

FMFI UK

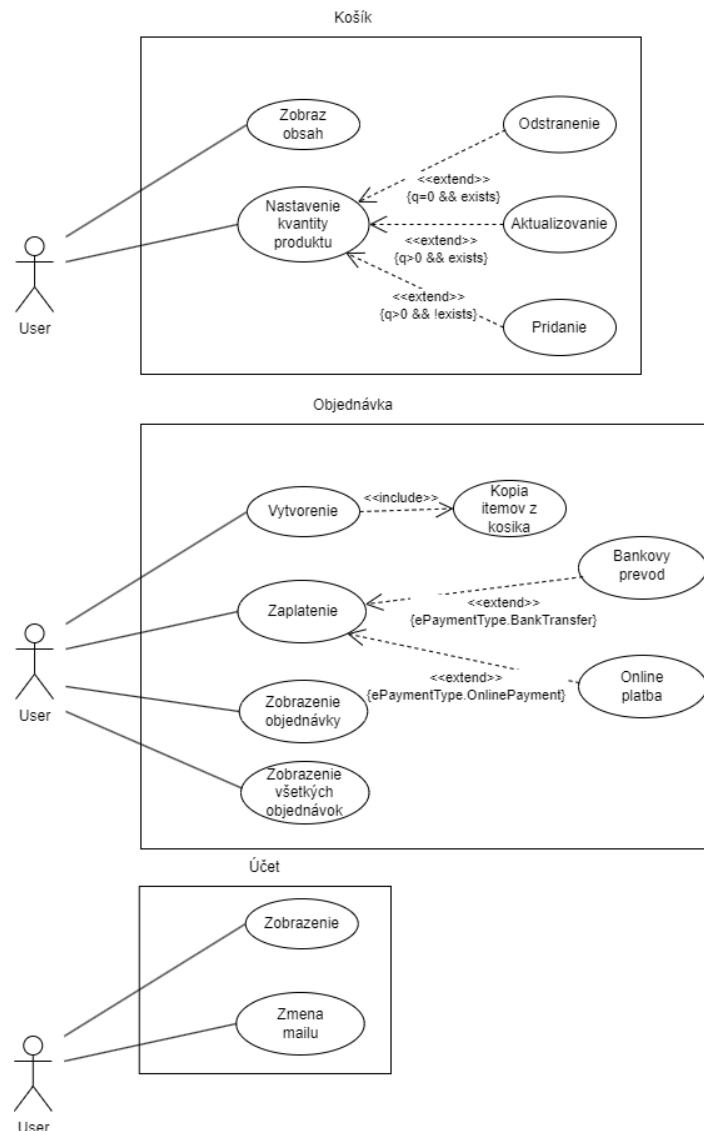
Online Food Ordering System  
Semestrálny projekt z ASwS 2023

## Úvod

Systém slúži na objednávanie produktov (jedla). Zákazníci majú košíky, do ktorých môžu pridávať a odstraňovať produkty. Z košíka je možné vytvoriť objednávku, zaplatiť za ňu a sledovať jej status.

Exceptions ktoré sú thrownuté v kóde sa s pomocou middleware pošlú ako error message v jsone.

## Use case



// Z pohľadu používateľa to závisí od implementácie a flowu frontendu. Pre používateľa sa v tomto prípade (resp. v každom prípade) iba vráti json ako odpoveď, teda scénar končí na prvom kroku.

1. Use case: Nastavenie kvantity produktu

Actor: User

Precondition: prihlásený používateľ

API má parametre ID produktu a kvantitu. Ak produkt neexistuje -> throw exception. Ak kvantita < 0 -> throw exception. Ak je produkt v košíku a kvantita je 0, pridá sa do košíka. Ak je produkt v košíku a kvantita nastavená na 0, odstráni sa z košíka. Ak produkt nie je v košíku, tak sa pridá.

2. Use case: Nastavenie kvantity produktu

Actor: User

Precondition: prihlásený používateľ, existujúci nezaplatený order

Podľa vybraného typu platby (parameter enum s 2ma možnosťami) sa vykoná platba.

3. Use case: Aktualizácia mailu

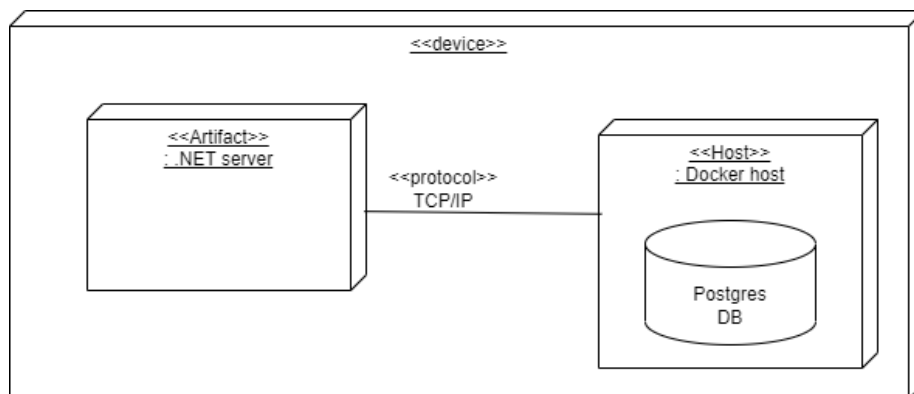
Actor: User

Precondition: prihlásený používateľ

Parameter email. Skontroluje sa validita emailu a ak je validný, aktualizuje sa.

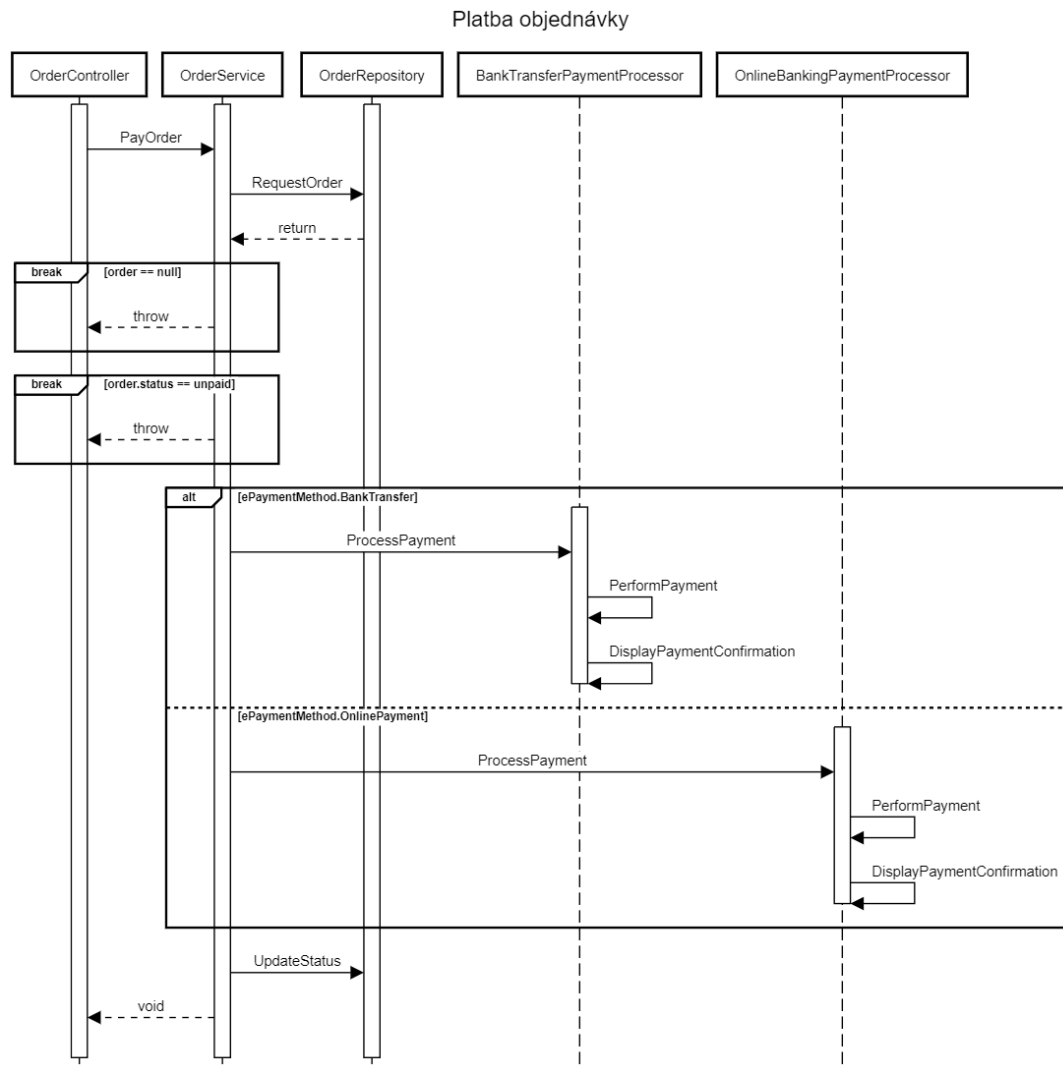
## Architektúra systému

Na zariadení (v mojom prípade Windows, no .NET Core a Docker bežia na Linuxe aj MacOS) beží .NET Core backend server a Postgres DB je deploynutá cez Docker.

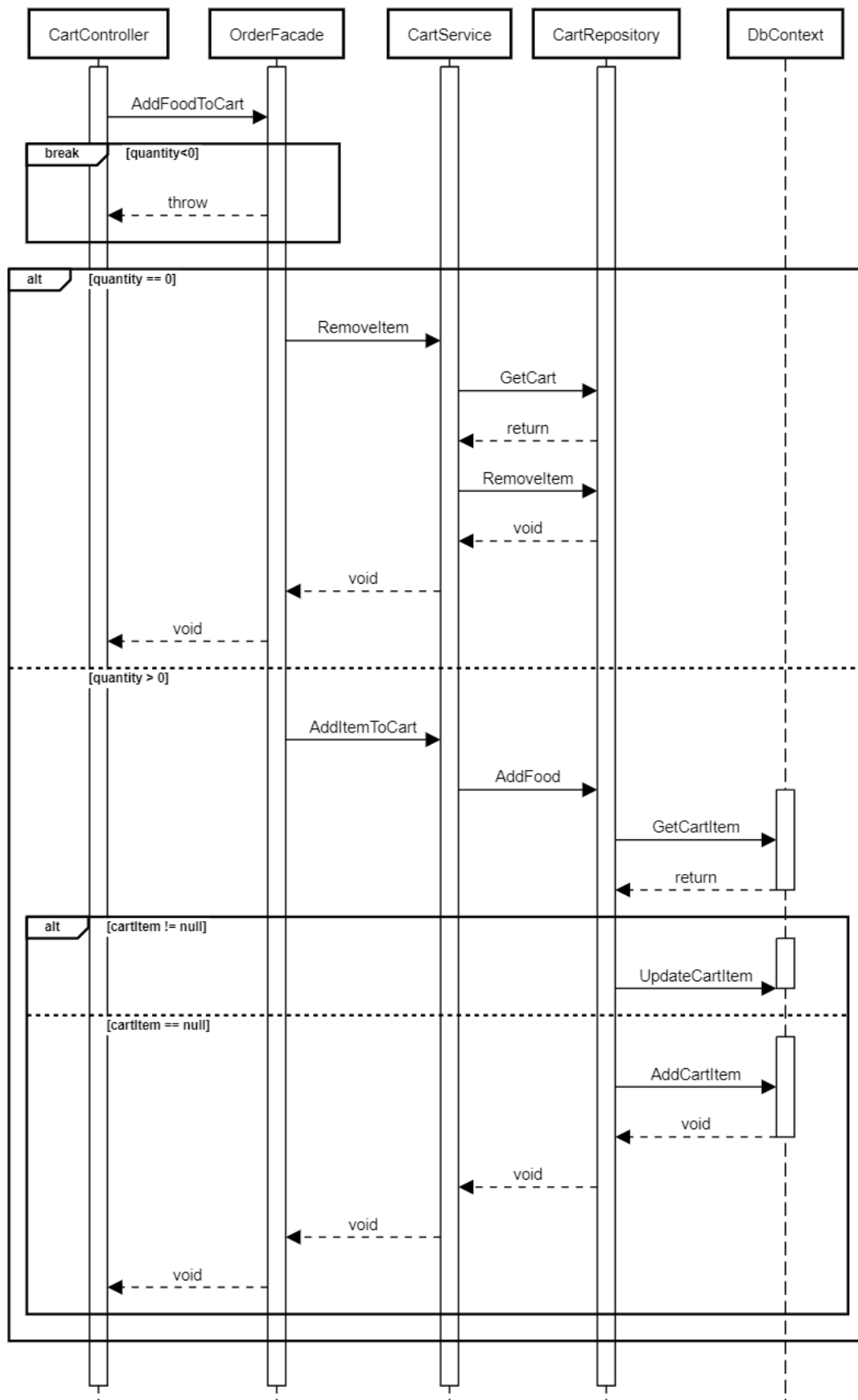


# Analýza

## Sequence diagram

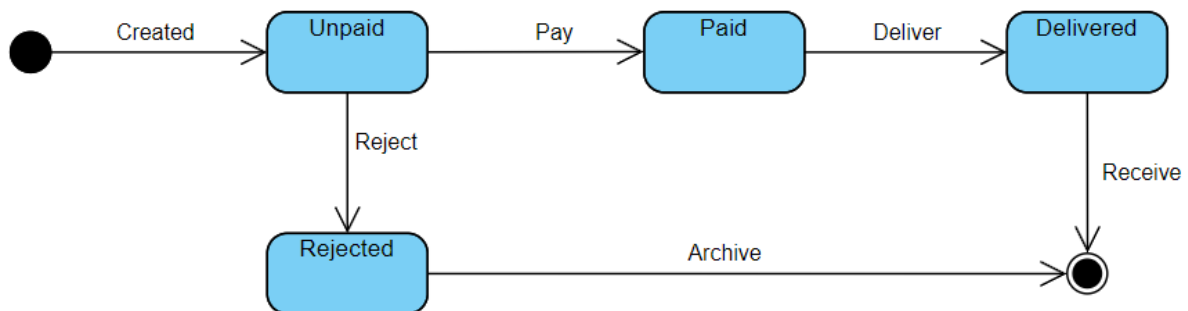


## Aktualizácia košíka

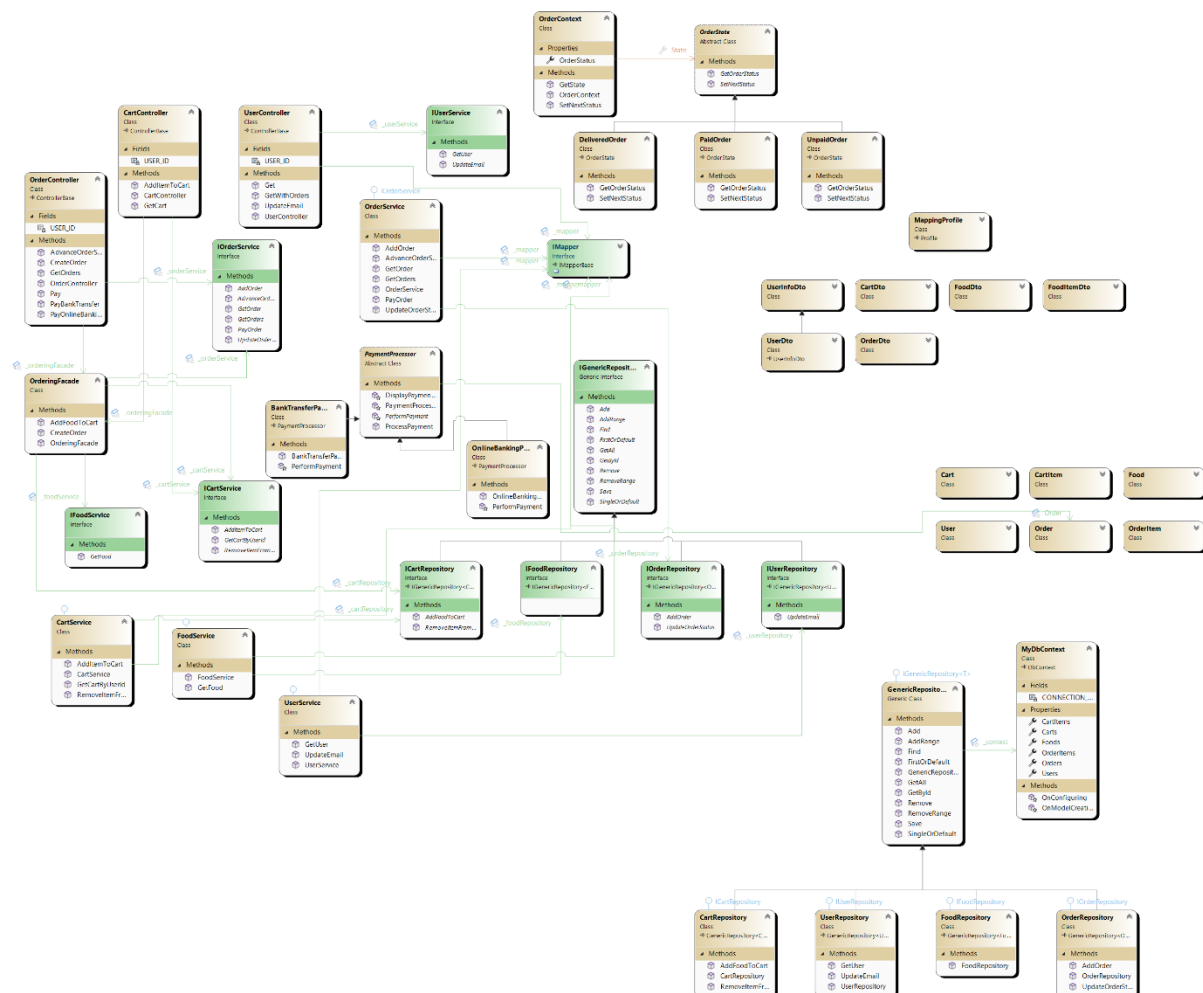


## State diagram

## Order

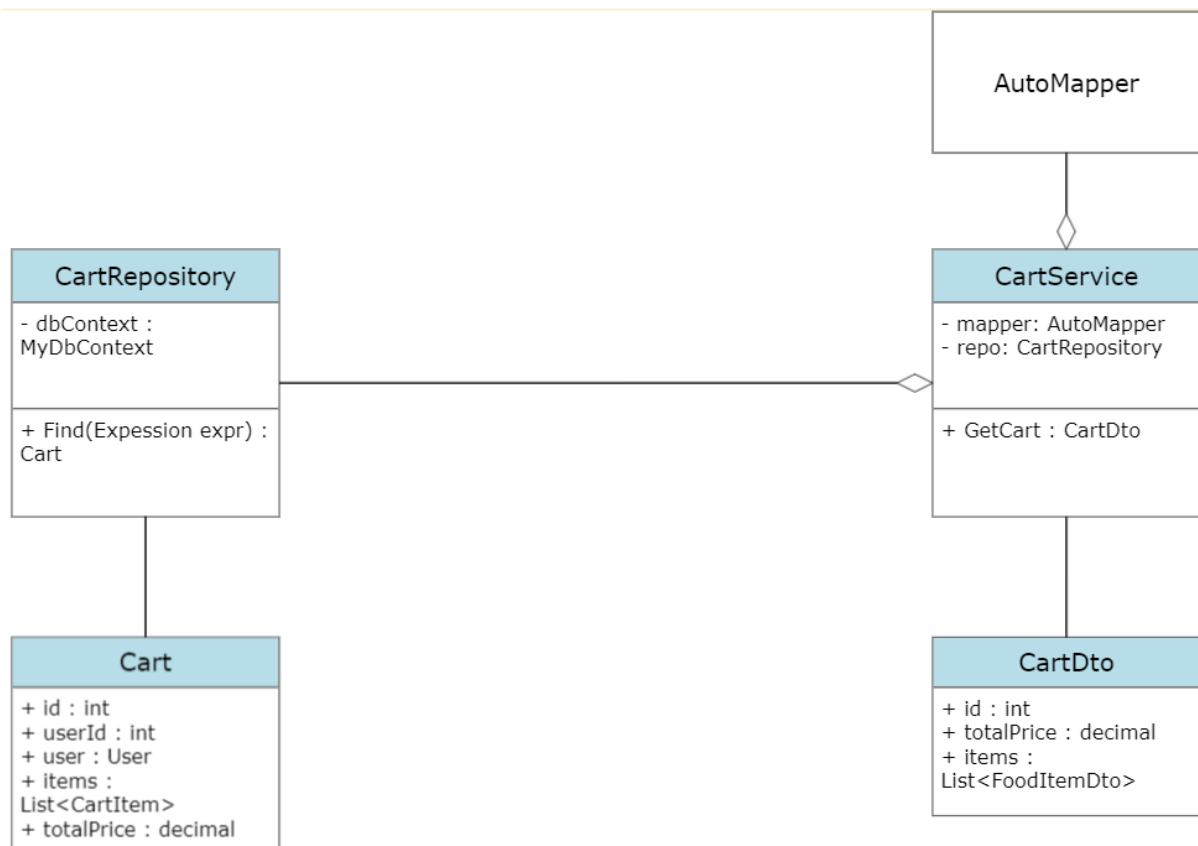


## Návrh a implementácia, identifikácia vzorov




## Class diagram

DTO




```

public class MappingProfile : Profile
{
     YelovSK
    public MappingProfile()
    {
        CreateMap<Order, OrderDto>();
        CreateMap<User, UserDto>();
        CreateMap<User, UserInfoDto>();
        CreateMap<UserDto, UserInfoDto>();
        CreateMap<Cart, CartDto>();
        CreateMap<CartItem, FoodItemDto>();
        CreateMap<OrderItem, FoodItemDto>();
        CreateMap<Food, FoodDto>();
    }
}

```

```

public class CartService : ICartService
{
    private readonly ICartRepository _cartRepository;
    private readonly IMapper _mapper;

     YelovSK
    public CartService(ICartRepository cartRepository, IMapper mapper)
    {
        _cartRepository = cartRepository;
        _mapper = mapper;
    }
}

```

```

public CartDto GetCartByUserId(int userId)
{
    var cart = _cartRepository.SingleOrDefault(expression: i:Cart => i.UserId == userId);

    if (cart == null)
    {
        throw new FoodOrderingException(message: "User not found");
    }

    return _mapper.Map<Cart, CartDto>(cart);
}

```



```

public class CartDto
{
    public int Id { get; set; }

    YelovSK
    public decimal TotalPrice => Items.Sum(f:FoodItemDto => f.Food.Price * f.Quantity);

    1 usage
    public List<FoodItemDto> Items { get; set; } = new();
}

```

```

public class Cart
{
    2 usages
    public int Id { get; set; }

    [Required]
    [ForeignKey(nameof(User))]
    4 usages
    public int UserId { get; set; }

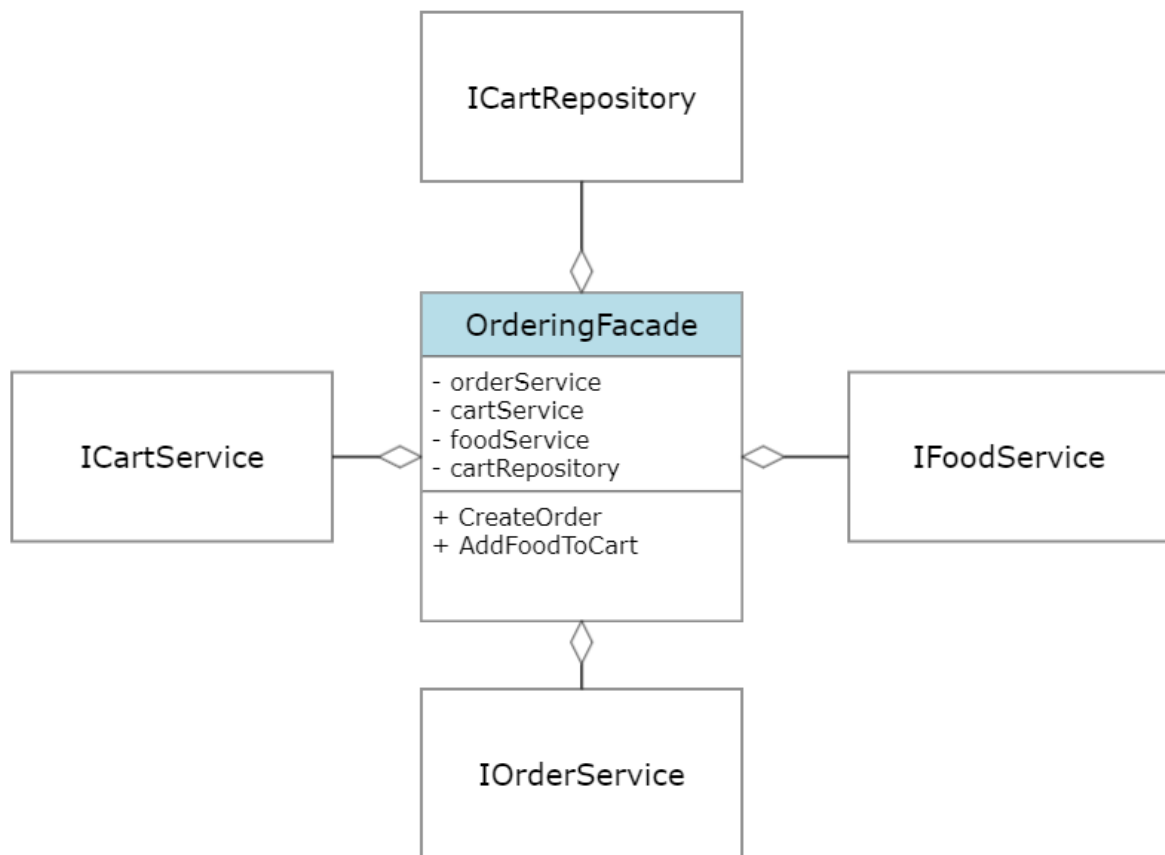
    2 usages
    public User User { get; set; }

    4 usages
    public ICollection<CartItem> Items { get; set; }

    YelovSK
    public decimal TotalPrice => Items.Sum(f:CartItem => f.Food.Price * f.Quantity);
}

```

## Facade



```
public class OrderingFacade
{
    private readonly IOrderService _orderService;
    private readonly ICartService _cartService;
    private readonly IFoodService _foodService;
    private readonly ICartRepository _cartRepository;

    public OrderingFacade(IOrderService orderService, ICartService cartService, IFoodService foodService, ICartRepository cartRepository)
    {
        _orderService = orderService;
        _cartService = cartService;
        _foodService = foodService;
        _cartRepository = cartRepository;
    }

    public OrderDto CreateOrder(int userId){...}

    public void AddFoodToCart(int userId, int foodId, int quantity){...}
}
```

Class diagram pre vzor

Implementácia vzorov v scenároch prípadov použitia