**Постановка задачи** (слайд 2)

Целью работы является создание клиент-серверного приложения для автоматизации продажи игровых товаров через Интернет. Эту задачу было бы правильно разделить на несколько подзадач, которые вы можете увидеть на слайде, я отмечу некоторые из них

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

2) Создание web-приложения:

3) Организация способа оплаты и предоставления оплаченного товара.

Используемые инструменты:

Для реализации задачи на стороне сервера было решено воспользоваться стеком технологий, которые предлагает Microsoft; В качестве архитектурного каркаса web-приложения используется asp.net mvc 5 fraemwork. 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 объектов.

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

Я пользуюсь подходом Inversion of Control (инверсия управления) — это некий абстрактный принцип, набор рекомендаций для написания слабо связанного кода.

Dependency Injection (внедрение зависимостей) — это одна из реализаций этого принципа (есть еще Factory Method, Service Locator).

IoC-контейнер — это библиотека, которая позволяет упростить и автоматизировать написание кода с использованием данного подхода

Согласно подходу инверсии управления если у нас есть клиент, который использует некий сервис, то он должен делать это не напрямую, а через посредника.

Для решения этой задачи мной был выбрана библиотека Ninject. Далее показываете на MVC жизненный цикл запроса, я переопределил стандартный MVC Dependecy Resolution, на написанную мною реализацию с использованием библиотеки Ninject, это класс NinjectDependencyResolver.cs (будьте готовы ответить на вопрос, чем Ninject круче)

**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 ("Оформить заказ") - сформировать заказ и завершить сеанс покупки.

Для сохранения и извлечения объектов Cart используется сессию Asp.NET и привязка моделей.

**Платженая система** (слайд 8)

В качестве платежной системы была выбрана весьма популярная платформа в России – Яндекс Деньги. Я зарегистрировал свое приложение в системе Яндекс.Деньги, заполнил всю необходимую информацию, после чего я получил готовую html разметку которую было необходимо использовать в моем приложении. Где надо указать счет для получения платежа и сумму платежа.

Хотелось бы отметить, что в программе предусмотрена возможность подключения любых других платежных систем со схожим принципом оплаты. (опять же можете это либо в речь либо оставить на потом, потому что про это полюбому будет вопрос, лучше уж гарантированно получить вопрос на это место)

(слайд 9-12) Процесс покупки. Товар добавляется в корзину, далее форма ввода данных покупателя, форма оплаты товара на сервере Яндекс-деньги. а в конце, так же отправляется письмо

**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 аутентификации используется класс *FormsAuthentication*, а так же фильтер *Authorize.*

Данные хранятся в коллекции IEnumerable, реализованы методы edit( id game), для сохранения, перегруженный edit(game, image = null) для создания нового товара, Delete для удаления. Для сохранения изменений в хранилище был добавлен метод saveGame и DeleteGame в класс репозитория. Сообщение tempData после фиксации изменения.

**Заключение**

В результате работы была спроектирована гибкая архитектура приложения, подготовленная к изменениям и модификациям в будущем. Приложение использует современные технологии, как на клиентской части, так и на серверной.

Рассмотрены проблемы проектирования модели данных и применена методика дизайна архитектуры на основе данных предметной области.

Спроектированный архитектурный каркас позволяет писать слабосвязанные компоненты, которые поддаются как модульному тестированию, так и подходу написания качественного программного обеспечения – разработку через тестирование.

Во время разработки проводилось тестирование создаваемых компонентов.

Изучены технологии привязки моделей и внедрение зависимостей, *Framework* *Bootstrap* и движок представление *Razor*.