# My P0 Redo:

**Basically, I need to create a project using a** **repository design pattern** **EF Core** **database** **first** **on premise Application.**

**These are the requirements, but I will Implement all that I can: including the non-required ones.**

**This is a Pizza Ordering System for several Pizza Stores That belong to one chain, Dominos.**

**This is not a mobile app, but an on-Premises Application that can be accessed from the location of the user. For this level of the project, the user must literally download this app on their computer to use it. Even thought the DB is on my PC, for this Scenario I must pretend that its actually on a Server that can be accessed over the internet by all Persons involved. This is the worst implementation possible in 2020 but its make-believe as a starting point for me to complete the P0 project.**

**Repository Design Pattern provides a level of abstraction. Provides CRUD methods, when done right it cures the coupling problem and allows for easier unit testing.**

# pizzabox

## architecture

* [solution] PizzaBox.sln
  + [project - console] PizzaBox.Client.csproj
  + [project - classlib] PizzaBox.Domain.csproj
    - [folder] Abstracts --optional
    - [folder] Interfaces --optional
    - [folder] Models
  + [project - classlib ] PizzaBox.Storing.csproj
    - [folder] Repositories
  + [project - xunit] PizzaBox.Testing.csproj
    - [folder] Mocks --optional
    - [folder] Specs --optional

## Requirements

### Restaurant

* [required] should exist at least 1 store
* [required] should be able to view its completed orders
* [optional] should be able to view its sales
* [optional] should be able to view its inventory
* [optional] should be able to view its users

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Interface** | **Fields** | **Notes** |
| **Restaurant** | Interface IRestaurant | * **Restaurant Id** * **Restaurant Name** * **Phone Number** * **Location** | * **What Distinguishes the Restaurant is the location, because these are all Dominos Restaurants.** * **Restaurant has child classes Person, Sale (Pizza), Inventory, Orders, Employee even the Customer (they will subscribe to the app), Components** * **Restaurant should be an Interface because it HAS all of these. The classes for example ARE or follow the IS A.** |

### Credentials

* User Registers to use application with login and password and this will give access to all Dominos Locations
* User should be able to sign out of the application

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **ICredentials** | Class ICredentials: Customer | * **Credentials Id** * **Pass Pin** | * **Customer Has Credentials** |

### **User Entities\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* [required] should be able to view its order history
* [required] should be able to only order from 1 location/24-hour period
* [required] should be able to only order if an account exists
* [optional] should be able to only order 1 time within a 2-hour period

### Person

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **class** | **fields** | **notes** |
| **Person** | Abstract Class Person: IRestaurant | * **Person Id** * **implements all from IRestaurant** * **First Name,** * **Last Name,** * **Phone Number,** * **Email Address** | **Customer, Employee ARE Persons** |

### Employee

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee is a Person** | Abstract Class Employee: Person | * **Employee Id** * **Inherits all from Person Class, indirectly implements IRestaurant** | **Inherits Directly from Person (passively implements IRestaurant)** |

### Manager

|  |  |  |  |
| --- | --- | --- | --- |
| **Manager is a Employee** | Class Manager: Employee | * **Manager Id,** * **Inherits all from Employee Class** | **Inherits Directly from Employee (passively from Person and passively implements IRestaurant)** |

### Attendant

|  |  |  |  |
| --- | --- | --- | --- |
| **Attendant is an Employee** | Class Attendant: Employee | * **Attendant Id** * **Inherits all from Employee class** | **Inherits Directly from Employee (passively from Person and passively implements IRestaurant)** |

### Customer

|  |  |  |  |
| --- | --- | --- | --- |
| **Customer is a Person** | Abstract Class Customer: Person | * **Customer Id** * **Inherits all from Person Class** | **Inherits Directly from Person (passively implements IRestaurant)**  **Customer HAS Credentials** |

### PizzaSaleOrder

* [required] should be able to view its pizzas
* [required] should be able to compute its cost
* [required] should be able to limit its cost to no more than $250[optional] should be able to limit its pizza count to no more than 100

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **PizzaSaleOrder:** | Public class PizzaSaleOrder: IRestaurant | * **Sale Id** * **TimeCountDown** * **SaleDateTime** * **SumOfPizzaOrders** * **CountOfPizzaOrders** | * **Implements IRestaurant** * **Sales KEEP TRACK of Pizzas** * **No Inheritance here** |

NOTE: Components of Pizzas should be in tables that only have related items. This minimizes decoupling. Putting all PizzaParts in once Pizza could cause complications like coupling and redundancy.

### Component

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Component** | Public abstract Component: IRestaurant | * **Component Id** * **Component Type** | * **Implements IRestaurant** * **All the toppings ARE Components** |

### Crust

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Crust** | Public abstract class Crust: Component | * **Crust Id** | * **Inherits Directly from Component** * **A publicly accessible class to be access by Pizza.** |

### Sauce

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Sauce** | Public abstract class Sauce: Component | * **Sauce Id** | * **Inherits Directly from Component** * **A publicly accessible class to be access by Pizza.** |

### Cheese

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Cheese** | Public abstract class Cheese: Component | * **Cheese Id** | * **Inherits Directly from Component** * **A publicly accessible class to be access by Pizza.** |

### Vegetable

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Vegetable** | Public abstract class Vegetable: Component | * **Vegetable Id** | * **Inherits Directly from Component** * **A publicly accessible class to be access by Pizza.** |

### Meat

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Meat** | Public abstract class Meat: Component | * **Meat Id** | * **Inherits Directly from Component** * **A publicly accessible class to be access by Pizza.** |

### Pizza

* [required] should be able to have a crust
* [required] should be able to have a size
* [required] should be able to compute its cost
* [optional] should be able to have at least 2 default toppings, including cheese and sauce
* [optional] should be able to limit its toppings to no more than 5

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Pizza** | Public class Pizza: IRestaurant | * **Pizza ID** * **Crust Type** * **Size** * **Price** | * **Inherits Implements IPizzaOrder and indirectly implements IRestaurant** * **Pizza has PresetPizzas** |

### PresetPizza NO TABLE NEEDED HERE. USE BL.

### Inventory

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Class** | **Fields** | **Notes** |
| **Inventory** | Class Inventory | * **Inventory Id** * **(implements fields from Restaurant)** * **lbsOfCrustBatter int** * **lbsOfCheese int** * **lbsOfSauce** * **lbsOfCrust** * **lbsOfMeat** * **lbsofVegetables** | * **Implements IRestaurant** |

## technologies

* .NET Core - C#
* .NET Core - EF / Serializer
* .NET Core - xUnit

## timelines

### Jan-22

## showcase (as many as you can implement)

* [required] as a user i should be able to signin
* [required] as a user i should be able to view a list of locations
* [required] as a user i should be able to select a location
* [required] as a user i should be able to make an order
* [required] as a user i should be able to choose preset pizza(s)
* [required] as a user i should be able to select a crust
* [required] as a user i should be able to select a size
* [required] as a user i should be able to view my order history
* [required] as a user i should be able to signout
* [required] as a store i should be able to view my order history
* [optional] as a user i should be able to register
* [optional] as a user i should be able to choose custom pizza(s)
* [optional] as a user i should be able to select a set of toppings
* [optional] as a user i should be able to preview my order
* [optional] as a user i should be able to confirm my order
* [optional] as a store i should be able to view my sales by day and by month history