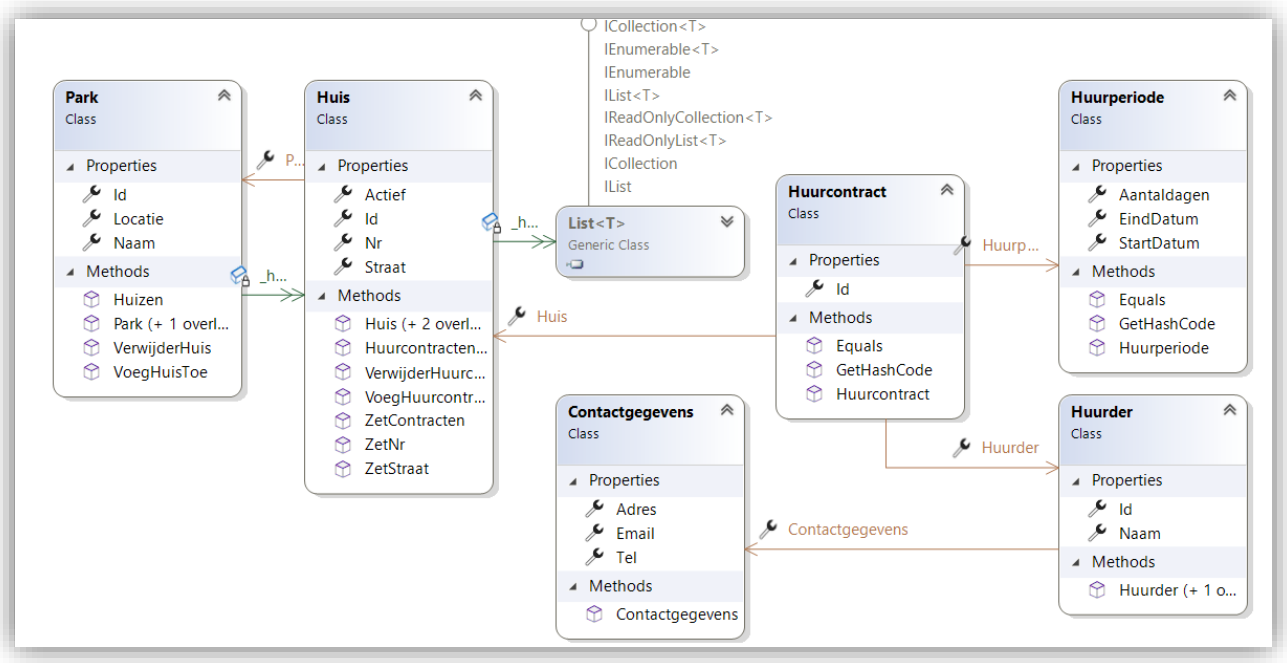
# Opgave bouwen van een datalaag met Entity Framework Core

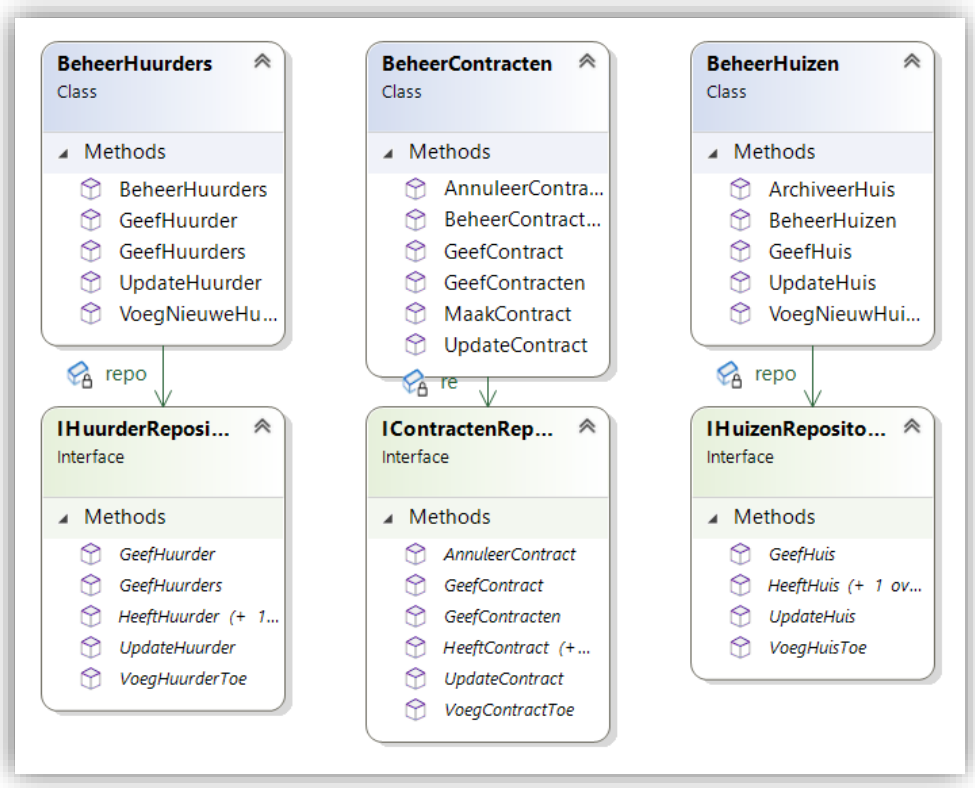
## De businesslaag

Het domeinmodel ziet er als volgt uit :

We hebben een (vakantie)park dat verschillende Huizen kan bevatten en waarbij elk huis ook weet tot welk park het behoort. Elk van deze huizen kan verhuurt worden voor een bepaalde periode. Om deze huur te beschrijven hebben we een klasse Huurcontract die enerzijds verwijst naar een Huis, maar ook naar een Huurder en een Huurperiode.  De klasse Huis houdt ook de Huurcontracten bij en doet dat in een dictionary waarbij de sleutel de Huurder is en de Value een lijst van contracten die bij deze hurder horen (voor dit huis).

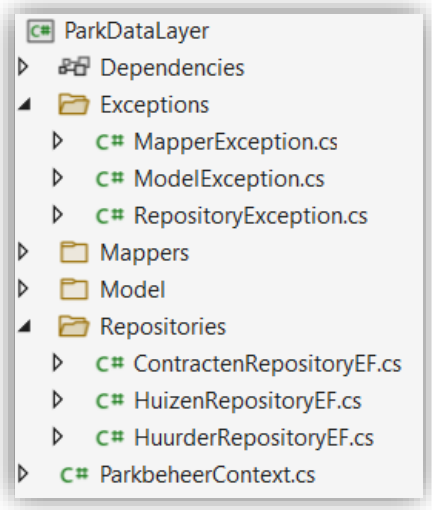



Het beheer van de verschillende entiteiten gebeurt door de klassen BeheerHuurders, BeheerContracten en BeheerHuizen die telkens gebruik maken van een aantal repositories uit de datalaag die door middel van de volgende interfaces zijn beschreven : IHuurderRepository, IHuizenRepository en IContractenRepository.




## De datalaag

De opgave bestaat er nu uit om de datalaag op te bouwen en daarin onderscheiden we drie verschillende taken. De structuur van de datalaag (class library) moet er als volgt uuit zien. Je maakt folders aan voor het model, de repositories, mappers en exceptions. Voor de exceptions zullen we ons beperken tot 3 exceptions, zo is er slechts één exception voor de verschillende repositories en ook maar één exception voor de mappers en het model.



## Datamodel

De verschillende entiteiten moeten aan de volgende voorwaarden voldoen :

### *Park*

- De primary key (Id) is een string met max lengte 20.
- De naam van het park is een verplicht veld en is maximaal 250 characters lang.
- De locatie is niet verplicht en maximaal 500 characters lang.

### *Huis*

- Heeft een integer Id die door de databank wordt aangemaakt.
- Heeft een veld straat die leeg kan zijn en een maximale lengte van 250 characters.
- Een nummer van het type int die altijd moet worden ingevuld.
- Een veld die aangeeft of het huis nog actief kan worden verhuurd (veld actief van het type bit, moet altijd worden ingevuld).

### *Huurder*

- Beschikt ook over een integer Id die door de databank wordt aangemaakt.
- Heeft een naam die niet null mag zijn en met een maximale lengte van 100.
- Verder zijn er nog contactgegevens (telefoon, email en adres) die elk een maximale lengte van 100 characters mogen hebben.  Geen van deze velden is verplicht.

### *Huurcontract*

- Hier is er een primary key (Id) die bestaat uit een string met maximale lengte 25.
- Er is een start- en eindatum die beiden moeten worden ingevuld.
- Een veld die het aantal dagen weergeeft dat het huis wordt verhuurd (integer en mag ook niet null zijn).

## Repositories

Te implementeren interfaces.

```
public interface IHuizenRepository
{
    2 references
    bool HeeftHuis(string straat, int nummer, Park park);
    2 references
    void VoegHuisToe(Huis h);
    3 references
    bool HeeftHuis(int id);
    3 references
    void UpdateHuis(Huis huis);
    2 references
    Huis GeefHuis(int id);
}
```

```
public interface IHuurderRepository
{
    2 references
    void VoegHuurderToe(Huurder h);
    2 references
    bool HeeftHuurder(string naam, Contactgegevens contact);
    2 references
    bool HeeftHuurder(int id);
    2 references
    void UpdateHuurder(Huurder huurder);
    2 references
    Huurder GeefHuurder(int id);
    2 references
    List<Huurder> GeefHuurders(string naam);
}
```

```
public interface IContractenRepository
{
    2 references
    bool HeeftContract(DateTime startDatum, int huurderid, int huisid);
    2 references
    void VoegContractToe(Huurcontract contract);
    2 references
    void AnnuleerContract(Huurcontract contract);
    2 references
    bool HeeftContract(string id);
    2 references
    void UpdateContract(Huurcontract contract);
    2 references
    Huurcontract GeefContract(string id);
    2 references
    List<Huurcontract> GeefContracten(DateTime dtBegin, DateTime? dtEinde);
}
```

- AnnuleerContract komt er op neer dat het contract moet worden verwijderd.
- Bij GeefContracten moeten de contracten getoont worden die een startdatum hebben groter dan of gelijk aan dtBegin en als er een einddatum wordt meegegeven (dtEinde) die niet null is, dan moet de startdatum van de contracten binnen deze periode liggen.

## Mappers

De laatste taak bestaat erin om een aantal mapper-klassen te voorzien die de datalaag-entiteiten kunnen omzetten naar business-laag entiteiten en omgekeerd.

Het project kunnen jullie vinden op GitHub,

https://github.com/tvdewiel/SolutionParkbeheerOpgave

Het DataLayer project is reeds aangemaakt, samen met de folderstructuur. Ook de exceptions zijn reeds voorzien het is dus aan jullie om het model te maken, de mappers en de repository-klassen aan te vullen.

Jullie moeten via een demo (console-app) de werking kunnen aantonen.

Veel succes

Tom