }

      public string Id { get; set; }

      [StringLength(256)]
      public string Email { get; set; }

      public bool EmailConfirmed { get; set; }

      public string PasswordHash { get; set; }

      public string SecurityStamp { get; set; }

      public string PhoneNumber { get; set; }

      public int Eggs { get; set; }

      public bool PhoneNumberConfirmed { get; set; }

      public bool TwoFactorEnabled { get; set; }

      public DateTime? LockoutEndDateUtc { get; set; }
```

```
        public bool LockoutEnabled { get; set; }

        public int AccessFailedCount { get; set; }

        [Required]
        [StringLength(256)]
        public string UserName { get; set; }

        public string UserRealName { get; set; }

        public string UserRealSurname { get; set; }

        public string SocietyName { get; set; }

        public int? Rating { get; set; }

        public string City { get; set; }

        public virtual Picture Selfie {get; set; }
        public virtual Device MobileDevice { get; set; }
        public virtual ICollection<House> SubHouses { get; set; }
        public virtual ICollection<Auction> HisAuctions { get; set; }
        public virtual ICollection<Product> HisProducts { get; set; }

        public virtual ICollection<AspNetUserClaim> AspNetUserClaims { get; set; }

        public virtual ICollection<AspNetUserLogin> AspNetUserLogins { get; set; }

        public virtual ICollection<AspNetRole> AspNetRoles { get; set; }

        public virtual ICollection<AspNetUser> Friends { get; set; }

        public virtual HouseHolder HisHouseHolder { get; set; }
        public virtual ICollection<Interest> Interests { get; set; }

        public virtual ICollection<PrivateMessage> ChatMessages { get; set; }
    }
}
```

1.4.2. Класс Auction

```
using System;
using System.Collections.Generic;

namespace CocktionMVC.Models.DAL
{
```

```
public class Auction
{
    public Auction()
    {
        BidProducts = new HashSet<Product>();
        Houses = new HashSet<House>();
        UsersBids = new HashSet<BidCluster>();
    }

    public Auction(bool isActive, Product product,
        bool winnerChosen, ToteBoard tote, AspNetUser user)
    {
        BidProducts = new HashSet<Product>();
        Houses = new HashSet<House>();
        UsersBids = new HashSet<BidCluster>();
        IsActive = isActive;
        SellProduct = product;
        WinnerChosen = winnerChosen;
        AuctionToteBoard = tote;
        Owner = user;
    }

    public int Id { get; set; }

    public DateTime EndTime { get; set; }

    public DateTime StartTime { get; set; }

    public bool IsActive { get; set; }

    public bool WinnerChosen { get; set; }

    public int SellProductId { get; set; }
    public int Rating { get; set; }
    public virtual ToteBoard AuctionToteBoard { get; set; }

    public virtual Product LeadProduct { get; set; }
    public virtual AspNetUser Owner { get; set; }

    public virtual Product SellProduct { get; set; }
    public virtual ICollection<BidCluster> UsersBids { get; set; }
    public virtual ICollection<House> Houses { get; set; }

    public virtual ICollection<Product> BidProducts { get; set; }
}
}
```

1.4.3. Класс BidCluster

```
using System.Collections.Generic;

namespace CocktionMVC.Models.DAL
{
   /// <summary>
   /// Моделирует сущность ставки (товарной) на аукционе.
   /// Нужен для того, чтобы можно было понять, в какие группы
   /// чувачок объединяет товары. И, чтобы можно было их спокойно выводить
   /// на экраны.
   /// </summary>
   public class BidCluster
   {
      public BidCluster(string userId, Auction auction) : this()
      {
         UserId = userId;
         HostAuction = auction;
      }

      public BidCluster()
      {
         Products = new HashSet<Product>();
      }
      public int Id { get; set; }
      public virtual Auction HostAuction { get; set; }

      public string UserId { get; set; }

      public virtual ICollection<Product> Products { get; set; }
   }
}
```

1.4.4. Класс Product

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace CocktionMVC.Models.DAL
{
   public class Product
   {
      public Product()
      {
         BidClusters = new HashSet<BidCluster>();
      }
```

```
        public Product(string name, string description, string category,
            bool isOnAuctionAsALot, Picture photo, AspNetUser user, Auction auction)
            : this(name,description, category, isOnAuctionAsALot, user)
        {
            Photo = photo;
            SelfAuction = auction;
        }

        public Product(string name, string description, string category,
            bool isOnAuctionAsALot, AspNetUser user) : this()
        {
            Name = name;
            Description = description;
            Category = category;
            IsOnAuctionAsALot = isOnAuctionAsALot;
            Owner = user;
        }

        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        public string Name { get; set; }

        [Required]
        [StringLength(350)]
        public string Description { get; set; }

        [Required]
        [StringLength(50)]
        public string Category { get; set; }

        public bool IsOnAuctionAsALot { get; set; }

        public virtual AspNetUser Owner { get; set; }

        public virtual ICollection<BidCluster> BidClusters { get; set; }

        public virtual Picture Photo { get; set; }

        public virtual Auction SelfAuction { get; set; }
    }
}
```

1.4.5. Класс CocktionContext

```
using System.Data.Entity;

namespace CocktionMVC.Models.DAL
{
   public class CocktionContext : DbContext
   {
      public CocktionContext()
         : base("name=DefaultConnection")
      {
      }
      public virtual DbSet<ToteBoard> ToteBoards { get; set; }
      public virtual DbSet<UsersFeedback> Feedbacks { get; set; }
      public virtual DbSet<C__MigrationHistory> C__MigrationHistory { get; set; }
      public virtual DbSet<AspNetRole> AspNetRoles { get; set; }
      public virtual DbSet<AspNetUserClaim> AspNetUserClaims { get; set; }
      public virtual DbSet<AspNetUserLogin> AspNetUserLogins { get; set; }
      public virtual DbSet<AspNetUser> AspNetUsers { get; set; }
      public virtual DbSet<Auction> Auctions { get; set; }
      public virtual DbSet<Product> Products { get; set; }
      public virtual DbSet<BidCluster> AuctionBids { get; set; }
      public virtual DbSet<ToteEntity> ToteEntities { get; set; }
      public virtual DbSet<ToteResult> ToteResults { get; set; }
      public virtual DbSet<House> Houses { get; set; }
      public virtual DbSet<ForumPost> Posts { get; set; }

      public virtual DbSet<Interest> Interests { get; set; }

      public virtual DbSet<HouseHolder> HouseHolders { get; set; }

      public virtual DbSet<Picture> Pictures { get; set; }

      public virtual DbSet<PrivateMessage> Messages { get; set; }

      public virtual DbSet<Device> Devices { get; set; }

      protected override void OnModelCreating(DbModelBuilder modelBuilder)
      {
         modelBuilder.Entity<Interest>()
            .HasMany(e => e.Subscribers);

         modelBuilder.Entity<AspNetUser>()
            .HasMany(e => e.Friends);

         modelBuilder.Entity<HouseHolder>()
            .HasMany(e => e.Houses);
```

```
modelBuilder.Entity<HouseHolder>()
   .HasMany(e => e.Users);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.Interests);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.ChatMessages);

modelBuilder.Entity<House>()
   .HasMany(e => e.Auctions);

modelBuilder.Entity<Auction>()
   .HasMany(e => e.Houses);

modelBuilder.Entity<House>()
   .HasMany(e => e.Posts);

modelBuilder.Entity<House>()
   .HasMany(e => e.Inhabitants);

modelBuilder.Entity<AspNetRole>()
   .HasMany(e => e.AspNetUsers)
   .WithMany(e => e.AspNetRoles)
   .Map(m => m.ToTable("AspNetUserRoles").MapLeftKey("RoleId").MapRightKey("UserId"));

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.AspNetUserClaims)
   .WithRequired(e => e.AspNetUser)
   .HasForeignKey(e => e.UserId);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.AspNetUserLogins)
   .WithRequired(e => e.AspNetUser)
   .HasForeignKey(e => e.UserId);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.HisAuctions);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.HisProducts);

modelBuilder.Entity<AspNetUser>()
   .HasMany(e => e.SubHouses);

//modelBuilder.Entity<Auction>()
//    .HasMany(e => e.BidProducts)  //Products1 ---> BidProducts
```

```
        //    .WithMany(e => e.BidAuctions)   //Auctions2 ---> BidAuctions
        //    .Map(m =>
m.ToTable("ProductAuction1").MapLeftKey("BidAuctions_Id").MapRightKey("BidProducts_Id"));



        modelBuilder.Entity<Auction>()
           .HasMany(e => e.UsersBids);

        modelBuilder.Entity<PrivateMessage>()
          .HasMany(e => e.Talkers);

        modelBuilder.Entity<BidCluster>()
           .HasRequired(e => e.HostAuction);

        modelBuilder.Entity<BidCluster>()
           .HasMany(e => e.Products);

        modelBuilder.Entity<Product>()
           .HasMany(e => e.BidClusters);

        modelBuilder.Entity<Product>()
           .Property(e => e.Name)
           .IsFixedLength();

        modelBuilder.Entity<Product>()
           .Property(e => e.Description)
           .IsFixedLength();

        modelBuilder.Entity<Product>()
           .Property(e => e.Category)
           .IsFixedLength();

        modelBuilder.Entity<Auction>()
           .HasRequired(e => e.AuctionToteBoard);

        modelBuilder.Entity<ToteBoard>()
           .HasMany(e => e.Bids);

        modelBuilder.Entity<ToteBoard>()
           .HasMany(e => e.ToteResultsForUsers);

        //modelBuilder.Entity<Auction>()
        //    .HasMany(e => e.BidProducts)
        //    .WithRequired(e => e.SelfAuction)
        //    .WillCascadeOnDelete(true);
```

```
            modelBuilder.Entity<Auction>()
                .HasMany(e => e.BidProducts);

            //modelBuilder.Entity<Product>()
            //    .HasRequired(e => e.SelfAuction)
            //    .WithMany(e => e.BidProducts)
            //    .WillCascadeOnDelete(true);

            //modelBuilder.Entity<Product>()
            //    .HasMany(e => e.Auctions)
            //    .WithRequired(e => e.SellProduct)
            //    .HasForeignKey(e => e.SellProductId)
            //    .WillCascadeOnDelete(false);


        }
    }
}
```

1.4.6. Класс Device

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CocktionMVC.Models.DAL
{
    public class Device
    {
        public int Id { get; set; }

        /// <summary>
        ///  ios/android/wp
        /// </summary>
        public string Type { get; set; }

        public string Token { get; set; }

        public Device() { }

        public Device(string type, string token)
        {
            Type = type;
            Token = token;
        }
```

```
    }
}
```

1.4.7. Класс Interest

```
using System.Collections.Generic;

namespace CocktionMVC.Models.DAL
{
    public class Interest
    {
        public Interest()
        {
            Subscribers = new List<AspNetUser>();
        }

        public Interest(string name, Picture photoCard)
        {
            Subscribers = new List<AspNetUser>();
            Name = name;
            Photocard = photoCard;
        }
        public int Id { get; set; }
        public string Name { get; set; }
        public virtual Picture Photocard { get; set; }
        public virtual ICollection<AspNetUser> Subscribers { get; set; }
    }
}
```

1.4.8. Класс Picture

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CocktionMVC.Models.DAL
{
    public class Picture
    {
        public Picture() { }
        public Picture(string fileName)
        {
            FileName = fileName;
        }
```

```
        public int Id { get; set; }

        public string FilePath { get; set; }

        public string FileName { get; set; }
    }
}
```

1.4.9. Класс PrivateMessage

```
using System;
using System.Collections.Generic;
using System.Web.UI.WebControls;

namespace CocktionMVC.Models.DAL
{
    public class PrivateMessage
    {
        public int Id { get; set; }
        public string Content { get; set; }
        public string ReceiverName { get; set; }
        public string AuthorName { get; set; }
        public DateTime DateOfPublishing { get; set; }

        public PrivateMessage(string message, string author, string receiver,
            DateTime date)
        {
            Content = message;
            AuthorName = author;
            ReceiverName = receiver;
            DateOfPublishing = date;
        }

        public PrivateMessage()
        {

        }

        public virtual ICollection<AspNetUser> Talkers { get; set; }
    }
}
```

1.4.10. Класс UsersFeedback

```
using System;
```

```
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Web;

namespace CocktionMVC.Models.DAL
{
   public class UsersFeedback
   {
      public int Id { get; set; }
      public string AuthorsName { get; set; }
      public string AuthorsSurname { get; set; }
      public string AuthorsId { get; set; }
      public string UsersId { get; set; }
      public string Message { get; set; }

      public UsersFeedback()
      { }

      public UsersFeedback(string authorsName, string authorsSurname, string authorsId,
         string usersId, string message)
      {
         AuthorsName = authorsName;
         AuthorsSurname = authorsSurname;
         AuthorsId = authorsId;
         UsersId = usersId;
         Message = message;
      }
   }
}
```

1.4.11. Класс ForumPost

```
namespace CocktionMVC.Models.DAL
{
   /// <summary>
   /// Моделирует пост на форуме
   /// </summary>
   public class ForumPost
   {
      public ForumPost(string message, string authorName)
      {
         Message = message;
         AuthorName = authorName;
         Likes = 0;
      }
```

```
        public ForumPost()
        { }

        /// <summary>
        /// Айдишник для базы данных
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Автор поста
        /// </summary>
        public string AuthorName { get; set; }

        /// <summary>
        /// Само сообщение
        /// </summary>
        public string Message { get; set; }

        /// <summary>
        /// Количество лайков
        /// </summary>
        public int Likes { get; set; }

        public virtual House HostHouse { get; set; }
    }
}
```

1.4.12. Класс House

```
using System.Collections.Generic;

namespace CocktionMVC.Models.DAL
{
    /// <summary>
    /// Моделирует аукционный дом!!!
    /// </summary>
    public class House
    {
        public House()
        {
            Auctions = new HashSet<Auction>();
            Posts = new HashSet<ForumPost>();
            Inhabitants = new HashSet<AspNetUser>();
        }
```

```
/// <summary>
/// Конструирует дом по заданным параметрам
/// </summary>
/// <param name="adress">Адрес дома</param>
/// <param name="faculty">Факультет / обшага</param>
/// <param name="holder">Холдер, к которому принадлежит дом</param>
/// <param name="photo">Фоточка, которая нужна для дома</param>
public House(string adress, string faculty, HouseHolder holder, Picture photo)
{
   Auctions = new HashSet<Auction>();
   Posts = new HashSet<ForumPost>();
   Adress = adress;
   Faculty = faculty;
   Likes = 0;
   Rating = 0;
   Holder = holder;
   Portrait = photo;
}

/// <summary>
/// Айдишник для базы данных
/// </summary>
public int Id { get; set; }

/// <summary>
/// Факультет, но если добавляется общага -
/// пишем сюда общагу
/// </summary>
public string Faculty { get; set; }

/// <summary>
/// Адрес
/// </summary>
public string Adress { get; set; }

/// <summary>
/// Количество лайков
/// </summary>
public int Likes { get; set; }

/// <summary>
/// Рейтинг
/// </summary>
public int Rating { get; set; }

public string Description { get; set; }
```

```
        /// <summary>
        /// Аукционы на этой площадке
        /// </summary>
        public virtual ICollection<Auction> Auctions { get; set; }

        /// <summary>
        /// Сообщения для форума.
        /// </summary>
        public virtual ICollection<ForumPost> Posts { get; set; }

        public virtual ICollection<AspNetUser> Inhabitants { get; set; }

        public virtual Picture Portrait { get; set; }

        public virtual HouseHolder Holder { get; set; }
    }
}
```

1.4.13. Класс HouseHolder

```
using System.Collections.Generic;
using CocktionMVC.Models.DAL;

namespace CocktionMVC.Models
{
    /// <summary>
    /// Модель для отображения хаус холдера
    ///
    /// Хаус холдер - контейнер для домов
    /// </summary>
    public class HouseHolder
    {
        public HouseHolder()
        {
            Houses = new HashSet<House>();
            Users = new HashSet<AspNetUser>();
        }

        public HouseHolder(string name, string city, Picture photo)
        {
            Houses = new HashSet<House>();
            Users = new HashSet<AspNetUser>();
            Name = name;
            PhotoCard = photo;
            City = city;
        }
```

```
        /// <summary>
        /// Айдишник для базы данных
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Название холдера
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Город, в котором нахоидтся холдер
        /// </summary>
        public string City { get; set; }

        /// <summary>
        /// Дома, которые находятся в этом хаусхолдере
        /// </summary>
        public virtual ICollection<House> Houses { get; set; }

        /// <summary>
        /// Пользователи в хаус холдере
        /// </summary>
        public virtual ICollection<AspNetUser> Users { get; set; }

        /// <summary>
        /// Аваторчка холдера
        /// </summary>
        public virtual Picture PhotoCard { get; set; }
    }
}
```

1.4.14. Класс ToteBoard

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using CocktionMVC.Functions;

namespace CocktionMVC.Models.DAL
{
    /// <summary>
    /// Класс, который реализует тотализатор.
    /// </summary>
    public class ToteBoard
```

```
{
    public ToteBoard()
    {
        IsActive = false;
        Bids = new HashSet<ToteEntity>();
        ToteResultsForUsers = new HashSet<ToteResult>();
    }

    /// <summary>
    /// Айдишник для базы данных.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Индикатор активности тотализатора
    /// </summary>
    public bool IsActive { get; set; }

    /// <summary>
    /// Общее количество яиц.
    /// </summary>
    public int TotalEggs { get; set; }

    /// <summary>
    /// Аукцион, для которого организован тотализатор.
    /// </summary>
    public virtual Auction ToteAuction { get; set; }

    /// <summary>
    /// результаты тотализатора с точки зрения пользователй
    /// </summary>
    public virtual ICollection<ToteResult> ToteResultsForUsers { get; set; }
    public virtual ICollection<ToteEntity> Bids { get; set; }

    /// <summary>
    /// Метод для подсчета коэффициента для конкретного продукта
    ///
    /// Коэффициент = яица на продукте / общее количество яиц в тотале
    /// </summary>
    /// <param name="productId">Айдишник продукта</param>
    /// <returns>Коэффициент на аукционе для продукта</returns>
    public double CountCoefficientForProduct(int productId)
    {
        var eggsCollection = (from x in Bids
                              where x.ProductId == productId
                              select x);
        double coefficient = 0;
```

```
    foreach (var egg in eggsCollection)
    {
        coefficient += egg.EggsAmount;
    }
    coefficient /= this.TotalEggs;
    return coefficient;
}

/// <summary>
/// Метод для подсчета коэффициента для конкретного пользователя
/// </summary>
/// <param name="userName">Имя пользователя</param>
/// <returns>Коэффициент пользователя</returns>
private double CountCoefficientForUser(string userId)
{

    //Ставку можно делать только один раз
    //выбираем все ставки данного пользователя
    var eggsCollection = (from x in Bids
                          where x.UserId == userId
                          select x);

    double coefficient = 0; //вводим коэффициент

    //суммируем все его ставки (она должна быть одна, но не важно)
    foreach (var i in eggsCollection)
        coefficient += i.EggsAmount;

    //выбираем все ставки, которые поставили на данный продукт
    var eggsOnProductCollection = (from x in Bids
                                   where x.ProductId == eggsCollection.First().ProductId
                                   select x);
    //суммируем количество яиц, поставленных на данный продукт
    double eggsOnProduct = 0;
    foreach (var i in eggsOnProductCollection)
        eggsOnProduct += i.EggsAmount;

    //рассчитываем коэффициент
    coefficient /= eggsOnProduct;
    return coefficient;
}

/// <summary>
/// Метод для расчета прибыли для конкретного пользователя
/// </summary>
/// <param name="userName">Имя пользователя</param>
/// <returns></returns>
```

```
    private int CountProfitForUser(string userId)
    {
        return (int) CountCoefficientForUser(userId) * this.TotalEggs;
    }

    /// <summary>
    /// Метод для расчета количества яиц во всем
    /// тотализаторе
    /// </summary>
    /// <returns>Количесто яиц в тоталихаторе</returns>
    private void CountTotalAmountOfEggs()
    {
        int sum = 0;
        foreach (var i in Bids)
        {
            sum += i.EggsAmount;
        }
        this.TotalEggs = sum;
    }

    /// <summary>
    /// Формирует словарь, содержащий имя каждого продукта
    /// и расчет коэффициента для него.
    /// </summary>
    public Dictionary<string, double> CountAllCoefficientsForProducts()
    {
        Dictionary<string, double> data = new Dictionary<string, double>();
        foreach (var i in ToteAuction.BidProducts)
        {
            data.Add(i.Name, CountCoefficientForProduct(i.Id));
        }
        return data;
    }

    /// <summary>
    /// Добавляет ставку пользователя
    /// </summary>
    /// <param name="userId">Айди пользователя</param>
    /// <param name="eggsAmount">Количество яиц</param>
    /// <param name="productId">Айдишник товара</param>
    public async Task<bool> SetRateForUser(int auctionId, AspNetUser user, int eggsAmount, int
productId, CocktionContext db)
    {
        if (IsEnoughEggs(user.Eggs, eggsAmount))
        {
            Bids.Add(new ToteEntity
            {
```

```
            UserId = user.Id,
            EggsAmount = eggsAmount,
            ProductId = productId
        });
        user.Eggs -= eggsAmount;
        CountTotalAmountOfEggs();
        AuctionHub.UpdateToteBoard(auctionId, CountAllCoefficientsForProducts());
        await DbItemsAdder.SaveDb(db);
        return true;
    }
    AuctionHub.UpdateToteBoard(auctionId, CountAllCoefficientsForProducts());
    return false;
}

/// <summary>
/// Метод для проверки достаточности яиц на счету у пользователя
/// </summary>
/// <param name="userEggs">Количество яиц у пользователя на балансе</param>
/// <param name="rateEggs">Количество яиц, которые он хочет поставить</param>
/// <returns>true, если достаточно, false иначе</returns>
private bool IsEnoughEggs(int userEggs, int rateEggs)
{
    if (userEggs >= rateEggs)
        return true;
    return false;
}

/// <summary>
/// Метод, завершающий тотализатор
/// </summary>
/// <param name="winProductId">Айдишник продукта - победителя</param>
public void FinishTote(int winProductId, CocktionContext db)
{
    //Рассчитать для каждого из пользователей результат аукциона.
    foreach (var i in Bids)
    {
        SetResult(winProductId, i, db);
    }
}

/// <summary>
///  Метод устанавливает результат тотализатора
///  для каждого пользователя
/// </summary>
/// <param name="userId"></param>
/// <param name="winProductId"></param>
private void SetResult(int winProductId, ToteEntity bid, CocktionContext db)
```

```
        {
            //Проверка на успешность
            ToteResult result;
            if (bid.ProductId == winProductId)
            {//если чувак поставил на победителя
                int profit = CountProfitForUser(bid.UserId);
                db.AspNetUsers.Find(bid.UserId).Eggs += profit;
                result = new ToteResult
                {
                    UserId = bid.UserId,
                    IsSuccessful = true,
                    Profit = profit //считаем прибыль, потому что чел успешен
                };
            }
            else
            {//если чувак неправильно поставил
                result = new ToteResult
                {
                    UserId = bid.UserId,
                    IsSuccessful = false,
                    Profit = -bid.EggsAmount //просто добавляем ему отрицательную прибыль!
                };
            }
            //записать все это дело в таблицу с результами
            this.ToteResultsForUsers.Add(result);
        }

    }
}
```

1.4.15. Класс ToteEntity

```
namespace CocktionMVC.Models.DAL
{
    /// <summary>
    /// Сущность, олицетворяющая ставку яйцами на аукционе.
    /// </summary>
    public class ToteEntity
    {
        /// <summary>
        /// Номер для хранения в базе данных
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Айди пользователя
```

```
        /// </summary>
        public string UserId { get; set; }

        /// <summary>
        /// Количество яиц
        /// </summary>
        public int EggsAmount { get; set; }

        /// <summary>
        /// Айдишник продукта
        /// </summary>
        public int ProductId { get; set; }

        /// <summary>
        /// связь с отцом-тотализатором
        /// </summary>
        public virtual ToteBoard OwnerTote { get; set; }
    }
}
```

1.4.16. Класс ToteResult

```
namespace CocktionMVC.Models.DAL
{
    /// <summary>
    /// Класс, содержащий в себе результаты аукциона
    /// </summary>
    public class ToteResult
    {
        /// <summary>
        /// Айди для базы данных
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Айди человека, к которому относится результат
        /// </summary>
        public string UserId { get; set; }

        /// <summary>
        /// Его выручка в яйцах
        /// </summary>
        public int Profit { get; set; }

        /// <summary>
        /// Успешнометр
```

```
        /// </summary>
        public bool IsSuccessful { get; set; }

        /// <summary>
        /// Тотализатор - владелец это ставки
        /// </summary>
        public virtual ToteBoard OwnerTote { get; set; }
    }
}
```

1.4.17. Класс AuctionHub

```
using System.Collections.Generic;
using System.Threading.Tasks;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.SignalR;
using WebGrease.Extensions;
using CocktionMVC.Functions.Managers;

namespace CocktionMVC
{

    /// <summary>
    /// Хаб используется для работы со страницой
    /// CurrentAuction. Он посылает сообщения,
    /// проводит транзакции с окончанием аукциона
    /// и т.д.
    /// </summary>
    public class AuctionHub : Hub
    {
        /// <summary>
        /// Посылает сообщения всем клиентам в данной
        /// группе
        /// </summary>
        /// <param name="name">Имя пользователя</param>
        /// <param name="message">Текст сообщения</param>
        /// <param name="auctionId">Id аукциона</param>
        public void Send(string name, string message, int auctionId)
        {
            Clients.Group(auctionId.ToString()).addNewMessageToPage(name, message);
        }

        /// <summary>
        /// Асинхронно добавляет новую группу
        /// на сервере
```

```
        /// </summary>
        /// <param name="auctionId">Айди аукциона</param>
        public void AddNewRoom(int auctionId)
        {
            Groups.Add(Context.ConnectionId, auctionId.ToString());
        }

        /// <summary>
        /// Добавляет товары в диаграмму на всех клиентах, где
        /// открыта страничка
        /// </summary>
        /// <param name="name">Название товара</param>
        /// <param name="fileName">Название файла с фотографией</param>
        /// <param name="auctionId">Айди аукциона</param>
        public static void AddNodesToClients(string name, string fileName, int auctionId, int productId)
        {
            //Добавляем клиентам всей группы Нодики
            IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
            context.Clients.Group(auctionId.ToString()).addNodesToPages(fileName, name, productId);
        }

        /// <summary>
        /// Добавляет довесок к товару на всех клиентах, где
        /// открыта страничка с аукционом
        /// </summary>
        /// <param name="name">Название товара</param>
        /// <param name="fileName">Название файла с фотографией</param>
        /// <param name="auctionId">Айдишник аукциона</param>
        /// <param name="parentProductId">айдишник товара, к которому добавлен довесок</param>
        /// <param name="childProductId">Айдишник товара, который добавляется в кач-ве довеска</
param>
        public static void AddExtraNodeToClients(string name, string fileName, int auctionId, int
parentProductId,
                                 int childProductId)
        {
            IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();

            context.Clients.Group(auctionId.ToString()).addExtraNodesToPages(fileName, name,
parentProductId, childProductId);
        }

        /// <summary>
        /// Метод посылает айдишник лидера на все клиенты
        /// </summary>
        /// <param name="leaderId">Айдишник лидера</param>
        /// <param name="auctionId">Айдишник аукциона, на котором все это происходит</param>
        public static void SetLider(string leaderId, int auctionId, string liderName)
```

```
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        context.Clients.Group(auctionId.ToString()).showLeaderOnPage(leaderId, liderName);
    }

    public async Task JoinRoom(int auctionId)
    {
        await Groups.Add(Context.ConnectionId, auctionId.ToString());
    }


    /// <summary>
    /// Метод заканчивает аукцион
    /// </summary>
    /// <param name="auctionId">Айди аукциона, который нужно закончить</param>
    public async Task FinishAuction(int auctionId)
    {
        //заканчивать здесь аукцион
        CocktionContext db = new CocktionContext();
        await FinishAuctionManager.FalseAuctionStatus(db, auctionId);
        Clients.Group(auctionId.ToString()).finishAuction();
    }

    /// <summary>
    /// Метод заканчивает аукцион с мобильника
    /// </summary>
    /// <param name="auctionId">Айди аукциона, который нужно закончить</param>
    public static async Task FinishAuctionMobile(int auctionId)
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        //заканчивать здесь аукцион
        CocktionContext db = new CocktionContext();
        await FinishAuctionManager.FalseAuctionStatus(db, auctionId);
        context.Clients.Group(auctionId.ToString()).finishAuction();
    }

    /// <summary>
    /// Метод для отправки на клиент всех значений тотализатора
    /// в данный момент времени
    /// </summary>
    /// <param name="auctionId">Айди аукциона, участникам которого необходимо сообщить</
param>
    public static void UpdateToteBoard(int auctionId, Dictionary<string, double> data)
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        string respond;
        respond = DataFormatter.DictionaryConverter(data);
```

```
            context.Clients.Group(auctionId.ToString()).updateToteBoard(respond);
        }

        class ToteString
        {
            public ToteString(string[] text)
            {
                Text = text;
            }
            public string[] Text { get; set; }
        }

        public void GetTote(int auctionId)
        {
            CocktionContext db = new CocktionContext();
            var auction = db.Auctions.Find(auctionId);
            UpdateToteBoard(auctionId, auction.AuctionToteBoard.CountAllCoefficientsForProducts());
        }

    }
}
```

1.4.18. Класс AuctionHub

```
using System.Collections.Generic;
using System.Threading.Tasks;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.SignalR;
using WebGrease.Extensions;
using CocktionMVC.Functions.Managers;

namespace CocktionMVC
{

    /// <summary>
    /// Хаб используется для работы со страницой
    /// CurrentAuction. Он посылает сообщения,
    /// проводит транзакции с окончанием аукциона
    /// и т.д.
    /// </summary>
    public class AuctionHub : Hub
    {
        /// <summary>
        /// Посылает сообщения всем клиентам в данной
        /// группе
```

```
    /// </summary>
    /// <param name="name">Имя пользователя</param>
    /// <param name="message">Текст сообщения</param>
    /// <param name="auctionId">Id аукциона</param>
    public void Send(string name, string message, int auctionId)
    {
        Clients.Group(auctionId.ToString()).addNewMessageToPage(name, message);
    }

    /// <summary>
    /// Асинхронно добавляет новую группу
    /// на сервере
    /// </summary>
    /// <param name="auctionId">Айди аукциона</param>
    public void AddNewRoom(int auctionId)
    {
        Groups.Add(Context.ConnectionId, auctionId.ToString());
    }

    /// <summary>
    /// Добавляет товары в диаграмму на всех клиентах, где
    /// открыта страничка
    /// </summary>
    /// <param name="name">Название товара</param>
    /// <param name="fileName">Название файла с фотографией</param>
    /// <param name="auctionId">Айди аукциона</param>
    public static void AddNodesToClients(string name, string fileName, int auctionId, int productId)
    {
        //Добавляем клиентам всей группы Нодики
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        context.Clients.Group(auctionId.ToString()).addNodesToPages(fileName, name, productId);
    }

    /// <summary>
    /// Добавляет довесок к товару на всех клиентах, где
    /// открыта страничка с аукционом
    /// </summary>
    /// <param name="name">Название товара</param>
    /// <param name="fileName">Название файла с фотографией</param>
    /// <param name="auctionId">Айдишник аукциона</param>
    /// <param name="parentProductId">айдишник товара, к которому добавлен довесок</param>
    /// <param name="childProductId">Айдишник товара, который добавляется в кач-ве довеска</
param>
    public static void AddExtraNodeToClients(string name, string fileName, int auctionId, int
parentProductId,
                                             int childProductId)
    {
```

```
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();

        context.Clients.Group(auctionId.ToString()).addExtraNodesToPages(fileName, name,
parentProductId, childProductId);
    }

    /// <summary>
    /// Метод посылает айдишник лидера на все клиенты
    /// </summary>
    /// <param name="leaderId">Айдишник лидера</param>
    /// <param name="auctionId">Айдишник аукциона, на котором все это происходит</param>
    public static void SetLider(string leaderId, int auctionId, string liderName)
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        context.Clients.Group(auctionId.ToString()).showLeaderOnPage(leaderId, liderName);
    }

    public async Task JoinRoom(int auctionId)
    {
        await Groups.Add(Context.ConnectionId, auctionId.ToString());
    }


    /// <summary>
    /// Метод заканчивает аукцион
    /// </summary>
    /// <param name="auctionId">Айди аукциона, который нужно закончить</param>
    public async Task FinishAuction(int auctionId)
    {
        //заканчивать здесь аукцион
        CocktionContext db = new CocktionContext();
        await FinishAuctionManager.FalseAuctionStatus(db, auctionId);
        Clients.Group(auctionId.ToString()).finishAuction();
    }

    /// <summary>
    /// Метод заканчивает аукцион с мобильника
    /// </summary>
    /// <param name="auctionId">Айди аукциона, который нужно закончить</param>
    public static async Task FinishAuctionMobile(int auctionId)
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
        //заканчивать здесь аукцион
        CocktionContext db = new CocktionContext();
        await FinishAuctionManager.FalseAuctionStatus(db, auctionId);
        context.Clients.Group(auctionId.ToString()).finishAuction();
    }
```

```
        /// <summary>
        /// Метод для отправки на клиент всех значений тотализатора
        /// в данный момент времени
        /// </summary>
        /// <param name="auctionId">Айди аукциона, участникам которого необходимо сообщить</
param>
        public static void UpdateToteBoard(int auctionId, Dictionary<string, double> data)
        {
            IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionHub>();
            string respond;
            respond = DataFormatter.DictionaryConverter(data);
            context.Clients.Group(auctionId.ToString()).updateToteBoard(respond);
        }

        class ToteString
        {
            public ToteString(string[] text)
            {
                Text = text;
            }
            public string[] Text { get; set; }
        }

        public void GetTote(int auctionId)
        {
            CocktionContext db = new CocktionContext();
            var auction = db.Auctions.Find(auctionId);
            UpdateToteBoard(auctionId, auction.AuctionToteBoard.CountAllCoefficientsForProducts());
        }

    }
}
```

1.4.19. Класс AuctionListHub

```
using Microsoft.AspNet.SignalR;
using System;

namespace CocktionMVC.Models.Hubs
{
    public class AuctionListHub : Hub
    {
        /// <summary>
        /// Метод обновляет страничку с аукционами при
        /// добавлении нового аукциона на страницу.
```

```
        /// </summary>
        /// <param name="name">Имя товара для аукциона</param>
        /// <param name="description">Описание товара для аукциона</param>
        /// <param name="category">Категория товара для аукциона</param>
        /// <param name="photoName">Имя фотки</param>
        public static void UpdateList(string name, string description, string category, DateTime time,
            string photoName, int id)
        {
            IHubContext context = GlobalHost.ConnectionManager.GetHubContext<AuctionListHub>();
            context.Clients.All.addAuctionToTheList(name, description, category,time.ToLongTimeString(),
                photoName, id);
        }
    }
}
```

1.4.20. Класс MessageHub

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Helpers;
using System.Web.Http.Results;
using CocktionMVC.Functions;
using CocktionMVC.Models.DAL;
using Microsoft.AspNet.SignalR;
using Newtonsoft.Json;

namespace CocktionMVC.Models.Hubs
{
    public class MessageHub : Hub
    {
        /// <summary>
        ///
        /// </summary>
        /// <param name="message"></param>
        /// <param name="author">Автор сообщения</param>
        /// <param name="userName">Имя пользователя, которому надо отправить сообщение</param>
        /// <param name="authorId">Айдишник того, кто отправляет</param>
        /// <param name="receiverId">Айдишник того, кому отправляют</param>
        /// <returns></returns>
        public async Task Send(string message, string author, string userName,
            string authorId, string receiverId)
        {
            try
            {
```

```
            Clients.Group(userName).addNewMessageToPage(author, message,
DateTime.Now.ToShortDateString(), receiverId);
            CocktionContext db = new CocktionContext();
            var userA = db.AspNetUsers.Find(authorId);
            var userB = db.AspNetUsers.Find(receiverId);

            if (userB.MobileDevice != null)
               Notificator.SendNotification(userB.MobileDevice, message, 1);

            PrivateMessage privateMessage = new PrivateMessage(message, userA.UserName,
userB.UserName,
               DateTime.Now);
            userA.ChatMessages.Add(privateMessage);
            userB.ChatMessages.Add(privateMessage);
            db.Messages.Add(privateMessage);
            await DbItemsAdder.SaveDb(db);
         }
         catch
         {
         }

      }

      /// <summary>
      /// Асинхронно добавляет новую группу
      /// на сервере
      /// </summary>
      /// <param name="auctionId">Айди аукциона</param>
      public void AddNewRoom(string userName)
      {
         Groups.Add(Context.ConnectionId, userName);
      }

      /// <summary>
      /// Получает список людей, которые когда-либо общались
      /// с данным пользователем
      /// </summary>
      /// <param name="userName">Имя пользователя, для которого надо все найти</param>
      public void GetListOfReceivers(string userId)
      {
         CocktionContext db = new CocktionContext();
         var user = db.AspNetUsers.Find(userId);
         var msg = (from x in user.ChatMessages
               select x.AuthorName == user.UserName ? x.ReceiverName : x.AuthorName).ToList();
         List<string> authors = new List<string>();
         foreach (var message in msg)
         {
```

```
            if (!authors.Contains(message) && message != user.UserName)
                authors.Add(message);
        }
        authors.ForEach(x => Clients.Group(user.UserName).addAuthors(x));

    }

    /// <summary>
    /// Получает сообщения чата, в котором участвует данный пользователь
    /// </summary>
    /// <param name="userName">Имя пользователя, с которым общение происходит</param>
    /// <param name="thisUserName">Имя данного пользователя</param>
    public void GetMessages(string responderName, string thisUserId)
    {
        CocktionContext db = new CocktionContext();
        var user = db.AspNetUsers.Find(thisUserId);
        var userB = db.AspNetUsers.First(x => x.UserName == responderName);
        var msg = (from x in user.ChatMessages
                   where (x.AuthorName == user.UserName && x.ReceiverName == userB.UserName) ||
                   (x.AuthorName == userB.UserName && x.ReceiverName == user.UserName)
                   select x).ToList();

        foreach (var message in msg)
        {
            Clients.Group(user.UserName).appendMessageToPage(message.AuthorName,
message.Content, message.DateOfPublishing.ToShortDateString(),
                userB.Id);
        }
    }
  }
}
```

1.4.21. Класс IdentityModels

```
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;

namespace CocktionMVC.Models
{
  // You can add profile data for the user by adding more properties to your ApplicationUser class, please
visit http://go.microsoft.com/fwlink/?LinkID=317594 to learn more.
  public class ApplicationUser : IdentityUser
  {
    public string UserRealName { get; set; }
    public string UserRealSurname { get; set; }
```

```
        public string PhoneNumber { get; set; }
        public int Eggs { get; set; }



        public int Rating { get; set; }
        public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser>
manager)
        {
            // Note the authenticationType must match the one defined in
CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            return userIdentity;
        }
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {

        public ApplicationDbContext()
            : base("DefaultConnection", false)
        {
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }
    }
}
```

1.4.22. Класс BidChecker

```
 // ReSharper disable once CheckNamespace
namespace CocktionMVC.Models.JsonModels
{
    /// <summary>
    /// Данная модель служит для отправки логического статуса
    /// о том, есть ставка уже у этого пользователя или нет!
    /// </summary>
    public class BidChecker
    {
        /// <summary>
        /// Индикатор того, поставлена ставка или нет!
        /// </summary>
```

```
        public bool HaveBid { get; set; }
    }
}
```

1.4.23. Класс BidSeller

```
 // ReSharper disable once CheckNamespace
namespace CocktionMVC.Models.JsonModels
{
    /// <summary>
    /// Моделирует пользователя при окончании аукциона
    /// </summary>
    public class BidSeller
    {
        /// <summary>
        /// Имя пользователя
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Тип посетителя аукциона.
        /// Всего их существует несколько типов.
        /// 1)Owner_undfnd - владелец, который не выбрал победителя
        /// 2)Winner - победитель аукциона
        /// 3)Owner - владелец аукциона
        /// 4)Looser - проигравший человек
        /// 5)Info - любой другой пользователь
        /// (неавторизованный / не участник аукциона)
        /// </summary>
        public string Type { get; set; }

        /// <summary>
        /// Сообщение, которое будет ему показано
        /// на экране при завершении аукциона
        /// </summary>
        public string Message { get; set; }

        /// <summary>
        /// Айдишник пользователя, для которого
        /// был создан объект BidSeller
        /// </summary>
        public string Id { get; set; }

        /// <summary>
        /// Прибыль, которую пользователь получил по результат тотализатора
        /// </summary>
        public int? ProfitFromTote { get; set; }
```

```
        /// <summary>
        /// Количество яиц, которое он получил.
        /// </summary>
        public int? CurrentEggsAmount { get; set; }
    }
}
```

1.4.24. Класс ForumPostMobile

```
namespace CocktionMVC.Models.JsonModels.HouseRelatedModels
{
    /// <summary>
    /// Контейнер для сообщений с форума
    /// </summary>
    public class ForumPostMobile
    {
        public ForumPostMobile() { }

        public ForumPostMobile(string author, string message, int likes)
        {
            authorName = author;
            this.message = message;
            this.likes = likes;
        }

        /// <summary>
        /// Имя автора
        /// </summary>
        public string authorName { get; set; }

        /// <summary>
        /// Сам сообщение
        /// </summary>
        public string message { get; set; }

        /// <summary>
        /// Количество лайков
        /// </summary>
        public int likes { get; set; }
    }
}
```

1.4.25. Класс ForumRespond

```
 // ReSharper disable once CheckNamespace
namespace CocktionMVC.Models.JsonModels
{
    /// <summary>
    /// Моделирует ответ с сервера о статусе добавления
    /// сообщения на форум
    /// </summary>
    public class ForumRespond
    {
        /// <summary>
        /// Статус отправки
        /// "Success" если успех
        /// "Failed" если провал
        /// </summary>
        public string Status { get; set; }

        /// <summary>
        /// Автор сообщения
        /// </summary>
        public string Author { get; set; }

        /// <summary>
        /// Собирает вариант, в котором важен автор
        /// </summary>
        /// <param name="status">Стутус отправки</param>
        /// <param name="author">Автор сообщения</param>
        public ForumRespond(string status, string author)
        {
            Author = author;
            Status = status;
        }

        /// <summary>
        /// Собирает вариант, в котором важен только статус
        /// </summary>
        /// <param name="status">Статус отправки</param>
        public ForumRespond(string status)
        {
            Status = status;
        }
    }
}
```

1.4.26. Класс Guild

namespace CocktionMVC.Models.JsonModels.HouseRelatedModels

```
{
    /// <summary>
    /// Контейнер для информации о дом-содержащей организации
    /// </summary>
    public class Guild
    {
        public Guild()
        {

        }

        public Guild(string name, string photoPath, int id)
        {
            this.title = name;
            this.photoPath = photoPath;
            this.id = id;
        }

        /// <summary>
        /// Название дома
        /// </summary>
        public string title { get; set; }

        /// <summary>
        /// Путь к фоточке
        /// </summary>
        public string photoPath { get; set; }

        /// <summary>
        /// Айдишник организации
        /// </summary>
        public int id { get; set; }

    }
}
```

1.4.26. Класс GuildsHouse

```
namespace CocktionMVC.Models.JsonModels.HouseRelatedModels
{
    /// <summary>
    /// Контейнер с информацией о доме для мобильного клиента
    /// </summary>
    public class GuildsHouse
    {
        public GuildsHouse() { }
```

```
        public GuildsHouse(string houseName, int houseId, string houseAdress, string housePhotoPath,
            bool isSubscribed)
        {
            title = houseName;
            id = houseId;
            adress = houseAdress;
            photoPath = housePhotoPath;
            this.isSubscribed = isSubscribed;
        }
        /// <summary>
        /// Имя дома (факультет/общага)
        /// </summary>
        public string title { get; set; }

        /// <summary>
        /// Айдишник дома
        /// </summary>
        public int id { get; set; }

        /// <summary>
        /// Адрес дома
        /// </summary>
        public string adress { get; set; }

        /// <summary>
        /// Путь к фотке дома
        /// </summary>
        public string photoPath { get; set; }

        public bool isSubscribed { get; set; }
    }
}
```

1.4.27. Класс HouseMobile

```
namespace CocktionMVC.Models.JsonModels.HouseRelatedModels
{
    /// <summary>
    /// Контейнер для данных о доме для посылки на мобилку
    /// </summary>
    public class HouseMobile
    {
        public HouseMobile() { }

        /// <summary>
```

```
/// Конструирует инфу о доме по всем полям
/// </summary>
/// <param name="photoPath">Путь к фотке для дома</param>
/// <param name="likes">Лайки дома</param>
/// <param name="dislikes">Дизлайки дома</param>
/// <param name="rating">Рэйтинг дома</param>
/// <param name="peopleIn">Люди в доме</param>
/// <param name="auctionsIn">Аукционов в доме</param>
/// <param name="description">Описание дома</param>
public HouseMobile(string name ,string photoPath, int likes, int dislikes, int rating, int peopleIn,
   int auctionsIn, string description, bool isSubscribed)
{
   this.title = name;
   this.photoPath = photoPath;
   this.likes = likes;
   this.dislikes = dislikes;
   this.rating = rating;
   peopleAmount = peopleIn;
   auctionsAmount = auctionsIn;
   this.description = description;
   this.isSubscribed = isSubscribed;
}

/// <summary>
/// Путь к фоточке
/// </summary>
public string photoPath { get; set; }

/// <summary>
/// Количество лайков
/// </summary>
public int likes { get; set; }

/// <summary>
/// Количество дизлайков
/// </summary>
public int dislikes { get; set; }

/// <summary>
/// Рэйтинг дома
/// </summary>
public int rating { get; set; }

/// <summary>
/// Количество человек в доме
/// </summary>
public int peopleAmount { get; set; }
```

```
        /// <summary>
        /// Количество аукционов в доме
        /// </summary>
        public int auctionsAmount { get; set; }

        /// <summary>
        /// Описание дома
        /// </summary>
        public string description { get; set; }

        public bool isSubscribed { get; set; }

        public string title { get; set; }
    }
}
```

1.4.28. Класс AuctionInfo

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
    /// Класс для модели аукциона, которая будет отсылаться
    /// мобильным клиентам.
    /// </summary>
    public class AuctionInfo
    {

        /// <summary>
        /// Описание аукциона
        /// </summary>
        public string description { get; set; }

        /// <summary>
        /// Название аукциона
        /// </summary>
        public string title { get; set; }

        /// <summary>
        /// Путь к картинке для аукциона
        /// </summary>
        public string photoPath { get; set; }

        /// <summary>
        /// Время окончания аукциона в минутах от текущего
        /// </summary>
```

```
        public int endTime { get; set; }

        /// <summary>
        /// Айдишник аукциона
        /// </summary>
        public int auctionId { get; set; }

        public int leaderId { get; set; }
        public string category { get; set; }
        public bool isActive { get; set; }
        public UserInfo owner { get; set; }

        public AuctionInfo(string description, int endTime, string photoPath, string title,
            int auctionId, int leaderId, string category, UserInfo info, bool isActive)
        {
            this.description = description;
            this.endTime = endTime;
            this.photoPath = photoPath;
            this.title = title;
            this.auctionId = auctionId;
            this.leaderId = leaderId;
            this.category = category;
            this.owner = info;
            this.isActive = isActive;
        }

        public AuctionInfo()
        {
        }
    }
}
```

1.4.29. Класс HouseInfo

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
    /// Контейнер для информации о доме, которая посылается на мобильный клиент
    /// </summary>
    public class HouseInfo
    {
        public string title { get; set; }
        public string adress { get; set; }
        public int rating { get; set; }
        public int likes { get; set; }
        public bool isSubscribed { get; set; }
```

```
    }
}
```

1.4.30. Класс LoginCredentials

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
    /// Логин/Пароль контейнер для мобильника
    /// </summary>
    public class LoginCredentials
    {
        /// <summary>
        /// Почтовый адрес, он же - логин
        /// </summary>
        public string email { get; set; }

        /// <summary>
        /// Пароль
        /// </summary>
        public string password { get; set; }

    }
}
```

1.4.31. Класс ProductInfo

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
    /// Класс для вывода информации о продукте
    /// на мобильный клиент
    /// </summary>
    public class  ProductInfo
    {
        /// <summary>
        /// Наименование продукта
        /// </summary>
        public string title { get; set; }

        /// <summary>
        /// Описание продукта
        /// </summary>
        public string description { get; set; }
```

```
        /// <summary>
        /// Название файла
        /// </summary>
        public string photoPath { get; set; }

        /// <summary>
        /// Айди товара ставки
        /// </summary>
        public int id { get; set; }

        /// <summary>
        /// Категория товара
        /// </summary>
        public string category { get; set; }

    }
}
```

1.4.32. Класс StatusAndPhotoPath

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
    /// Контейнер для информации со статусом добавления аукциона
    /// с мобильника
    /// </summary>
    public class StatusAndPhotoPath
    {
        /// <summary>
        /// Статус
        /// </summary>
        public string Status { get; set; }

        /// <summary>
        /// Путь к фотке
        /// </summary>
        public string PhotoPath { get; set; }
     }
}
```

1.4.33. Класс TokenResponse

```
namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    /// <summary>
```

```
    /// Контейнер для токена для мобильника
    /// </summary>
    public class TokenResponse
    {
        /// <summary>
        /// Токен, который шлем в ответочку телефону
        /// </summary>
        public string Token { get; set; }
    }
}
```

1.4.34. Класс UserInfo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CocktionMVC.Models.JsonModels.MobileClientModels
{
    public class UserInfo
    {
        public string id { get; set; }
        public string title { get; set; }

        public string photoPath { get; set; }

        public bool isInformator { get; set; }

        public UserInfo(string id, string name, string photoPath, bool isInformator)
        {
            this.id = id;
            this.title = name;
            this.photoPath = @"http://cocktion.com/Images/Thumbnails/" + photoPath;
            this.isInformator = isInformator;
        }
    }
}
```

1.4.35. Класс ProductMVCInfo

```
namespace CocktionMVC.Models.JsonModels
{
    /// <summary>
    /// Контейнер для информации о товаре
```

```
    /// </summary>
    public class ProductMVCInfo
    {
        /// <summary>
        /// Наименование продукта
        /// </summary>
        public string Name { get; set; }

        /// <summary>
        /// Описание продукта
        /// </summary>
        public string Description { get; set; }

        /// <summary>
        /// Название файла
        /// </summary>
        public string FileName { get; set; }

        /// <summary>
        /// Категория товара
        /// </summary>
        public string Category { get; set; }

    }
}
```

1.4.36. Класс StatusHolder

```
namespace CocktionMVC.Models.JsonModels
{
    /// <summary>
    /// Контенер для стандартного ответа
    /// сервера.
    ///
    /// СИСТЕМНЫЙ КЛАСС
    /// </summary>
    public class StatusHolder
    {
        public StatusHolder() { }

        public StatusHolder(bool isSuccessful)
        {
            if (isSuccessful)
                Status = "Success";
            else
                Status = "Failure";
```

```
        }
        public string Status { get; set; }
    }

    public class StatusHolderWithError : StatusHolder
    {
        public string Message { get; set; }

        public StatusHolderWithError(bool isSuccessful, string message)
            : base(isSuccessful)
        {
            Message = message;
        }
    }
}
```

1.4.37. Класс ToteEggsInfo

```
namespace CocktionMVC.Models.JsonModels.ToteRelatedModels
{
    /// <summary>
    ///  Контейнер для информации о яйцах
    /// </summary>
    public class ToteEggsInfo
    {

        /// <summary>
        /// Статус
        /// </summary>
        public bool  Status { get; set; }

        /// <summary>
        /// Количество яиц у пользователя
        /// </summary>
        public int UsersAmountOfEggs { get; set; }
    }
}
```

1.4.38. Класс AccountViewModels

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace CocktionMVC.Models
{
```

```
    public class ExternalLoginConfirmationViewModel
    {
        [Required(ErrorMessage="Ну никак нельзя обойтись без почтового адреса!")]
        [EmailAddress(ErrorMessage="Введенный имейл недействителен!")]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required(ErrorMessage="Необходимо ввести свое имя!")]
        [StringLength(25, ErrorMessage = "Странно, что ваше имя превышает 25 символов...")]
        [Display(Name = "Ваше имя")]
        public string UserRealName { get; set; }

        [Required(ErrorMessage = "Необходимо ввести свою фамилию")]
        [StringLength(25, ErrorMessage = "Странно, что ваша фамилия превышает 25 символов...")]
        [Display(Name = "Ваша фамилия")]
        public string UserRealSurname { get; set; }

        [Required(ErrorMessage = "Необходимо ввести номер телефона!")]
        [RegularExpression("^[0-9\\-\\+]{11,12}$", ErrorMessage="Ваш номер телефона
недействителен!")]
        public string PhoneNumber { get; set; }

    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }
```

```
        [Display(Name = "Запомнить этот браузер?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Электро-почта")]
        public string Email { get; set; }
    }

    public class LoginViewModel
    {
        [Required]
        [Display(Name = "Электро-почта")]
        [EmailAddress]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Display(Name = "Запомнить меня?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required(ErrorMessage = "Ну никак нельзя обойтись без почтового адреса!")]
        [EmailAddress(ErrorMessage = "Введенный имейл не действителен!")]
        [Display(Name = "Электро-почта")]
        public string Email { get; set; }

        //регулярка для телефона "^[0-9\\-\\+]{11,12}$"

        [Required(ErrorMessage = "Необходимо придумать пароль")]
        [StringLength(100, ErrorMessage = "Пароль должен быть как минимум {2} значный.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Required(ErrorMessage="Необходимо подтвердить пароль!")]
        [DataType(DataType.Password)]
```

```
        [Display(Name = "Пароль еще раз")]
        [Compare("Password", ErrorMessage = "Упс, пароли не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ResetPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Электро-почта")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Пароль должен быть как минимум 6 значный",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтвердить пароль")]
        [Compare("Password", ErrorMessage = "Упс, пароли не совпадают.")]
        public string ConfirmPassword { get; set; }

        public string Code { get; set; }
    }

    public class ForgotPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Электро-почта")]
        public string Email { get; set; }
    }
}
```

1.4.39. Класс ManageViewModels

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNet.Identity;
using Microsoft.Owin.Security;

namespace CocktionMVC.Models
{
```

```
    public class IndexViewModel
    {
        public bool HasPassword { get; set; }
        public IList<UserLoginInfo> Logins { get; set; }
        public string PhoneNumber { get; set; }
        public bool TwoFactor { get; set; }
        public bool BrowserRemembered { get; set; }
    }

    public class ManageLoginsViewModel
    {
        public IList<UserLoginInfo> CurrentLogins { get; set; }
        public IList<AuthenticationDescription> OtherLogins { get; set; }
    }

    public class FactorViewModel
    {
        public string Purpose { get; set; }
    }

    public class SetPasswordViewModel
    {
        [Required]
        [StringLength(100, ErrorMessage = "Пароль должен быть как минимум {2} символьный.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Новый пароль")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтвердить новый пароль")]
        [Compare("NewPassword", ErrorMessage = "Упс, пароли не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ChangePasswordViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Текущий пароль")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Пароль должен быть как минимум {2} символьный.",
MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Новый пароль")]
```

```
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтвердить новый пароль")]
        [Compare("NewPassword", ErrorMessage = "Упс, пароли не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class AddPhoneNumberViewModel
    {
        [Required]
        [Phone]
        [Display(Name = "Номер телефона")]
        public string Number { get; set; }
    }

    public class VerifyPhoneNumberViewModel
    {
        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }

        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string PhoneNumber { get; set; }
    }

    public class ConfigureTwoFactorViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
    }
}
```

1.4.40. Класс ProfileViewModels

```
using System.Collections.Generic;
using System.ComponentModel;
using CocktionMVC.Models.DAL;

namespace CocktionMVC.Models.ViewModels
{
    /// <summary>
    /// Модель для просмотра профиля пользователя
    /// </summary>
```

```
public class ProfileViewModel
{
    /// <summary>
    /// TODO Сделать интересы пользователя и подписки
    /// </summary>
    public ProfileViewModel()
    {
    }

    public ProfileViewModel(int eggsAmount, int auctionsAmount, int productsAmount,
        string surname, string name, int? rating, int daysWithUs, List<Auction> hisAuctions, string
userId,
        List<UsersFeedback> feeds, List<House> houses)
    {
        EggsAmount = eggsAmount;
        AuctionsAmount = auctionsAmount;
        ProductsAmount = productsAmount;
        Surname = surname;
        Name = name;
        Rating = rating;
        DaysWithUsAmount = daysWithUs;
        HisAuctions = hisAuctions;
        UserId = userId;
        Feeds = feeds;
        Houses = houses;
        Users = new List<AspNetUser>();
    }

    public ProfileViewModel(int eggsAmount, int auctionsAmount, int productsAmount,
      string surname, string name, int? rating, int daysWithUs, List<Auction> hisAuctions, string userId,
      List<UsersFeedback> feeds, List<House> houses, List<AspNetUser> friends)
    {
        EggsAmount = eggsAmount;
        AuctionsAmount = auctionsAmount;
        ProductsAmount = productsAmount;
        Surname = surname;
        Name = name;
        Rating = rating;
        DaysWithUsAmount = daysWithUs;
        HisAuctions = hisAuctions;
        UserId = userId;
        Feeds = feeds;
        Houses = houses;
        Users = friends;
    }

    public List<AspNetUser> Users { get; set; }
```

```
        public List<House> Houses { get; set; }
        public List<UsersFeedback> Feeds { get; set; }
        public string UserId { get; set; }
        public int EggsAmount { get; set; }
        public int AuctionsAmount { get; set; }
        public int ProductsAmount { get; set; }
        public string Surname { get; set; }
        public string Name { get; set; }
        public int? Rating { get; set; }
        public int DaysWithUsAmount { get; set; }
        public List<Auction> HisAuctions { get; set; }
    }
}
```

1.5. Код каскадных таблиц стилей

1.5.1. Код таблицы createAuctionStyle

```
/*Стили для формы для создания аукциона*/

/*Контейнеры с чекбоксами*/
#typeOfAuctionContainer, #timeContainer {
    text-align: center;
    font-size: medium;
    border: dotted 1px #48b8e5;
    border-radius: 3px;
    padding-bottom: 3px;
}

h3,h4{
    text-align: center;
}

#facultyToChoose {
    text-align: left;
}

#goToAuctionBtn {
    float: right;
}

/*кнопка отправки и ввод файла*/
#fileAndButtonContainer{
    text-align: center;
```

```
}

/*Стили формы для ввода данных текстовых и радио батонов*/
#inputs{
   border-bottom: solid 1px #48b8e5;
   border-left: solid 1px #48b8e5;
   border-top: solid 1px #48b8e5;
   background-color: white;
   border-top-left-radius: 3px;
   border-bottom-left-radius: 3px;
   padding-bottom: 10px;
   color: black;
}

/*Форма для стилей, которые получает форма для выбора домов*/
#houses{
   border-top: solid 1px #48b8e5;
   border-bottom: solid 1px #48b8e5;
   border-right: solid 1px #48b8e5;
   border-left: solid 1px #48b8e5;
   background-color: white;
   border-top-right-radius: 3px;
   border-bottom-right-radius: 3px;
   padding-bottom: 10px;
   height: 446px;
   overflow: scroll;
   text-align: center;
      color: black;

}
```

1.5.2. Код таблицы currentAuctionStyle

```
#timer {
   text-align: center;
}

#typeOfAuctionContainer {
   border: solid 1px rgb(204, 204, 204);
   border-radius: 3px;
}

/*Для контейнера, который содержит всю правую часть
странички, (не болталку)*/
#globalContainer{
   border-left: dashed 1px lightgray;
```

```
    padding: 10px;
}


#inputProductTextBoxes{
    border: solid 1px lightblue;
    padding: 10px;
    border-radius: 10px;
}

#productBidContainer, #unauthorizedUserInfo{
    text-align: center;
}

/*Стили блоков, содержащих основную информацию об
аукционе, чате, ставках, тотализаторе:)*/
#firstInfoContainer, #secondInfoContainer{
    border: solid 1px lightgray;
    padding: 10px;
    border-radius: 3px;
    margin-right: 3%;
}

/*Стили блоков, содержащих основную информацию об
аукционе, чате, ставках, тотализаторе:)*/
#firstInfoContainer, #secondInfoContainer{
    border: solid 1px lightgray;
    padding: 10px;
    border-radius: 3px;
    margin-right: 3%;
}

#chatMessagesContainer, #coefficientsTable{
    width: 100%;
    height: 200px;
    border: solid 1px lightblue;
    border-radius: 3px;
    padding: 10px;
    overflow: scroll;
}

#auctionInfoContainer, #productsContainer {
    height: auto;
    width: 100%;
    border: solid 1px lightblue;
    border-radius: 3px;
    padding: 10px;
```

```
}

#productPhotoContainer, #productInfoContainer {
   margin: 3%
}
```

1.5.3. Код houseAuctionStyle

```
#auctionPanel{
   border-bottom-left-radius: 3px;
   border-bottom-right-radius: 3px;
   border: solid 1px lightgray;
   padding: 5px;
}


/*Это ссылка на аукцион в том, что добавляется в таблицу*/
#auctionLink {
   text-align: center;
}

/*Это контейнер для аукциончика*/
.auction{
   border: dotted 1px lightgray;
   border-radius: 3px;
   padding: 3px;
}


.row {
   margin: 3px;
}

#auctionsControlPanel{
   background-color: #48b8e5;
   text-align: center;
   border-top-left-radius: 3px;
   border-top-right-radius: 3px;
   border-right: solid 1px lightgray;
   border-left: solid 1px lightgray;
   border-top: solid 1px lightgray;
   padding: 5px;
   color: white;
}
```

1.5.4. Код indexStyle

```
#auctionPanel{
    border-bottom-left-radius: 3px;
    border-bottom-right-radius: 3px;
    border: solid 1px #1c94c4;
    padding: 5px;
}

#auctionsControlPanel{
    background-color: #48b8e5;
    text-align: center;
    color: white;
    border-top-left-radius: 3px;
    border-top-right-radius: 3px;
    border-right: solid 1px #1c94c4;
    border-left: solid 1px #1c94c4;
    border-top: solid 1px #1c94c4;
    padding: 5px;
}

.row {
    margin: 3px;
}
/*Это ссылка на аукцион в том, что добавляется в таблицу*/
#auctionLink {
    text-align: center;
}

/*Это контейнер для аукционщика*/
.auction{
    border: dotted 1px #1c94c4;
    border-radius: 3px;
    padding: 3px;
}
```

1.5.5. Код таблицы getCurrentAuctionStyle

```
p {
    font-family: monospace;
}

h3 {
    text-align: center;
    font-family: monospace;
}
```

```
#houseDescriptionContainer {
    border: 1px solid black;
    padding: 10px;
    font-family: monospace;
    border-radius: 3px;
}

#secondInfoColumn {
    border: 1px solid black;
    padding: 10px;
    border-radius: 3px;
}

#messageContainer {
    margin-bottom: 10px;
}

.author {
    border-bottom: solid 1px lightgray;
}

.postContainer {
    border: dotted 1px lightgray;
    padding: 5px;
    border-radius: 3px;
    margin: 5px;
}


#messageEnterContainer {
    border-top: 2px dotted black;
    padding-top: 30px;
}

textarea {
    width: 100%;
    height: 100%;
}

#addCommentButton {
    float: right;
}
```

1.5.6. Код таблицы universityHousesStyle

```
.house{
    border: dotted 1px lightgray;
    border-radius: 3px;
    padding: 5px;
}

#universityHousesControlPanel{
    background-color: whitesmoke;
    text-align: center;
    border-top-left-radius: 3px;
    border-top-right-radius: 3px;
    border-right: solid 1px lightgray;
    border-left: solid 1px lightgray;
    border-top: solid 1px lightgray;
    margin-left: 3px;
    padding: 5px;
    margin-right: 3px;
}

#universityHousesContainer{
    text-align: center;
    border-bottom-left-radius: 3px;
    border-bottom-right-radius: 3px;
    border: solid 1px lightgray;
    margin-left: 3px;
    margin-right: 3px;
    padding: 5px;
}
```

1.5.7. Код таблицы mainStyle

```
header {
    text-align: center;
    color: #fff;
    background: #18bc9c;
    background-color: #1c94c4;
}

header .container {
    padding-top: 100px;
    padding-bottom: 50px;
}

header img {
    display: block;
    margin: 0 auto 20px;
```

```
}

header .intro-text .name {
    display: block;
    text-transform: uppercase;
    font-family: Montserrat,"Helvetica Neue",Helvetica,Arial,sans-serif;
    font-size: 2em;
    font-weight: 700;
}

header .intro-text .skills {
    font-size: 1.25em;
    font-weight: 300;
}

@media(min-width:768px) {
    header .container {
        padding-top: 200px;
        padding-bottom: 100px;
    }

    header .intro-text .name {
        font-size: 4.75em;
    }

    header .intro-text .skills {
        font-size: 1.75em;
    }
}

@media(min-width:768px) {
    header .container {
        padding-top: 80px;
        padding-bottom: 50px;
    }

    header .intro-text .name {
        font-size: 2.75em;
    }

    header .intro-text .skills {
        font-size: 1.75em;
    }
}


#generalInfo{
```

```
    text-align: center;
    margin-top: 20px;
    margin-bottom: 80px;
}


footer {
    color: #fff;
}

footer h3 {
    margin-bottom: 30px;
}

footer .footer-above {
    padding-top: 50px;
    background-color: #2c3e50;
}

footer .footer-col {
    margin-bottom: 50px;
}

footer .footer-below {
    padding: 25px 0;
    background-color: #233140;
}

.body-content {
    padding-left: 0px;
    padding-right: 0px;
}
```

1.5.8. Код таблицы ProfileStyle

```
h3, h4,h5 {
    text-align: center;
}
#userInformation, #subscribedHouses, #usersAuctions{
    background-color: transparent;
    border-top-left-radius: 3px;
    border-bottom-left-radius: 3px;
    border-color: black;
    border: groove 1px;
}
```

```
#userInformation {
    height: 250px;
    align-items: center;
    background-color: white;
}

#userLifePos {
    border-bottom-right-radius: 3px;
    border-top-right-radius: 3px;
    border-top: groove 1px;
    border-bottom: groove 1px;
    border-color: lightgray;
    border-right: groove 1px;
    background-color: white;
    height: 250px;
    align-items: flex-start;
}

#interests {
    padding: 1px;
}
.house{
    margin-left: 5px;
}


/*Это ссылка на аукцион в том, что добавляется в таблицу*/
#auctionLink {
    text-align: center;
}

/*Это контейнер для аукционщика*/
.auction{
    border: dotted 1px lightgray;
    border-radius: 3px;
    padding: 3px;
    margin-left: 5px;
}

#auctionHeader, #housesHeader {
    border-bottom: dotted 1px gray;
    padding: 10px;
}

#feedbackFormContainer {
    border: solid 1px gray;
```

```
    border-radius: 3px;
    padding-bottom: 10px;
}

textarea{
    width:100%;
    height: 100px;
}

.postAuthor {
    border-bottom: solid 1px lightgray;
}

#sendFeedbackBtn {
    float: right;
}

#feedbackPosts{
    border: solid 1px lightblue;
    border-radius: 3px;
    padding: 10px;
}

.feedbackPost{
    border: dotted 1px lightgray;
    padding: 5px;
    border-radius: 3px;
}

#userFullInformation {
    border: solid 1px lightgray;
    background-color: #f5f5f5;
}


/*Вся форма с сообщениями целиком */
#messagesWindow{
    border-right: 1px solid lightgray;
    border-top:  1px solid lightgray;
    border-bottom:  1px solid lightgray;
    border-top-right-radius: 3px;
    border-bottom-right-radius: 3px;
}

#inputMessageForm{
    margin-bottom: 5px;
}
```

```
/*Контейнер, в котором находятся текстовые сообщение*/
#messages{
    background-color: #ffffff;
    border: 1px solid lightgray;
    margin-top: 5px;
    border-radius: 3px;
    margin-bottom: 5px;
    max-height: 300px;
    overflow-y: scroll;
    overflow-wrap: break-word;
    height: 385px;
    padding: 10px;
}

#usersList{
    border: 1px solid lightgray;
    border-top-left-radius: 3px;
    border-bottom-left-radius: 3px;
    overflow-y: scroll;
    height: 398px;
}
```

1.5.9. Код таблицы registerStyle

```
#formContainer {
    background-color: white;
    border: solid 1px lightblue;
    border-radius: 3px;
    padding: 5px;
    text-align: center;
}

h4 {
    border-bottom: solid 1px lightgray;
    margin-bottom: 5px;
}
```

1.5.10. Код таблицы usersStyle

```
.userInfo{
    border: solid 1px lightgray;
    border-radius: 3px;
    padding: 10px;
    background-color: white;
```

```
}


#usersHere{
   border: solid 1px lightgray;
   border-top-left-radius: 3px;
   border-bottom-left-radius: 3px;
   padding: 10px;
}

h3,h4{
   text-align: center;
}

img{
   margin-left: 3%;
}

#filterContainer{
   text-align: center;
   border-top: solid 1px lightgray;
   border-bottom: solid 1px lightgray;
   border-right: solid 1px lightgray;
   background-color: #f5f5f5;
   border-top-right-radius: 3px;
   border-bottom-right-radius: 3px;
   padding-bottom: 5px;
   padding-top: 5px;
}
```

1.6. Код JavaScript функций

1.6.1. Код функции createAuctionFormProcessor

```
/**
 * Обрабатывает клики на радио батоны.
 * Функция нужна для того, чтобы снимать клики с других радио-батонов,
 * чтобы в своей области была только одна кликнута.
 *
 * Объединены по принципу:
 * ProductType: book, product, service;
 * TimeBound: 1day, 1dayTime, 1weekTime;
 *
 * @param btn кнопка, которая кликается
 * 11.04.2015
 */
```

```
function auctionRadioCheckHandler(btn) {
   switch (btn.id) {
      case 'bookRadio':
         productType = btn.id;
         $('#productRadio').prop('checked', false);
         $('#serviceRadio').prop('checked', false);
         break;
      case 'productRadio':
         productType = btn.id;
         $('#bookRadio').prop('checked', false);
         $('#serviceRadio').prop('checked', false);
         break;
      case 'serviceRadio':
         productType = btn.id;
         $("#bookRadio").prop('checked', false);
         $("#productRadio").prop('checked', false);
         break;
      case '4hoursTime':
         timeBound = btn.id;
         $('#1dayTime').prop('checked', false);
         $('#1weekTime').prop('checked', false);
         break;
      case '1dayTime':
         timeBound = btn.id;
         $("#4hoursTime").prop('checked', false);
         $("#1weekTime").prop('checked', false);
         break;
      case '1weekTime':
         timeBound = btn.id;
         $("#4hoursTime").prop('checked', false);
         $('#1dayTime').prop('checked', false);
         break;
   }
}
var timeBound = ''; //хранит данные о кликнутом чекбоксе о времени
var productType = ''; //хранит данные о типе продукта  чекнутого

/**
 * Проверяет, что указаны границы по времени и тип продаваемой вещи.
 * Выделяет красным цветом форму, в которую ничего не введено.
 * Возвращает старый цвет, если все введено корректно
 *
 * Возвращает true, если все правильно отмечено
 *          false, если нет.
 * @returns {boolean}
 */
function checkRadio() {
```

```
    var timeBoundIsEmpty;
    var productTypeIsEmpty;
    var canSend = true;
    //проверка timeBound
    if (timeBound == "") {
        timeBoundIsEmpty = true;
        $("#timeContainer").css('border-color', 'red');
        canSend = false;
    } else {
        timeBoundIsEmpty = false;
        $("#timeContainer").css('border-color', '#48b8e5');
    }

    //проверка productType
    if (productType == '') {
        productTypeIsEmpty = true;
        canSend = false;
        $("#typeOfAuctionContainer").css('border-color', 'red');
    } else {
        productTypeIsEmpty = false;
        $("#typeOfAuctionContainer").css('border-color', '#48b8e5');
    }

    return canSend;
}

/**
 * Обрабатывает событие onchange поля productName.
 * Если она не пустая -> устанавливает ей старый цвет.
 */
function changeInputColor() {
    if ($("#productName").val().length != 0) {
        $("#productName").css('border-color', 'rgb(204, 204, 204)');
    }
}

/**
 * Возвращает тип товара в зависимости от названия кнопки
 * @param productType
 * @returns {string}
 */
function getProductType(productType) {
    switch (productType) {
        case 'bookRadio':
            return 'Книга';
            break;
        case 'productRadio':
            return 'Вещь';
```

```
            break;
        case 'serviceRadio':
            return 'Услуга';
            break;
    }
}

/**
 * Проверяет поле для ввода названия продукта.
 * Если оно непустое - меняет его цвет на красный.
 * Возвращает true, если все введено корректно
 *          false, если нет.
 * @returns {boolean}
 */
function checkInput() {
    var productName = $("#productName").val();
    if (productName.length == 0) {
        $("#productName").css('border-color', 'red');
        return false;
    } else {
        return true;
    }
}

function checkUniversity() {
    if (ids.length > 1) {
        $("#universitySearch").css('border-color', 'rgb(204, 204, 204)');
        return true;
    } else {
        $("#universitySearch").css('border-color', 'red');
        return false;
    }
}


/**
 * Отправляет данные на севрер. Проверяет все поля на корректность.
 */
function sendData() {
    var canSend = checkRadio() & checkInput();
    createStringOfHouses();
    canSend = canSend & checkUniversity();
    if (canSend) {
        //TODO проверить дома, когда будут
        //отправить все на сервер
        $("#progressBar").show();
        $("#createAuctionBtn").hide();
```

```
var formData = new FormData();

//добавляю файл
var fileInput = document.getElementById('fileInput');
if (fileInput.files.length > 0) {
   formData.append(fileInput.files[0].name, fileInput.files[0]);
}

//добавляем временные границы
formData.append("timeBound", timeBound);

//добавляю поле с именем
var name = $("#productName").val();
formData.append("name", name);

//добавляю поле с опмсание
var description = $("#descriptionName").val();
formData.append("description", description);

//добавляю поле с категорией
var typeOfProductToSend = getProductType(productType);
formData.append('category', typeOfProductToSend);

formData.append('housesIds', ids);

//создаю запрос
var xhr = new XMLHttpRequest();

//если все хорошо -
xhr.upload.onprogress = function(e) {
   $('#bar').css('width', (e.loaded / e.total) * 100 + '%');
}

xhr.onreadystatechange = function() {
   if (xhr.readyState == 4 && xhr.status == 200) {
      $('#progressBar').hide(); //прячем прогресс бар
      clearForm(); //очищаем форму
      var auction = JSON.parse(xhr.responseText); //парсим ответ от сервера
      link = 'http://cocktion.com/Auction/CurrentAuction/' + auction.Id; //фигачим ссылочку
      $("#readyInfo").show();
      $("#readyInfo").append("<a href=\"" + link + "\">" + 'Перейти на аукцион!' + "</a><br/>");
      $("#createAuctionBtn").show();
      $("#universitySearch").val("");
      $("#universitySearchResults").empty();
      $("#facultyToChoose").empty();
      $('#chooseFacultyRadio').prop('checked', false);
      $('#allFacultyRadio').prop('checked', false);
   };
```

```
        }

        xhr.open('POST', '/BidAuctionCreator/CreateAuction');
        xhr.send(formData); //отправка данных
    }
}

var link;

function goToAuction() {
    window.location = link;
}

/**
 * Очищает чекбокс по айдишнику
 * @param checkBoxId
 */
function clearCheckboxes(checkBoxId) {
    switch (checkBoxId) {
        case 'bookRadio':
            $('#bookRadio').prop('checked', false);
            break;
        case 'productRadio':
            $('#productRadio').prop('checked', false);
            break;
        case 'serviceRadio':
            $('#serviceRadio').prop('checked', false);
            break;
        case '4hoursTime':
            $('#4hoursTime').prop('checked', false);
            break;
        case '1dayTime':
            $("#1dayTime").prop('checked', false);
            break;
        case '1weekTime':
            $("#1weekTime").prop('checked', false);
            break;

    }
}

/**
 * Чистит всю форму после отправки
 */
function clearForm() {
    $("#productName").val("");
    $("#descriptionName").val("");
    $("#fileInput").val("");
```

```
    clearCheckboxes(timeBound);
    clearCheckboxes(productType);
}
```

1.6.2. Код функции search

```
/*Ищет университет в базе данных
  Динамически выводит результат на экран
*/
function searchUniversity() {
    $("#universitySearchResults").empty();
    $('#chooseFacultyRadio').prop('checked', false);
    $('#allFacultyRadio').prop('checked', false);
    $("#facultyToChoose").empty();
    if ($("#universitySearch").val().length > 0) {
        var xhr = new XMLHttpRequest();


        var formData = new FormData();
        formData.append('university', $("#universitySearch").val().trim());

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var searchResults = JSON.parse(xhr.responseText);
                if (searchResults.IsSearchEmpty == true) {
                    if (!document.getElementById('emptySearch')) {
                        $("#universitySearchResults").append(generateSearchRespond('Тю-тю...', false));
                    }
                } else {
                    var univLength = searchResults.Names.length;
                    for (var i = 0; i < univLength; i++) {
                        if (!document.getElementById(searchResults.Names[i])) {
                            $("#universitySearchResults").append(generateSearchRespond(searchResults.Names[i],
true));
                        }
                    }
                }
            };
        }

        xhr.open('POST', '/Search/SearchUniversity');
        xhr.send(formData); //отправка данных
    }
}

function generateSearchRespond(respondString, status) {
    if (status) {
```

```
        return "<div onclick=\"onClickHandler(this.id)\" onmouseout=\"onMouseOutHandler(this.id)\"
onmouseover=\"onMouseOverHandler(this.id)\" class =\"searchRespond\" id=\"" + respondString + "\">"
+
        respondString + "</div>";
    }
    return "<div  class =\"searchRespond\" id=\"emptySearch\">" +
        respondString + "</div>";
}

function onMouseOverHandler(searchResult) {
    $("#" + searchResult).css('background-color', 'blue');
}

function onMouseOutHandler(param) {
    $("#" + param).css('background-color', 'whitesmoke');
}

function onClickHandler(param) {
    $("#universitySearch").val(param);
    $("#universitySearchResults").empty();
}

function constructFacultyToAppend(facultyName, facultyId) {
    return "<p><input type=\"checkbox\" onclick=\"addCheckedId(this)\" id=\"" + facultyId + "\"> " +
facultyName + "</p>";
}

function getFacultyList() {
    var xhr = new XMLHttpRequest();
    $("#facultyToChoose").empty();
    var formData = new FormData();
    formData.append('university', $("#universitySearch").val().trim());


    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
            var faculties = JSON.parse(xhr.responseText);
            if (faculties.IsEmpty == false) {
                $("#universitySearch").css('border-color', 'rgb(204, 204, 204)');
                for (var i = 0; i < faculties.Names.length; i++) {
                    $("#facultyToChoose").append(constructFacultyToAppend(faculties.Names[i],
faculties.Ids[i]));
                }
            } else {
                $("#universitySearch").css('border-color', 'red');
            }
        }
    }
```

```
    xhr.open('POST', '/Search/GetFacultyList');
    xhr.send(formData); //отправка данных
}

function facultyRadioCheckHandler(btn) {
    $("#universitySearchResults").empty();
    switch (btn.id) {
        case 'allFacultyRadio':
        housesIds = new Array();
        $('#chooseFacultyRadio').prop('checked', false);
        $("#facultyToChoose").empty();
        break;
    case 'chooseFacultyRadio':
        $('#allFacultyRadio').prop('checked', false);
        getFacultyList();
        break;
    }
}

var housesIds = new Array();

function addCheckedId(btn) {
    if (btn.checked) {
        housesIds.push(btn.id);
    }
}

var ids = "?";

function createStringOfHouses() {
    if (housesIds.length == 0) {
        ids = $("#universitySearch").val();
    } else {
        for (var i = 0; i < housesIds.length; i++) {
            ids += (housesIds[i] + '!');
        }
    }
    return ids;
}
```

1.6.3. Код функции auctionEndLogic

```
/*Печатает на экране пользователя результаты аукциона,
в зависимости от типа пользователя.
*/
function printAuctionResults(obj) {
```

```
    if (obj.Type == 'Owner') {
        $('#auctionInfoContainer').empty();
        $('#auctionInfoContainer').append(obj.Message);
        $("#showAuctionInfoBtn").click();
    }//end of if
    else if (obj.Type == 'Winner') {
        $('#auctionInfoContainer').empty();
        $('#auctionInfoContainer').append(obj.Message);
        $("#showAuctionInfoBtn").click();
    }//end of if
    else if (obj.Type == 'Info') {
        $('#timer').append(obj.Message);
    }
    else if (obj.Type == 'Looser') {
        $('#timer').append(obj.Message);
    }
    else if (obj.Type == 'Owner_undfnd') {
        $('#timer').append(obj.Message);
    }
}

/*Запрашивает с сервера результаты аукциона
*/
function getAuctionResults() {
    var winnerNameRequest = new XMLHttpRequest();
    var idSender = new FormData();
    idSender.append('auctionId', auctionId);
    winnerNameRequest.open('POST', '/AuctionRealTime/SendAuctionResults');
    winnerNameRequest.send(idSender);
    winnerNameRequest.onreadystatechange = function () {
        if (winnerNameRequest.readyState == 4 && winnerNameRequest.status == 200) {
            var info = JSON.parse(winnerNameRequest.responseText);
            $('#timer').empty();
            printAuctionResults(info);
        };
    };
    return false;
};

/*Запускается, когда таймер заканчивает работу*/
function sayThatFinished() {
    chat.server.finishAuction(auctionId);
};
```

1.6.4. Код функции bidFunctions

```
function checkBidForm() {
```

```
    if ($("#bidName").val().length == 0) {
        $("#bidName").css('border-color', 'red');
        return false;
    } else {
        $("#bidName").css('border-color', 'rgb(204, 204, 204)');
        return true;
    }
}

function checkCategory() {
    if (category.length == 0) {
        $("#typeOfAuctionContainer").css('border', 'solid 1px red');
        return false;
    } else {
        $("#typeOfAuctionContainer").css('border', 'solid 1px rgb(204, 204, 204)');
        return true;
    }
}

var category = "";

function radioProcessor(btn) {
    switch (btn.id) {
    case 'bookRadio':
        $('#productRadio').prop('checked', false);
        $('#serviceRadio').prop('checked', false);
        category = 'Книга';
        break;
    case 'productRadio':
        $('#bookRadio').prop('checked', false);
        $('#serviceRadio').prop('checked', false);
        category = 'Вещь';
        break;
    case 'serviceRadio':
        $("#bookRadio").prop('checked', false);
        $("#productRadio").prop('checked', false);
        category = 'Услуга';
        break;
    }
}

//Скрипт, добавляющий ставку к аукциону.
function addBid(auctionId) {
    if (checkBidForm() & checkCategory()) {
        var formData = new FormData();
        $("#progressBar").show();

        //добавляю файл
```

```
//добавляю файл
var fileInput = document.getElementById('fileInput');
if (fileInput.files.length > 0) {
    formData.append(fileInput.files[0].name, fileInput.files[0]);
}

//добавляю поле с именем
var bidName = document.getElementById('bidName').value;
formData.append("name", bidName);

//добавляю поле с опмсание
var bidDescription = document.getElementById('bidDescription').value;
formData.append("description", bidDescription);

formData.append('category', category);

//adding a field with auctionId
formData.append('auctionId', auctionId);

//создаю запрос
var xhr = new XMLHttpRequest();

//если все хорошо
xhr.upload.onprogress = function(e) {
    $('#bar').css('width', (e.loaded / e.total) * 100 + '%');
}

xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        //получение статуса с сервера при отправке
        // document.getElementById('updater').outerText = 'успешно добавлено';
        //TODO добавить код взаимодействия с пользователем

        //Зачистка полей
        $('#bidName').val("");
        $('#bidDescription').val("");
        $('#bidCategory').val("");
        $("#progressBar").hide();
        $("#fileInput").val("");
        // $("#addExtraBid").show();
        //добавление нодика ко всем остальным клиентам
        // chat.server.addNodesToClients(name1, filename1, auctionId, xhr.responseText);
    };
}

xhr.open('POST', '/BidAuctionCreator/AddProductBet');
xhr.send(formData); //отправка данных
```

```
        return false;
    }
}

function addExtraBid(auctionId) {
    var formData = new FormData();

    //добавляю файл
    var fileInput = document.getElementById('extraFileInput');
    filename1 = fileInput.files[0].name;
    formData.append(fileInput.files[0].name, fileInput.files[0]);

    //добавляю поле с именем
    var bidName = document.getElementById('extraBidName').value;
    name1 = bidName;
    formData.append("name", bidName);

    //добавляю поле с опмсание
    var bidDescription = document.getElementById('extraBidDescription').value;
    formData.append("description", bidDescription);

    //добавляю поле с категорией
    var bidCategory = document.getElementById('extraBidCategory').value;
    formData.append('category', bidCategory);

    //adding a field with auctionId
    formData.append('auctionId', auctionId);

    //создаю запрос
    var xhr = new XMLHttpRequest();
    xhr.open('POST', '/BidAuctionCreator/AddExtraBid');
    xhr.send(formData); //отправка данных
    //если все хорошо
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            //получение статуса с сервера при отправке
            document.getElementById('extraBidContainer').outerText = 'успешно добавлен довесок';
            //добавление нодика ко всем остальным клиентам
            //var response = JSON.parse(xhr.responseText);
            //var parentId = response.FirstBidId;
            //var childId = response.ThisBidId;
            //chat.server.addNodesToClients(name1, filename1, auctionId, xhr.responseText);
            //chat.server.addExtraNodeToClients(name1, filename1, auctionId, parentId, childId);
        };

    }
}
function showExtraBidAdder() {
```

```
    $("#extraBidContainer").show();
}

//Функция проверяет на то, что чувак сделал ставочку
function check(modelId) {
    var xhr = new XMLHttpRequest();
    var data = new FormData();
    data.append("auctionId", modelId);
    xhr.open("POST", "/AuctionRealTime/CheckIfUserBidded");
    xhr.send(data);
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            var info = JSON.parse(xhr.responseText);
            $("#nodeInfo").append(info.HaveBid);
        };
    };
};
```

1.6.6. Код функции btnsOnClickHandler

```
//TODO Засунуть функцию такую в другое место...
//при нажатии на кнопку выбрать лидера
function chooseLider() {
    var data = new FormData();
    data.append('auctionId', auctionId);
    data.append('productId', nodeId);
    winnerId = nodeId;
    var request = new XMLHttpRequest();
    request.open('POST', '/AuctionRealTime/AddLider');
    request.send(data);
    request.onreadystatechange = function () {
        if (request.readyState == 4 && request.status == 200) {
            var resp = JSON.parse(request.responseText);
            if (resp.Status == 'True') {
                $("#chooseLeaderInfo").empty();
                $("#chooseLeaderInfo").append("Лидер успешно выбран:)");
                isChoosed = true;
            } else {
                alert("Извини, мы облажались... Попробуй еще раз...")
                alert(resp.Status);
            }
        }
    };
    return false;
};

//Функция для завершения аукциона
```

```
function finishAuction() {
    if (isChoosed == true) {
        sayThatFinished();
    } else {
        alert("Необходимо выбрать победителя, прежде чем завершать аукцион:)");
    }
};

//посылает ставку яичную
function makeEggBet() {
    if (checkIfCanSendEggs()) {
        $("#toteProgress").show();
        $("#eggInfo").empty();
        $("#eggsAmountTextbox").hide();
        $("#sendEggsBtn").hide();
        var formData = new FormData();
        var eggsAmnt = $('#eggsAmountTextbox').val();
        formData.append('eggsAmount', eggsAmnt);
        formData.append('auctionId', auctionId);
        formData.append('productId', nodeId);
        var req = new XMLHttpRequest();
        req.onreadystatechange = function() {
            if (req.readyState == 4 && req.status == 200) {
                var info = JSON.parse(req.responseText);
                sayResultOfAddingToteBet(info.Status, info.UsersAmountOfEggs);
                $("#eggInfo").show();
                $("#eggsAmountTextbox").val("");
                $("#toteProgress").hide();
                $("#eggsAmountTextbox").show();
                $("#sendEggsBtn").show();
            };
        };

        req.upload.onprogress = function(e) {
            $('#toteBar').css('width', (e.loaded / e.total) * 100 + '%');
        }

        req.open('POST', '/AuctionRealTime/AddToteRate');
        req.send(formData);
    }
}

function eggsTextBoxOnChange() {
    if ($("#eggsAmountTextbox").val().length != 0) {
        $("#eggsAmountTextbox").css('border-color', 'rgb(204, 204, 204)');
    }
}
```

```
function checkIfCanSendEggs() {
    if ($("#eggsAmountTextbox").val().length == 0) {
        $("#eggsAmountTextbox").css('border-color', 'red');
        return false;
    }
    return true;
}

function sayResultOfAddingToteBet(status, eggs) {
    if (status == true) {
        $("#eggInfo").append("<p>Ставка поставлена, осталось яиц: " + eggs + "</p>");
    } else {
        $("#eggInfo").append("<p>К сожалению, мне не хватило этого количества яиц для такой ставки
(" + eggs + ")</p>");
    }
}

//Показывает контейнер с товарами на аукционе
function showProducts(btn, isOwner) {
    if (isOwner == 'True') {
        btn.focus();
        $("#productsContainer").show();

        $("#toteBetsContainer").hide();
        $("#auctionInfoContainer").hide();

    } else {
        btn.focus();
        $("#productsContainer").show();
        $("#toteBetsContainer").hide();
        $("#auctionInfoContainer").hide();
    }
}

//Показывает информацию о ставках на аукционе
function showToteBets(btn) {
    btn.focus();
    $("#toteBetsContainer").show();
    $("#productsContainer").hide();
    $("#auctionInfoContainer").hide();
}

//Показывает окошко с информацией об аукционе
function showAuctionInfo(btn) {
    btn.focus();
    $("#auctionInfoContainer").show();
    $("#productsContainer").hide();
    $("#toteBetsContainer").hide();
```

```
}

//Показывает контейнер для добавления своего товара на
//аукцион
function showProductBids(btn, userStatus, userName) {
    btn.focus();
    if (userStatus == 'True') {
        if (userName == ownerName) {
            $('#productBidErrorInfo').empty();
            $('#productBidErrorInfo').show();
            $('#productBidErrorInfo').append('<p>Вы же владелец;) Вы уже добавили)</p>');
            $("#productBidContainer").hide();
            $("#chatContainer").hide();
            $('#stopAuction').show();
        } else {
            $("#productBidContainer").show();
            $("#chatContainer").hide();
        }
    } else {
        $('#productBidErrorInfo').show();
        $('#productBidErrorInfo').empty();
        $("#productBidContainer").hide();
        $("#chatContainer").hide();
        $('#productBidErrorInfo').append('<p>Для того, чтобы принять участие в аукционе, пожалуйста,
авторизуйтесь или зарегистрируйтесь:) </p>');
    }
}

//Показывает контейнер с чатом
function showChat(btn, userStatus) {
    btn.focus();
    if (userStatus == 'True') {
        $("#chatContainer").show();
        $("#productBidContainer").hide();
        $('#productBidErrorInfo').hide();
    } else {
        $('#productBidErrorInfo').empty();
        $('#productBidErrorInfo').show();
        $("#chatContainer").show();
        $("#productBidContainer").hide();
        $("#chatControlPanel").hide();
        $('#productBidErrorInfo').append('<p>Чтобы отправлять сообщения, нужно быть
авторизованным!</p>')
    }
}
```

1.6.7. Код функции hubConnection

```
function initialiseHub(modelId, isUseAuth, userIdentityName) {
    chat = $.connection.auctionHub;
    //Функция для добавления сообщения на страничку
    chat.client.addNewMessageToPage = function (name, message) {
        var textToAppend = '<p><b>' + name + ': </b>' + message + '</p>';
        $("#chatMessagesContainer").append(textToAppend);
    };

    //Функция для отображения информации о тотализаторе
    chat.client.updateToteBoard = function(data)
    {
        $('#coefficientsTable').empty();
        $('#coefficientsTable').append("<p>"+data + "</p>");
    };

    chat.client.addExtraNodesToPages = function (fileName, name, parentProductId, childProductId) {
        nodes.add([{ id: childProductId, label: name.trim(), shape: 'circularImage', image: 'http://
cocktion.com/Images/Thumbnails/' + fileName }]);
        edges.add([{ from: parentProductId, to: childProductId }]);
    }

    //Функция для добавления товара на все страницы аукциона
    chat.client.addNodesToPages = function (fileName, name, productId){
        nodes.add([{ id: productId, label: name.trim(), shape: 'circularImage', image: 'http://cocktion.com/
Images/Thumbnails/' + fileName }]);
        edges.add([{ from: i, to: productId }]);

    };

    //функция для объявления лидера на всех страничках
    //ВО ВРЕМЯ АУКЦИОНА!!!!!!!
    chat.client.showLeaderOnPage = function (leaderId, liderName){
        $('#leaderHolder').empty();
        $('#leaderHolder').append('<p><b>Лидер аукциона: </b>' + liderName+'</p>');
        isChosen = true;
    };

    chat.client.finishAuction = function()
    {
        $n.countdown('stop');
        getAuctionResults();
    };

    $.connection.hub.start().done(function () {
        chat.server.addNewRoom(modelId);
        chat.server.getTote(modelId);
```

```
        $('#sendMessageBtn').click(function() {
            if (isUseAuth == 'True') {
                chat.server.send(userIdentityName, $("#enterMessageTextBox").val(), modelId);
                $("#enterMessageTextBox").val("").focus();
            } else {
                alert("Я же предупредил, что вам нужно авторизоваться;)");
            }
        });
    });
}
```

1.6.8. Код функции nodeProcessor

```
//Функция для получения информации о нодике
function getNodeInfo(nodeId) {
    var req = new XMLHttpRequest();
    var data = new FormData();
    data.append("Id", nodeId);
    req.open('POST', '/AuctionRealTime/SendInfoAboutProduct');
    req.send(data);
    req.onreadystatechange = function () {
        if (req.readyState == 4 && req.status == 200) {
            var info = JSON.parse(req.responseText);
            $('#productPhotoContainer').empty();
            $('#productPhotoContainer').append(
                '<img class=\"img-circle\" src=\"' + 'http://cocktion.com/Images/Thumbnails/' + info.FileName
+ "\">"
            );

            $("#productInfoContainer").empty();
            $("#productInfoContainer").append('<p><b>Что: </b>' + info.Name + '</p>');
            $("#productInfoContainer").append('<p><b>Конкретно: </b>' + info.Description + '</p>');
            $("#productInfoContainer").append('<p><b>Категория: </b>' + info.Category + '</p>');

            $("#showNodeInfoBtn").click();
        };
    }
    req.onprogress = function () {
        $('#nodeInfo').append('<p>Идет получение информации </p>');
    }
};



//если выбран какой-то из товаров
function nodeSelected(properties) {
    nodeId = properties.nodes[0];
```

```
    //проверка того, зашел зарегистрированный пользователь
    //или нет
    if (nodeId != null) {
        getNodeInfo(nodeId);
        if (userStatus == 'True') { //если зарегистрирован
            if (userName == ownerName) {
                //проверяем выбрал ли или нет
                if (isChoosed == false) {
                    $('#chooseLeaderForm').show();
                    $('#chooseLeaderInfo').empty();
                    $('#chooseLeaderInfo').append('<p>Выбрать этот товар лидером?</p>');
                } else {
                    $('#chooseLeaderForm').show();
                    $('#chooseLeaderInfo').empty();
                    $('#chooseLeaderInfo').append('<p>Сменить выбор на этот?</p>');
                }
            }
        }
        if (nodeId == i) {
            $("#addToteBetContainer").hide();
        } else {
            $("#addToteBetContainer").show();
        }
    }
}

//Функция добавляет кружочек на "карту" аукциона.
function addNode(fileName, name, productId) {
    nodes.add([{ id: productId, label: name.trim(), shape: 'circularImage', image: 'http://cocktion.com/
Images/Thumbnails/' + fileName }]);
    edges.add([{ from: i, to: productId }]);
};

/*Функция добавляет дополнительную связь между довеском
        и тем товаром, к которому он приклеен
        fileName - название файла на сервер
        name - имя товара, которое надо отобразить
        parentId - айдишник товара, к которому надо приклеить
        childId - айдишник товара, который надо приклеить к
        parent'y
        */
function addExtraNode(fileName, name, parentId, childId) {
    nodes.add([{ id: childId, label: name.trim(), shape: 'circularImage', image: 'http://cocktion.com/Images/
Thumbnails/' + fileName }]);
    edges.add([{ from: parentId, to: childId }]);
};
```

```
var edges = new vis.DataSet();
var nodes = new vis.DataSet();
//если выбран победитель аукциона
var nodeId;

function initialiseNetwork(sellProductName, sellProductPhotoName) {
    // create an array with nodes
   // var nodes = new vis.DataSet();
    nodes.add([
        { id: i, label: sellProductName.trim(), shape: 'circularImage', image: 'http://cocktion.com/Images/
Thumbnails/' + sellProductPhotoName}
    ]);
    // create an array with edges
    //var edges = new vis.DataSet();
    // create a network
    var container = document.getElementById('mynetwork');
    var data = {
        nodes: nodes,
        edges: edges,
    };
    var options = {
        width: '100%',
        height: '600px'
    }
    var network = new vis.Network(container, data, options);
    network.on('select', nodeSelected);
}
```

1.6.9. Код функции auctionAdder

```
//Добавляет ячейку с информацией об аукционе в таблицу
//containerName - название элемента, в котором содержится таблица
//name - наименование аукциона
//description - описание аукциона
//date - дата окончания аукциона
//imgSrc - путь к фотке аукциона
//link - ссылка на этот аукцион
function addCellToTheGrid(containerName, name, description, date, imgSrc, link) {
    if (auctionColCounter < 4) {
        if (auctionColCounter == 0) {
            addAuctionRow(containerName, auctionRowCounter);
            appendAuctionInfo(auctionRowCounter, name, description, date, imgSrc, link);
            auctionColCounter++;
        }
        else {
            appendAuctionInfo(auctionRowCounter, name, description, date, imgSrc, link);
            auctionColCounter++;
```

```
        }
    }
    else {
        auctionColCounter = 0;
        ++auctionRowCounter;
        addAuctionRow(containerName, auctionRowCounter);
        appendAuctionInfo(auctionRowCounter, name, description, date, imgSrc, link);
        auctionColCounter++;
    }
}

//Добавляет col с аукционом в нужный row
//nameOfRow - название row, в который нужно вставить
//description - описание аукциона
//date - дата окончания
//imgSrc - путь к фотке
//link - ссылка на аукцион
function appendAuctionInfo(nameOfRow, name, description, date, imgSrc, link) {
    document.getElementById('auction'+nameOfRow).innerHTML += ("<div class=\"col-md-3\"><div
class=\"auction\"> " +
        "<img class=\"img-circle\" src=\"" + imgSrc + "\" >" +
    "<p><b>Продается: </b>" + name + "</p>" +
    "<p><b>Описание: </b>" + description.trim() + "</p>" +
    "<p><b>Кончится: </b>" + date + "</p>" +
    "<p id=\"auctionLink\"><a href=\"" + link + "\">Заглянуть</a></p>" +
    "</div></div>");
}

//Добавляет новый div (class=row) на страничку, для того, чтобы в нее потом запихивать колонки
function addAuctionRow(containerName, nameOfRow) {
    //$("#panelContainer").append("<div class=\"row\" id=\"" + nameOfRow+"\">добавил!</div>");
    document.getElementById(containerName).innerHTML += "<div class=\"row\" id=\"auction" +
nameOfRow + "\"></div></br>";
}

var auctionRowCounter = 0;
var auctionColCounter = 0;

//Вписывает в указаныний элемент информацию о том, что никто ничего на аукционе не продает
//10.04.2015
function sayThatEmpty(elementName) {
    document.getElementById(elementName).innerHTML = "<p>К сожалению, в данный момент
ничего не продается ;(</p>";
}
```

1.6.10. Код функции hubConnection

```
//Коннектится к хабу и добавляет свеженькие аукциончики в список
$(function () {
   var adder = $.connection.auctionListHub;
   adder.client.addAuctionToTheList = function (name, description, category, time, photoPath, id) {
      addCellToTheGrid('auctionPanel', name, description,
            time, 'http://cocktion.com/Images/Thumbnails/' + photoPath,
            'http://cocktion.com/Auction/CurrentAuction/' + id);
   }
   $.connection.hub.start().done(function () {
   });
});
```

1.6.11. Код функции commenter

```
//Добавляет комментарий к аукционному дому
function comment(modelId) {
   if ($("#message").val().length != 0)
   {
      var formData = new FormData();
      var messageData = $("#message").val();
      $("#message").val("");
      formData.append("message", messageData);
      formData.append("houseId", modelId);
      var xhr = new XMLHttpRequest();
      xhr.open("POST", "/AuctionHouse/AddComment");
      xhr.send(formData);
      xhr.onreadystatechange = function () {
         if (xhr.readyState == 4 && xhr.status == 200) {
            var respond = JSON.parse(xhr.responseText);
            if (respond.Status == "Success") {
               $("#messageContainer").append("<div id=\"postContainer\"><div id=\"author\"><p><b>" +
respond.Author + "</b> сказал сегодня:</p></div><div id=\"messageInstance\"><p>" + messageData +
"</p></div></div><br /><br />");
            } else
               $("#messageContainer").prepend("<p>" + "Не удалось!" + "</p>");

         }
      }
   }
   else
   {
      alert("Необходимо ввести сообщение свое!");
   }
}
```

1.6.12. Код функции commenter

```
//Добавляет комментарий к аукционному дому
function comment(modelId) {
   if ($("#message").val().length != 0)
   {
      var formData = new FormData();
      var messageData = $("#message").val();
      $("#message").val("");
      formData.append("message", messageData);
      formData.append("houseId", modelId);
      var xhr = new XMLHttpRequest();
      xhr.open("POST", "/AuctionHouse/AddComment");
      xhr.send(formData);
      xhr.onreadystatechange = function () {
         if (xhr.readyState == 4 && xhr.status == 200) {
            var respond = JSON.parse(xhr.responseText);
            if (respond.Status == "Success") {
               $("#messageContainer").append("<div id=\"postContainer\"><div id=\"author\"><p><b>" +
respond.Author + "</b> сказал сегодня:</p></div><div id=\"messageInstance\"><p>" + messageData +
"</p></div></div><br /><br />");
            } else
               $("#messageContainer").prepend("<p>" + "Не удалось!" + "</p>");

         }
      }
   }
   else
   {
      alert("Необходимо ввести сообщение свое!");
   }
}
```

1.6.12. Код функции scroller

```
//число после оффсета - время в милисекундах
//скроллит при нажатии на кнопочку к верхнему элементу
function scrollToTheTop() {
   $('html, body').animate({
      scrollTop: $("#top").offset().top
   }, 1000);
};

function scrollToTheForum() {
   $('html, body').animate({
      scrollTop: $("#messageEnter").offset().top
```

```
    }, 1000);
}
```

1.6.13. Код функции subscriber

```
function  checkSubscription(modelId) {
    var formData = new FormData();
    formData.append('modelId', modelId);
    var xhr = new XMLHttpRequest();
    var response;
    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            response = JSON.parse(xhr.responseText);
            if (response.Status == "Success") {
                document.getElementById('subscribe').innerHTML = "<p>В вашем колхозе  <span class=
\"glyphicon glyphicon-ok\"></span></p>" +
                    "<p>Удалить из колхоза<button class=\"btn btn-default\" onclick=
\"unsubscribeFromHouse()\">" +
                    "<span class=\"glyphicon glyphicon-remove\"></span></button></p>"

            } else {
                document.getElementById('subscribe').innerHTML = "<p>Добавить в колхоз " + "<button
class=\"btn btn-default\" onclick=\"subscribeOnHouse()\">" +
                "<span class=\"glyphicon glyphicon-plus\"></span></button></p>";
            }
        };
    }

    xhr.open('POST', '/Subscription/CheckHouseSubscription');
    xhr.send(formData); //отправка данных
}

function unsubscribeFromHouse() {
    var formData = new FormData();
    formData.append('houseId', houseId);

    var xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            var response = JSON.parse(xhr.responseText);
            if (response.Status) {
                document.getElementById('subscribe').innerHTML = "<p>Удален из колхоза! <span class=
\"glyphicon glyphicon-ok\"></span></p>";
            } else {
                document.getElementById('subscribe').innerHTML = "<p>Попробуйте еще раз ;(</p>";
```

```
        }
    };
  }

  xhr.open('POST', '/Subscription/UsubscribeFromHouse');
  xhr.send(formData); //отправка данных
}


var houseId;
function subscribeOnHouse() {
  var formData = new FormData();
  formData.append('houseId', houseId);
  var xhr = new XMLHttpRequest();

  xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      var response = JSON.parse(xhr.responseText);
      if (response.Status) {
        document.getElementById('subscribe').innerHTML = "<p>В вашем колхозе  <span class=
\"glyphicon glyphicon-ok\"></span></p>";
      } else {
        document.getElementById('subscribe').innerHTML = "<p>Попробуйте еще раз ;(</p>";
      }
    };
  }

  xhr.open('POST', '/Subscription/SubscribeOnHouse');
  xhr.send(formData); //отправка данных
}
```

1.6.14. Код функции printHousesInCells

```
/**
 * Created by aleksandrlazarenko on 10.04.15.
 */

//Распределяет новую ячейку по сетки со столбцами
//nameOfRow - название строки, в которую надо вставить все
//nameOfFaculty - название факультета
//adress - адрес факультета
//link - ссылочка на факультет
//imgSrc - путь к фоточке
//containerName - название контейнера, в который надо вставить
function addHouseCell(containerName, nameOfFaculty, adress, link, imgSrc) {
  if (housesColCounter < 4) {
    if (housesColCounter == 0) {
```

```
            addHouseRow(containerName, housesRowCounter);
            appendHouseInfo(housesRowCounter, nameOfFaculty, adress, link, imgSrc);
            housesColCounter++;
        }
        else {
            appendHouseInfo(housesRowCounter, nameOfFaculty, adress, link, imgSrc);
            housesColCounter++;
        }
    }
    else {
        housesColCounter = 0;
        ++housesRowCounter;
        addHouseRow(containerName, housesRowCounter);
        appendHouseInfo(housesRowCounter, nameOfFaculty, adress, link, imgSrc);
        housesColCounter++;
    }
}

//Добавляет непосредственно саму информацию о доме в ячейку
//nameOfRow - название строки, в которую надо вставить все
//nameOfFaculty - название факультета
//adress - адрес факультета
//link - ссылочка на факультет
//imgSrc - путь к фоточке
function appendHouseInfo(nameOfRow, nameOfFaculty, adress, link, imgSrc) {
    document.getElementById(nameOfRow).innerHTML += ("<div class=\"col-md-3\"><div class=
\"house\"> " +
    "<img src=\"" + imgSrc + "\" class= \"img-thumbnail\">" + //вставляем фотографию
    "<p><b>Факультет: </b>" + nameOfFaculty + "</p>" +    //название факультета
    "<p><b>Адрес: </b>" + adress + "</p>" + //адрес факультета
    "<p><a href=\"" + link + "\">Заглянуть</a></p></div></div>"); //ссылочка
}

//Добавляет новый div (class=row) на страничку, для того, чтобы в нее потом запихивать колонки
//containerName - название контейнера, в который надо вставить
//nameOfRow - название строчки, в которую надо вставить что-то
function addHouseRow(containerName, nameOfRow) {
    document.getElementById(containerName).innerHTML += "<div class=\"row\" id=\"" + nameOfRow
+ "\"></div></br>";
}

var housesRowCounter = 0;
var housesColCounter = 0;
```

1.6.15. Код функции printHolders

```
function addHolderCell(containerName, holderName, link, imgSrc) {
    if (holderColCounter < 4) {
```

```
        if (holderColCounter == 0) {
            addHolderRow(containerName, holderRowCounter);
            appendHolderInfo(holderRowCounter, holderName, link, imgSrc);
            holderColCounter++;
        }
        else {
            appendHolderInfo(holderRowCounter, holderName, link, imgSrc);
            holderColCounter++;
        }
    }
    else {
        holderColCounter = 0;
        ++holderRowCounter;
        addHolderRow(containerName, holderRowCounter);
        appendHolderInfo(holderRowCounter, holderName, link, imgSrc);
        holderColCounter++;
    }
}

//Добавляет непосредственно саму информацию о доме в ячейку
//nameOfRow - название строки, в которую надо вставить все
//nameOfFaculty - название факультета
//adress - адрес факультета
//link - ссылочка на факультет
//imgSrc - путь к фоточке
function appendHolderInfo(nameOfRow, nameOfHolder, link, imgSrc) {
    document.getElementById(nameOfRow).innerHTML += ("<div class=\"col-md-3\"><div class=
\"holder\"> " +
    "<img src=\"" + imgSrc + "\" >" + //вставляем фотографию
    "<p><b>Название: </b>" + nameOfHolder + "</p>" +    //название факультета
    "<p><a href=\"" + link + "\">Заглянуть</a></p></div></div>"); //ссылочка
}

//Добавляет новый div (class=row) на страничку, для того, чтобы в нее потом запихивать колонки
//containerName - название контейнера, в который надо вставить
//nameOfRow - название строчки, в которую надо вставить что-то
function addHolderRow(containerName, nameOfRow) {
    document.getElementById(containerName).innerHTML += "<div class=\"row\" id=\"" + nameOfRow
+ "\"></div></br>";
}

var holderRowCounter = 0;
var holderColCounter = 0;
```

1.6.16. Код функции formVerifier

//Проверяет форму для ввода нового дома

```
function verifyHouseForm() {
   if ($("#description").val().length == 0) {
      alert("Не введено описание дома!");
      return false;
   } else if ($("#faculty").val().length == 0) {
      alert("Не введен факультет/ общага!");
      return false;
   } else if ($("#adress").val().length == 0) {
      alert("Не введен адрес!");
      return false;
   }  else if ($("#housePhoto").val().length == 0) {
      alert('Нет фотографии дома!!!');
      return false;
   }
   else if ($("#selector").val() == -1) {
      alert('Не выбран холдер!');
      return false;
   }
   return true;
}

//првоеряет форму с холдером на корректность
function verifyHolderForm() {
   if ($("#holderName").val().length == 0) {
      alert("Не введено название холдера");
      return false;
   } else if ($("#holderPhoto").val().length == 0) {
      alert('Не введена фотография дома');
      return false;
   } else if ($("#holderCity").val().length == 0) {
      alert('Не введен город ');
      return false;
   }
   else
      return true;
}
```

1.6.17. Код функции holderSender

```
//посылает на сервак holder
function sendHolderToServer() {

   if (verifyHolderForm()) {
      var confirmString = "Название: " + $("#holderName").val() + "\n" + "город: " +
         $("#holderCity").val() + "\n" + "Все окей? Добавляю?";
      if (confirm(confirmString)) {
```

```
        var formData = new FormData();
        var xhr = new XMLHttpRequest();
        formData.append("holderName", $("#holderName").val());
        formData.append("holderCity", $("#holderCity").val());
        var fileInput = document.getElementById('holderPhoto');
        if (fileInput.files.length > 0) {
           formData.append(fileInput.files[0].name, fileInput.files[0]);
        }

        xhr.onreadystatechange = function () {
           if (xhr.readyState == 4 && xhr.status == 200) {
              alert("Ты все четко добавил;)");
              $("#holderName").val("");
              $("#holderCity").val("");
              $("#holderPhoto").val("");
              $('#holderBar').css('width', '0%');
              var respond = JSON.parse(xhr.responseText);
              alert(respond.Status);
           }
        }

        xhr.upload.onprogress = function (e) {
           $('#holderBar').css('width', (e.loaded / e.total) * 100 + '%');
        }

        xhr.open("POST", "/EditHouses/AddHolder");
        xhr.send(formData);
     }
  }
}
```

1.6.18. Код функции houseSender

```
//Посылает новый дом на сервак
function sendHouseToServer() {
   if (verifyHouseForm()) {

      var confirmString = "Описание: " + $("#description").val() + "\n" + "Факультет: " + $
("#faculty").val() +
         "\n" + "Адрес: " + $("#adress").val() + "\n" + "Все окей? Добавляю?";
      if (confirm(confirmString)) {

         var formData = new FormData();
         var xhr = new XMLHttpRequest();

         formData.append("description", $("#description").val());
```

```
        formData.append("faculty", $("#faculty").val());
        formData.append("adress", $("#adress").val());
        formData.append("holderId", $("#selector").val())
        var fileInput = document.getElementById('housePhoto');
        if (fileInput.files.length > 0) {
            formData.append(fileInput.files[0].name, fileInput.files[0]);
        }

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                alert("Ты все четко добавил;)");
                $("#description").val("");
                $("#faculty").val("");
                $("#adress").val("");
                $("#housePhoto").val("");
                $('#houseBar').css('width', '0%');
                var respond = JSON.parse(xhr.responseText);
                alert(respond.Status);
            }//end of if
        }//end of onreadystatechange

        xhr.upload.onprogress = function (e) {
            $('#houseBar').css('width', (e.loaded / e.total) * 100 + '%');
        }

        xhr.open("POST", "/EditHouses/AddHouse");
        xhr.send(formData);
    }//end of if(confirm...
}//enf of of(verify..
}//end of sendHouse...

function deleteHouse(id, houseName) {

    var confirmString = "Удалить дом " + houseName + "?";
    if (confirm(confirmString)) {
        var formData = new FormData();
        formData.append("houseId", id);
        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    alert("Дом " + houseName + " успешно удален");
                } else {
                    alert("Произошла какая-то ошибка при удалении");
                }
            } //end of if
```

```
        } //end of onreadystatechange

        xhr.open("POST", "/EditHouses/DeleteHouse");
        xhr.send(formData);
    }
}

function deleteHolder(id, holderName) {
    var confirmString = "Удалить холдер " + holderName + "?";
    if (confirm(confirmString)) {
        var formData = new FormData();
        formData.append("holderId", id);
        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function() {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    alert("Холдер " + holderName + " успешно удален!");
                } else {
                    alert("Произошла какая-то ошибка при удалении");
                }
            } //end of if
        } //end of onreadystatechange

        xhr.open("POST", "/EditHouses/DeleteHolder");
        xhr.send(formData);
    }
}

function showEditHouseForm(id) {
    $("#house_" + id).show();
}

function editHouse(id) {
    var houseName = $("#faculty_" + id).val();
    var houseAdress = $("#adress_" + id).val();
    var houseDescription = $("#description_" + id).val();
    var confirmString = "Измениить дом?";

    if (confirm(confirmString)) {
        var formData = new FormData();

        var fileInput = document.getElementById('housePhoto_' + id);
        if (fileInput.files.length > 0) {
            formData.append(fileInput.files[0].name, fileInput.files[0]);
        }
```

```
        formData.append("houseName", houseName);
        formData.append("houseAdress", houseAdress);
        formData.append("houseDescription", houseDescription);
        formData.append("houseId", id);
        var xhr = new XMLHttpRequest();

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    alert("Дом успешно изменен!");
                } else {
                    alert("Произошла какая-то ошибка при изменении");
                }
            } //end of if
        } //end of onreadystatechange

        xhr.open("POST", "/EditHouses/EditHouse");
        xhr.send(formData);
    }
}

function showEditHolderPanel(id) {
    $("#holder_" + id).show();
}

function editHolder(id) {
    var holderName = $("#holderName_" + id).val();
    var holderCity = $("#holderCity_" + id).val();
    var confirmString = "Измениить холдер?";

    if (confirm(confirmString)) {
        var formData = new FormData();

        var fileInput = document.getElementById('holderPhoto_' + id);
        if (fileInput.files.length > 0) {
            formData.append(fileInput.files[0].name, fileInput.files[0]);
        }

        formData.append("holderName", holderName);
        formData.append("holderCity", holderCity);
        formData.append("holderId", id);
        var xhr = new XMLHttpRequest();
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    alert("Холдер успешно изменен!");
```

```
            } else {
                alert("Произошла какая-то ошибка при изменении");
            }
        } //end of if
    } //end of onreadystatechange

    xhr.open("POST", "/EditHouses/EditHolder");
    xhr.send(formData);
  }
}
```

1.6.19. Код функции EditInterests

```
function checkInterest() {
  if ($("#name").val().length == 0) {
    alert("Не введено название холдера");
    return false;
  } else if ($("#fileInput").val().length == 0) {
    alert('Не введена фотография интереса');
    return false;
  }
  else
    return true;
}

function sendInterest() {
  if (checkInterest()) {
    var confirmString = "Название: " + $("#name").val() + "\n" + "Все окей? Добавляю?";
    if (confirm(confirmString)) {
      var formData = new FormData();
      var xhr = new XMLHttpRequest();
      formData.append("name", $("#name").val());
      var fileInput = document.getElementById('fileInput');
      if (fileInput.files.length > 0) {
        formData.append(fileInput.files[0].name, fileInput.files[0]);
      }

      xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
          $("#name").val("");
          $("#fileInput").val("");
          $('#interestBar').css('width', '0%');
          var respond = JSON.parse(xhr.responseText);
          alert(respond.Status);
        }
      }
```

```
            xhr.upload.onprogress = function (e) {
                $('#interestBar').css('width', (e.loaded / e.total) * 100 + '%');
            }

            xhr.open("POST", "/EditInterests/AddInterest");
            xhr.send(formData);
        }
    }
}
```

1.6.20. Код функции unauthMainScripts

```
/**
 * Created by aleksandrlazarenko on 12.04.15.
 */
$(function () {
    var nodes = [
        { id: 1, label: 'Коки', shape: 'circularImage', image: 'http://cocktion.com/Content/SiteImages/
cocky.jpg' },
        { id: 2, label: 'Книга', shape: 'circularImage', image: 'http://cocktion.com/Content/SiteImages/
book.jpg' },
        { id: 3, label: 'Услуга', shape: 'circularImage', image: 'http://cocktion.com/Content/SiteImages/
service.jpg' },
        { id: 4, label: 'Вещь', shape: 'circularImage', image: 'http://cocktion.com/Content/SiteImages/
stuff.png' }];

    // create an array with edges
    var edges = [
        { from: 1, to: 2 },
        { from: 1, to: 3 },
        { from: 1, to: 4 }
    ];

    // create a network
    var container = document.getElementById('mynetwork');
    var data = {
        nodes: nodes,
        edges: edges
    };
    var options = {
        width: '100%',
        height: '100%',
        physics: {
            barnesHut: {
                enabled: true,
                gravitationalConstant: -4000,
                springLength: 250
```

```
        }
      }
   };
   var network = new vis.Network(container, data, options);
   network.on('select', selectHandler);
})

//Скрывает дивы в зав-ти от того, какой попросят показать
function hideForms(formToShowName) {
   switch (formToShowName) {
      case 'cocky':
         $("#welcomeText").hide();
         $("#cockyText").show();
         $("#bookText").hide();
         $("#stuffText").hide();
         $("#serviceText").hide();
         break;
      case 'book':
         $("#welcomeText").hide();
         $("#cockyText").hide();
         $("#bookText").show();
         $("#stuffText").hide();
         $("#serviceText").hide();
         break;
      case 'stuff':
         $("#welcomeText").hide();
         $("#cockyText").hide();
         $("#bookText").hide();
         $("#stuffText").show();
         $("#serviceText").hide();
         break;
      case 'service':
         $("#welcomeText").hide();
         $("#cockyText").hide();
         $("#bookText").hide();
         $("#stuffText").hide();
         $("#serviceText").show();
         break;
   }
}

//Показывает ту или иную область на экране в зависимости от нажатого нодика
function selectHandler(properties) {
   var nodeId = properties.nodes[0];
   switch (nodeId) {
      case '1':
         hideForms('cocky');
         break;
```

```
        case '2':
            hideForms('book');
            break;
        case '3':
            hideForms('service');
            break;
        case '4':
            hideForms('stuff');
            break;
    }
}
```

1.6.21. Код функции addInformation

```
function showEditInfoForm() {
    $("#editInformationContainer").show();
}

function sendInfo(){
    if (checkInputs()) {
        var formData = new FormData();
        var xhr = new XMLHttpRequest();
        $("#infoBarContainer").show();
        formData.append('name', $("#name").val());
        formData.append('surname', $("#surname").val());
        formData.append('school', $("#school").val());

        xhr.upload.onprogress = function (e) {
            $('#infoBar').css('width', (e.loaded / e.total) * 100 + '%');
        }

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    $("#userRates").append("<p>Информация обновлена</p>");
                    $("#infoBarContainer").hide();
                } else {
                    $("#infoBar").css('background-color', 'red');
                    $("#infoBar").css('width', '100%');
                }
            };
        }

        xhr.open('POST', '/Profile/EditProfileInformation');
        xhr.send(formData); //отправка данных
    }
```

```
}

function showInterestsForm() {
    $("#showInterestsFormBtn").hide();
    $("#interestsToChoose").show();
}

var interestsIds = "";
function addId(check) {
    $("#sendInterestsBtn").show();
    interestsIds += check.id;
}

function sendInt() {
    var formData = new FormData();
    var xhr = new XMLHttpRequest();

    formData.append('ids', interestsIds);

    xhr.upload.onprogress = function (e) {
        $('#interestsBar').css('width', (e.loaded / e.total) * 100 + '%');
    }

    xhr.onreadystatechange = function () {
        if (xhr.readyState == 4 && xhr.status == 200) {
            var respond = JSON.parse(xhr.responseText);
            if (respond.Status == "Success") {
                $("#userRates").append("<p>Информация обновлена</p>");
                $("#interestsBarContainer").hide();
            } else {
                $("#interestsBar").css('background-color', 'red');
                $("#interestsBar").css('width', '100%');
            }
        };
    }

    xhr.open('POST', '/Profile/AddInterests');
    xhr.send(formData); //отправка данных
}


function checkInputs() {
    var school = $("#school").val();
    var name = $("name").val();
    var surname = $("surname").val();
    if (school || name || surname) {
        return true;
    } else {
```

```
        $("#errorField").show();
        $("#errorField").empty();
        $("#errorField").append("<p>Нужно добавить хотьк какую-нибудь информацию </p>");
    }
}

function handleInfoChange() {
    $("#errorField").empty();
    $("#errorField").hide();
}
```

1.6.22. Код функции addPhoto

```
//Добавляет пользователю аватарку
function showPhotoForm() {
    $("#addPhotoContainer").show();
}
function addPhoto() {
    if (fileInput.files.length > 0) {
        $("#progressBarContainer").show();
        var formData = new FormData();
        formData.append(fileInput.files[0].name, fileInput.files[0]);
        var xhr = new XMLHttpRequest();

        xhr.upload.onprogress = function (e) {
            $('#bar').css('width', (e.loaded / e.total) * 100 + '%');
        }

        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
                var respond = JSON.parse(xhr.responseText);
                if (respond.Status == "Success") {
                    $("#avatar").attr('src', 'http://cocktion.com/Images/Thumbnails/' + respond.FileName);
                    $('#bar').css('width', 0 + '%');
                    $("#progressBarContainer").hide();
                } else {
                    $('#bar').css('background-color', 'red');
                    $('#bar').css('width', '100%');
                }
            };
        }

        xhr.open('POST', '/Profile/AddPhotoToUser');
        xhr.send(formData); //отправка данных
    } else {
        $("#addPhotoBtn").attr('disabled', 'disabled');
    }
```

```
}

function showAddPhotoButton() {
   fileInput = document.getElementById('usersPhoto');
   if (fileInput.files.length > 0) {
      $("#addPhotoBtn").removeAttr('disabled');
   }
}

var fileInput;
```

1.6.23. Код функции chatLogic

```
//Подсвечивает пользователя, при наведении на него мыши
function highlightUser(divelem) {
   divelem.style.backgroundColor = 'aliceblue';
}

//убирает подсветку при отведении мыши
function dehighlightUser(divelem) {
   divelem.style.backgroundColor = 'white';
}



//генерирует строку для вывода в чате
function generateMessageString(author, message, date) {
   return "<p><b>" + author + ": </b>" + message +" " + date + "</p>";
}

//добавляет пользователя в список людей, с которыми ты счас общаешься
function addUserToList(username, photoPath) {
   var textToAppend = "<div class=\"userToSpeak row\"" +"value=\""+username+"\""+" onclick=
\"showHisMessages(this, '@Model.Id')\"" +
         "onmouseout=\"dehighlightUser(this)\" onmouseover=\"highlightUser(this)\">" +
         "<div class=\"col-md-6\"><p>" + username + "</p></div></div>";
   $("#usersList").append(textToAppend);
   return textToAppend;
}
```

1.6.24. Код функции opinionAdder

```
//Отправляет мнение пользователя на сервер
function addOpinion(userId) {
   var post = $('#writeFeedbackTextarea').val();//получаем текст в textarea
```

```
    //TODO здесь сделать код, который будет отправлять на сервак всю необходимую информацию
    var xhr = new XMLHttpRequest();
    var formData = new FormData();
    formData.append('message', post);
    formData.append('userId', userId);
    xhr.open('POST', '/Profile/AddUsersFeedback');
    xhr.send(formData);
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
            var obj = JSON.parse(xhr.responseText);
            if (obj.Status == 'success') {
                var name = obj.Name;
                var surname = obj.Surname;
                //Обновляем формочку на клиенте
                $('#feedbackPosts').append(
                    "<div class=\"feedbackPost\">" + "<p class=\"postAuthor\"><b>" + name + ' ' + surname +
"</b> сказал: </p>" +
                    "<p>" + post + "<p/>" + "</div>" + "</br>"
                );

                //очищаем поле для ввода текста
                $('#writeFeedbackTextarea').val('');
            } else {
                alert('К сожалению, произошла какая-то ошибка');
            }
        }
    }
}
```

1.6.25. Код функции subscribeProcessor

```
function checkSubscription(userId) {
    var formData = new FormData();
    formData.append('userId', userId);
    var xhr = new XMLHttpRequest();
    var response;
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
            response = JSON.parse(xhr.responseText);
            if (response.Status == "Success") {
                document.getElementById('subscribe').innerHTML = "<p>У вас в информаторах! <span
class=\"glyphicon glyphicon-ok\"></span></p>";
            } else {
                document.getElementById('subscribe').innerHTML = "<p>Добавить в информаторы " +
"<button class=\"btn btn-default\" onclick=\"subscribeOnUser('" + userId.toString() + "')\">" +
                    "<span class=\"glyphicon glyphicon-plus\"></span></button></p>";
```

```
        }
    };
  }

  xhr.open('POST', '/Subscription/CheckUsersSubscription');
  xhr.send(formData); //отправка данных
}



function subscribeOnUser(userId) {
  var formData = new FormData();
  formData.append('userId', userId);
  var xhr = new XMLHttpRequest();
  var response;
  xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      response = JSON.parse(xhr.responseText);
      if (response.Status == "Success") {
        document.getElementById('subscribe').innerHTML = "<p>У вас в информаторах! <span
class=\"glyphicon glyphicon-ok\"></span></p>";
      } else {
        document.getElementById('subscribe').innerHTML = "<p>Попробуйте еще раз,
пожалуйста... <span class=\"glyphicon glyphicon-ok\"></span></p>";

      }
    };
  }

  xhr.open('POST', '/Subscription/SubscribeOnUser');
  xhr.send(formData); //отправка данных
}
```

1.6.26. Код функции subscriptionProcessor

```
function subscribeOnUser(userId) {
  var formData = new FormData();
  formData.append('userId', userId);
  var xhr = new XMLHttpRequest();
  var response;
  xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      response = JSON.parse(xhr.responseText);
      if (response.Status == "Success") {
        document.getElementById(userId).parentElement.innerHTML = "<p>У вас в информаторах!
<span class=\"glyphicon glyphicon-ok\"></span></p>";
      } else {
```

```
            document.getElementById(userId).innerHTML = "<p>Попробуйте еще раз, пожалуйста...
<span class=\"glyphicon glyphicon-ok\"></span></p>";

        }
    };
}

xhr.open('POST', '/Subscription/SubscribeOnUser');
xhr.send(formData); //отправка данных
}
```

1.6.27. Код функции userPrinter

```
//Функция добавляет пользователя
//name - имя пользователя
//surname - фамилия пользователя
//photoPath - путь к фотке
//rating - рэйтинг пользователя
//eggs - яйца, которые пользователь заработал
//auctionsAmount - количество аукционов
function addUser(name, surname, id ,photoPath, rating, eggs, auctionsAmount, isFirst, isSubscribed) {
    if (!isFirst) {
        $("#usersHere").append(
            "<div class=\"userInfo\"><div class=\"row\"><div class=\"col-md-4\"> <img src=\"" + photoPath
+ "\"class=\"img-thumbnail\">" +
            "</div><div class=\"col-md-8\"> <p><b><a href=\"http://cocktion.com/Users/GetUser/"+id
+"\">" + name + ' ' + surname + "</a></b></p><p><b>Рейтинг:</b>" + rating +
            "</p> <p><b>Яйца:</b>" + eggs + "</p>" + "<p><b>Аукционов: </b>" + auctionsAmount +
"</p>" +
            "</div> </div>"+displaySubscriptionStatus(id, isSubscribed)+" </div><br/>" //добавить кнопку
дисплея кнопки
        );
    } else {
        $("#usersHere").append(
            "<div class=\"userInfo\"><div class=\"row\"><div class=\"col-md-4\"> <img src=\"" + photoPath
+ "\"class=\"img-thumbnail\">" +
            "</div><div class=\"col-md-8\"> <p><b><a href=\"http://cocktion.com/Users/GetUser/" + id +
"\">" + name + ' ' + surname + "</a></b></p><p><b>Рейтинг:</b>" + rating +
            "</p> <p><b>Яйца:</b>" + eggs + "</p>" + "<p><b>Аукционов: </b>" + auctionsAmount +
"</p>" +
            "</div> </div>" + displaySubscriptionStatus(id, isSubscribed) + " </div>"
        );
    }
}
//<div class=\"col-md-4\"><p><button class=\"btn btn-default\"></button></p></div>
```

```
function displaySubscriptionStatus(id, isSubscribed) {
   if (isSubscribed == 'True') {
      //если подписан - паказываем ему, что у него в колхозе все в поряде
      return "<p>У вас в информаторах! <span class=\"glyphicon glyphicon-ok\"></span></p>";
   } else {
      //если не подписан - показываем кнопку
      return "<br/><p><button class=\"btn btn-default\" onclick=\"subscribeOnUser('" +
         id+"')\" id=\""+id+"\">Подписаться</button></p>";
   }
}

//проверяет чекбоксы ф чильтрах
function sexCheckboxCheckedHandler(checkId) {
   switch (checkId.id) {
      case 'male':
         $("#female").prop('checked', false);
         break;
      case 'female':
         $("#male").prop('checked', false);
         break;
   }
}
```

1.6.28. Код функции globalSearchFunctions

```
function sendSearchString() {
   $('#progressBarContainer').show();
      if (canSendSearchString()) {
      $("#auctionsContainer").empty();
      auctionRowCounter = 0;
      auctionColCounter = 0;
      var formData = new FormData();
      formData.append('searchString', $("#searchField").val());

      var xhr = new XMLHttpRequest();
      xhr.upload.onprogress = function (e) {
         $('#bar').css('width', (e.loaded / e.total) * 100 + '%');
      }

      xhr.onreadystatechange = function () {
         if (xhr.readyState == 4 && xhr.status == 200) {
            var respond = JSON.parse(xhr.responseText);
            if (!respond.IsEmpty) {
               //если в коллекции что-то есть - выводим все пользователям на страничку
               for (var i = 0; i < respond.Auctions.length; i++) {
                  addCellToTheGrid('auctionsContainer', respond.Auctions[i].Name,
```

```
                    respond.Auctions[i].Description, 'через 3 дня', 'http://cocktion.com/Images/
Thumbnails/' +
                    respond.Auctions[i].Photo, 'http://cocktion.com/Auction/CurrentAuction/' +
                    respond.Auctions[i].Id);
                }
            } else {
                //если в коллекции ничего нет - выводим сообщенеи
                $("#auctionsContainer").append("<p>Такого еще никто не продавал...</p>");
            }
            $('#bar').css('width', 0 + '%');
            $('#progressBarContainer').hide();
        };
    }

    xhr.open('POST', '/Search/SearchEverywhere');
    xhr.send(formData); //отправка данных
  }
}




function canSendSearchString() {
  var searchString = $("#searchField").val();
  if (searchString.length == 0) {
    $("#searchField").css('border-color', 'red');
    return false;
  } else {
    $("#searchField").css('border-color', 'rgb(204, 204, 204)');
    return true;
  }
}
```

1.7. Коды представлений

1.7.1. Код представления Login

```
@using CocktionMVC.Models
@model LoginViewModel
@{
  ViewBag.Title = "Вход в систему";
}

<h2 style="text-align: center">@ViewBag.Title</h2>
<div class="row">
  <div class="col-md-3"></div>
```

```
<div class="col-md-6" style="border: solid 1px #48b8e5">
    <section id="loginForm">
        @using (Html.BeginForm("Login", "Account", new { ReturnUrl = ViewBag.ReturnUrl },
FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
        {
            @Html.AntiForgeryToken()
            <h4 style="text-align: center">Мы рады, что Вы вернулись!</h4>
            <hr />
            @Html.ValidationSummary(true, "", new { @class = "text-danger" })
            <div class="form-group">
                @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
                <div class="col-md-10">
                    @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
                    @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
                <div class="col-md-10">
                    @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
                    @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-offset-2 col-md-10">
                    <div class="checkbox">
                        @Html.CheckBoxFor(m => m.RememberMe)
                        @Html.LabelFor(m => m.RememberMe)
                    </div>
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-5"></div>
                <div class="col-md-2">
                    <input type="submit" value="Войти" style="text-align: center" class="btn btn-default" />
                </div>
                <div class="col-md-5"></div>
            </div>
            <p style="text-align: center">
                @Html.ActionLink("Зарегистрироваться", "Register")
            </p>
            <p style="text-align: center">
                @Html.ActionLink("Забыли свой пароль?", "ForgotPassword")
            </p>
        }
    </section>
</div>
<div class="col-md-3"></div>
```

```
    <!--
    <div class="col-md-4">
        <section id="socialLoginForm">
            @Html.Partial("_ExternalLoginsListPartial", new ExternalLoginListViewModel { ReturnUrl =
ViewBag.ReturnUrl })
        </section>
    </div>
        -->
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.7.2. Код представления Register

```
@model CocktionMVC.Models.RegisterViewModel
@{
    ViewBag.Title = "Зарегистрироваться";
}
@using Recaptcha.Web.Mvc;
<link rel="stylesheet" href="~/Content/UsersRelatedStyles/registerStyle.css" />
<div class="row">
    <div class="col-md-3"></div>
    <div class="col-md-6" id="formContainer">
        <h4>Я - Регистратор! :)</h4>
        @using (Html.BeginForm("Register", "Account", FormMethod.Post, new { role = "form" }))
        {
            @Html.AntiForgeryToken()
            @Html.ValidationSummary("", new { @class = "text-danger" })
            <br />
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control", @placeholder = "Электро-
почта" })
            <br />

            @Html.PasswordFor(m => m.Password, new { @class = "form-control", @placeholder =
"Парольчик" })
            <br />

            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control", @placeholder =
"И еще раз парольчик, чтобы я не забыл..." })
            <br />

            @Html.Recaptcha(language: "ru", theme: Recaptcha.Web.RecaptchaTheme.Clean)
            <br />

            <input type="submit" class="btn btn-default" value="Погнали!" />
```

```
        }
    </div>
    <div class="col-md-3"></div>
</div>
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.7.3. Код представления Create

```
@model List<CocktionMVC.Models.DAL.House>
@{
    ViewBag.Title = "Создать аукцион";
}
<script src="~/Scripts/MyLib/Auction/Create/createAuctionFormProcessor.js"></script>
<script src ="/Scripts/MyLib/Auction/Create/search.js"></script>
<link rel="stylesheet" href="~/Content/AuctionStyles/createAuctionStyle.css"/>

<div id="createAuctionFormContainer" class="row">
    <div class="col-md-2"><!--Для разметки --></div>
    <div class="col-md-8" id="form" class="row">
        <!--Контейнер для формы, создающей аукцион.-->
        <h3>Создайте самый классный аукцион :)</h3>
        <div class="col-md-6" id="inputs">
            <!--Форма для внесения данных о товаре и аукционе-->
            <h4>Что продаем?</h4>
            <input type="text" class="form-control" onchange="changeInputColor()"
                   id="productName" placeholder="Название товара (обязательно поле)">
            <br/>
            <input type="text" class="form-control" id="descriptionName" placeholder="Описание">
            <br/>
            <div id="typeOfAuctionContainer">
                <!--Чекбоксы для типов товара-->
                <h5>Тип предмета</h5>
                <label class="radio-inline">
                    <input type="radio" id="bookRadio" onclick="auctionRadioCheckHandler(this)"> Книга
                </label>
                <label class="radio-inline">
                    <input type="radio" id="productRadio" onclick="auctionRadioCheckHandler(this)"> Вещь
                </label>
                <label class="radio-inline">
                    <input type="radio" id="serviceRadio" onclick="auctionRadioCheckHandler(this)"> Услуга
                </label>
            </div>
            <br/>
            <div id="timeContainer">
                <!--Чекбоксы для временных ограничений -->
```

```
            <!--4 часа / 1 день / 1 неделя -->
            <h5>Длительность аукциона</h5>
            <label class="radio-inline">
              <input type="radio" id="4hoursTime" onclick="auctionRadioCheckHandler(this)"> 4 часа
            </label>
            <label class="radio-inline">
              <input type="radio" id="1dayTime" onclick="auctionRadioCheckHandler(this)"> 1 день
            </label>
            <label class="radio-inline">
              <input type="radio" id="1weekTime" onclick="auctionRadioCheckHandler(this)"> 1 неделя
            </label>
          </div>
          <br/>
          <div id="fileAndButtonContainer">
            <!--Контейнр для файла и кнопок -->
            <input type="file" id="fileInput" accept="image/*">
            <br />
            <div class="progress" hidden="" id="progressBar">
              <!--Прогресс бар-->
              <div class="progress-bar progress-bar-success" id="bar" role="progressbar"
                aria-valuenow="0" aria-valuemin="0" aria-valuemax="100" style="width: 0%">
                <!--Здесь можно впендюрить текстовый индикатор процентов загрузки-->
              </div>
            </div>
            <div hidden="" id="readyInfo"></div>
            <button id="createAuctionBtn" onclick=" sendData() " class="btn btn-default">Создать
аукцион</button>
          </div>
        </div>
        <div class="col-md-6" id="houses">
          <!-- Контейнер для выбора домов -->
          <h4>Где продаем?</h4>
          <div id="universitySearchContainer">
            <input type="text" oninput="searchUniversity()" id="universitySearch" class="form-control"
placeholder="Университет">
          </div>
          <div id="universitySearchResults">
          </div>
          <div id="facultyToChoose">

          </div>
          <label class="radio-inline">
            <input type="radio" id="allFacultyRadio" onclick="facultyRadioCheckHandler(this)"> На
всех факультетах
          </label>
          <label class="radio-inline">
            <input type="radio" id="chooseFacultyRadio" onclick=" facultyRadioCheckHandler(this) " />
Выбрать факультеты
```

```
            </label>
        </div>
    </div>
    <div class="col-md-2"><!--Для разметки --></div>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval");
}
```

1.7.4. Код представления CurrentAuction

```
@{
    ViewBag.Title = "CurrentAuction";
}
@using System.Linq
@model CocktionMVC.Models.DAL.Auction

<script type="text/javascript" src="~/Scripts/Jquery/jquery-2.1.3.js"></script>
<script type="text/javascript" src="~/Scripts/MyLib/Vis/vis.min.js"></script>
<script type="text/javascript" src="~/Scripts/Jquery/jquery.countdown.js"></script>
<script type="text/javascript" src="~/Scripts/MyLib/Auction/CurrentAuction/auctionEndLogic.js"></
script>
<script type="text/javascript" src="~/Scripts/MyLib/Auction/CurrentAuction/bidFunctions.js"></script>
<script type="text/javascript" src="~/Scripts/MyLib/Auction/CurrentAuction/
btnsOnClickHandlers.js"></script>
<script type="text/javascript" src="~/Scripts/MyLib/Auction/CurrentAuction/nodeProcessor.js"></script>
<script src="~/Scripts/Jquery/jquery.signalR-2.2.0.min.js"></script>
<script src="~/signalr/hubs"></script>
<script src="~/Scripts/MyLib/Auction/CurrentAuction/hubConnection.js"></script>

<link rel="stylesheet" href="~/Content/AuctionStyles/currentAuctionStyle.css" />

<body>
    <div class="row">
        <div class="col-md-8">
            <div id="timer"></div>
            <div id="mynetwork"></div>
        </div>
        <div class="col-md-4" id="globalContainer">
            <!--Контеенер для тоталика, инфы, чата, формы с загрузчиком-->
            <br />
            <div class="btn-group">
                <button class="btn btn-default" id="showAuctionInfoBtn"
onclick="showAuctionInfo(this)">Аукцион</button>
                <button class="btn btn-default" onclick="showProducts(this) "
                        id="showNodeInfoBtn">
```

```
            Товар
        </button>
        <button class="btn btn-default" onclick=" showToteBets(this) ">Тотализатор</button>
    </div>

    <div id="firstInfoContainer">
        <div id="auctionInfoContainer">
            <h4>Выставлено на торг: @Model.SellProduct.Name</h4>
            <p>
                <b>Описание: </b>@Model.SellProduct.Description
            </p>
            <p><b>Хозяин: </b>@Model.Owner.UserName</p>
            <div id="leaderHolder">
                @{
                    if (Model.LeadProduct != null)
                    {
                        <p>
                            <b>Лидер аукциона: </b>@Model.LeadProduct.Name
                        </p>
                    }
                    else
                    {
                        <p>
                            <b>Лидер аукциона: </b>не выбран.
                        </p>
                    }
                }
            </div>
            <p>
                <b>Всего ставок: </b>@Model.UsersBids.Count()
            </p>
            <p>
                <b>Дом: </b>красивый и классный
            </p>
            @{

                <p><b>Рейтинг: </b>@Model.Rating</p>

                if (!Model.Owner.EmailConfirmed)
                {
                    <p><b>Статус владельца: </b>не подтвердил себя</p>
                }
}
            <div id="auctionEnderHidden" hidden="">
                <button class="btn btn-danger" onclick="finishAuction()"
id="endAuctionBtn">Закончить аукцион</button>
            </div>
        </div>
```

```
        <div id="productsContainer" hidden="">
          <div class="row">
            <div class="col-md-5" id="productPhotoContainer">
              <img class="img-thumbnail" src="http://cocktion.com/Images/Thumbnails/
@Model.SellProduct.Photo.FileName">
              <p>Фотокарточка</p>
            </div>
            <div class="col-md-5" id="productInfoContainer">
              <p>
                <b>Что: </b>@Model.SellProduct.Name
              </p>
              <p>
                <b>Конкретно: </b>@Model.SellProduct.Description
              </p>
              <p>
                <b>Категория: </b>@Model.SellProduct.Category
              </p>
            </div>
          </div>
          <div hidden="" id="eggInfo"></div>
          <div class="progress" hidden="" id="toteProgress">
            <div class="progress-bar progress-bar-info" role="progressbar" aria-valuenow="0" aria-
valuemin="0"
              aria-valuemax="100" id="toteBar" style="width: 0%">
            </div>
          </div>
          <div class="input-group" id="addToteBetContainer">
            <!--Форма ввода количества яиц для отправки на аукциончик-->
            <input type="text" onchange="eggsTextBoxOnChange()" class="form-control"
id="eggsAmountTextbox">
            <span class="input-group-btn">
              <button class="btn btn-default" type="button" onclick="makeEggBet()"
id="sendEggsBtn"
                      title="На cocktion есть настоящий тотализтатор, который позволяет выйграть
некоторое количество яиц">
                Поставь!
              </button>
            </span>
          </div>
          <div id="chooseLeaderForm" hidden="">
            <div id="chooseLeaderInfo"></div>
            <button type="button" class="btn btn-success" onclick="chooseLider()"
id="chooseLeaderBtn">Выбрать лидера</button>
          </div>
        </div>
        <div id="toteBetsContainer" hidden="">
          <!--Скрытая в начале форма-->
```

```
        <h4 style="text-align: center">Таблица коэффициентов</h4>
        <div id="coefficientsTable">
          <!--Таблица коэффициентов для аукциона-->
          <p>Попробуйте сделать ставку :)</p>
        </div>
        <br />
      </div>
    </div>
    <br />
    <div class="btn-group">
      <button class="btn btn-default" onclick=" showProductBids(this, userStatus, userName) "
          id="showProductBidsBtn">
        Ставка
      </button>
      <button class="btn btn-default" onclick=" showChat(this, userStatus) "
          id="showChatBtn">
        Беседа
      </button>
    </div>
    <div id="secondInfoContainer">
      <!--Контейнер для ставки и чата-->
      <div id="chatContainer">
        <div id="chatMessagesContainer">
          <!--Здесь будут сообщения для чата -->
          <p>
            <b>Коки: </b>Привет, друг! Это наш чат:) Здесь можно общаться и
            беседовать;)
          </p>
        </div>
        <br />
        <div class="input-group" id="chatControlPanel">
          <!--Поле для ввода и кнопочка -->
          <input type="text" class="form-control" id="enterMessageTextBox">
          <span class="input-group-btn">
            <button class="btn btn-default"
                type="button" id="sendMessageBtn" title="Кокает:)">
              Отправь!
            </button>
          </span>
        </div><!-- /input-group -->
      </div>
      <div id="productBidContainer" hidden="">
        <!--Скрыто с глаз в начале -->
        <h4>Поставь свою вещь</h4>
        <div id="inputProductTextBoxes">
          <input type="text" id="bidName"
              class="form-control" placeholder="Название товара">
          <br />
```

```
                <input type="text" id="bidDescription" placeholder="Описание товара"
                    class="form-control">
                <br />
                <div id="typeOfAuctionContainer">
                  <!--Чекбоксы для типов товара-->
                  <h5>Тип предмета</h5>
                  <label class="radio-inline">
                    <input type="radio" id="bookRadio" onclick="radioProcessor(this)"> Книга
                  </label>
                  <label class="radio-inline">
                    <input type="radio" id="productRadio" onclick="radioProcessor(this)"> Вещь
                  </label>
                  <label class="radio-inline">
                    <input type="radio" id="serviceRadio" onclick="radioProcessor(this)"> Услуга
                  </label>
                </div>
                <br/>
                <input type="file" name="docs" id="fileInput" />
              </div>
              <br/>
              <div class="progress" hidden="" id="progressBar">
                <div class="progress-bar progress-bar-success" id="bar" role="progressbar"
                    aria-valuenow="0" aria-valuemin="0" aria-valuemax="100" style="width:0%">
                  <!--Здесь можно впендюрить текстовый индикатор процентов загрузки-->
                </div>
              </div>
              <button class="btn btn-default" onclick=" addBid(auctionId) "
id="sendProductBidBtn">Отправить товар</button>
            </div>
            <div id="productBidErrorInfo" hidden="">
              <!--Это для добавления сообщений (чтобы не показывать форму)-->
            </div>
          </div>
          <!-- Добавлялка дополнительной ставки
          <div id="extraBidContainer">
            <p>Введите наименование довеска</p>
            <input type="text" id="extraBidName"/>
            <p>Введите Описание довеска</p>
            <input type="text" id="extraBidDescription"/>
            <p>Введите категорию довеска</p>
            <input type="text" id="extraBidCategory"/>
            <p>Можете прикрепить файлик</p>
            <input type="file" id="extraFileInput">
            <button type="button" id="submitExtraBid" class="btn btn-danger"
onclick="addExtraBid(@Model.Id)">Послать довесок</button>
          </div>
          <button type="button" class="btn btn-success" id="addExtraBid"
onclick="showExtraBidAdder()">Добавить довесок</button>
```

```
            -->
        </div>
    </div>

    <script>
    //Здесь содержится таймер и проверка
    //на то, что человек кнопку увидит
    var chat;
    var isChoosed = false;
    var auctionId = @Model.Id;
    var ownerId;
    var winnerId;
    var i = @Model.SellProduct.Id;
    var userStatus = '@User.Identity.IsAuthenticated';
    var userName = '@User.Identity.Name';
    var ownerName = '@Model.Owner.UserName';
    var $n = $('#timer');
    if ('@Model.IsActive' == "True")
    {
        $n.countdown('@Model.EndTime.Year/@Model.EndTime.Month/@Model.EndTime.Day
@Model.EndTime.Hour:@Model.EndTime.Minute:@Model.EndTime.Second')
            .on('update.countdown',function(event) {
                var totalHours = event.offset.totalDays * 24 + event.offset.hours;
                var $this = $(this).html(event.strftime('До конца: '
                    + totalHours + ' : '
                    + '<span>%M</span> : '
                    + '<span>%S</span>'));
            }).on('finish.countdown', function(event) {
                sayThatFinished();
            });
    }
    else
    {
        getAuctionResults();
    }

    //Занимается инициализацией скриптов
    $(document).ready(function () {
        $('#enterMessageTextBox').keypress(function (e) {
            if (e.keyCode == 13)
                $('#sendMessageBtn').click();
        });
        $("#showProductBidsBtn").click();
        $("#showAuctionInfoBtn").click();

        if ("@Model.WinnerChosen" == "True") {
            isChoosed = true;
        }
```

```
        if (userName == ownerName) {
            $("#auctionEnderHidden").show();
            $("#addToteBetContainer").hide();
        }
        initialiseNetwork('@Model.SellProduct.Name', '@Model.SellProduct.Photo.FileName');
        initialiseHub('@Model.Id', '@User.Identity.IsAuthenticated', '@User.Identity.Name');
    });

    </script>

    @{

        if (Model.BidProducts != null)
        {
            foreach (var i in Model.BidProducts)
            {
                <script>
                    addNode('@i.Photo.FileName', '@i.Name', '@i.Id');
                </script>
            }
        }
    }
</body>
```

1.7.5. Код представления Auction/Index

```
@{
    ViewBag.Title = "Аукционы";
}
@model List<CocktionMVC.Models.DAL.Auction>
<script src="~/Scripts/Jquery/jquery-2.1.3.min.js"></script>
<script src="~/Scripts/Jquery/jquery.signalR-2.2.0.min.js"></script>
<script src="~/signalr/hubs"></script>
<script src="~/Scripts/MyLib/Auction/Index/hubConnection.js"></script>
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>
<link rel="stylesheet" href="~/Content/AuctionStyles/indexStyle.css"/>
<div id="auctionsControlPanel">
    <div id="controls" class="row">
        <div id="filter" class="col-md-12" style="text-align: center">
            <h4>Все активные аукционы в данный момент времени</h4>
        </div>
    </div>
</div>
<div id="auctionPanel"></div>

@{
```

```
   if (Model.Count != 0)
   {
      foreach (var auction in Model)
      {
         <script>
            addCellToTheGrid('auctionPanel', '@auction.SellProduct.Name.Trim().Replace("\n", " ")',
'@auction.SellProduct.Description.Trim().Replace("\n", " ")',
               '@auction.EndTime', 'http://cocktion.com/Images/Thumbnails/
@auction.SellProduct.Photo.FileName',
               'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
         </script>
      }
   }
   else
   {
      <script>
         sayThatEmpty('auctionPanel');
      </script>
   }
}
```

1.7.6. Код представления MyAuctions

```
@{
   ViewBag.Title = "MyAuctions";
}
@using CocktionMVC.Models.DAL
@model List<Auction>
<h2>MyAuctions</h2>
@using (Html.BeginForm())
{
   foreach (var i in Model)
   {
      <p>@i.SellProduct.Name</p>
      <p>@i.SellProduct.Description</p>
      <img src="@i.SellProduct.Photo.FilePath" width="60" height="40" />
      <p></p>
   }
}
```

1.7.7. Код представления GetCurrentAuctionHouse

```
@{
   ViewBag.Title = "Конкретный дом";
}
@using CocktionMVC.Models.DAL
```

```
@using Microsoft.AspNet.Identity
@model House
<h3>Домик в @Model.Holder.Name</h3>

<script src="~/Scripts/MyLib/AuctionHouse/GetCurrentAuctionHouse/commenter.js"></script>
<script src="~/Scripts/MyLib/AuctionHouse/GetCurrentAuctionHouse/scroller.js"></script>
<script src="~/Scripts/MyLib/AuctionHouse/GetCurrentAuctionHouse/subscriber.js"></script>

<link rel="stylesheet" href="~/Content/HouseRelatedStyles/getCurrentAuctionHouseStyle.css"/>

<div class="row" id ="top">
   <!--Контейнер для первой части страницы-->
   <div class="col-md-3" id="houseDescriptionContainer">
      <!--Фотка и описание доам-->
      <!--Фотография-->
      <img src="http://cocktion.com/Images/Thumbnails/@Model.Portrait.FileName" class="img-
responsive" />
      <br />
      <div>
         <p>
            <b>Описание:</b> @Model.Description
         </p>
         <p>
            <b>Адрес:</b> @Model.Adress
         </p>
         <script>houseId = @Model.Id</script>
         <p><b>Факультет:</b> @Model.Faculty</p>
      </div>
      <br />
   </div>
   <div class="col-md-7">
      <div class="row" id="imageLinksContainer">
         <!--Тут находятся ссылочки на странички с разными аукционами-->
         <div class="col-md-6">
            <!--Картинка с ссылкой на прошедшие аукционы-->
            <a href="http://cocktion.com/AuctionShower/ShowOldAuctions/@Model.Id">
               <img src="~/Content/SiteImages/deadSun.png" class="img-thumbnail" />
            </a>
            <p style="text-align: center">Прошедшие аукционы</p>
         </div>
         <div class="col-md-6">
            <!--Картинка с ссылкой на активные аукционы-->
            <a href="http://cocktion.com/AuctionShower/ShowActiveAuctions/@Model.Id">
               <img src="~/Content/SiteImages/happySun.png" class="img-thumbnail" />
            </a>
            <p style="text-align: center">Активные аукционы</p>
         </div>
      </div>
```

```
    </div>
    <div class="col-md-2" id="secondInfoColumn">
        <!--Лайки, рейтинг и кнопка-->
        <p><b>Рейтинг:</b> @Model.Rating</p>
        <br />
        <p><b>Жители:</b> @Model.Inhabitants.Count</p>
        <br />
        <p><b>Количество аукционов:</b> @Model.Auctions.Count</p>
        <div id="subscribe">
            @{
                if (User.Identity.IsAuthenticated)
                {
                    <script>
                        checkSubscription(@Model.Id);
                    </script>
                }
                else
                {
                    <script>
                        document.getElementById('subscribe').innerHTML = "<p>Чтобы добавлять дома в свой
колхоз, нужно быть авторизованным!</p>";
                    </script>
                }
            }
        </div>

    </div>
</div>
<!--
    <div class="row">
    <div class="col-md-2"></div>
    <!--Контейнер для описания
    <div id="houseDescriptionContainer" class="col-md-8">
        <p>
            <b>Описание:</b> @Model.Description
        </p>
        <p><b>Адрес:</b> @Model.Adress
        </p>
        <script>houseId = @Model.Id</script>
        <p><b>Факультет:</b> @Model.Faculty</p>
    </div>
    <div class="col-md-2"></div>
</div>-->

<div class="row">
    <!--Здесь начинается форум-->
    <div class="col-md-3">
        <h3>Подписчики</h3>
```

```
      <div id="subscribersList">
        @foreach (var sub in Model.Inhabitants)
        {
          <p>@sub.UserName</p>
        }
      </div>
    </div>
    <div id="forumContainer" class="col-md-6">
      <h3 style="text-align: center">Дискуссия</h3>
      <div id="messageContainer">
        <!--Тут отображаются сообщения с форума-->
        @if (Model.Posts.Count > 0)
        {
          foreach (var post in Model.Posts)
          {
            <div class="postContainer">
              <div class="author">
                <p>
                  <b>@post.AuthorName</b> сказал сегодня:
                </p>
              </div>
              <div class="messageInstance">
                <p>@post.Message</p>
              </div>
            </div>
          }
        }
        else
        {
          <p>Нет сообщений никаких!</p>
        }
      </div>
      <div id="messageEnterContainer">
        <div id="messageEnter">
          <p>
            <b>Написать коммент</b>
          </p>
          <textarea rows="3" id="message"> </textarea>
          <button id="addCommentButton" onclick="comment(@Model.Id)" title=" appppp!!!!"
class="btn btn-default">Царапнуть</button>
        </div>
      </div>
    </div>
    <div class="col-md-3">
    </div>
</div>
```

1.7.8. Код представления GetUniversityHouses

```
@{
   ViewBag.Title = "Дома в " + Model[0].Holder.Name;
}
@model List<CocktionMVC.Models.DAL.House>
<script src="~/Scripts/MyLib/AuctionHouse/GetUniversityHouses/printHousesInCells.js"></script>
<link rel="stylesheet" href="~/Content/HouseRelatedStyles/universityHousesStyle.css"/>
<div id="universityHousesControlPanel">
   <div class="row">
      <div class="col-md-4"><p>Какая-то опция</p></div>
      <div class="col-md-4"><h4>@Model[0].Holder.Name</h4></div>
      <div class="col-md-4"><p>Еще какая-то опция</p></div>
   </div>
</div>
<div id="universityHousesContainer"></div>

@{
   foreach (var house in Model)
   {
      <script>
         addHouseCell('universityHousesContainer', '@house.Faculty', '@house.Adress',
            'http://cocktion.com/AuctionHouse/GetCurrentAuctionHouse/@house.Id',
            'http://cocktion.com/Images/Thumbnails/@house.Portrait.FileName');
      </script>

   }
}
```

1.7.9. Код представления AuctionHouse/Index

```
@{
   ViewBag.Title = "Дома, в которых живут самые крутые аукционы!";
}
@model List<CocktionMVC.Models.HouseHolder>
<script src="~/Scripts/MyLib/AuctionHouse/printHolders.js"></script>
<style>
  #holdersContainer {
     text-align: center;
  }
</style>
<h3>Выбери свой любимый университет</h3>
<div id="holdersContainer">
</div>

@{
```

```
    foreach (var holder in Model)
    {
        <script>
            addHolderCell('holdersContainer', '@holder.Name',
                'http://cocktion.com/AuctionHouse/GetUniversityHouses/@holder.Id', 'http://cocktion.com/
Images/Thumbnails/@holder.PhotoCard.FileName');
        </script>
    }
}
```

1.7.10. Код представления ShowActiveAuctions

```
@{
    ViewBag.Title = "Активные аукционы";
}
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>
<link rel="stylesheet" href="~/Content/AuctionStyles/houseAuctionsStyle.css" />
@model List<CocktionMVC.Models.DAL.Auction>
<div id="auctionsControlPanel">
    <div id="controls" class="row">
        <div id="headerContainer" class="col-md-3">
        </div>
        <div id="filter" class="col-md-6">
            <h4>Активные аукционы на данной площадке</h4>
        </div>
        <div id="secondFilter" class="col-md-3">
        </div>
    </div>
</div>
<div id="auctionPanel">

</div>

@{
    if (Model.Count != 0)
    {
        foreach (var auction in Model)
        {
            <script>
                addCellToTheGrid('auctionPanel', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                    '@auction.EndTime', 'http://cocktion.com/Images/Thumbnails/
@auction.SellProduct.Photo.FileName',
                    'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
            </script>
        }
    }
```

```
    else
    {
        <script>
            sayThatEmpty('auctionPanel');
        </script>
    }
}
```

1.7.11. Код представления ShowOldAuctions

```
@{
    ViewBag.Title = "История аукционов";
}
@model List<CocktionMVC.Models.DAL.Auction>
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>
<link rel="stylesheet" href="~/Content/AuctionStyles/houseAuctionsStyle.css"/>
<div id="auctionsControlPanel">
    <div id="controls" class="row">
        <div id="headerContainer" class="col-md-4">
        </div>
        <div id="filter" class="col-md-4">
            <h4>История прошедших аукционов</h4>
        </div>
        <div id="secondFilter" class="col-md-4">
        </div>
    </div>
</div>
<div id="auctionPanel">

</div>

@{
    if (Model.Count != 0)
    {
        foreach (var auction in Model)
        {
            <script>
                addCellToTheGrid('auctionPanel', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                    'уже кончился!', 'http://cocktion.com/Images/Thumbnails/
@auction.SellProduct.Photo.FileName',
                    'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
            </script>
        }
    }
```

```
    else
    {
        <script>
            sayThatEmpty('auctionPanel');
        </script>
    }
}
```

1.7.12. Код представления EditHouses/Index

```
@{
    ViewBag.Title = "Index";
}
@using CocktionMVC.Models.DAL
@model Tuple<List<House>, List<CocktionMVC.Models.HouseHolder>>
<script src="~/Scripts/Jquery/jquery-2.1.3.min.js"></script>
<script src="~/Scripts/MyLib/EditHouses/formVerifier.js"></script>
<script src="~/Scripts/MyLib/EditHouses/houseSender.js"></script>
<script src="~/Scripts/MyLib/EditHouses/holderSender.js"></script>
<div class="row">
    <div class="col-lg-3">
        <div id="addHouseForm">
            <h3>Добавить новый дом</h3>
            Факультет <br />
            <input type="text" id="faculty" /> <br />
            Адрес <br />
            <input type="text" id="adress"/> <br/>
            Описание <br/>
            <input type="text" id="description"/><br/>
            <input type="file" id="housePhoto"/>
            <select id="selector">
                <option value="-1">----</option>
                @{
                    foreach (var holder in Model.Item2)
                    {
                        <option value="@holder.Id">@holder.Name</option>
                    }
                }
            </select>
            <div class="progress">
                <div class="progress-bar progress-bar-info" role="progressbar" id="houseBar" aria-
valuenow="20" aria-valuemin="0" aria-valuemax="100" style="width:0%">
                </div>
            </div>
            <button id="addHouseBtn" class="btn btn-default" onclick="sendHouseToServer()" title="Жги,
детка!">Добавить дом</button>
        </div>
```

```
    </div>
    <div class="col-lg-3">
        <h3>Существующие дома</h3>
        <div id="houseList">
            @{
                if (Model.Item1.Count > 0)
                {
                    foreach (var item in Model.Item1)
                    {
                        if (item.Holder != null)
                        {
                            <div class="house" style="border: solid 1px black">
                                <img src="~/Images/Thumbnails/@item.Portrait.FileName" class="img-thumbnail" /
>
                                <p>Холдер: @item.Holder.Name</p>
                                <p>Факультет: @item.Faculty</p>
                                <p>Адрес: @item.Adress</p>
                                <p>Описание: @item.Description</p>
                                <a  onclick="deleteHouse('@item.Id', '@item.Faculty') ">Удалить</a><br/>
                                <a onclick="showEditHouseForm('@item.Id')">Редактировать</a>
                                <div id="house_@item.Id" hidden="">
                                    <p>Если нужно обновить только одно поле - вписывай одно нужное,
остальные
                                        можно оставить пустыми</p>
                                    Факультет <br />
                                    <input type="text" id="faculty_@item.Id" /> <br />
                                    Адрес <br />
                                    <input type="text" id="adress_@item.Id" /> <br />
                                    Описание <br />
                                    <input type="text" id="description_@item.Id"/><br/>
                                    <input type="file" id="housePhoto_@item.Id" />
                                    <button onclick="editHouse('@item.Id')">Изменить данные</button>
                                </div>
                            </div>
                            <br/>
                        }
                        @*else
                        {
                            <script>deleteHouse('@item.Id', '@item.Faculty')</script>
                        }*@
                    }
                }
                else
                {
                    <p>Нет домов</p>
                }
            }
        </div>
```

```
</div>
<div class="col-lg-3">
    <h3>Существующие хаусхолдеры</h3>
    @{
        if (Model.Item2.Count > 0)
        {
            foreach (var item in Model.Item2)
            {
                <div class="holder" style="border: solid 1px black">
                    <img src="~/Images/Thumbnails/@item.PhotoCard.FileName" class="img-thumbnail"/>
                    <p>Холдер: @item.Name, Город: @item.City</p>
                    <a onclick=" deleteHolder('@item.Id', '@item.Name') ">Удалить</a>
                    <a onclick="showEditHolderPanel('@item.Id')">Редактировать</a>
                    <div id="holder_@item.Id" hidden="">
                        Название <br />
                        <input type="text" id="holderName_@item.Id"/><br/>
                        Город <br />
                        <input type="text" id="holderCity_@item.Id"/><br/>
                        Фотка <br />
                        <input type="file" id="holderPhoto_@item.Id"/>
                        <button onclick="editHolder('@item.Id') ">Изменить данные</button>
                    </div>
                </div>
                <br/>
            }
        }
        else
        {
            <p>Нет холдеров</p>
        }
    }

</div>
<div class="col-lg-3">
    <div id="addHouseHolder">
        <h3>Добавить новый хаусхолдер</h3>
        Название холдера <br/>
        <input type="text" id="holderName"/><br/>
        Город<br />
        <input type="text" id="holderCity"/><br />
        Фотка<br />
        <input type="file" id="holderPhoto"/><br/>
        <div class="progress">
            <div class="progress-bar" id="holderBar" role="progressbar" aria-valuenow="60" aria-
valuemin="0" aria-valuemax="100" style="width: 0%;">
            </div>
        </div>
```

```
        <button id="addHolderBtn" class="btn btn-default" onclick="sendHolderToServer()">Добавить холдер</button>
      </div>
   </div>
</div>
```

1.7.13. Код представления EditInterests/Index

```
@{
   ViewBag.Title = "Index";
}
@model List<CocktionMVC.Models.DAL.Interest>
<script src="~/Scripts/MyLib/EditInterests/EditInterests.js"></script>
<h3>Редактирование интересов</h3>
<div class="row">
   <div class="col-md-3">
      <h4>Добавить новый</h4>
      <div class="input-group">
         <input type="text" id="name" class="form-control" placeholder="Название интереса">
         <br />
         <input type="file" id="fileInput"/>
         <div class="progress">
            <div class="progress-bar progress-bar-info" role="progressbar" id="interestBar" aria-valuenow="20" aria-valuemin="0" aria-valuemax="100" style="width: 0%">
            </div>
         </div>
         <br/>
         <button onclick="sendInterest()" class="btn btn-default">Добавить интерес</button>
      </div>
   </div>
   <div class="col-md-3">
      @{
         foreach (var interest in Model)
         {
            <p>@interest.Name</p>
            <img src="http://cocktion.com/Images/Thumbnails/@interest.Photocard.FileName">
            <br/>
         }
      }
   </div>
   <div class="col-md-3"></div>
   <div class="col-md-3"></div>
</div>
```

1.7.14. Код представления Home/Index

```
@{
   Layout = null;
 }
<head>
   <title>Главная страница</title>
   <link rel="stylesheet" href="~/Content/BootstrapCss/bootstrap.min.css" />

   @Scripts.Render("~/bundles/modernizr")
   @Scripts.Render("~/bundles/jquery")
   @Scripts.Render("~/bundles/bootstrap")
   <link rel="stylesheet" href="~/Content/MainPageStyles/mainStyle.css" />
</head>
<body>
   <div class="navbar navbar-default navbar-fixed-top" style="height:60px">
      <div class="container">
         <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse">
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("Главная", "Index", "Home", new { area = "" }, new { @class = "navbar-
brand" })
         </div>
         <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
               <li>@Html.ActionLink("Аукционы", "Index", "Auction")</li>
               <li>@Html.ActionLink("Продать", "Create", "Auction")</li>
               <li>@Html.ActionLink("Дома", "Index", "AuctionHouse")</li>
               <li>@Html.ActionLink("Люди", "Index", "Users")</li>
               <li>@Html.ActionLink("Поиск", "Index", "Search")</li>
            </ul>
            @Html.Partial("_LoginPartial")
         </div>
      </div>
   </div>

   <header>
      <div class="container">
         <div class="row">
            <div class="col-lg-12">
               <img class="img-responsive" src="~/Content/SiteImages/collaboration.png" alt="">
               <div class="intro-text">
                  <span class="name">Общайтесь, меняйтесь, радуйтесь!</span>
                  <hr>
```

```
                <span class="skills">Web Application for Student's Collaboration</span>
              </div>
            </div>
          </div>
        </div>
    </header>

    <div class="container" id="generalInfo">
      <div class="row">
        <h1>Что здесь можно делать?</h1>
        <hr style="columns: ;">
        <div class="col-lg-4">
          <img src="~/Content/SiteImages/talk.png" height="370px">
          <h2>Общаться со студентами из своего университета</h2>
        </div>
        <div class="col-lg-4">
          <img src="~/Content/SiteImages/exchange.png" height="370px">
          <h2>Меняться с теми, кто оценил вашу вещь по достоинству</h2>
        </div>
        <div class="col-lg-4">
          <img src="~/Content/SiteImages/neighbours.png" height="370px">
          <h2>Смотреть, что есть на обмен в общежитии и на факультете</h2>
        </div>
      </div>
    </div>

    <footer class="text-center">
      <div class="footer-below">
        <div class="container">
          <div class="row">
            <div class="col-lg-12">
              Web Application for Student's Collaboration, 2015. Created by Alexander Lazarenko.
            </div>
          </div>
        </div>
      </div>
    </footer>
</body>
```

1.7.15. Код представления VerifyEmail

```
@{
    ViewBag.Title = "Подтверждение почтового адреса";
}

<h3>Пожалуйста, подтвердите свою электро-почту</h3>
```

```
<p>
   Коки заботится о Вас, поэтому необходимо подтвердить тот факт, что данная почта - ваша, ведь вы тоже
   хотите иметь дело только с настоящими людьми?)
</p>
```

1.7.16. Код представления Profile/Index

```
@{
   ViewBag.Title = "Index";
   CocktionContext db = new CocktionContext();
}
@using CocktionMVC.Models.DAL;
@using CocktionMVC.Models.ViewModels
@using Microsoft.AspNet.Identity
@model AspNetUser

<link rel="stylesheet" href="~/Content/UsersRelatedStyles/ProfileStyle.css"/>
<script src="~/Scripts/MyLib/AuctionHouse/GetUniversityHouses/printHousesInCells.js"></script>
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>
<script src="~/Scripts/MyLib/Profile/addPhoto.js"></script>
<script src="~/Scripts/MyLib/Profile/addInformation.js"></script>
<script src="~/Scripts/Jquery/jquery.signalR-2.2.0.min.js"></script>
<script src="~/signalr/hubs"></script>
<script src="~/Scripts/MyLib/Profile/chatLogic.js"></script>

<script>
   var chat = $.connection.messageHub;
   var responderName = "";
   var thisUserName = '@Model.UserName';
   var responderId = null;
   var thisUserId = '@Model.Id';
   //Функция используется для передачи сообщений, которые посылаются людьми
   chat.client.addNewMessageToPage = function (author, message,date, receiverId) {

      //если в табличке есть пользователь с именм author, но
      //responderName != authorName, то тогда мы его хайлайтим
      if ($("div[value='" + author + "']").length > 0) {
         if (responderName != author) { //не открыто окно с сообщениями от этого пользователя
            $n = $("div[value='" + author + "']")[0];
            $n.style.backgroundColor = 'darkturquoise';
         } else { //открыто окно с сообщениями от этого автора
            responderId = receiverId;
            $("#messages").append(generateMessageString(author, message, date));
            $("#messages").animate({ scrollTop: $("#messages")[0].scrollHeight }, 100);
         }
      } else if(thisUserName != author) {
```

```
            addUserToList(author, 'dfd');
            $n = $("div[value='" + author + "']")[0];
            $n.style.backgroundColor = 'darkturquoise';
        }
    };

    //функция вставляет сообщения, которые приходят с сервера
    chat.client.appendMessageToPage = function (author, message,date, receiverId) {
        responderId = receiverId;
        $("#messages").append(generateMessageString(author, message, date));
        $("#messages").animate({ scrollTop: $("#messages")[0].scrollHeight }, 100);
    };

    //добавляет авторов в колонку справа
    chat.client.addAuthors = function(author) {
        addUserToList(author, 'Тип фоточка');
    };

    $.connection.hub.start().done(function () {

        //добавляем комнату
        chat.server.addNewRoom('@Model.UserName');

        //получаем список получателей
        chat.server.getListOfReceivers('@Model.Id');

        //обрабатываем клик мышкой
        $('#sendMessageBtn').click(function () {
            if (responderName.length > 0) {
                var textToAppend = '<p><b>' + '@Model.UserName' + ': </b>' + $("#messageContainer").val()
+ "@DateTime.Now.ToShortDateString()"+'</p>';
                $("#messages").append(textToAppend);
                chat.server.send($("#messageContainer").val(), '@Model.UserName', responderName,
'@User.Identity.GetUserId()',
                    responderId);
                $("#messageContainer").val("");
                $("#messages").animate({ scrollTop: $("#messages")[0].scrollHeight }, 100);
            }
        });
    });
</script>
<script>//отпавка по кнопке энтер
    $(document).ready(function () {
        $('#messageContainer').keypress(function (e) {
            if (e.keyCode == 13)
                $('#sendMessageBtn').click();
        });
    });
```

```
    //подгружает с сервера сообщения пользователя
    //надо бы эту функцию умнее сделать!!!
    function showHisMessages(divelem) {
        var name = divelem.getAttribute('value');
        responderName = name;
        $("#messages").empty();
        $("#userHeader").empty();
        $("#userHeader").append(name);
        chat.server.getMessages(name, thisUserId);
        $("#messages").animate({ scrollTop: $("#messages")[0].scrollHeight }, 100);
    }
    </script>
<div id="chatFormContainer">
    <div class="row">
        <div class="col-md-5" id="usersList">
            <!--Тут будет список людей, с которыми ты можешь общаться-->
            <!--Сделать запрос на сервер, чтобы получить список всех пользователей, с которыми он в
диалоге-->
        </div>
        <div class="col-md-7" id="messagesWindow">
            <!--Здесь будет непосредственно контейнер для сообщений-->
            <h4 id="userHeader">Диалог с </h4>
            <div id="messages">
            </div>
            <div id="inputMessageForm">
                <!--Группа для ввода сообщения-->
                <div class="input-group">
                    <input type="text" class="form-control" id="messageContainer">
                    <span class="input-group-btn">
                        <button class="btn btn-default" id="sendMessageBtn" type="button">Отправить</
button>
                    </span>
                </div><!-- /input-group -->
            </div>
        </div>
    </div>
</div>

<div class="row"> <!--Верхняя часть страницы, имя, рейтинг, фотка и интересы-->
    <br/>
    <!--Фотка и информация о пользователе-->
    <div class="col-md-3" style="align-items:flex-start">
        <!--Контейнер для аватарки-->
        @{
            if (Model.Selfie == null)
            {
                <img id="avatar" src="~/Content/SiteImages/anonPhoto.jpg" class="img-thumbnail" />
            }
```

```
        else
        {
            <img id="avatar" src="~/Images/Thumbnails/@Model.Selfie.FileName" class="img-
thumbnail" />
        }
    }
    <p><button class="btn btn-default" onclick="showPhotoForm() ">Добавить аватарку</button>
    </p>
    <p><button class="btn btn-default" onclick="showEditInfoForm()">Добавить информацию о
себе</button>
    </p>
    <div id="editInformationContainer" hidden="">
        <input type="text" oninput="handleInfoChange()" id="name" class="form-control"
placeholder="Имя">
        <br />
        <input type="text" oninput="handleInfoChange()" id="surname" class="form-control"
placeholder="Фамилия">
        <br />
        <input type="text" oninput="handleInfoChange()" id="school" class="form-control"
placeholder="Университет/школа">
        <div id="errorField" hidden=""></div>
        <button id="editInfoBtn" onclick=" sendInfo() ">Добавить инфу</button>
        <div class="progress" id="infoBarContainer" hidden="">
            <div class="progress-bar" id="infoBar" role="progressbar" aria-valuenow="60" aria-
valuemin="0" aria-valuemax="100" style="width: 0%;">
            </div>
        </div>
    </div>
    <div id="addPhotoContainer" hidden="">
        <input type="file" onchange="showAddPhotoButton()" id="usersPhoto"/>
        <button class="btn btn-default" id="addPhotoBtn" onclick="addPhoto() " disabled="">Сменить
фотку</button>
        <div class="progress" id="progressBarContainer" hidden="">
            <div class="progress-bar" id="bar" role="progressbar" aria-valuenow="60" aria-valuemin="0"
aria-valuemax="100" style="width: 0%;">
            </div>
        </div>
    </div>
  </div>
  <div class="col-md-2" id="userInformation">
    <!--Контейнер для информации-->
    <div id="userName">
        <!--Контейнер для имени пользователя-->
        @{
            if (!string.IsNullOrEmpty(Model.UserRealSurname) && !
string.IsNullOrEmpty(Model.UserRealName))
            {
                <h4>@Model.UserRealSurname @Model.UserRealName</h4>
```

```
            }
            else
            {
               <h4>@Model.UserName</h4>
            }
         }
      </div>
      <br/>
      <div id="userRates">
         <!--Контейнер для информации о пользователе-->
         <p><b>Рейтинг:</b> @Model.Rating</p>
         <p><b>Яиц: </b> @Model.Eggs</p>
         <p><b>Аукционов сделал:</b> @Model.HisAuctions.Count</p>
         @{
            if (!string.IsNullOrEmpty(Model.SocietyName))
            {
               <p><b>Общество: </b>@Model.SocietyName</p>
            }
         }
         <p><b>Ставок сделал: </b>@Model.HisProducts.Count</p>
      </div>
   </div>
   <div class="col-md-7" id="userLifePos">
      <h4>Жизненная позиция пользователя</h4>
      <div class="row" id="interests">
         @{
            if (Model.Interests.Count != 0)
            {
               <h3>Ваши интересы</h3>
               foreach (var i in Model.Interests.ToList())
               {
                  <p>@i.Name</p>
               }
            }
}
         <p><button id="showInterestsFormBtn" onclick="showInterestsForm()">Добавить интересы</
button></p>
         <div id="interestsToChoose" hidden="">
            @if (Model.Interests.Count != db.Interests.Count())
            {
               foreach (var i in db.Interests.ToList())
               {
                  if (!Model.Interests.Contains(i))
                  {
                     <p><input type="checkbox" id="int_@i.Id" onclick=" addId(this) "/>@i.Name</p>
                  }
               }
            }
```

```
            else
            {
                <p>Больше нечего добавлять!</p>
            }
            <div class="progress" id="interestsBarContainer" hidden="">
                <div class="progress-bar" id="interestsBar" role="progressbar" aria-valuenow="60" aria-
valuemin="0" aria-valuemax="100" style="width: 0%;">
                </div>
            </div>
            <button id="sendInterestsBtn" onclick=" sendInt() " hidden="">Добавить еще интересов</
button>
        </div>
    </div>
</div>
</div>
<br />
<br />

<div class="row" id="subscribedHouses"> <!--Информация о домах, на которые подписан
пользователь-->
    <!--Дома, на которые подписан пользователь-->
    <h3 id="housesHeader">Житель следующих домов</h3>


    @{
        if (Model.SubHouses.Count != 0)
        {
            foreach (var house in Model.SubHouses)
            {
                <script>
                    addHouseCell('subscribedHouses', '@house.Faculty', '@house.Adress',
                        'http://cocktion.com/AuctionHouse/GetCurrentAuctionHouse/@house.Id',
                        'http://cocktion.com/Images/Thumbnails/@house.Portrait.FileName');
                </script>
            }
        }
        else
        {
            <script>$("#subscribedHouses").append("<p>К сожеланию, вы еще не подписались на дома ;
(</p>")</script>
        }
    }
</div>
<br />
<br/>

<div class="row" id="usersAuctions"> <!--Информация об аукционах, которые намутил
пользователь-->
```

```
    <!--Аукционы пользователя-->
    <h3 id="auctionHeader">Владелец прекрасных аукционов</h3>
@{
    if (Model.HisAuctions.Count != 0)
    {
        foreach (var auction in Model.HisAuctions)
        {
            if (auction.EndTime >= DateTime.Now && auction.IsActive)
            {
                <script>
                    addCellToTheGrid('usersAuctions', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                        '@auction.EndTime', 'http://cocktion.com/Images/Thumbnails/
@auction.SellProduct.Photo.FileName',
                        'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
                </script>
            }
            else
            {
                <script>
                    addCellToTheGrid('usersAuctions', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                        'уже', 'http://cocktion.com/Images/Thumbnails/@auction.SellProduct.Photo.FileName',
                        'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
                </script>
            }

        }
    }
    else
    {
        <script>
            sayThatEmpty('usersAuctions');
        </script>
    }
}
</div>
<br/>
<br/>

<script>
    function unsubscribeFromUser(userId) {
        var formData = new FormData();
        formData.append('userId', userId);
        var xhr = new XMLHttpRequest();
        var response;
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4 && xhr.status == 200) {
```

```
            response = JSON.parse(xhr.responseText);
            if (response.Status == "Success") {
                document.getElementById("id_"+userId).innerHTML = "<p>Удален</p>";
            } else {
                document.getElementById("id_" + userId).innerHTML = "<p>Попробуйте еще раз,
пожалуйста... <span class=\"glyphicon glyphicon-ok\"></span></p>";

            }
        };
    }

    xhr.open('POST', '/Subscription/UnsubscribeFromUser');
    xhr.send(formData); //отправка данных
  }
</script>

<div class="row" id="usersSubs"> <!--Подписки пользователя на людей-->
  <h4 style="text-align: center">Ваши подписки</h4>
  <div class="col-md-4"></div>
  <div class="col-md-4">
    @{
      foreach (var user in Model.Friends)
      {
        <p id="id_@user.Id">@user.UserName <button
onclick="unsubscribeFromUser('@user.Id')">Удалить</button></p>
      }
    }
  </div>
  <div id="col-md-4"></div>
</div>

<div class="row" id="usersFeedback"> <!--То, что пишут о пользователе другие пользователе-->
  <!--Отзывы о пользователе-->
  <div class="col-md-3" id="feedbackDescription">
    <p>
      Здесь можно разместить свой отзыв о данном пользователе. Если
      данный пользователь вас когда-либо обманывал, то напиши о нем свой плохой отзыв
      и заявите о своих недовольствах администрации. Мы разберемся и примем меры:)
      Все ради Вас!
    </p>
  </div>
  <div class="col-md-6" id="feedbackFormContainer">
    <h5 id="oppinionHeader">Мнения</h5>
    <div id="feedbackPosts"><!--Контейнер для сообщений -->
      @{
        var feeds = (from x in db.Feedbacks
          where x.UsersId == Model.Id
          select x).ToList();
```

```
        }
        @if (feeds.Count != 0)
        {
            foreach (UsersFeedback feed in feeds)
            {
                <div class="feedbackPost">
                    <p class="postAuthor"><b>@feed.AuthorsName @feed.AuthorsSurname</b> сказал
                    </p>
                    <p>@feed.Message</p>
                </div>
                <br />
            }
        }
        else
        {
            <div class="feedbackPost" id="defaultFeedback">
                <p class="postAuthor">
                    <b>Коки Кок</b> сказал
                </p>
                <p>К сожалению, никто еще не оставил отзыв об этом пользователе</p>
            </div>
            <br />
        }
    </div>
</div>
<div class="col-md-3">
    <p>Не надо стесняяяться!</p>
    <button>Наверх</button>
</div>
</div>
```

1.7.17. Код представления Search/Index

```
@{
    ViewBag.Title = "Поиск";
}
<link rel="stylesheet" href="~/Content/globalSearch.css"/>
<script src="~/Scripts/MyLib/globalSearchFunctions.js"></script>
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>

<h2>Ищи все, что хочешь!</h2>

<!--Контейнер для поля с поиском-->
<div class="row">
    <div class="col-md-4"></div>
    <div class="col-md-4">
```

```
      <div class="input-group" id="searchContainer">
         <input type="text" id="searchField" class="form-control">
         <span class="input-group-btn">
            <button onclick=" sendSearchString() " class="btn btn-default" type="button">Погнали!</
button>
         </span>
      </div><!-- /input-group -->
      <div class="progress" id="progressBarContainer" hidden="">
         <div class="progress-bar" id="bar" role="progressbar" aria-valuenow="0" aria-valuemin="0"
aria-valuemax="100" style="width: 0%;">
         </div>
      </div>
   </div>
   <div class="col-md-4"></div>
</div>

<div id="auctionsContainer"></div>
```

1.7.18. Код представления Shared/_Layout

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>@ViewBag.Title - cocktion.com</title>
   <link rel="stylesheet" href="~/Content/Site.css"/>
   <link rel="stylesheet" href="~/Content/style.css" />
   <link rel="stylesheet" href="~/Content/BootstrapCss/bootstrap.min.css" />
   <script src="~/Scripts/MyLib/layoutFunctions.js"></script>

   @Scripts.Render("~/bundles/modernizr")
   @Scripts.Render("~/bundles/jquery")
   @Scripts.Render("~/bundles/bootstrap")
   <style>

   </style>

</head>
<body>
   <div class="navbar navbar-default navbar-fixed-top">
      <div class="container">
         <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse">
               <span class="icon-bar"></span>
               <span class="icon-bar"></span>
```

```
            <span class="icon-bar"></span>
          </button>
          @Html.ActionLink("Главная", "Index", "Home", new { area = "" }, new { @class = "navbar-
brand" })
        </div>
        <div class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li>@Html.ActionLink("Аукционы", "Index", "Auction")</li>
            <li>@Html.ActionLink("Продать", "Create", "Auction")</li>
            <li>@Html.ActionLink("Дома", "Index", "AuctionHouse")</li>
            <li>@Html.ActionLink("Люди", "Index", "Users")</li>
            <li>@Html.ActionLink("Поиск", "Index", "Search")</li>
          </ul>
          @Html.Partial("_LoginPartial")
        </div>
      </div>
    </div>
    <div class="container body-content">
      @RenderBody()
      <hr />
      <footer>
        <p style="text-align: center">Web Application for Student's Collaboration, 2015. Created by
Alexander Lazarenko.</p>
      </footer>
    </div>


    @RenderSection("scripts", required: false)
</body>
</html>
```

1.7.19. Код представления GetUser

```
@{
    ViewBag.Title = "Index";
}
@using CocktionMVC.Models.DAL
@using CocktionMVC.Models.ViewModels
@using Microsoft.AspNet.Identity
@model AspNetUser
<link rel="stylesheet" href="~/Content/UsersRelatedStyles/ProfileStyle.css" />
<script src="~/Scripts/MyLib/Profile/opinionAdder.js"></script>
<script src="~/Scripts/MyLib/Auction/Index/auctionAdder.js"></script>
<script src="~/Scripts/MyLib/AuctionHouse/GetUniversityHouses/printHousesInCells.js"></script>
<script src="~/Scripts/MyLib/Profile/subscribeProcessor.js"></script>
<script src="~/Scripts/Jquery/jquery.signalR-2.2.0.min.js"></script>
<script src="~/signalr/hubs"></script>
```

```
<div class="row" id="userFullInformation">
    <br />
    <!--Фотка и информация о пользователе-->
    <div class="col-md-3" style="align-items: flex-start">
        <!--Контейнер для аватарки-->
        @{
            if (Model.Selfie == null)
            {
                <img src="~/Content/SiteImages/anonPhoto.jpg" class="img-thumbnail" />
            }
            else
            {
                <img src="http://cocktion.com/Images/Thumbnails/@Model.Selfie.FileName" class="img-
thumbnail" />
            }
        }
        <div id="subscribe">

        </div>
        <script>
            checkSubscription("@Model.Id");
        </script>
    </div>
    <div class="col-md-2" id="userInformation" >
        <!--Контейнер для информации-->
        <div id="userName">
            <!--Контейнер для имени пользователя-->
            @{
                if (!string.IsNullOrEmpty(Model.UserRealSurname) && !
string.IsNullOrEmpty(Model.UserRealName))
                {
                    <h4>@Model.UserRealSurname @Model.UserRealName</h4>
                }
                else
                {
                    <h4>@Model.UserName</h4>
                }
            }
        </div>
        <br />
        <div id="userRates">
            <!--Контейнер для информации о пользователе-->
            <p><b>Рейтинг:</b> @Model.Rating</p>
            <p><b>Яиц: </b> @Model.Eggs</p>
            <p><b>Аукционов сделал:</b> @Model.HisAuctions.Count</p>
            <p>
```

```
            <b>Ставок сделал: </b> @Model.HisProducts.Count штук
        </p>
        @{
            if (Model.SocietyName != null)
            {
                <p><b>Общество: </b>@Model.SocietyName</p>
            }
        }
    </div>
</div>
<div class="col-md-7" id="userLifePos">
    <h4>Жизненная позиция пользователя</h4>
    <div class="row" id="interests">
        <h4>Интересы</h4>
        @{
            if (Model.Interests.Count == 0)
            {
                <p>Пользователь еще не определился с интересами</p>
            }
            else
            {
                foreach (var interest in Model.Interests)
                {
                    <p>@interest.Name</p>
                }
            }
        }
    </div>
</div>
</div>

<br />
<br />
<div class="row" >
    <div class="col-lg-3"></div>
    <div class="col-lg-6" id="gmsc">
        <p>Здесь можно написать сообщение данному пользователю</p>
        <div id="chatMessagesContainer" style="min-height: 200px"></div>
        <input type="text" class="form-control" id="inputMessage" /><br />
        <button class="btn btn-default" id="sendMessageBtn">Отправить</button>
        <br />
    </div>
    <div class="col-lg-3"></div>
</div>
<br />

<style>
    #chatMessagesContainer {
```

```
        border: solid 1px lightgray;
    }
    #gmsc{
        border: solid 1px lightblue;
    }
</style>

<script>
    $(document).ready(function () {
        $('#gmsc').keypress(function (e) {
            if (e.keyCode == 13)
                $('#sendMessageBtn').click();
        });
    });

    var chat = $.connection.messageHub;
    chat.client.addNewMessageToPage = function (author, message) {
        var textToAppend = '<p><b>' + author + ': </b>' + message + '</p>';
        $("#chatMessagesContainer").append(textToAppend);
    };

    $.connection.hub.start().done(function () {
        chat.server.addNewRoom('@User.Identity.GetUserName()');
        $('#sendMessageBtn').click(function () {
            var textToAppend = '<p><b>' + '@User.Identity.GetUserName()' + ': </b>' + $
("#inputMessage").val() + '</p>';
            $("#chatMessagesContainer").append(textToAppend);
            //1ый параметр - сам текст сообщений
            //2ой - имя автора сообщения
            //3ий - имя пользователя, которому надо отправить
            //айдишник текущего
            //айдишник того, которому надо отправить
            chat.server.send($("#inputMessage").val(), '@User.Identity.GetUserName()',
'@Model.UserName',
                '@User.Identity.GetUserId()', '@Model.Id');
            $("#inputMessage").val("");
        });
    });
</script>



<div class="row" id="subscribedHouses">
    <!--Дома, на которые подписан пользователь-->
    <h3 id="housesHeader">Житель следующих домов</h3>
</div>

@{
```

```
    if (Model.SubHouses.Count != 0)
    {
        foreach (var house in Model.SubHouses)
        {
            <script>
                addHouseCell('subscribedHouses', '@house.Faculty', '@house.Adress',
                    'http://cocktion.com/AuctionHouse/GetCurrentAuctionHouse/@house.Id',
                    'http://cocktion.com/Images/Thumbnails/@house.Portrait.FileName');
            </script>
        }
    }
    else
    {
        <script>$("#subscribedHouses").append("<p>К сожеланию, вы еще не подписались на дома ;(</
p>")</script>
    }
}

<br />
<br />

<div class="row" id="usersAuctions">
    <!--Аукционы пользователя-->
    <h3 id="auctionHeader">Владелец прекрасных аукционов</h3>
</div>
@{
    if (Model.HisAuctions.Count != 0)
    {
        foreach (var auction in Model.HisAuctions)
        {
            if (auction.EndTime >= DateTime.Now && auction.IsActive)
            {
                <script>
                    addCellToTheGrid('usersAuctions', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                        '@auction.EndTime', 'http://cocktion.com/Images/Thumbnails/
@auction.SellProduct.Photo.FileName',
                        'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
                </script>
            }
            else
            {
                <script>
                    addCellToTheGrid('usersAuctions', '@auction.SellProduct.Name.Replace("\n", " ")',
'@auction.SellProduct.Description.Replace("\n", " ")',
                        'уже', 'http://cocktion.com/Images/Thumbnails/@auction.SellProduct.Photo.FileName',
                        'http://cocktion.com/Auction/CurrentAuction/@auction.Id');
                </script>
```

```
        }

    }
  }
  else
  {
    <script>
      sayThatEmpty('usersAuctions');
    </script>
  }
}

<br />
<br />
<div class="row" id="usersFeedback">
  <!--Отзывы о пользователе-->
  <div class="col-md-3" id="feedbackDescription">
    <p>
      Здесь можно разместить свой отзыв о данном пользователе. Если
      данный пользователь вас когда-либо обманывал, то напиши о нем свой плохой отзыв
      и заявите о своих недовольствах администрации. Мы разберемся и примем меры:)
      Все ради Вас!
    </p>
  </div>
  <div class="col-md-6" id="feedbackFormContainer">
    <h5 id="oppinionHeader">Мнения</h5>
    <div id="feedbackPosts">
      <!--Контейнер для сообщений -->
      @{
        CocktionContext db = new CocktionContext();
        var feeds = (from x in db.Feedbacks
          where x.UsersId == Model.Id
          select x).ToList();
      }
      @if (feeds.Count != 0)
      {
        foreach (UsersFeedback feed in feeds)
        {
          <div class="feedbackPost">
            <p class="postAuthor">
              <b>@feed.AuthorsName @feed.AuthorsSurname</b> сказал
            </p>
            <p>@feed.Message</p>
          </div>
          <br />
        }
      }
      else
```

```
        {
          <div class="feedbackPost">
            <p class="postAuthor">
              <b>Коки Кок</b> сказал
            </p>
            <p>К сожалению, никто еще не оставил отзыв об этом пользователе</p>
            <br />
          </div>
        }
      </div>
      <div id="writeFeedbackForm">
        <!--Контейнер с формой для отправки своего мнения-->
        <h5>Напиши своё</h5>
        <textarea id="writeFeedbackTextarea"></textarea>
        <button class="btn btn-success" onclick="addOpinion('@Model.Id')"
id="sendFeedbackBtn">Отправить отзыв</button>
      </div>
   </div>
   <div class="col-md-3">
   </div>
</div>
```

1.7.20. Код представления Users/Index

```
@{
   ViewBag.Title = "Cocktioneers";
}
@using CocktionMVC.Models.DAL
@using Microsoft.AspNet.Identity
@model List<CocktionMVC.Models.DAL.AspNetUser>
<h3>Пользователи</h3>
<link rel="stylesheet" href="~/Content/UsersRelatedStyles/usersStyle.css"/>
<script src="~/Scripts/MyLib/Users/userPrinter.js"></script>
<script src="~/Scripts/MyLib/Users/subscriptionProcessor.js"></script>
<div class="row">
   <div class="col-md-3"></div>

   <div class="col-md-6" id="usersHere">
      <!--Контейнер для информации о пользователе -->
      @{
         int amountOfUsers = Model.Count;
         CocktionContext db = new CocktionContext();
         var user = db.AspNetUsers.Find(User.Identity.GetUserId());
         int stopIndex = amountOfUsers - 1;
         if (User.Identity.IsAuthenticated)
```

```
        {
            for (int i = 0; i < amountOfUsers; i++)
            {
                if (String.IsNullOrEmpty(Model[i].UserRealSurname) &
String.IsNullOrEmpty(Model[i].UserRealName))
                {
                    continue;
                }
                if (i == stopIndex)
                {
                    if (Model[i].Selfie == null)
                    {
                        <script>
                            addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'Content/SiteImages/anonPhoto.jpg',
                                '@Model[i].Rating', '@Model[i].Eggs', '150', true,
'@user.Friends.Contains(db.AspNetUsers.Find(Model[i].Id))');
                        </script>
                    }
                    else
                    {
                        <script>
                            addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'http://cocktion.com/Images/Thumbnails/@Model[i].Selfie.FileName',
                                '@Model[i].Rating', '@Model[i].Eggs', '150', true,
'@user.Friends.Contains(db.AspNetUsers.Find(Model[i].Id))');
                        </script>
                    }
                }
                else
                {
                    if (Model[i].Selfie == null)
                    {
                        <script>
                            addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'Content/SiteImages/anonPhoto.jpg',
                                '@Model[i].Rating', '@Model[i].Eggs', '150', false,
'@user.Friends.Contains(db.AspNetUsers.Find(Model[i].Id))');
                        </script>
                    }
                    else
                    {
                        <script>
                            addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'http://cocktion.com/Images/Thumbnails/@Model[i].Selfie.FileName',
                                '@Model[i].Rating', '@Model[i].Eggs', '150', false,
'@user.Friends.Contains(db.AspNetUsers.Find(Model[i].Id))');
                        </script>
```

```
                }
            }
        }
    }
    else
    {
        for (int i = 0; i < amountOfUsers; i++)
        {
            if (i == stopIndex)
            {
                if (Model[i].Selfie == null)
                {
                    <script>
                        addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'Content/SiteImages/anonPhoto.jpg',
                            '@Model[i].Rating', '@Model[i].Eggs', '150', true, 'False');
                    </script>
                }
                else
                {
                    <script>
                        addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'http://cocktion.com/Images/Thumbnails/@Model[i].Selfie.FileName',
                            '@Model[i].Rating', '@Model[i].Eggs', '150', true, 'False');
                    </script>
                }
            }
            else
            {
                if (Model[i].Selfie == null)
                {
                    <script>
                        addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'Content/SiteImages/anonPhoto.jpg',
                            '@Model[i].Rating', '@Model[i].Eggs', '150', false, 'False');
                    </script>
                }
                else
                {
                    <script>
                        addUser('@Model[i].UserRealName', '@Model[i].UserRealSurname', '@Model[i].Id',
'http://cocktion.com/Images/Thumbnails/@Model[i].Selfie.FileName',
                            '@Model[i].Rating', '@Model[i].Eggs', '150', false, 'False');
                    </script>
                }
            }
        }
    }
```

```
}
    </div>
    <!--
    <div class="col-md-3" id="filterContainer">
        <p>Отфильтруйте :)</p>
        <!--Контейнер для фильтра
        <div class="userFilters">
            <input type="text" id="city" class="form-control" placeholder="Город">
            <br/>
            <input type="text" id="university" class="form-control" placeholder="Университет">
            <br/>
            <input type="text" id="faculty" class="form-control" placeholder="Факультет">
            <h5>Дома</h5>
            <label class="checkbox-inline">
                <input type="checkbox"> В тех же, где и я.
            </label>
        </div>
    </div>
-->
    <div class="col-md-3"></div>
</div>
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

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