**Scrum**

Produktplanung (Product Backlog) -> Liste von Anforderungen

**Rollen:**  
Kunde: Wünsche, Anforderungen und Ideen

Product Owner: Verwaltet Kundenanforderungen und nimmt Input von allen Seiten

Scrum Master: Konflikte lösen, Entwickler betreuen, implementiert Scrum Prozess

Entwickler: Liefert das Produkt und hält QA-Standards ein

Artefakte: Dokumente, Grafiken usw.

**Backlog:**  
Der Backlog kann jederzeit angepasst werden.

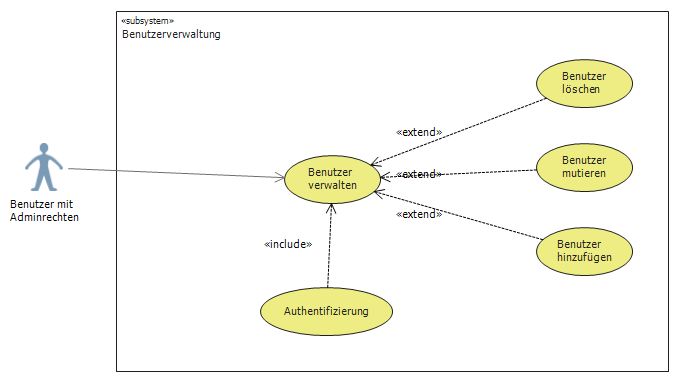
User-Stories: Wer will was und warum?

Prioriät und Aufwand: In Stunden oder Story Points (Planning Poker)

PBI (Product Backlog Item): Aufwand von Entwickler geschätzt

DoD: Definition of Done. Abschlussanforderungen an eine Story. Wird von PO überprüft

**Anwendungsfall-Diagramm (Use-Case)**Elemente: Akteur, Anwendungsfall, Systemgrenze



**TFS (Team Foundation Server)**Vorteile: Ein zentrales Repository, Eine tiefe Integration verschiedener Rollen, Eine offene und flexible Schnittstelle für Anpassungen

Features: Versionskontrolle

TFS-Buid: Die Buildengine des TFS unterstützt das automatische Erstellen des entwickelten Produkts („build“), optional z.B auch mit Dokumentation. Dabei kann man auch Unittests ausführen und Statistiken bzw. Berichte („reports“) generieren lassen.

**Entity Framework**Domänen-Schicht, Datenbankabfragen, Model First, Code First, Database First

**Create Beispiel**

using (var ctx = new CodeFirstMovieDBContext())

{

  ctx.Genres.Add(new Genre { GenreName = "Drama" });

  ctx.Directors.Add(new Director { DirectorName = "Ridley Scott" });

  ctx.Movies.Add(new Movie { MovieName = "Gladiator", PlayTime = 150,  
     DirectorId = 1, GenreId = 2 });

  ctx.SaveChanges();

}

**Read Beispiel**

using (var ctx = new CodeFirstMovieDBContext())

{

  System.Console.WriteLine("Es befinden sich folgende Filme in der DB:");

   var movies = ctx.Movies;

  foreach (Movie movie in movies)

     {

        System.Console.WriteLine(movie.MovieName);

     }

}

**Update Beispiel**

using (var ctx = new CodeFirstMovieDBContext())

{

  System.Console.WriteLine("Es befinden sich folgende Filme in der DB:");

  var movies = ctx.Movies;

  foreach (Movie movie in movies)

  {

     System.Console.WriteLine(movie.MovieName);

  }

  Movie movieToUpdate = (from x in ctx.Movies where x.MovieName == "Pulp

Fiction" select x).First();

  movieToUpdate.MovieName = "Avengers - Age of Ultron";

  ctx.SaveChanges();

  System.Console.WriteLine("Neu befinden sich folgende Filme in der DB:");

  var movies2 = ctx.Movies;

  foreach (Movie movie in movies2)

  {

     System.Console.WriteLine(movie.MovieName);

  }

**LINQ**Bietet mit Language-Integrated Query (LINQ)-Unterstützung die Möglichkeit Queries zu erstellen für die im konzeptionellen Modell definierten Entitätstypen.

**Unit-Testing**

**namespace** Rezeptsammlung.Test

**{**

**[**TestClass**]**

**public** class ValidateRecipeTest

**{**

**[**TestMethod**]**

**public** void CheckRecipeTest**()**

**{**

           // Arrange

           List**<**string**>** errorMessages **=** **new** List**<**string**>();**

           List**<**Rezept\_Zutat**>** recipeIngreds **=** **new** List**<**Rezept\_Zutat**>();**

           List**<**Rezept\_Schlagwort**>** recipeTags **=** **new** List**<**Rezept\_Schlagwort**>();**

           Rezept recipe **=** **new** Rezept**();**

           recipe**.**Titel **=** "Titel"**;**

           recipe**.**Autor **=** "Autor"**;**

           recipe**.**Beschreibung **=** "Beschreibung"**;**

           recipe**.**Quelle **=** "Quelle"**;**

           recipe**.**Schwierigkeit **=** 1**;**

           recipe**.**Herkunftsland\_ID **=** 1**;**

           recipe**.**Kategorie\_ID **=** 1**;**

           recipe**.**Bewertung **=** 10**;**

           recipe**.**Saison\_ID **=** 3**;**

           recipe**.**Dauer **=** 40**;**

           recipeIngreds**.**Add**(new** Rezept\_Zutat **{** Rezept\_ID **=** 1**,** Menge **=** "100g" **});**

           recipeTags**.**Add**(new** Rezept\_Schlagwort **{** Rezept\_ID **=** 1**,** Schlagwort\_ID **=** 1 **});**

           // Act

           ValidateRecipe validator **=** **new** ValidateRecipe**(**recipe**,** recipeIngreds**,** recipeTags**,** **false);**

           errorMessages **=** validator**.**errorMsgs**;**

           // Assert

           Assert**.**AreEqual**(**errorMessages**.**Count**,** 0**);**

**}**

**}**

**}**

**WPF :**

Windows Presentation Foundation -> Grafikframework des .NET-Frameworks, auf XAML aufgebaut

<Label Content=**"Zutat"** HorizontalAlignment=**"Left"** Margin=**"151,404,0,0"** VerticalAlignment=**"Top"**/>

<ComboBox x:Name=**"cboRecipeIngredient"** HorizontalAlignment=**"Left"** Margin=**"231,404,0,0"** VerticalAlignment=**"Top"** Width=**"120"**/>

<Label Content=**"Menge"** HorizontalAlignment=**"Left"** Margin=**"151,435,0,0"** VerticalAlignment=**"Top"**/>

<TextBox x:Name=**"txtRecipeAmount"** HorizontalAlignment=**"Left"** Height=**"23"** Margin=**"231,437,0,0"** TextWrapping=**"Wrap"** VerticalAlignment=**"Top"** Width=**"120"**/>

<ListView x:Name=**"lstAddedIngredient"** HorizontalAlignment=**"Left"** Height=**"110"** Margin=**"413,404,0,0"** VerticalAlignment=**"Top"**

Width=**"506"** IsSynchronizedWithCurrentItem=**"True"** ScrollViewer.HorizontalScrollBarVisibility=**"Disabled"**>

<ListView.View>

<GridView>

<GridViewColumn Header=**"Zutat"**  DisplayMemberBinding=**"{Binding Zutat.ZutatName}"**

Width=**"300"**/>

<GridViewColumn Header=**"Menge"** DisplayMemberBinding=**"{Binding Menge}"** Width=**"206"**/>

</GridView>