**Постановка задачи** (слайд 2)

Целью работы является создание клиент-серверного приложения для автоматизации продажи игровых аккаунтов, ключей через Интернет. Эту задачу было бы правильно разделить на несколько подзадач:

1) Ознакомление с различными технологиями разработки web-приложений.

2) Разработка структуры приложения и базы данных для хранения информации о продукции магазина, покупках и покупателях.

3) Создание web-приложения:

Реализация каталога товаров разделённых по категориям;

Реализация корзины пользователя;

Предоставление информации о возможности оплаты;

4) Организация способа оплаты и предоставления оплаченного товара.

Используемые инструменты:

Для реализации задачи на стороне сервера было решено воспользоваться стеком технологий, которые предлагает Microsoft; а именно технологией asp.net, которая является частью платформы .net, использующая все её возможности.

В качестве архитектурного каркаса web-приложения используется asp.net mvc 5 fraemwork.

Этот framework следует архитектурному шаблону MVC, который предназначен для разделения модели данных, бизнес-логики и интерфейса пользователя.

Хранилищем данных выступи Microsoft sql server.

В качестве orm фреймворка применяется Microsoft entity framework

Структура данных.

**Сущности и связи** (слайд 3)

В соответствии с требованиями, в модели данных выделим следующие сущности: Товар, Категория, Заказ, Пользователь и корзина.

Чтобы сформировать корзину, необходимо знать id пользователя, товар, количество.

Чтобы сформировать заказ, необходимо знать какие товары присутствуют в заказе и их цену.

**Проектирование модели данных.**

Существует набор принципов, называемы предметно-ориентированным проектированием (DDD), некоторые элементы которого реализованы в приложении. Правильное применение этих принципов приводит к созданию программных абстракций, которые называются модели предметных областей. Они решают проблемы объектов предметных областей – Доменных обектов.

Эта концепция выделяет следующие элементы построения моделей предметной области:

Сущность

Объект-значение

Множество

Сервис

Репозиторий

В соответствии с концептуальной моделью данных можно выделить следующие классы доменных объектов (слайд 4)

**Диаграмма вариантов использования**(слайд5)

была разработана use case diagram, которая представлена на слайде

**Сквозная функциональность** (слайд 6)

Сквозная функциональность это аспекты дизайна, которые могут применятся ко всем слоям, компонентам и уровням, например:

Аутентификация и авторизация

Свзяь

Управление исключениями

Один из важнейших компонентов сквозной функциональности Внедрение зависимостей (слайд 6)

Контроллеры не являются переиспользуемыми компонентами, они не должны содержать бизнес логику. Т.о. контроллеры имеют внешние зависимости в виде сервисных компонентов модели данных и репозиториев. Для ослабления связи между компонентами было решено воспользоваться подходом Внедрение зависимости.

Первый этап заключается в подготовке Ninject к использованию. Для этого создается экземпляр ядра Ninject.

Второй этап процесса состоит в конфигурировании ядра Ninject с целью предоставления сведений о том, какие объекты реализации должны применяться для каждого интерфейса, с которым придется работать.

Последний этап — это действительное использование Ninject

**Корзина товаров**

Кнопка добавления в корзину ("Добавить в корзину") будет отображаться рядом с каждым товаром в каталоге. Щелчок на ней будет приводить к выводу сводки по товарам, которые уже были выбраны пользователем, включая общую стоимость. В этой точке пользователь может с помощью кнопки продолжения покупки ("Продолжить покупку") вернуться в каталог товаров, а с помощью кнопки перехода к оплате ("Перейти к оплате") - сформировать заказ и завершить сеанс покупки.

Корзина для покупок является частью предметной области приложения, поэтому для представления корзины имеет смысл создать сущность в модели предметной области.

Для сохранения и извлечения объектов Cart применяется средство состояния сеанса ASP.NET

**Платженая система** (слайд 8)

Первым делом необходимо подключить HTTP-уведомления. Это можно сделать в личном кабинете по адресу <https://sp-money.yandex.ru/myservices/online.xml>

Во-первых, в поле адреса нам надо ввести адрес того ресурса, который будет обрабатывать приходящие от Яндекса уведомления об оплате. На данный момент у нас ничего нет, но потом можно изменить. Только не надо устанавливать адреса типа localhost:xxxx. Нужен не локальный адрес, доступный извне. Запросы будут отправляться с помощью http-метода POST.

Во-вторых, яндекс нам дает некоторое секретное слово, которое мы можем посмотреть, нажав на кнопку "Показать секрет". Оно нам понадобится для проверки уведомлений в целях безопасности.

И в третьих, нужно установить отметку в поле "Отправлять уведомления"

у нас в приложении есть модель order, описывающая заказ пользователя

форма , которая передается на сервер яндекса (слайд 9)

Для отправки формы нам надо указать на форме ряд параметров.

Параметр receiver указывает кошелек в ЯД, на который клиент будет переводить деньги.

Поле label (<input name="label" value="@Model.OrderId" type="hidden">) выступает в качестве идентификатора платежа в нашей системе. Таким идентификатором является Id заказа.

Параметр quickpay-form указывает на тип платежей, в нашем случае он имеет значение "shop".

Параметр targets содержит название платежа, которое будет отображаться клиенту при оплате.

Параметр sum указывает на сумму, которую вводит клиент.

Параметр successURL указывает на ресурс, на который будет переадресация после успешной оплаты. Здесь задан тот же адрес, что указывался выше для получения http-уведомления. Однако в данном случае редирект будет идити с помощью запроса GET, то есть два метода будут обрабатывать разные типы запросов, хотя и будут называться одинаково.

Для параметра successURL мы можем задать метод: Checkout() – заказ оплачен

[HttpGet]

public string Checkout()

{

    return "<p>заказ оплачен</p>";

}

Суть метода, который и будет обрабатывать полученные HTTP-уведомления о платеже:

* проверяем хэш
* создаем класс для сравнения строк
* если хэши идентичны, добавляем данные о заказе в бд

От яндекса в http-уведомлении мы получаем ряд параметров о платеже. Однако есть одна проблема: этому же методу может обратиться любой и послать какие угодно параметры. Для этого нам и нужно секретное слово. Оно указывается в качестве значения строковой переменной key. Оно позволяет сгенерировать хэш пароля с использованием алгоритма sha1.

После успешной оплаты клиента переадресует на метод Cart/checkout обрабатывающего запрос Get. А на метод Cart/checkout, обрабатывающий запрос POST, будет отправлено http-уведомление.

**Покупка товара**

Покупка подтверждается электронным сообщением на почту, указанную пользователем при покупке и присылается ключ авторизации игры. Отправка сообщения реализовано через встроенный класс SmtpClient.

**Панель администратора**

По общепринятым нормам, администратор должен иметь возможность использования CRUD операций через пользовательский интерфейс, для этого была реализована панель администратора.

Для перехода к панели администратора нужно перейти по адресу /Admin/index и пройти аутентификацию.

Для поддержки средств администрирования был создан MVC 5 Controller с именем AdminController.

В конструкторе AdminController объявлена зависимость от интерфейса IGameRepository, которую Ninject будет распознавать при создании экземпляров. Метод действия Index() вызывает метод View(), чтобы выбрать стандартное представление для действия, передавая ему в качестве модели представления набор товаров из базы данных.

Для отображения списка товаров было создано представление Index с вариантом шаблона List.