## Herhalingsoefening ADO.NET + ASP.NET Identity

## Deel 1

## Doelstellingen

* ASP.NET MVC Identity opzetten en configureren
* Herhaling ADO.NET

## Omschrijving

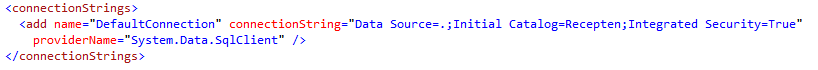
We gaan geen applicatie schrijven waarbij koks recepten kunnen opladen. Deze recepten zitten telkens in een bepaalde categorie. Enkel gebruikers in de “Kok” rol kunnen een recept toevoegen. Iedereen kan de recepten zien, maar commentaar geven kan enkel als gewone gebruiker.

## Database

Maak een nieuwe database aan op je eigen SQL server en noemen deze recepten.

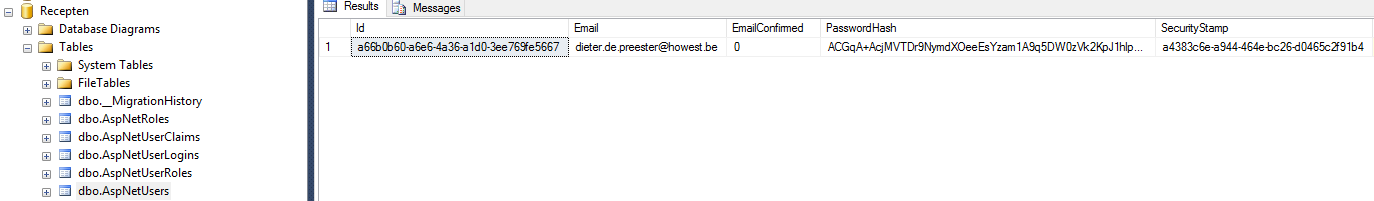
## Connectionstring installeren

Wijzig de connectionstring zodat deze naar je eigen SQL server gaat verwijzen en de recepten database. Waar moeten we dit aanpassen?



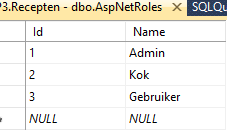
## Registreren Admin + aanmaken tabellen

Om de tabellen van ASP.NET Identity aan te maken moeten we eerst een gebruiker registeren. Registreer de “Admin” gebruiker. Later zullen we deze in een rol koppelen.

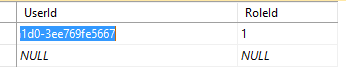


## Toevoegen rollen + koppelen gebruiker

Voeg 3 rollen toe in de tabel “AspNetRoles”.



Open tabel AspNetUsers en maak een kopie van het “Id”. Open tabel AspNetUserRoles en koppel de gebruiker aan de role Admin

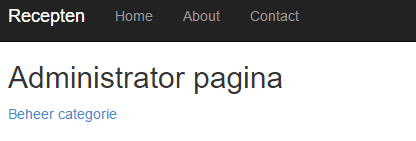


## Models

Maak een database schema en noteer welke tabellen je nodig hebt samen met de relaties. Toon dit aan de docent en na goedkeuring mag u zelf de modellen aanmaken net als de tabellen.

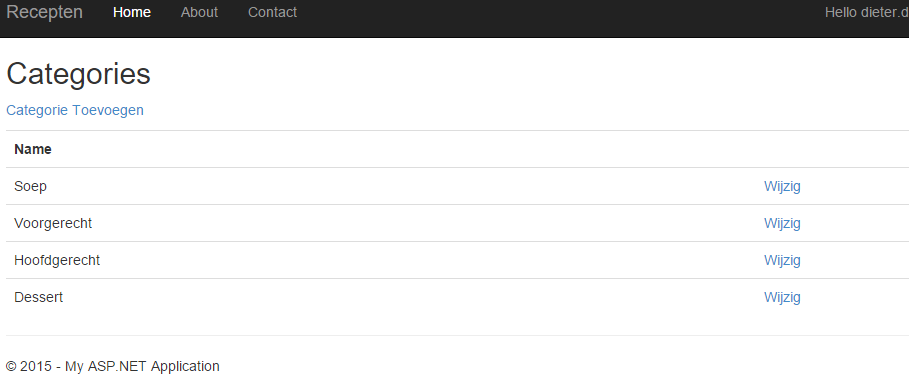
## Admin Controller

Enkel system beheerders mogen een categorie kunnen toevoegen. Voeg een “AdminController” toe. Deze controller zal functies bevatten die voor gebruikers in de “Admin” role. Hoe ga je die beveiligen? Voeg een Index actie toe waarin we de mogelijke admin acties zullen weergeven. Dit zit er in eerste instantie als volgt uit:



## Categorie beheer

De eerste functionaliteit die we gaan toevoegen is voor het beheren van een categorie. Dit scherm zal een overzicht tonen van de aanwezig categorieën. Daarnaast kunnen we nieuwe toevoegen of hernoemen. Verwijderen kan niet.



U zou nu zelf instaat moeten zijn om volgende zaken te maken:

* Repository maken met daarin
  + Toevoegen categorie
  + Ophalen van alle categorieën
  + Wijzigen van een categorie
* Uitwerken van de GUI voor deze functionaliteit

## Registratie kok van restaurant

Er zijn twee soorten registratie mogelijkheden. Ofwel ben je een kok van een restaurant ofwel een gewone gebruiker die commentaar mag geven. Een kok moet opgeven in welk restaurant hij werkt. Het restaurant zit dus eigenlijk in het profiel van de ingelogde gebruiker. We moeten deze dus uitbreiden. Ga naar de map models en zoek daar “IdentityModel.cs” . Daar is de klasse ApplicationUser aanwezig. Deze klasse bevat na inloggen de gebruiker. We kunnen deze dus uitbreiden met de property “Restaurant”.

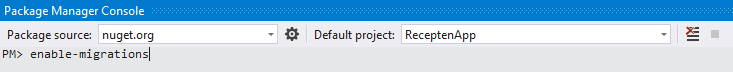


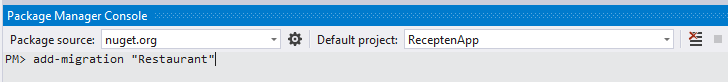
Nu moeten we de database nog aanpassen. Dit moeten we doen via Entity Framework migraties (veel meer hierover in Semester 4).

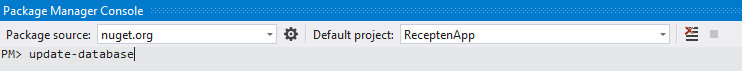
Open de package manager console via:

* Tools 🡺 NuGet Package Manager 🡺 Package Manager Console

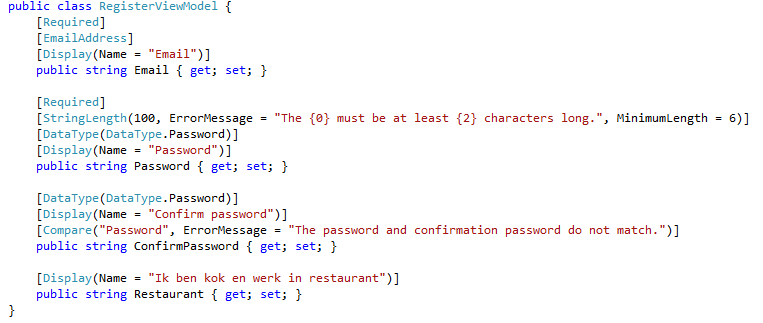
Voer daarna volgende commando’s uit en controleer in de database of de kolom is aangemaakt.

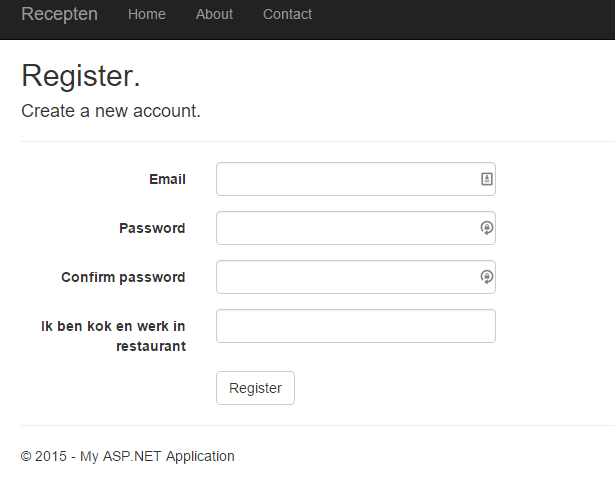






Bij het registreren van een gebruiker moeten we nu de mogelijkheid geven om een restaurant in te vullen. Hiervoor moeten we het login scherm wijzigen. Zoek de file “Register.cshtml” in de map Views/Account. Kijk wel viewmodel men gebruikt. Als u het type van het viewmodel weet dan kan u de klasse vinden in de file “AccountViewModels” in de map “Models”. Open deze file en zoek de klasse “RegisterViewModel”. Nu moeten we deze uitbreiden zodat we de naam van het Restaurant kunnen invullen.



Start de applicatie op en ga naar het registratiescherm dat er nu ongeveer als volgt moet uitzien: 

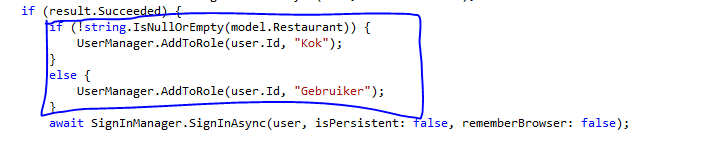
Nu moeten we nog het effectief registeren aanpassen. Alles rond ASP.NET Identity kan u terugvinden in de controller “AccountController”. Open deze en zoek de methode Register.



Wijzig onderstaande code zodat we ook het restaurant mee opslaan in de database.



Daarnaast moeten we ook nog de nieuwe gebruiker in de juiste rol steken. Mensen met een restaurant zitten in de rol “Kok”. Mensen zonder restaurant zijn gewone gebruikers. We maken hiervoor gebruik van de “UserManager” klasse. Bekijk deze klasse goed. Ze bevat veel methodes voor het beheren van gebruikers. Voeg hiervoor volgende code toe in de methode “Register”.



Test daarna de registratie uit en controleer of het restaurant is opgeslagen en of de gebruikers gekoppeld is aan de juiste rol. Maak ook een gebruiker aan die geen kok is.

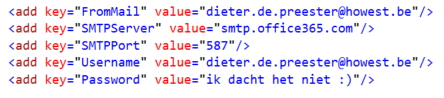
## Validatie E-Mail na registratie

Bij een moderne site gaan we na het registeren altijd een e-mail sturen naar het opgegeven e-mailadres tijdens de registratie. Dit doen we om zeker te zijn dat men geen vals e-mailadres heeft opgegeven. Deze functionaliteit is standaard aanwezig in ASP.NET Identity.

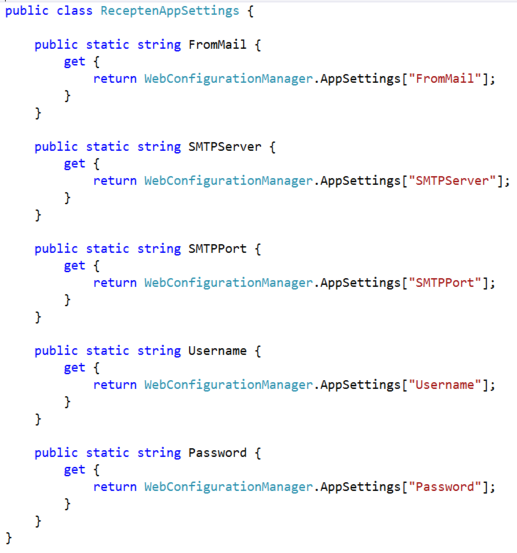
Open de file “IdentityConfig.cs” in de map App\_Start. Deze file bevat code die we kunnen gebruiken voor het configureren van ASP.NET Identity. Zoek de klasse “EmailService”. Dez klasse bevat 1 methode SendAsync. Deze methode zal validatie mails en password reset mails versturen. Om mails te kunnen versturen hebben we volgende instellingen nodig:

* From Mail adres
* SMTP Server
* Poort
* Paswoord

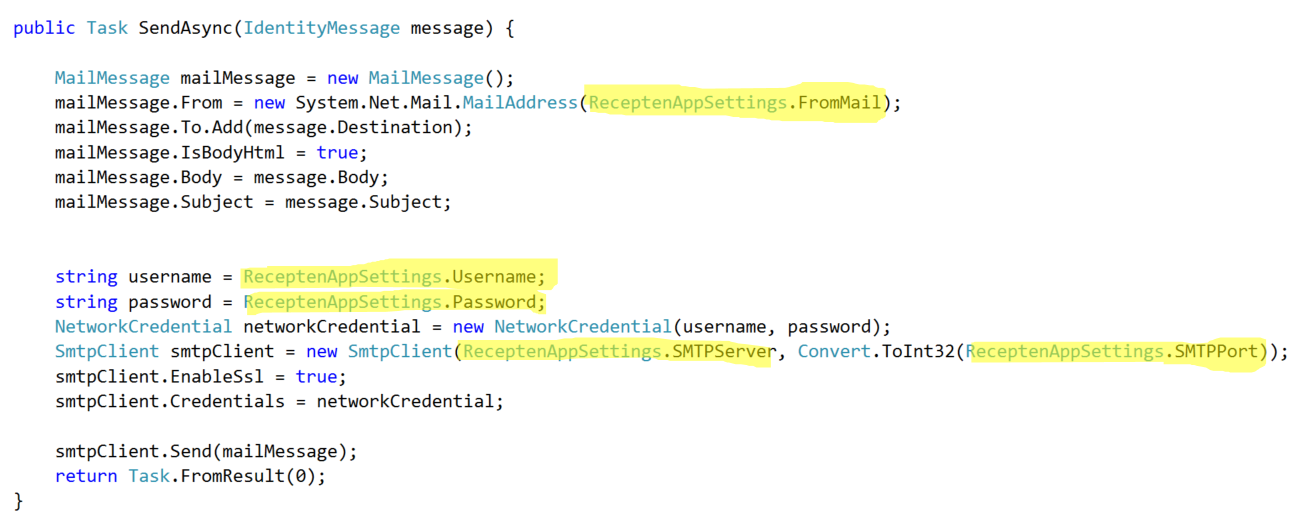
We gaan deze instellingen NIET opslaan in code maar in de web.config. Open web.config en ga naar het onderdeel <appSettings>. Hier gaan we deze configureerbare parameters opslaan. We kiezen hiervoor telkens een unieke key. Via deze key kunnen we de waarden dan opvragen. Wij maken gebruik van de Office 365 SMTP server. Thuis moet u misschien gebruik maken van de SMTP van Telenet of Proximus (of de provider waar u klant bent).



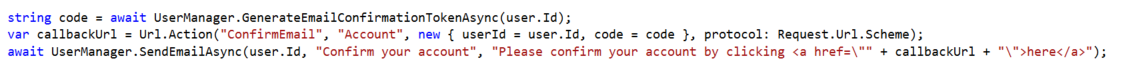
Het opvragen van deze waarden gaan we doen via een centrale klasse “ReceptenAppSettings”. Maak een map Settings aan in het project en voeg een nieuwe C# file toe “ReceptenAppSettings.cs”. Voorzien nu voor iedere op te halen settings een **static** property. In deze property halen we via de klasse WebConfigurationManager klasse uit .NET de juiste setting op. Bij mij zit dit er als volgt uit:



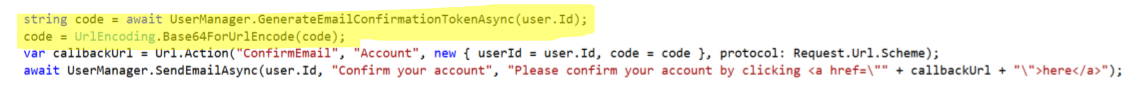
Nu kunnen we de code schrijven die effectief de e-mail zal versturen. Open “EmailService.cs” in de file “IdentityConfig.cs”. Het versturen van e-mails in .NET gaat via de klasse SmtpClient. Deze klasse bevat een methode “Send” die een MailMessage object verwacht. In deze code vragen we de juiste parameters op via onze settings klasse.



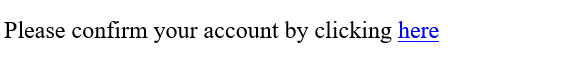
Als laatste moeten we na registratie de mail nog versturen. Open “AccountController.cs” en ga naar de “Register” methode die een HTTPOST doet. Plaatst volgende code UIT commentaar:



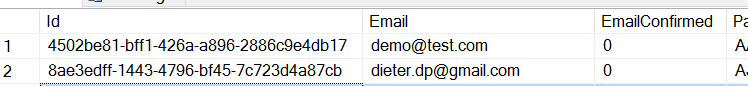
Bovenstaande code is verantwoordelijk voor het versturen van e-mails na registratie. We moeten hier echter oppassen. Er zit een bug in de code bij het genereren van de uniek code waardoor het systeem niet altijd zal werken. We lossen dit op door de gegenereerde code te encoderen. Voeg de klasse “UrlEncoding” toe in de folder “Helpers”. U kan deze klasse downloaden van LEHO. Wijzig nu het versturen van de e-mail als volgt zodat we de code encoderen.



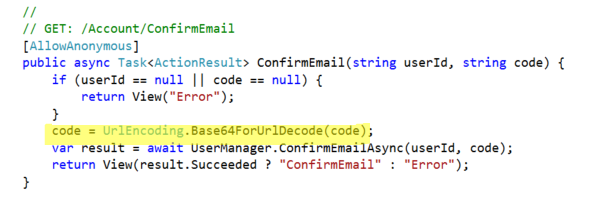
Registreert nu een nieuw gebruiker. Als dit lukt zal u een mail ontvangen die er als volgt uitziet:



In de database ziet dit er als volgt uit, EmailConfirmed is false:



Als je nu zou klikken op de link dan zal je een fout krijgen. We moeten namelijk nog een aanpassing doen in de methode “ConfirmEmail”. Daar moeten we de code terug decoderen.



## Password Reset (Afwerken Thuis)

Met wat u nu weet zou u nu zelf de “Password Reset” functionaliteit moeten kunnen uitwerken. Indien problemen zullen we u met plezier helpen tijdens het labo of in het monitoraat op vrijdag na de theorie.

## Toevoegen recepten

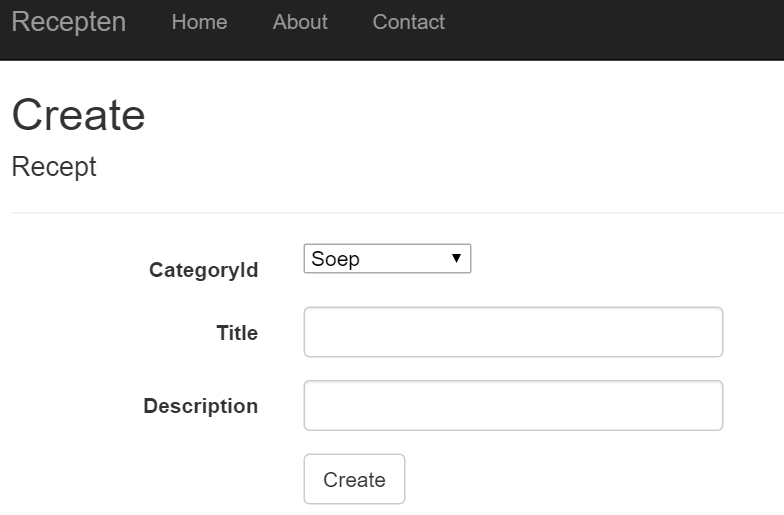
U zou nu zelf instaat moeten zijn om een nieuwe controller + pagina toe toevoegen om recepten aan te maken. U moet er wel voor zorgen dat enkel mensen in de rol “Kok” de recepten kunnen aanmaken.

ReceptController 🡺 Index

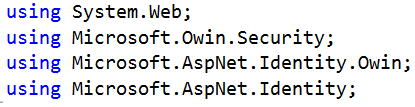
Deze actie moet alle recepten ophalen en weergeven. Zorg ervoor dat de actie alle recepten weergeven de default actie is bij het opstarten van de applicatie. Verwijder nu de HomeController en alle Home Views, we hebben deze niet meer nodig. Opgepast, deze methode mag wel door iedereen aangeroepen worden. Hoe kunnen we de rechten exclusief voor deze methode instellen zodat iedreen de recepten kan zien ?

ReceptController 🡺 Create

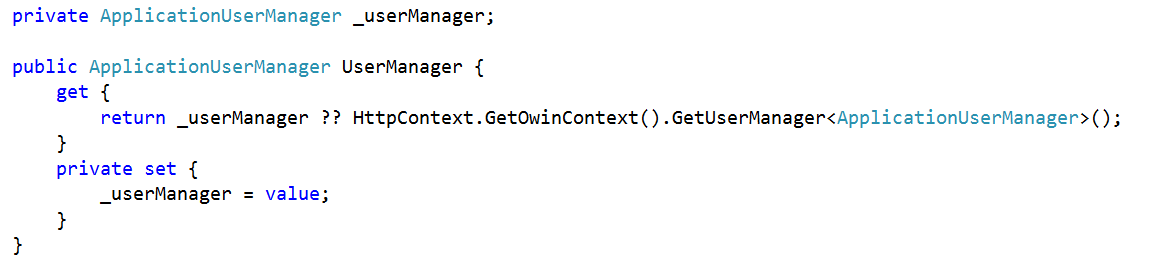
Deze actie mag enkel aangeroepen worden door mensen in de rol “Kok”. De view zal ook iets complexer zijn. De gebruiker moet een recept aanmaken en koppelen aan een categorie. De UI ziet er als volgt uit:



Schrijf een tweede methode die de POST van de form zal opvangen en het recept zal wegschrijven naar de database. Voordat we kunnen wegschrijven naar de database moeten we eerst het ID opvragen van de gebruiker die het recept wil toevoegen. Hiervoor moeten we praten met het ASP.NET Identity Systeem. Voeg volgende namespaces toe:

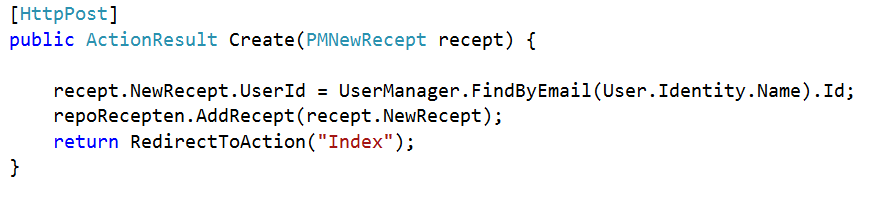


Daarna voegen we volgende private field en public property toe (je kan deze overnemen uit de AccountController):



Via de UserManager hebben we toegang tot een aantal methodes waarmee we kunnen communiceren met ASP.NET Identity.

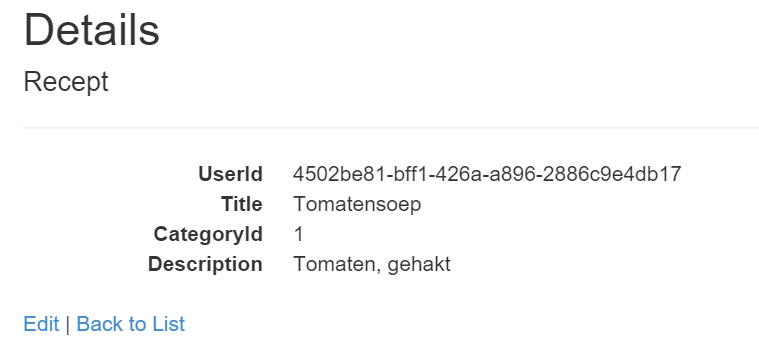
Voor het wegschrijven naar de database kunnen we nu het ID van de ingelogde gebruiker ophalen via UserManager.FindByEmail(username…). De huidige gebruiker kunnen we opvragen via de static klasse “User”. Na het wegschrijven doen we een redirect naar de Index pagina waardoor we in het overzicht van de recepten komen.



Het effectief wegschrijven van het recept zou u nu zelf moeten kunnen. Dit hebben we reeds een aantal keer gedaan.

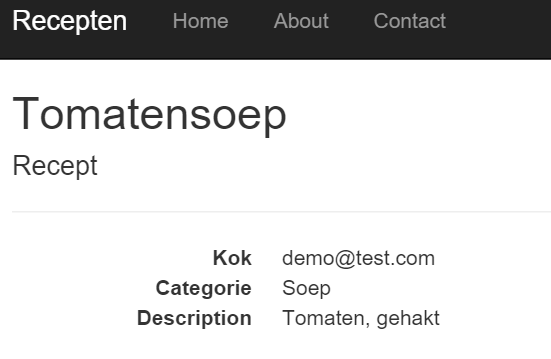
## Details Recept

In het overzicht scherm bij recepten staat er een link “Details”. Bij het klikken moeten we het volledig recept ophalen. Dit zal er in eerste instantie als volgt uitzien:



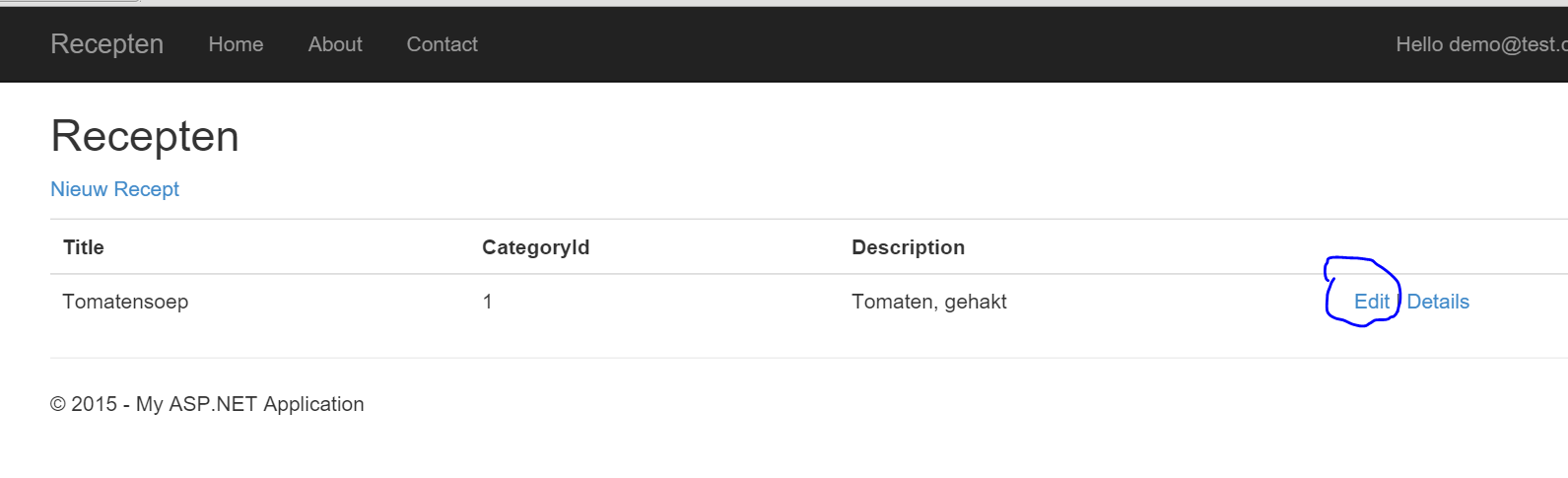
Wijzig het detail scherm zodat dit er als volgt uitziet:

* De naam van de kok die het recept heeft gemaakt
* De naam van de categorie

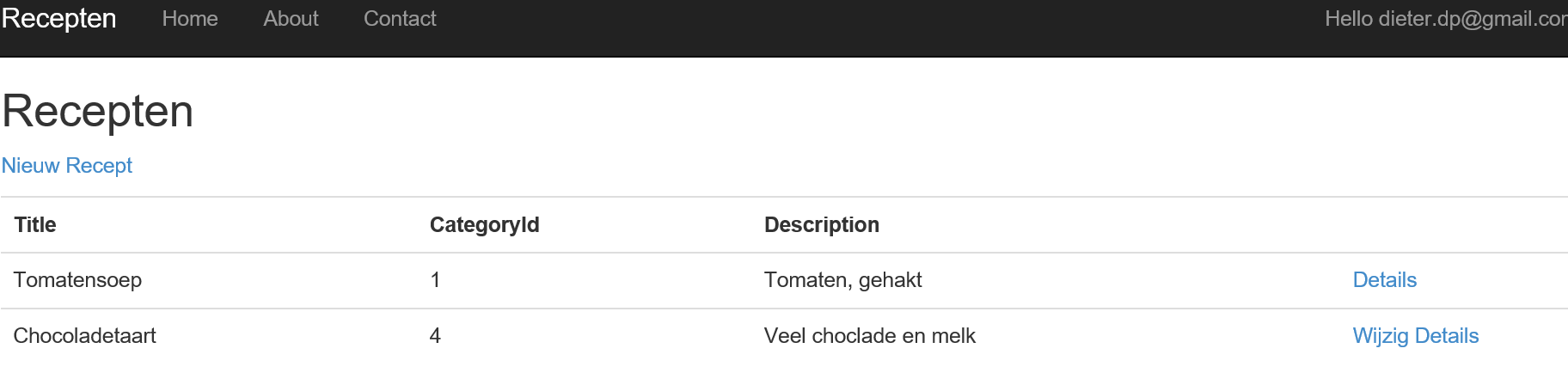


## Overzicht

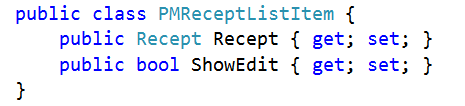
Het overzicht ziet er als volgt uit:



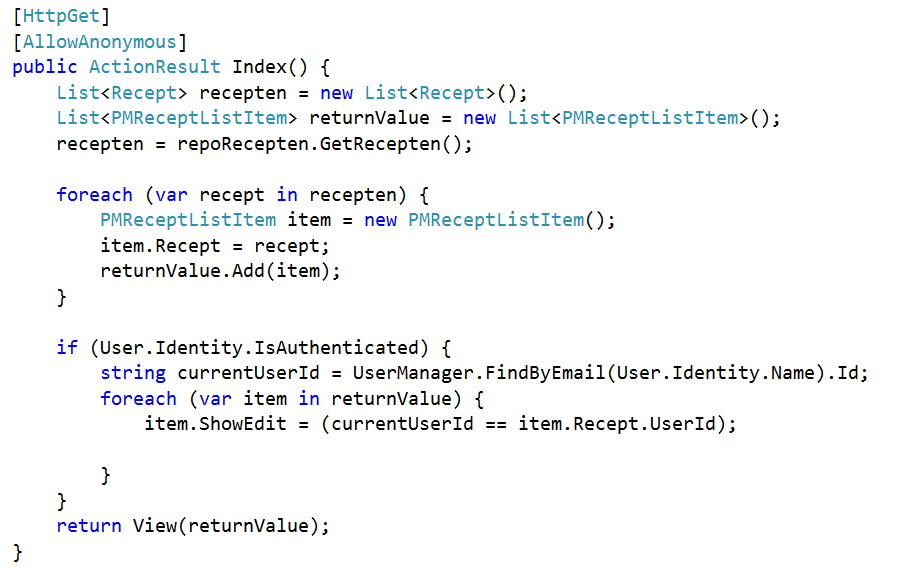
Wijzig dit scherm zodat enkel ingelogde gebruikers in de rol “Kok” het scherm edit zien. Daarnaast mag enkel de persoon die het recept heeft geschreven de knop edit zien.



Na het ophalen van alle recepten moeten we controleren per recept of de bezoeker het al dan niet mag kunnen wijzigen. We maken een nieuw klasse PMReceptListItem aan die er als volgt uitziet:



Deze klasse zal zowel ons recept bevatten als het al dan niet weergeven van de edit knop. In de methode Index moeten we nu eerst alle recepten ophalen. Deze overlopen en in een PMReceptListItem stoppen. Daarna moeten we deze overlopen en kijken via het UserId of we al dan niet mogen wijzigen. We moeten dit enkel doen als we ingelogd zijn. Dit controleren we via “User.Identity.IsAuthenticated”. Daarna vragen we het UserID op van de huidige ingelogde gebruiker. Dit moeten we dan per recept vergelijken met het ID die in het recept object zit. Op deze manier weten we of we de knop al dan niet mogen tonen.



Wijzig daarna de Index View zodat je gebruikt maakt van PMReceptListItem.

## Helper Code URL Encoding

public static class UrlEncoding {

public static string Base64ForUrlEncode(this string str) {

byte[] encbuff = Encoding.UTF8.GetBytes(str);

return HttpServerUtility.UrlTokenEncode(encbuff);

}

public static string Base64ForUrlDecode(this string str) {

byte[] decbuff = HttpServerUtility.UrlTokenDecode(str);

return Encoding.UTF8.GetString(decbuff);

}

}