

**рисунок 1 Примерная схема основных частей проекта**

В доменной модели 1,2,3 – это подсистемы

Короче, какая моя задумка была:

1. разбиваем программу на 3 основные подсистемы (сделал)
2. описываем примерно, какие функции в них будут, и что эти функции будут делать. Так же описываем взаимодействие подсистем между собой, тип какая подсистема к какой обращается и что они там делают (начал)
3. разбиваем подсистемы на классы и описываем их взаимодействие между собой (не успел)

Задачи доменной модели:  
авторизация  
покупка/продажа  
сохранения информации  
просмотр товаров  
просмотр пользователей  
распределение ролей

**ПУНКТ 1 Основные подсистемы**:

Подсистема №1, отвечающая за работу с БД. Обрабатывает, как и запросы к бд напрямую, так и вызовы методов из разряда: «покажи мне список всех купленных товаров, пользователем Вася»

Подсистема №2, отвечающая за хранение нынешнего пользователя и вызов методов, доступных ему.

Подсистема №3, отвечающая за создание новых пользователей, регистрацию и авторизацию пользователя.

Подсистема №4, отвечающая за обработку заказов

**ПУНКТ 2 Детальное описание основных подсистем**

**Подсистема №1:**

interface IDBRequestSystem

Интерфейс, которые инкапсулирует логику работы с БД. Т.е. остальная программа не знает, как работает бд. Ей и не надо.

Есть три таблицы(БД), в которых будет храниться информация.

То как будет устроен класс работы с бд – неважно. Главные чтобы он реализовывал следующий интерфейс.

Каталог Товаров (таблица товаров)

* Добавление товара (Выставление описания, цены и фотографий товара)

bool AddProduct(Product prod)

Добавляет объект prod в бд.

true – добавление произошло

false – добавление не произошло

* Изменение товара по id (описания, цены и фотографий товара)

bool ChangeProductByGuid(Guid id, Product prod)

Переносит поля prod, в поля товара в бд. Id Товара в бд не меняется.

true – изменение произошло

false - изменение не произошло

* Изменение товара по названию (описания, цены и фотографий товара)

ChangeProductByName(string name, Product prod)

Переносит поля prod, в поля товара в бд. название товара в бд не меняется

true – изменение произошло

false - изменение не произошло

* Удаление товара по id (описания, цены и фотографий товара)

bool DeleteProductByGuid(Guid id)

Удаляет товар с таким id из бд

true – удаление произошло

false - удаление не произошло

* Удаление товара по названию (описания, цены и фотографий товара)

DeleteProductByName(string name)

Удаляет товар с таким name из бд

true – удаление произошло

false - удаление не произошло

* Получение всего каталога товаров

List<Products> GetAllProducts()

Сделать проверку на отсутствие объектов в бд

* Получение товаров в наличии

List<Products> GetStockProduct()

Сделать проверку на отсутствие объектов в бд

* Поиск товара по Guid

Product GetProductByGuid(Guid id)

Возвращает соответствующий товар из бд

NULL (или что то похожее) если такого товара нет

* Поиск товара по названию

Product GetProductByName(string name)

Возвращает соответствующий товар из бд

NULL (или что то похожее) если такого товара нет

* Проверка товара на наличие по Guid

bool ContainsByGuid(Guid id)

True – товар есть в наличии

False – товара нет в наличии

* Проверка товара на наличие по навзанию

bool ContainsByName(string name)

True – товар есть в наличии

False – товара нет в наличии

* Получение товара, купленного определенным покупателем (покупатель определяется по login)

List<Product> GetCustomerOrdersByLogin(string name)

* Получение всего купленного товара

List<Product> GetAllBoughtProduct()

* Просмотр товара, проданного определенным продавцом по login

List <Product> GetSellerProductByLogin(string name)

Таблица заказов

* Добавление заказа

bool AddOrder(Order order)

true – добавление произошло

false – добавление не произошло

* Удаление заказа

bool DeleteOrder(Guid id)

true – удаление произошло  
false – удаление не произошло

* Изменение заказа

bool ChangeOrder(Guid id, Order order)

id – id заказа который нужно изменить

order – заказ которым будет заменен другой заказ

true – изменение произошло  
false – изменение не произошло

* Получение заказов определенного пользователя

List<Order> GetOrderByLogin(string login)

Возвращает список заказов определенного пользователя. Если заказов нет, или пользователя не существует – возвращает null

Таблица пользователей

* добавление пользователя

bool AddUser(User user)

true – добавление произошло

false – добавление не произошло

* Удаление пользователя по login

bool DeleteUser(string login)

true – удаление произошло  
false – удаление не произошло

* bool ChangeUser(string username, string login, string password, Role role)

true – изменение произошло  
false – изменение не произошло

* Получение пользователей по определенным ролям

List<User> GetUserByRoles(Role role)

* Получение всех пользователей

List<User> GetAllUsers()

* Получение пользователя по login

User GetCertainUser(string login)

* Проверка на наличие login в бд

bool ContainsUser(string login)

**Подсистема №2:**

CustomerRequestHandler :

обрабатывает запросы пользователя

+ userControlSystem : UserControlSystem

+ currentUser : User

вызывает соответствующий метод у currentUser

+ CancelOrder()

вызывает соответствующий метод админа, если пользователь админ

+ CreateUser()

вызывает соответствующий метод админа, если пользователь админ

+ DeleteUser()

**Подсистема №3**

UserControlSystem

класс отвечающий за создание новых пользователей, регистрацию и авторизацию

+ customerRequestHandler : CustomerRequestHandler

+ db : IDBRequestSystem

создание нового пользователя

+ bool CreateUser(string login, string password, Role role)

Пытается создать пользователя с таким логином. Если такой пользователь есть – возвращает false, иначе – true.   
Все поля, кроме тех, что переданы в метод, заполняет NULLом или чем то таким.  
Обращается к интерфейсу работы с бд

удаление существующего пользователя

+ bool DeleteUser(string login)  
Пытается удалить пользователя с таким login. Если такой пользователь есть – возвращает true, иначе – false.

изменение существующего пользователя

+ bool ChangeUserInfo(string login, string newLogin, string password)

Изменяет соответствующий поля у пользователя с login.  
Если какой то из переданных аргументов NULL, пустая строка или другой тип соответствующий ничему, то такое поле у пользователя не меняется.

Например : вызову я метод ChangeUserInfo(“Antonio”, “”, “Alfkjl123”) – у пользователя Antonio поменяется только пароль – станет “Alfkjl123”  
  
Сначала получает пользователя из бд по login. Далее изменяет соответствующие поля и передает их в метод изменения пользователя в бд

Вернет True – если пользователь с login существует.

Вернет False – если пользователь с login не существует.

вход

+ bool LogIn(string login)

получает пользователя из бд и устанавливает его в поле currentUser в CustomerRequestHandler

True – если такой пользователь есть

False – если такого пользователя нет

выход

+ void LogOut()

в поле currentUser в customerRequestHandler устанавливает NULL.

изменяет нынешнего пользователя на другого

+ bool SwitchUser(string login)

Комбинация методов LogOut и LogIn.

True – если такой пользователь существует

False – если такой пользователь не существует

User : interface

Класс пользователь. Содержит в себе базовый функционал присущий каждой роли пользователя системы.

+ Login : string

+ Password : string

+ CancelOrder()

+ UserOrders : List<Orders> зачем?

+ CurrentOrder : Order (не надо сохранять в бд)

+ Role : enum

+ AddProductInBucket()

+ DeleteProductInBucket()

+ CreateOrder()

Customer : User

Класс определяет возможности покупателя и, в соответствии им, реализует методы родителя.

+ override CreareOrder()

+ override CancelOrder()

Seller : User

Класс определяет возможности продавца и, в соответствии им, реализует методы родителя.

+ override CreareOrder()

+ override CancelOrder()

Admin : User

Класс определяет возможности администратора и, в соответствии им, реализует методы родителя.

+ override CreareOrder()

+ override CancelOrder()

+ CancelAnyOrder()

+ DeleteUser()

+ CreateUser()

**Подсистема №4**

OrderHandleSystem

Класс отвечающий за работу с заказами – их создание, обработку их изменений. Данный класс существует, поскольку при появлении необходимости изменения процесса создания заказа или изменения логики обработки заказов (добавления новых состояний, более сложные переключения между ними) изменения будут вносится более удобным образом, станет проще отлов ошибок и их правки.

Вернуть заказ  
+ bool ReturnOrder(Guid id)

Получает заказ из бд.

Поле State в order делает – cancelled  
Берет текущее время на компьютере и записывает в поле CancellationTime

Вызывает метод ChangeProductByGuid()

Создать заказ

+ Order CreateOrder(string creatorLogin, string buyerFullName , List<Guid> bucket)

Создаст объект класса Order и проинициализирует его поля:

Создаст объект класса Guid и перенесет его в поле Id.Fcre  
Запишет creatorLogin, buyerFullName, bucket в соответствующие поля.  
Возьмет текущее время на компьютере и записывает в поле CreationTime  
Поле State в order делает – finished  
В остальные поля запишет NULL

True – заказ существует

False – заказ не существует

Order :

Класс описывающий заказ. Заказ имеет 2 состояния:

1. завершен (пользователь оформил заказ. Он должен сохраняться в базу данных)
2. возвращен (пользователь вернул товар назад и забрал деньги. Он должен сохраняется в базу данных). В заказе содержится информация о состоянии заказа, том кто его сделал, когда сделал и что заказал.

Заказ переходит в состояние “завершен” одновременно с созданием объекта Order. (поскольку объект класса Order будет создавать только при создании заказа пользователем. Т.е. Order создастся тогда, когда пользователь тыкнет создать заказ. Разделение на создан и доставлен отсутствует, т.к. зачем усложнять себе жизнь).

Переход в состояние “возвращен” возможен только из состояния “завершен”

+ Id : Guid

+ CreatorUserNane: string

+ BuyerFullName : string

+ CreationTime : DateTime

+ CancellationTime : DateTime

+ Bucket : List<Guid>

+ State : enum

+ GetBucketPrice() : decimal

проходится по корзине (поле Bucket) и возвращает стоимость всех товаров в ней

decimal GetBucketPrice();

ProductCreator

Чисто служебный класс. Нужен для того, чтобы создавать объекты Product, для демонстрации работы программы.

Причина существования – отделить код по созданию товаров, при демонстрации работы программы, от общего кода, чтобы не захламлять его.

Метод, которые отвечает за создание товара и вызов методов бд для помещения товара туда.

+CreateProduct()

Метод, в котором будут создаваться конкретные товары, для демонстрации работы программы.

+Demonstration()

Product : abstract

Абстрактный класс. Хранит в себе данные, которые имеет любой товар

+ Id : Guid

+ Name : string

+ Price : decimal

+ Manufacturer : string

+ InStock : bool

Car : Product

Класс наследуемый от Product. Хранит в себе данные о машине.

+ model: enum

+ engine : enum

+ gearbox : enum

+ FuelTankCapacity: float

+ ManufactureDate : DateTime

+ Color : enum

+ WheelDrive : enum

+ Weight : float

+ Power : float

+ Torque : float

+ FuelConsumptionCity : float

+ FuelConsumptionHighway : float

Tires : Product

Класс наследуемый от Product. Хранит в себе данные о покрышках.

+ Season : enum

+ Width : float

+ ProfileHeight : float

+ ConstructionType : char

+ RimDiameter : float

+ LoadIndex : float

+ SpeedIndex : char

EngineOil : Product

Класс наследуемый от Product. Хранит в себе данные о моторном масле

+ Composition : string

+ Viscosity : string

+ EngineType : enum

+ Volume : float

Пользователи

* Просмотр всех пользователей.

Dict<string, string> GetAllUsers()

Вернуть словарь, содержащий всех пользователей (ключ — имя пользователя, значение — строка содержащая информацию о пользователе (роль и прочая информация))

* Просмотр пользователей по ролям (только админ, только покупатель, или вместе)

Dict<string, string>(string role)

Вернуть словарь, содержащий пользователей определенный роли или нескольких ролей (ключ — имя пользователя, значение — строка содержащая информацию о пользователе(роль и прочая информация))

* Добавление пользователя в БД

bool AddUser(string login, string password, string name, string famila, string phoneNumber, string email, string role)

Устанавливает соответствующие данные в соответствующие поля и, если создание пользователя прошло успешно (пользователя с таким login не существовало в БД до вызова функции), функция возвращает true, иначе false

* Изменение пользователя в БД

bool ChangeUser(string login, string password, string name, string famila, string phoneNumber, string email, string role)

Устанавливает соответствующие данные в соответствующие поля и, если изменение пользователя прошло успешно (пользователя с таким login существовал в БД до вызова функции), функция возвращает true, иначе false

* Удаление пользователя из БД

bool DeleteUser(string login)

Удаляет пользователя с конкретным id. Возвращает true если такой пользователь существовал до вызова функции, иначе false

* Просмотр информации о пользователе

string UserInformation(string login)

Возвращает строку, содержащую информацию о пользователе с конкретным id. Если пользователь не существует – возвращает пустую строку

МЕЛКИЕ, ПРИВЕТИК!!!!!

ВАШИ КУРАТОРЫ — ЛОХИ. НЕ ТО, ЧТО У 4 ГРУППЫ)

АВТОР РУСЯ ХИПСТЕР

Вопросы?

* + - 1. Кто будет проверять корректность ввода пользовательских данных?
      2. как сделать проверку на то, какие именно данные введены неверно?