

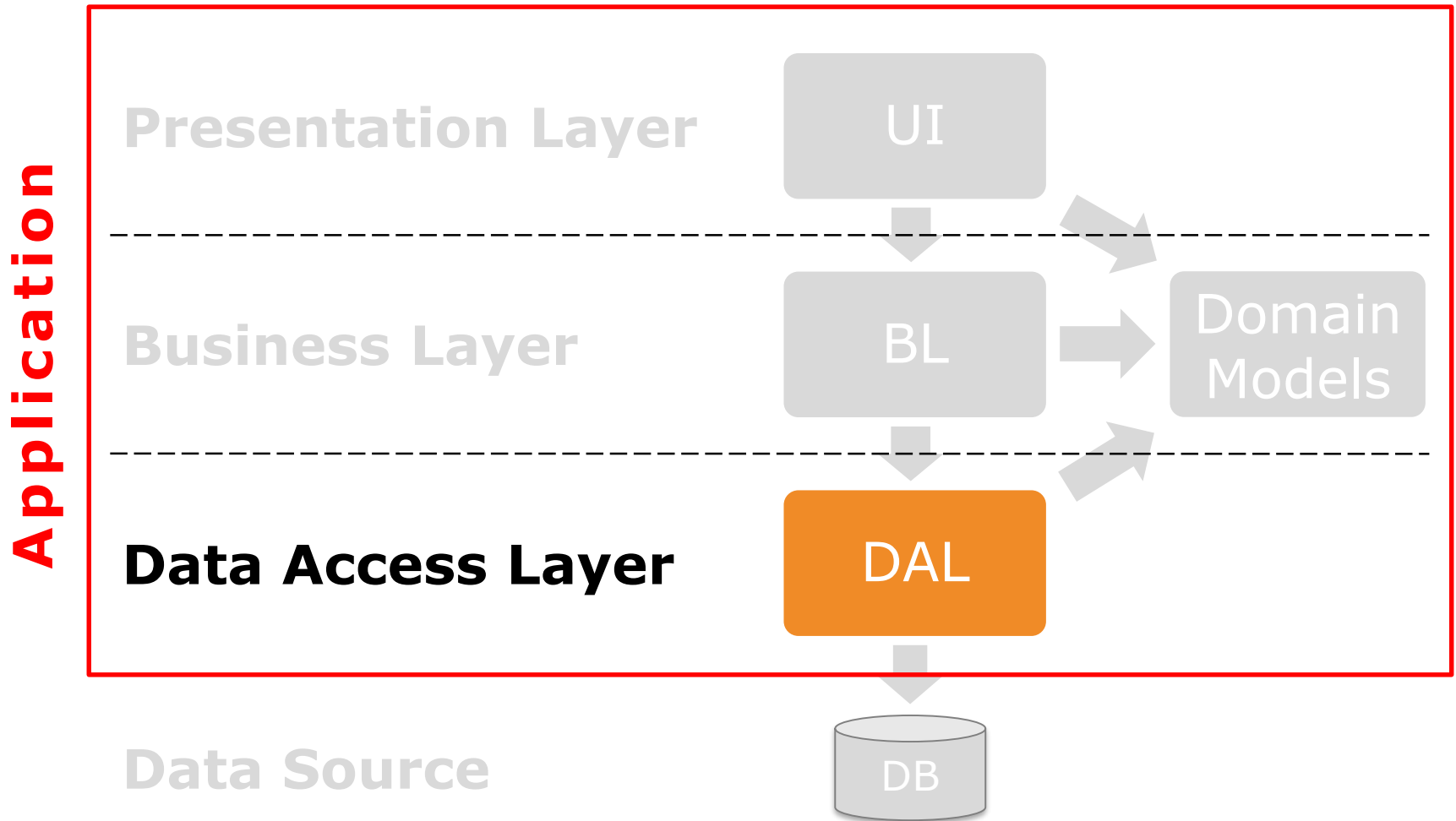
Example 'SupportCenter'

ADO.NET Entity Framework
Code First (New Database)

Info

- De applicatie moet zijn data bijhouden in een databank gebruikmakend van een ORM
- Als ORM-framework kiezen we voor ADO.NET Entity Framework, met als techniek Code First (New Database)
- Als datasource kiezen we voor een SQL Server

N-Tier architectuur



Info 'EF binnen N-Tier'

Voeg het Entity Framework toe

1. Aan welk(e) project(en)?

Volgens de n-tier architectuur is enkel DAL verantwoordelijk voor databank bewerkingen en mag dus enkel het project DAL een 'dependency' hebben op EF

2. Wat met App.config-settings?

- Project DAL is een 'Class Library' en is dus geen applicatie. Deze settings moeten dus in App.config van het project UI-CA voorzien worden. Dit heeft als gevolg dat het project UI-CA moet voorzien worden van het Entity Framework en hierop dus een dependency krijgt.
- Om dit te vermijden kan je, in de plaats van App.config, gebruik maken van de DbConfiguration-klasse binnen het project DAL!

DbContext

Oefening

- project DAL
 - Voeg het Entity Framework toe
 - Voeg **map 'EF'** toe
 - Voorzie een klasse '**TicketRepository**' die de interface 'ITicketRepository' implementeerd (maar werk de methoden nog niet uit)
 - Voorzie ook een EF-container klasse met de naam '**SupportCenterDbContext**' (erft over van 'DbContext')

EF.TicketRepository

```
...
using SC.BL.Domain;

namespace SC.DAL.EF
{
    public class TicketRepository : ITicketRepository
    {
        public Ticket CreateTicket(Ticket ticket)
        {
            throw new NotImplementedException();
        }

        public IEnumerable<Ticket> ReadTickets()
        {
            throw new NotImplementedException();
        }

        ...
    }
}
```

Oefening

- project DAL (map 'EF')
 - Onze repository moet gebruik gaan maken van 'EF Code First'
 - Voorzie hiervoor een EF-container klasse 'SupportCenterDbContext' (erft over van 'DbContext')
 - De repository moet intern gebruik maken van deze container klasse. Voorzie hiervoor een private veld 'ctx' dat in je instantieert via de default constructor
- Access modifier van de container klasse binnen onze N-Tier architectuur?

EF.SupportCenterDbContext

```
...  
using System.Data.Entity;  
  
namespace SC.DAL.EF  
{  
    internal class SupportCenterDbContext : DbContext  
    {  
  
    }  
}
```

EF.TicketRepository

```
...

namespace SC.DAL.EF
{
    public class TicketRepository : ITicketRepository
    {
        private SupportCenterDbContext ctx = null;

        public TicketRepository()
        {
            ctx = new SupportCenterDbContext();
        }

        ...
    }
}
```

Oefening

- Zorg er voor dat de te genereren databank de naam **'SupportCenterDB_EFCodeFirst'** krijgt en wordt aangemaakt op de SQL Server-instantie **'.\SQLSERVER2016'**
- Maak hierbij gebruik van de 'Application Configuration'-file om deze connectionstring (tevens met de naam 'SupportCenterDB_EFCodeFirst') te voorzien

ConnectionString

- Voeg **App.config** toe aan het **project UI-CA** (indien er nog geen is)
 - Voeg connectionstring toe met als naam 'SupportCenterDB_EFCodeFirst'

```
...  
<connectionStrings>  
    ...  
    <add name="SupportCenterDB_EFCodeFirst"  
        connectionString="Data Source=.\SQLSERVER2016;  
                           Initial Catalog=SupportCenterDB_EFCodeFirst;  
                           Integrated Security=True"  
        providerName="System.Data.SqlClient"/>  
</connectionStrings>  
...
```

EF.SupportCenterDbContext

```
...  
using System.Data.Entity;  
  
namespace SC.DAL.EF  
{  
    internal class SupportCenterDbContext : DbContext  
    {  
        public SupportCenterDbContext() : base("SupportCenterDB_EFCodeFirst")  
        {  
        }  
    }  
}
```

DbConfiguration

Oefening

- Voorzie een EF-configuration klasse zodat de configuratie via code kan gebeuren ipv via de application configuration file
 - Waarom?

Om te vermijden dat er in onze N-Tier architectuur voor het project UI-CA een referentie naar EF moet gelegd worden

EF.SupportCenterDbConfiguration

```
...
using System.Data.Entity;

namespace SC.DAL.EF
{
    internal class SupportCenterDbConfiguration : DbConfiguration
    {
        public SupportCenterDbConfiguration()
        {
            this.SetDefaultConnectionFactory(
                new System.Data.Entity.Infrastructure.SqlConnectionFactory());

            this.SetProviderServices("System.Data.SqlClient",
                System.Data.Entity.SqlServer.SqlProviderServices.Instance);
        }
    }
}
```


EF.SupportCenterDbContext

```
...  
using System.Data.Entity;  
  
namespace SC.DAL.EF  
{  
    [DbConfigurationType(typeof(SupportCenterDbConfiguration))]  
    internal class SupportCenterDbContext : DbContext  
    {  
        public SupportCenterDbContext() : base("SupportCenterDB_EFCodeFirst")  
        {  
        }  
    }  
}
```

Models/Entities

Oefening

- Zorg dat de domain models van de applicatie gebruikt worden als entiteiten voor EF
- 'unique identifiers' van models?
 - Ticket/HardwareTicket
 - TicketNumber: geen conventie, gebruik 'KeyAttribute'
 - TicketResponse
 - Id: door conventie

EF.SupportCenterDbContext

```
...
using System.Data.Entity;
using SC.BL.Domain;

namespace SC.DAL.EF
{
    [DbConfigurationType(typeof(SupportCenterDbConfiguration))]
    internal class SupportCenterDbContext : DbContext
    {
        public SupportCenterDbContext() : base("SupportCenterDB_EFCodeFirst")
        {
        }

        public DbSet<Ticket> Tickets { get; set; }
        public DbSet<HardwareTicket> HardwareTickets { get; set; }
        public DbSet<TicketResponse> TicketResponses { get; set; }
    }
}
```

Ticket

```
...  
using System.ComponentModel.DataAnnotations;  
  
namespace SC.BL.Domain  
{  
    public class Ticket  
    {  
        [Key]  
        public int TicketNumber { get; set; }  
  
        ...  
    }  
}
```

Initializer

Oefening

- Voorzie een database initializer 'SupportCenterDbInitializer' voor SupportCenterDbContext, die ervoor zorgt dat de databank wordt verwijderd en terug aangemaakt als een model wordt gewijzigd
 - erft over van 'DropCreateDatabaseIfModelChanges'

EF.SupportCenterDbInitializer

```
...  
using System.Data.Entity;  
...  
  
namespace SC.DAL.EF  
{  
    internal class SupportCenterDbInitializer  
        : DropCreateDatabaseIfModelChanges<SupportCenterDbContext>  
    {  
  
    }  
}
```


EF.SupportCenterDbConfiguration

```
...
using System.Data.Entity;

namespace SC.DAL.EF
{
    internal class SupportCenterDbConfiguration : DbConfiguration
    {
        public SupportCenterDbConfiguration()
        {
            this.SetDefaultConnectionFactory(
                new System.Data.Entity.Infrastructure.SqlConnectionFactory());

            this.SetProviderServices("System.Data.SqlClient",
                System.Data.Entity.SqlServer.SqlProviderServices.Instance);

            this.SetDatabaseInitializer<SupportCenterDbContext>(
                new SupportCenterDbInitializer());
        }
    }
}
```

Oefening

- Voorzie bij initialisatie volgende dummy data:
 - ticket (met TicketNumber '1')
 - "Ik kan mij niet aanmelden op de webmail"
 - Account '1'
 - 09/09/2012 13:05:59
 - 'Closed'
 - Responses
 - response met Id '1' door de helpdesk
 - » "Account is geblokkeerd"
 - » 09/09/2012 13:24:48
 - response met Id '2' door de helpdesk
 - » "Account terug in orde en nieuw paswoord ingesteld"
 - » 09/09/2012 13:29:11
 - response met Id '3' door klant
 - » "Aanmelden gelukt en paswoord gewijzigd"
 - » 10/09/2012 07:22:36

Oefening


- ticket (met TicketNumber '2')
 - "Geen internetverbinding"
 - Account '1'
 - 05/11/2012 09:45:13
 - 'Answerd'
 - Responses
 - response met Id '4' door de helpdesk
 - » "Controleer of de kabel goed is aangesloten"
 - » 05/11/2012 11:25:42
- hardwareticket (met TicketNumber '3')
 - "Blue screen!" op toestel "PC-123456"
 - Account '2'
 - 14/12/2012 19:05:02
 - 'Open'

EF.SupportCenterDbInitializer

```
...
namespace SC.DAL.EF
{
    internal class SupportCenterDbInitializer
        : DropCreateDatabaseIfModelChanges<SupportCenterDbContext>
    {
        protected override void Seed(SupportCenterDbContext context)
        {
            // Create first ticket with three responses
            Ticket t1 = new Ticket()
            {
                AccountId = 1,
                Text = "Ik kan mij niet aanmelden op de webmail",
                DateOpened = new DateTime(2012, 9, 9, 13, 5, 59),
                State = TicketState.Open,
                Responses = new List<TicketResponse>()
            };
            context.Tickets.Add(t1);
        }
    }
}
```




EF.SupportCenterDbInitializer




```
TicketResponse t1r1 = new TicketResponse()
{
    Ticket = t1,
    Text = "Account was geblokkeerd",
    Date = new DateTime(2012, 9, 9, 13, 24, 48),
    IsClientResponse = false
};
t1.Responses.Add(t1r1);

TicketResponse t1r2 = new TicketResponse()
{
    Ticket = t1,
    Text = "Account terug in orde en nieuw paswoord ingesteld",
    Date = new DateTime(2012, 9, 9, 13, 29, 11),
    IsClientResponse = false
};
t1.Responses.Add(t1r2);
```




EF.SupportCenterDbInitializer



```
TicketResponse t1r3 = new TicketResponse()
{
    Ticket = t1,
    Text = "Aanmelden gelukt en paswoord gewijzigd",
    Date = new DateTime(2012, 9, 10, 7, 22, 36),
    IsClientResponse = true
};
t1.Responses.Add(t1r3);
t1.State = TicketState.Closed;

// Create second ticket with one response
Ticket t2 = new Ticket()
{
    AccountId = 1,
    Text = "Geen internetverbinding",
    DateOpened = new DateTime(2012, 11, 5, 9, 45, 13),
    State = TicketState.Open,
    Responses = new List<TicketResponse>()
};
context.Tickets.Add(t2);
```



EF.SupportCenterDbInitializer



```
TicketResponse t2r1 = new TicketResponse()
{
    Ticket = t2,
    Text = "Controleer of de kabel goed is aangesloten",
    Date = new DateTime(2012, 11, 5, 11, 25, 42),
    IsClientResponse = false
};
t2.Responses.Add(t2r1);
t2.State = TicketState.Answered;

// Create hardware ticket without response
HardwareTicket ht1 = new HardwareTicket()
{
    AccountId = 2,
    Text = "Blue screen!",
    DateOpened = new DateTime(2012, 12, 14, 19, 5, 2),
    State = TicketState.Open,
    DeviceName = "PC-123456"
};
context.Tickets.Add(ht1);

// Save the changes in the context (all added entities) to the database
context.SaveChanges();
}
...
}
```

Repository

Oefening

- Zorg er nu voor dat de applicatie gebruik maakt van het Entity Framework CodeFirst
 - wijzig in de constructor van 'TicketManager' het initialiseren van het veld 'repo' naar een object van het type **'SC.DAL.EF.TicketRepository'**

'TicketManager.cs'

```
...
using SC.BL.Domain;
using SC.DAL;

namespace SC.BL
{
    public class TicketManager : ITicketManager
    {
        private readonly ITicketRepository repo;

        public TicketManager()
        {
            //repo = new TicketRepositoryHC();
            //repo = new SC.DAL.SqlClient.TicketRepository();
            repo = new SC.DAL.EF.TicketRepository();
        }

        ...
    }
}
```

Oefening

- Ondanks dat de logica van de repository 'EF.TicketRepository' nog niet voorzien is, is alles betreffende het Entity Framework wel uitgewerkt
- Plaats een breakpoint bij 'ReadTickets' van 'EF.TicketRepository' en start de applicatie
 - Kies menu-item '1) Toon alle tickets'
- Bekijk de databank via SQL Server Management Studio
 - De databank bestaat nog niet!!
- Ondanks dat 'EF.SupportCenterDbContext' al geïnstantieerd is (zie veld 'ctx'), worden EF-initializers pas aangeroepen bij de eerste actie die op de databank uitgevoerd moet worden

Oefening

- Zorg er voor dat de initialisatie van de databank wordt afgedwongen zonder dat er een actie op de databank moet uitgevoerd worden
 - de initialisatie moet enkel gebeuren indien nog niet eerder uitgevoerd

EF.TicketRepository

```
...
using System.Data.Entity;
using SC.BL.Domain;

namespace SC.DAL.EF
{
    public class TicketRepository : ITicketRepository
    {
        private SupportCenterDbContext ctx;

        public TicketRepository()
        {
            ctx = new SupportCenterDbContext();
            ctx.Database.Initialize(false);
        }

        ...
    }
}
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'ReadTickets'

ReadTickets

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public IEnumerable<Ticket> ReadTickets()
    {
        IEnumerable<Ticket> tickets = ctx.Tickets.AsEnumerable();
        return tickets;
    }
    ...
}
```

```
Keuze: 1
[1] Ik kan mij niet aanmelden op de webmail <0 antwoorden>
[2] Geen internetverbinding <0 antwoorden>
[3] Blue screen! <0 antwoorden>
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'ReadTicket'

ReadTicket

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public Ticket ReadTicket(int ticketNumber)
    {
        Ticket ticket = ctx.Tickets.Find(ticketNumber);
        return ticket;
    }
    ...
}
```

```
Keuze: 2
Ticketnummer: 1
Ticket      : 1
Gebruiker   : 1
Datum       : 09/09/2012
Status      : Closed
Vraag/probleem : Ik kan mij niet aanmelden op de webmail
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'CreateTicket'

CreateTicket

```
...
public class TicketRepository : ITicketRepository
{
    public Ticket CreateTicket(Ticket ticket)
    {
        ctx.Tickets.Add(ticket);
        ctx.SaveChanges();

        return ticket; // 'TicketNumber' has been created by the database!
    }

    ...
}
```

```
Keuze: 4
Is het een hardware probleem (j/n)? n
Gebruikersnummer: 1
Probleem: mailbox configuratie in Outlook is verdwenen
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'UpdateTicket'

UpdateTicket

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public void UpdateTicket(Ticket ticket)
    {
        // Make sure that 'ticket' is known by context
        // and has state 'Modified' before updating to database
        ctx.Entry(ticket).State = System.Data.Entity.EntityState.Modified;
        ctx.SaveChanges();
    }
    ...
}
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'UpdateTicketStateToClosed'

UpdateTicketStateToClosed

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public void UpdateTicketStateToClosed(int ticketNumber)
    {
        Ticket ticket = ctx.Tickets.Find(ticketNumber);
        ticket.State = TicketState.Closed;
        ctx.SaveChanges();
    }
    ...
}
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'DeleteTicket'

DeleteTicket

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public void DeleteTicket(int ticketNumber)
    {
        Ticket ticket = ctx.Tickets.Find(ticketNumber);
        ctx.Tickets.Remove(ticket);
        ctx.SaveChanges();
    }
    ...
}
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'ReadTicketResponsesOfTicket'

ReadTicketResponsesOfTicket

```
...
public class TicketRepository : ITicketRepository
{
    ...
    public IEnumerable<TicketResponse> ReadTicketResponsesOfTicket
                                   (int ticketNumber)
    {
        IEnumerable<TicketResponse> responses = ctx.TicketResponses
            .Where(r => r.Ticket.TicketNumber == ticketNumber)
            .AsEnumerable();

        return responses;
    }
    ...
}
```

```
Keuze: 3
Ticketnummer: 1
09/09/2012 Account was geblokkeerd
09/09/2012 Account terug in orde en nieuw paswoord ingesteld
10/09/2012 Aanmelden gelukt en paswoord gewijzigd (client)
```

Oefening

- Zorg in 'EF.TicketRepository' voor de implementatie van de methode 'CreateTicketResponse'

CreateTicketResponse

```
...  
public class TicketRepository : ITicketRepository  
{  
  
    ...  
    public TicketResponse CreateTicketResponse(TicketResponse response)  
    {  
        ctx.TicketResponses.Add(response);  
        ctx.SaveChanges();  
  
        return response; // 'Id' has been created by the database!  
    }  
    ...  
}
```

Geassocieerde data

Oefening

- Zorg ervoor dat als alle tickets worden opgevraagd in de console, ook het aantal antwoorden op het ticket getoond wordt
 - Probeer volgende uitwerkingen:
 - Eager-loading
 - Lazy-loading

Eager-loading

```
...
using System.Data.Entity;

...
public class TicketRepository : ITicketRepository
{
    ...
    public IEnumerable<Ticket> ReadTickets()
    {
        IEnumerable<Ticket> tickets = ctx.Tickets
            .Include(t => t.Responses)
            .AsEnumerable();

        return tickets;
    }
    ...
}
```

```
Keuze: 1
[1] Ik kan mij niet aanmelden op de webmail <3 antwoorden>
[2] Geen internetverbinding <1 antwoorden>
[3] Blue screen! <0 antwoorden>
```


Lazy-loading

```
...  
public class Ticket  
{  
    ...  
    public virtual ICollection<TicketResponse> Responses { get; set; }  
}
```

OPGELET: Standaard kan er in één connectie maar één DataReader te gelijk actief zijn!!

```
...  
public class TicketRepository : ITicketRepository  
{  
    ...  
    public IEnumerable<Ticket> ReadTickets()  
    {  
        IEnumerable<Ticket> tickets = ctx.Tickets.ToList();  
  
        return tickets;  
    }  
    ...  
}
```

OF
'MultipleActiveResultSets=true'
toevoegen aan de connectionstring

```
Keuze: 1  
[1] Ik kan mij niet aanmelden op de webmail <3 antwoorden>  
[2] Geen internetverbinding <1 antwoorden>  
[3] Blue screen! <0 antwoorden>
```

Database-schema manipulaties

Oefening

- Voorzie volgende manipulaties op de creatie van het DB-schema via de techniek 'Fluent API'
 - Conventies
 - Geen meervouden voor tabelnamen
 - Geen 'cascading delete' op required-relaties

EF.SupportCenterDbContext

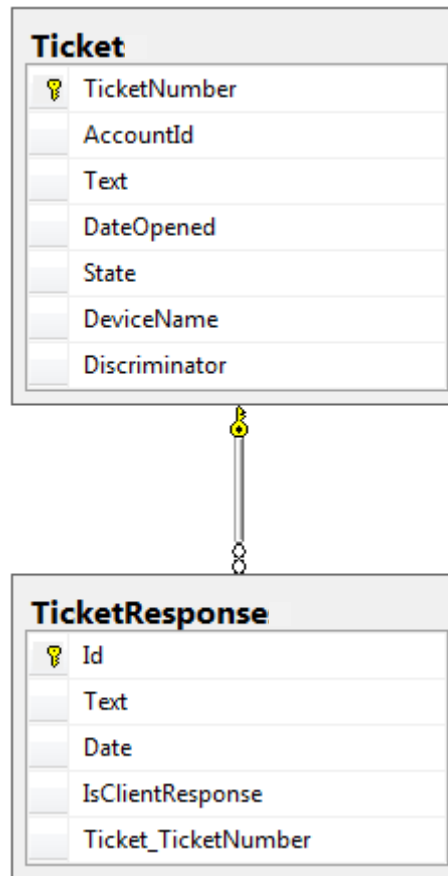
```
...
using System.Data.Entity.ModelConfiguration.Conventions;

namespace SC.DAL.EF
{
    [DbConfigurationType(typeof(SupportCenterDbConfiguration))]
    class SupportCenterDbContext : DbContext
    {
        ...

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            // Remove pluralizing tablenameames
            modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

            // Remove cascading delete for all required-relationships
            modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();
            modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();
        }
    }
}
```

DB-schema



Oefening

- Voorzie volgende manipulaties op de creatie van het DB-schema met 'Data Annotations'
 - Ticket
 - 'State' voorzien van een index


Ticket

```
...  
using System.ComponentModel.DataAnnotations.Schema;
```

```
namespace SC.BL.Domain  
{  
    public class Ticket  
    {  
        ...  
  
        [Index]  
        public TicketState State { get; set; }  
  
        ...  
    }  
}
```

OPGELET: Vereist een referentie naar EF in het Domain-project!

DB-schema

Ticket	
	TicketNumber
	AccountId
	Text
	DateOpened
	State
	DeviceName
	Discriminator

- dbo.Ticket
 - Columns
 - TicketNumber (PK, int, not null)
 - AccountId (int, not null)
 - Text (nvarchar(100), not null)
 - DateOpened (datetime, not null)
 - State (tinyint, not null)
 - DeviceName (nvarchar(max), null)
 - Discriminator (nvarchar(128), not null)
 - Keys
 - PK_dbo.Ticket
 - Constraints
 - Triggers
 - Indexes
 - IX_State (Non-Unique, Non-Clustered)
 - PK_dbo.Ticket (Clustered)

Oefening

- Waarom is 'Data Annotations' geen goede keuze in deze applicatie?
- Welke alternatieve techniek kan je beter gebruiken?

Oefening

- Wijzig de voorziene manipulaties op de creatie van het DB-schema voor 'Ticket' via de techniek 'Fluent API' ipv 'Data Annotations'
 - Ticket
 - 'TicketNumber' als unique identity
 - verwijder KeyAttribute
 - 'State' voorzien van een index
 - verwijder IndexAttribute
- Verwijder ook het 'Entity Framework' uit het project 'Domain'!

Ticket

```
...  
using System.ComponentModel.DataAnnotations.Schema;  
  
namespace SC.BL.Domain  
{  
    public class Ticket  
    {  
        [Key]  
        public int TicketNumber { get; set; }  
  
        ...  
  
        [Index]  
        public TicketState State { get; set; }  
  
        ...  
    }  
}
```

OPGELET: Het 'Key'-attribute is vanuit een domain-model (BL) toepasselijke informatie (metadata), net zoals andere validatie-attributen. Verder bevindt dit attribute zich ook in de assembly van het Validation Framework en niet in de EF-assembly. We zouden deze dus zonder problemen kunnen laten staan!

EF.SupportCenterDbContext

```
...
using System.Data.Entity.ModelConfiguration.Conventions;

namespace SC.DAL.EF
{
    [DbConfigurationType(typeof(SupportCenterDbConfiguration))]
    class SupportCenterDbContext : DbContext
    {
        ...

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            ...

            // 'Ticket.TicketNumber' as unique identifier
            modelBuilder.Entity<Ticket>().HasKey(t => t.TicketNumber);

            // 'Ticket.State' as index
            modelBuilder.Entity<Ticket>().Property(t => t.State)
                .HasColumnAnnotation("Index", new IndexAnnotation(new IndexAttribute()));
        }
    }
}
```