Example 'SupportCenter'

N-Tier Architectuur

15-9-2017

Karel de Grote Hogeschool

Info

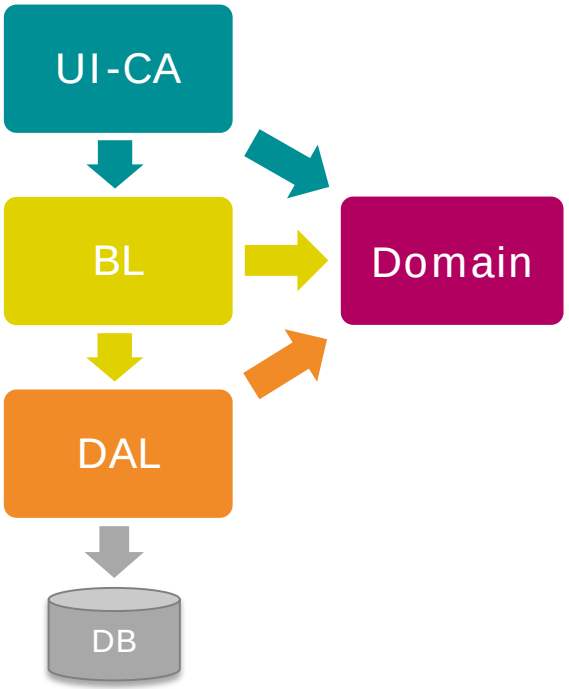
- Maak een applicatie om meldingen van technische problemen te registreren en op te volgen

- Pas N-Tier architectuur toe!

N-Tier Architectuur

Oefening

- Werk de structuur van de solution uit zodat dit overeenkomt met een n-tier architectuur



- solution 'SupportCenter'
- projecten:
  - UI-CA
  - BL
  - Domain
  - DAL

project 'Domain'

- Voorzie een nieuw project 'Domain' in een nieuw solution 'SupportCenter'
  - Type 'Class Library (.NET Framework)'
  - Properties:
    - naam v/d assembly 'SC.BL.Domain'
    - standaard namespace 'SC.BL.Domain'

OPGELET:
Verwijder 'Class1.cs'

project 'Domain'

New Project

- Recent
- Installed
  - Templates
    - Visual C#
      - Windows Classic Desktop
      - Web
      - .NET Core
      - .NET Standard
      - Cloud
      - Test
      - WCF
    - Visual Basic
    - SQL Server
    - Other Project Types

Not finding what you are looking for?
Open Visual Studio Installer

- Online

.NET Framework 4.5.2 | Sort by: Default | Search Installed Templates (Ctrl+E)

WPF App (.NET Framework)	Visual C#
Windows Forms App (.NET Framework)	Visual C#
Console App (.NET Core)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Standard)	Visual C#
Class Library (.NET Framework)	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
ASP.NET Core Web Application (.NET Core)	Visual C#
ASP.NET Core Web Application (.NET Framework)	Visual C#
Shared Project	Visual C#
Class Library (Portable)	Visual C#
WCF Service Application	Visual C#

Type: Visual C#

A project for creating a C# class library (.dll)

Name: Domain

Location: %dotNETDir%

Solution name: SupportCenter

Browse...

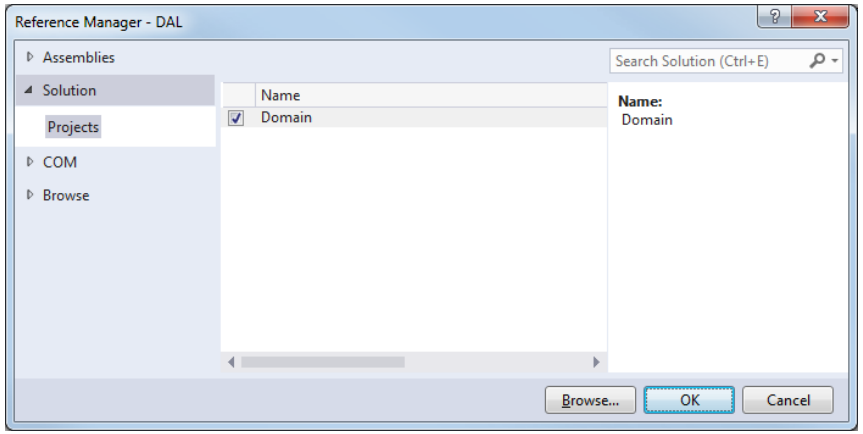
☑ Create directory for solution

☐ Add to Source Control

OK | Cancel

project 'DAL'

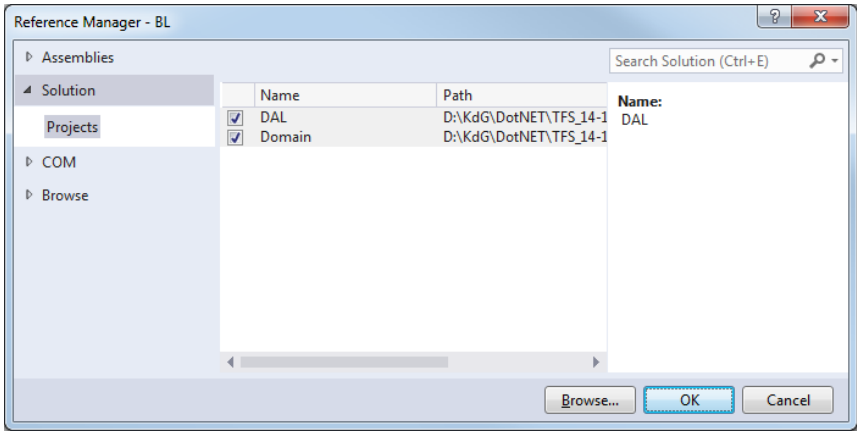
- Voeg een nieuw project 'DAL' toe
  - Type 'Class Library (.NET Framework)'
  - Properties:
    - naam v/d assembly 'SC.DAL'
    - standaard namespace 'SC.DAL'
  - References:
    - project 'Domain'



OPGELET:
Verwijder 'Class1.cs'

project 'BL'

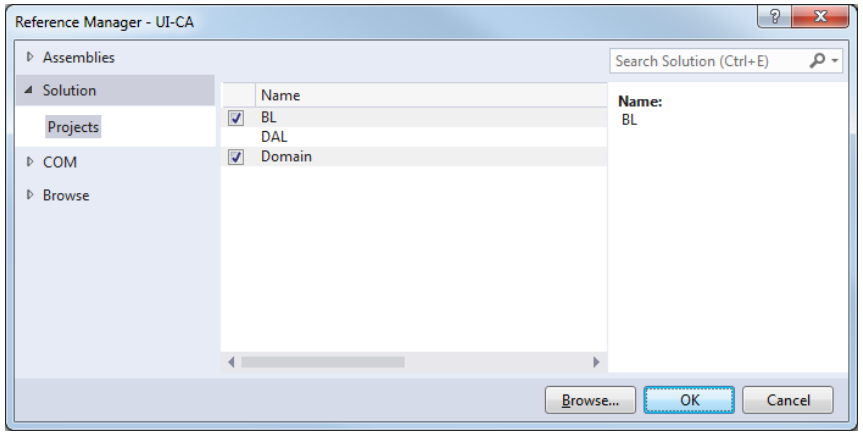
- Voeg een nieuw project 'BL' toe
  - Type 'Class Library (.NET Framework)'
  - Properties:
    - naam v/d assembly 'SC.BL'
    - standaard namespace 'SC.BL'
  - References:
    - project 'DAL'
    - project 'Domain'



OPGELET:
Verwijder 'Class1.cs'

project 'UI-CA'

- Voeg een nieuw project 'UI-CA' toe
  - Type 'Console Application (.NET Framework)'
    - StartUp-project!!
  - Properties:
    - naam v/d assembly 'SC.UI.CA'
    - standaard namespace 'SC.UI.CA'
  - References:
    - project 'BL'
    - project 'Domain'



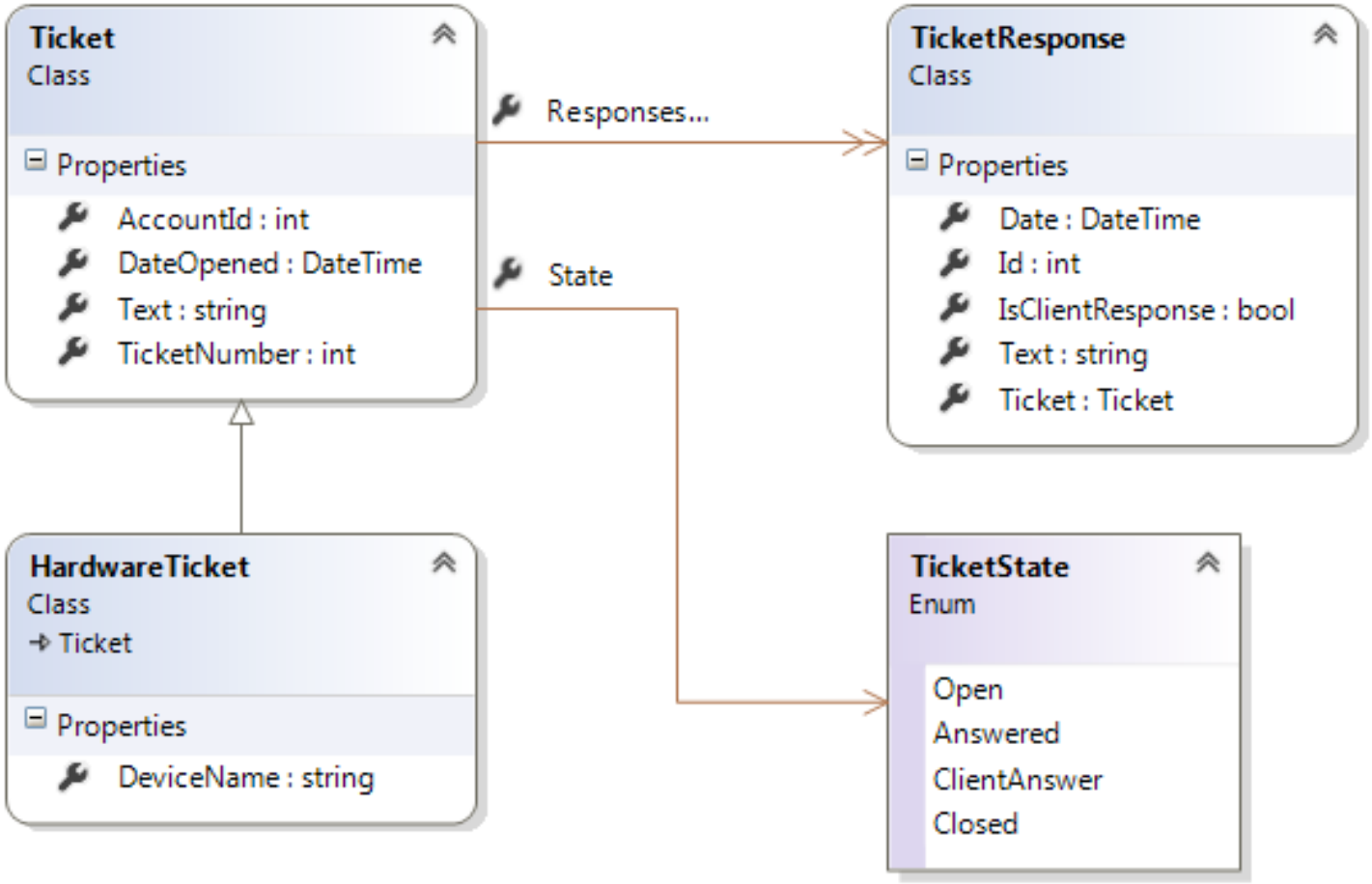
Domain models

project 'Domain'

Domain models

- Voorzie volgende models/entiteiten (zie diagram op de volgende dia)
  - class 'Ticket'
  - class 'HardwareTicket'
  - class 'TicketResponse'
  - enum 'TicketState'

Domain models - Diagram



'Ticket.cs'

```
using System;
...

namespace SC.BL.Domain
{
  public class Ticket
  {
    public int TicketNumber { get; set; }
    public int AccountId { get; set; }
    public string Text { get; set; }
    public DateTime DateOpened { get; set; }
    public TicketState State { get; set; }

    public ICollection<TicketResponse> Responses { get; set; }
  }
}
```

'HardwareTicket.cs'

```
using System;
...

namespace SC.BL.Domain
{
  public class HardwareTicket : Ticket
  {
    public string DeviceName { get; set; }
  }
}
```

'TicketResponse.cs'

```
using System;
...

namespace SC.BL.Domain
{
  public class TicketResponse
  {
    public int Id { get; set; }
    public string Text { get; set; }
    public DateTime Date { get; set; }
    public bool IsClientResponse { get; set; }

    public Ticket Ticket { get; set; }
  }
}
```

'TicketState.cs'

```
using System;
...

namespace SC.BL.Domain
{
  public enum TicketState : byte
  {
    Open = 1,
    Answered,
    ClientAnswer,
    Closed
  }
}
```

Data Access Layer

project 'DAL'
Hardcoded data

Oefening

- Voorzie een interface 'ITicketRepository' voor de koppeling vanuit BL
  - methoden:
    - CreateTicket
      - parameter 'ticket' (type 'Ticket')
      - return 'Ticket'
    - ReadTickets
      - return 'IEnumerable<Ticket>'

'ITicketRepository.cs'

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

namespace SC.DAL
{
  public interface ITicketRepository
  {
    Ticket CreateTicket(Ticket ticket);
    IEnumerable<Ticket> ReadTickets();
  }
}
```

Oefening

- Voorzie een publieke klasse 'TicketRepositoryHC' die de interface 'ITicketRepository' implementeert

- ...HC: hardcoded data!
  - twee private velden voor data:
    - tickets (type List<Ticket>)
    - responses (type List<TicketResponse>)
  - private Seed-methode om dummy data te initializeren
  - default constructor die de Seed-methode aanroept
  - CreateTicket moet ervoor zorgen dat het binnenkomende ticket een uniek ticketnummer krijgt (= het huidig aantal tickets + 1)
  - ReadTickets geeft de huidige lijst van tickets terug

'TicketRepositoryHC.cs'

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

namespace SC.DAL
{
  public class TicketRepositoryHC : ITicketRepository
  {
    private List<Ticket> tickets;
    private List<TicketResponse> responses;

    public TicketRepositoryHC()
    {
      Seed();
    }

    private void Seed()
    {
      // Initialize the basic data of the application and/or some dummy data
    }

    public Ticket CreateTicket(Ticket ticket)
    {
      ticket.TicketNumber = tickets.Count + 1;
      tickets.Add(ticket);
      return ticket;
    }

    public IEnumerable<Ticket> ReadTickets()
    {
      return tickets;
    }
  }
}
```

Oefening

- Voorzie 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'

'TicketRepositoryHC.cs'

```
...
namespace SC.DAL
{
  class TicketRepositoryHC : ITicketRepository
  {
    ...
    private void Seed()
    {
      tickets = new List<Ticket>();
      responses = new List<TicketResponse>();

      // Create first ticket with three responses
      Ticket t1 = new Ticket()
      {
        TicketNumber = tickets.Count + 1,
        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>()
      };
      tickets.Add(t1);
```

'TicketRepositoryHC.cs'

```
TicketResponse t1r1 = new TicketResponse()
{
  Id = responses.Count + 1,
  Ticket = t1,
  Text = "Account was geblokkeerd",
  Date = new DateTime(2012, 9, 9, 13, 24, 48),
  IsClientResponse = false
};
t1.Responses.Add(t1r1);
responses.Add(t1r1);

TicketResponse t1r2 = new TicketResponse()
{
  Id = responses.Count + 1,
  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);
responses.Add(t1r2);
```

'TicketRepositoryHC.cs'

```
TicketResponse t1r3 = new TicketResponse()
{
  Id = responses.Count + 1,
  Ticket = t1,
  Text = "Aanmelden gelukt en paswoord gewijzigd",
  Date = new DateTime(2012, 9, 10, 7, 22, 36),
  IsClientResponse = true
};
t1.Responses.Add(t1r3);
responses.Add(t1r3);
t1.State = TicketState.Closed;

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

'TicketRepositoryHC.cs'

```
      TicketResponse t2r1 = new TicketResponse()
      {
        Id = responses.Count + 1,
        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);
      responses.Add(t2r1);
      t2.State = TicketState.Answered;

      // Create hardware ticket without response
      HardwareTicket ht1 = new HardwareTicket()
      {
        TicketNumber = tickets.Count + 1,
        AccountId = 2,
        Text = "Blue screen!",
        DateOpened = new DateTime(2012, 12, 14, 19, 5, 2),
        State = TicketState.Open,
        DeviceName = "PC-123456"
      };
      tickets.Add(ht1);
    }
    …
  }
}
```

Business Layer

project 'BL'

Oefening

- Voorzie een interface 'ITicketManager' voor de koppeling vanuit UI-CA
  - Methoden
    - GetTickets: om alle tickets op te vragen
      - return 'IEnumerable<Ticket>'
    - AddTicket: om een ticket aan te maken
      - parameters 'accountID' (int), 'question' (string)
      - return 'Ticket'
    - AddTicket: om een hardwareticket aan te maken
      - parameters 'acountId' (int), 'device' (string) en 'problem' (string)
      - return 'Ticket'

'ITicketManager.cs'

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

namespace SC.BL
{
  public interface ITicketManager
  {
    IEnumerable<Ticket> GetTickets();
    Ticket AddTicket(int accountId, string question);
    Ticket AddTicket(int accountId, string device, string problem);
  }
}
```

Oefening

- Voorzie een publieke klasse 'TicketManager' die de interface 'ITicketManager' implementeert
- Voorzie in 'TicketManager' de koppeling met DAL:
  - private veld voor repository:
    - 'repo' (type 'ITicketRepository')
  - default-contructor om 'repo' te initialiseren met een instantie van 'TicketRepositoryHC'

'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();
    }

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

    public Ticket AddTicket(int accountId, string question)
    {
      throw new NotImplementedException();
    }

    public Ticket AddTicket(int accountId, string device, string problem)
    {
      throw new NotImplementedException();
    }
  }
}
```

Oefening

- Werk de methoden van de interface 'ITicketManager' uit:
  - GetTickets
    - vraag alle tickets op via ReadTickets van de repository
  - AddTicket(int accountId, string question)
    - creëer een nieuw Ticket-object en zorg dat dit wordt doorgegeven aan de repository
  - AddTicket(int accountId, string device, string problem)
    - creëer een nieuw HardwareTicket en zorg dat dit wordt doorgegeven aan de reposiroty

'TicketManager.cs'

```
...
namespace SC.BL
{
  pubic class TicketManager : ITicketManager
  {
    ...
    public IEnumerable<Ticket> GetTickets()
    {
      return repo.ReadTickets();
    }

    public Ticket AddTicket(int accountId, string question)
    {
      Ticket t = new Ticket()
      {
        AccountId = accountId,
        Text = question,
        DateOpened = DateTime.Now,
        State = TicketState.Open,
      };
      return this.AddTicket(t);
    }
```

'TicketManager.cs'

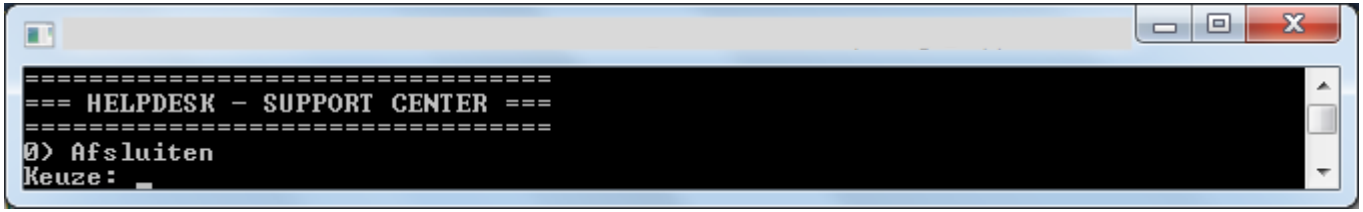
```
    public Ticket AddTicket(int accountId, string device, string problem)
    {
      Ticket t = new HardwareTicket()
      {
        AccountId = accountId,
        Text = problem,
        DateOpened = DateTime.Now,
        State = TicketState.Open,
        DeviceName = device
      };
      return this.AddTicket(t);
    }

    private Ticket AddTicket(Ticket ticket)
    {
      return repo.CreateTicket(ticket);
    }
  }
}
```

Presentation Layer

Console Application

Oefening

- Voorzie in 'Program' de koppeling met BL:
  - private static veld voor manager:
    - 'mgr' (type 'ITicketManager')
  - initialiseer in declaratie met een instantie van 'TicketManager'
- Zorg dat bij het opstarten van de console applicatie volgend menu getoond wordt:



  - de applicatie moet in een oneindige loop blijven zolang men niet kiest voor 'Afsluiten'

'Program.cs'

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

namespace SC.UI.CA
{
  class Program
  {
    private static bool quit = false;
    private static readonly ITicketManager mgr = new TicketManager();

    static void Main(string[] args)
    {
      while (!quit)
        ShowMenu();
    }

    private static void ShowMenu()
    {
      Console.WriteLine("==================================");
      Console.WriteLine("=== HELPDESK - SUPPORT CENTER ===");
      Console.WriteLine("==================================");
      Console.WriteLine("0) Afsluiten");
      DetectMenuAction();
    }
```

'Program.cs'

```
    private static void DetectMenuAction()
    {
      bool inValidAction;
      do
      {
        inValidAction = false;
        Console.Write("Keuze: ");
        string input = Console.ReadLine();
        int action;
        if (Int32.TryParse(input, out action))
        {
          if (action == 0)
            quit = true;
          else
          {
            Console.WriteLine("Geen geldige keuze!");
            inValidAction = true;
          }
        }
      } while (inValidAction);
    }
  }
}
```

Oefening

- Voeg volgende menu-items toe:

```
================================
=== HELPDESK - SUPPORT CENTER ===
================================
1) Toon alle tickets
2) Maak een nieuw ticket
0) Afsluiten
Keuze: _
```

  - Toon alle tickets (keuze '1')

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

    - Maak gebruik van een extension-methode 'GetInfo' (return string, zie screenshot) voor het type 'Ticket'

  - Maak een nieuw ticket (keuze '2')

```
Keuze: 2
Is het een hardware probleem (j/n)? j
Naam van het toestel: PC-987654
Gebruikersnummer: 1
Probleem: Muispad werkt niet meer
```

'ExtensionMethods.cs'

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

namespace SC.UI.CA.ExtensionMethods
{
  internal static class ExtensionMethods
  {
    internal static string GetInfo(this Ticket t)
    {
      return String.Format("[{0}] {1} ({2} antwoorden)"
                           , t.TicketNumber, t.Text
                           , t.Responses == null ? 0 : t.Responses.Count);
    }
  }
}
```

'Program.cs'

```
...
using SC.UI.CA.ExtensionMethods;
...
  class Program
  {
    ...
    private static void ShowMenu()
    {
      ...
      Console.WriteLine("1) Toon alle tickets");
      Console.WriteLine("2) Maak een nieuw ticket");
      Console.WriteLine("0) Afsluiten");
      DetectMenuAction();
    }

    private static void DetectMenuAction()
    {
      ...
        if (Int32.TryParse(input, out action))
        {
          switch (action)
          {
            case 1:
              PrintAllTickets(); break;
            case 2:
              ActionCreateTicket(); break;
            case 0:
              quit = true; return;
            default:
              Console.WriteLine("Geen geldige keuze!");
              inValidAction = true;
              break;
          }
        }
      ...
    }
```

'Program.cs'

```
    private static void PrintAllTickets()
    {
      foreach (var t in mgr.GetTickets())
        Console.WriteLine(t.GetInfo());
    }

    private static void ActionCreateTicket()
    {
      string device = "";
      Console.Write("Is het een hardware probleem (j/n)? ");
      bool isHardwareProblem = (Console.ReadLine().ToLower() == "j");
      if (isHardwareProblem)
      {
        Console.Write("Naam van het toestel: ");
        device = Console.ReadLine();
      }

      Console.Write("Gebruikersnummer: ");
      int accountNumber = Int32.Parse(Console.ReadLine());
      Console.Write("Probleem: ");
      string problem = Console.ReadLine();

      if (!isHardwareProblem)
        mgr.AddTicket(accountNumber, problem);
      else
        mgr.AddTicket(accountNumber, device, problem);
    }
  }
}
```

CRUD-logica

Oefening

- Werk de volledige CRUD-logica verder uit in DAL
  - 'ITicketRepository'

```
public interface ITicketRepository
{
  Ticket CreateTicket(Ticket ticket);
  IEnumerable<Ticket> ReadTickets();
  Ticket ReadTicket(int ticketNumber);
  void UpdateTicket(Ticket ticket);
  void DeleteTicket(int ticketNumber);

  IEnumerable<TicketResponse> ReadTicketResponsesOfTicket(int ticketNumber);
  TicketResponse CreateTicketResponse(TicketResponse response);
}
```

'TicketRepositoryHC.cs'

```
...
    public Ticket ReadTicket(int ticketNumber)
    {
      return tickets.Find(t => t.TicketNumber == ticketNumber);
    }

    public void UpdateTicket(Ticket ticket)
    {
      // Do nothing! All data lives in memory, so everything references the same objects!!
    }

    public void DeleteTicket(int ticketNumber)
    {
      this.responses.RemoveAll(r => r.Ticket.TicketNumber == ticketNumber);
      this.tickets.Remove(ReadTicket(ticketNumber));
    }

    public IEnumerable<TicketResponse> ReadTicketResponsesOfTicket(int ticketNumber)
    {
      return tickets.Find(t => t.TicketNumber == ticketNumber).Responses;
    }

    public TicketResponse CreateTicketResponse(TicketResponse response)
    {
      response.Id = responses.Count + 1;
      responses.Add(response);
      return response;
    }
...
```

Oefening

- Werk de volledige CRUD-logica verder uit in BL
  - 'ITicketManager'

```
public interface ITicketManager
{
  IEnumerable<Ticket> GetTickets();
  Ticket GetTicket(int ticketNumber);
  Ticket AddTicket(int accountId, string question);
  Ticket AddTicket(int accountId, string device, string problem);
  void ChangeTicket(Ticket ticket);
  void RemoveTicket(int ticketNumber);

  IEnumerable<TicketResponse> GetTicketResponses(int ticketNumber);
  TicketResponse AddTicketResponse(int ticketNumber, string response, bool isClientResponse);
}
```

TIP: 'AddTicketResponse' zoekt eerst het ticket adhv het ticketnummer en voegt hier een nieuwe ticketresponse aan toe. Vergeet niet de status van het ticket bij te werken.

'TicketManager.cs'

```
...

namespace SC.BL;
{
  public class TicketManager : ITicketManager
  {
    ...
    public Ticket GetTicket(int ticketNumber)
    {
      return repo.ReadTicket(ticketNumber);
    }

    public void ChangeTicket(Ticket ticket)
    {
      repo.UpdateTicket(ticket);
    }

    public void RemoveTicket(int ticketNumber)
    {
      repo.DeleteTicket(ticketNumber);
    }

    public IEnumerable<TicketResponse> GetTicketResponses(int ticketNumber)
    {
      return repo.ReadTicketResponsesOfTicket(ticketNumber);
    }
```

'TicketManager.cs'

```
public TicketResponse AddTicketResponse(int ticketNumber, string response, bool isClientResponse)
{
  Ticket ticketToAddResponseTo = this.GetTicket(ticketNumber);
  if (ticketToAddResponseTo != null)
  {
    // Create response
    TicketResponse newTicketResponse = new TicketResponse();
    newTicketResponse.Date = DateTime.Now;
    newTicketResponse.Text = response;
    newTicketResponse.IsClientResponse = isClientResponse;
    newTicketResponse.Ticket = ticketToAddResponseTo;

    // Add response to ticket
    var responses = this.GetTicketResponses(ticketNumber);

    if (responses != null)
      ticketToAddResponseTo.Responses = responses.ToList();
    else
      ticketToAddResponseTo.Responses = new List<TicketResponse>();

    ticketToAddResponseTo.Responses.Add(newTicketResponse);
```

'TicketManager.cs'

```
        // Change state of ticket
        if (isClientResponse)
          ticketToAddResponseTo.State = TicketState.ClientAnswer;
        else
          ticketToAddResponseTo.State = TicketState.Answered;

        // Save changes to repository
        repo.CreateTicketResponse(newTicketResponse);
        repo.UpdateTicket(ticketToAddResponseTo);

        return newTicketResponse;
      }
      else
        throw new ArgumentException("Ticketnumber '" + ticketNumber + "' not found!");
    }
  }
}
```

Oefening

- Program (in UI-CA)
  - Bereid het menu verder uit naar:

```
================================
=== HELPDESK - SUPPORT CENTER ===
================================
1) Toon alle tickets
2) Toon details van een ticket
3) Toon de antwoorden van een ticket
4) Maak een nieuw ticket
5) Geef een antwoord op een ticket
0) Afsluiten
Keuze: _
```

- Toon details van een ticket (keuze '2')

```
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

- Toon de antwoorden van een ticket (keuze '3')

```
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)
```

  – Voorzie extra extension-methode 'GetInfo' (return string, zie screenshot) voor het type 'TicketResponse'

- Maak een nieuw ticket (keuze '4' ipv '2')
- Geef een antwoord op een ticket (keuze '5')

```
Keuze: 5
Ticketnummer: 3
Antwoord: Breng het toestel binnen
```

UI-CA: 'Program.cs'

```
...
  class Program
  {
    ...
    private static void ShowMenu()
    {
      ...
      Console.WriteLine("1) Toon alle tickets");
      Console.WriteLine("2) Toon details van een ticket");
      Console.WriteLine("3) Toon de antwoorden van een ticket");
      Console.WriteLine("4) Maak een nieuw ticket");
      Console.WriteLine("5) Geef een antwoord op een ticket");
      Console.WriteLine("0) Afsluiten");
      DetectMenuAction();
    }

    private static void DetectMenuAction()
    {
      ...
            case 1:
              PrintAllTickets(); break;
            case 2:
              ActionShowTicketDetails(); break;
            case 3:
              ActionShowTicketResponses(); break;
            case 4:
              ActionCreateTicket(); break;
            case 5:
              ActionAddResponseToTicket(); break;
      ...
    }
```

UI-CA: 'Program.cs'

```
...

private static void ActionShowTicketDetails()
{
  Console.Write("Ticketnummer: ");
  int input = Int32.Parse(Console.ReadLine());

  Ticket t = mgr.GetTicket(input);
  PrintTicketDetails(t);
}

private static void PrintTicketDetails(Ticket ticket)
{
  Console.WriteLine("{0,-15}: {1}", "Ticket", ticket.TicketNumber);
  Console.WriteLine("{0,-15}: {1}", "Gebruiker", ticket.AccountId);
  Console.WriteLine("{0,-15}: {1}", "Datum", ticket.DateOpened.ToString("dd/MM/yyyy"));
  Console.WriteLine("{0,-15}: {1}", "Status", ticket.State);

  if (ticket is HardwareTicket)
    Console.WriteLine("{0,-15}: {1}", "Toestel", ((HardwareTicket)ticket).DeviceName);

  Console.WriteLine("{0,-15}: {1}", "Vraag/probleem", ticket.Text);
}

...
```

UI-CA: 'Program.cs'

```
  ...
  private static void ActionShowTicketResponses()
  {
    Console.Write("Ticketnummer: ");
    int input = Int32.Parse(Console.ReadLine());

    IEnumerable<TicketResponse> responses = mgr.GetTicketResponses(input);
    if (responses != null) PrintTicketResponses(responses);
  }

  private static void PrintTicketResponses(IEnumerable<TicketResponse> responses)
  {
    foreach (var r in responses)
      Console.WriteLine(r.GetInfo());
  }

  private static void ActionAddResponseToTicket()
  {
    Console.Write("Ticketnummer: ");
    int ticketNumber = Int32.Parse(Console.ReadLine());
    Console.Write("Antwoord: ");
    string response = Console.ReadLine();

    mgr.AddTicketResponse(ticketNumber, response, false);
  }
  ...
}
```

UI-CA: 'ExtensionMethods.cs'

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

namespace SC.UI.CA.ExtensionMethods
{
  internal static class ExtensionMethods
  {
    internal static string GetInfo(this Ticket t)
    {
      return String.Format("[{0}] {1} ({2} antwoorden)", t.TicketNumber, t.Text
                           , t.Responses == null ? 0 : t.Responses.Count);
    }

    internal static string GetInfo(this TicketResponse r)
    {
      return String.Format("{0:dd/MM/yyyy} {1}{2}", r.Date, r.Text
                           , r.IsClientResponse ? " (client)" : "");
    }
  }
}
```