Webapplicaties III

1 DienstenCheques Project

Alle "using" statements zijn weggelaten om hopelijk plaats te besparen tijdens het printen...

1.1 Structuur

Dit is de mappen structuur van het project, enkel de belangrijkste bestanden voor ons staan er in.

1.1.1 DienstenCheques

```
.
├── App_Start
|   ├── BundleConfig.cs
|   ├── FilterConfig.cs
|   ├── IdentityConfig.cs
|   ├── NinjectWebCommon.cs
|   ├── RouteConfig.cs
|   └── Startup.Auth.cs
├── Content
|   ├── Site.css
|   ├── bootstrap.css
|   └── bootstrap.min.css
├── Controllers
|   ├── AccountController.cs
|   ├── BestellingenController.cs
|   ├── HomeController.cs
|   └── ManageController.cs
├── Global.asax.cs
├── Infrastructure
|   └── GebruikerModelBinder.cs
├── Models
|   ├── DAL
|   |   ├── Mapper
|   |   |   ├── BestellingMapper.cs
|   |   |   ├── DienstenChequeMapper.cs
|   |   |   ├── GebruikerMapper.cs
|   |   |   └── PrestatieMapper.cs
|   |   ├── DienstenChequesContext.cs
|   |   ├── DienstenChequesInitializer.cs
|   |   └── GebruikersRepository.cs
|   ├── Domain
|   |   ├── ApplicationUser.cs
|   |   ├── Bestelling.cs
|   |   ├── DienstenCheque.cs
|   |   ├── Gebruiker.cs
|   |   ├── IGebruikersRepository.cs
|    |    ├── Onderneming.cs
|    |    ├── Prestatie.cs
|    |    └── PrestatieType.cs
|    ├── AccountViewModels.cs
|    └── ManageViewModels.cs
├── Scripts
|    ├── Bestellingen.js
|    ├── _references.js
|    ├── bootstrap.js
|    ├── bootstrap.min.js
|    ├── jquery-1.10.2.intellisense.js
|    ├── jquery-1.10.2.js
|    ├── jquery-1.10.2.min.js
|    ├── jquery-1.10.2.min.map
|    ├── jquery.validate-vsdoc.js
|    ├── jquery.validate.js
|    ├── jquery.validate.min.js
|    ├── jquery.validate.unobtrusive.js
|    ├── jquery.validate.unobtrusive.min.js
|    ├── modernizr-2.6.2.js
|    ├── respond.js
|    └── respond.min.js
├── Startup.cs
├── ViewModels
|    └── BestellingenViewModels.cs
├── Views
|    ├── Account
|    |    ├── ConfirmEmail.cshtml
|    |    ├── ExternalLoginConfirmation.cshtml
|    |    ├── ExternalLoginFailure.cshtml
|    |    ├── ForgotPassword.cshtml
|    |    ├── ForgotPasswordConfirmation.cshtml
|    |    ├── Login.cshtml
|    |    ├── Register.cshtml
|    |    ├── ResetPassword.cshtml
|    |    ├── ResetPasswordConfirmation.cshtml
|    |    ├── SendCode.cshtml
|    |    ├── VerifyCode.cshtml
|    |    └── _ExternalLoginsListPartial.cshtml
|    ├── Bestellingen
|    |    ├── Bestelling.cshtml
|    |    ├── Index.cshtml
|    |    └── Nieuw.cshtml
|    ├── Home
|    |    ├── About.cshtml
|    |    ├── Contact.cshtml
|    |    └── Index.cshtml
|    ├── Manage
|    |    ├── AddPhoneNumber.cshtml
|    |    ├── ChangePassword.cshtml
|    |    ├── Index.cshtml
|    |    ├── ManageLogins.cshtml
|    |    ├── SetPassword.cshtml
|    |    └── VerifyPhoneNumber.cshtml
|    ├── Shared
|    |    ├── Error.cshtml
|    |    ├── Lockout.cshtml
|   |   ├── _Layout.cshtml
|   |   └── _LoginPartial.cshtml
|   ├── Web.config
|   └── _ViewStart.cshtml
└── Web.config
```

1.1.2 DienstenCheques.Tests

```
.
├── Controllers
|   ├── BestellingenControllerTest.cs
|   ├── DummyContext.cs
|   └── HomeControllerTest.cs
└── Models
    ├── BestellingTest.cs
    ├── DienstenChequeTest.cs
    └── GebruikerTest.cs
```

1.2 DienstenCheques

1.2.1 App_Start

1.2.1.1 BundleConfig.cs

```
namespace DienstenCheques {
    public class BundleConfig {
        public static void RegisterBundles(BundleCollection bundles) {
            // Dit kan je toepassen voor alle files die je nodig hebt
            // In include, kan je meerdere items plaatsen
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                "~/Scripts/jquery-{version}.js"));
            bundles.Add(new StyleBundle("~/Content/css").Include(
              "~/Content/bootstrap.css",
              "~/Content/site.css"));
        }
    }
}
```

1.2.1.2 FilterConfig.cs

```
namespace DienstenCheques {
    public class FilterConfig {
        public static void RegisterGlobalFilters(
            GlobalFilterCollection filters
        ) {
            filters.Add(new HandleErrorAttribute());
        }
    }
}
```

1.2.1.3 IdentityConfig.cs

```
namespace DienstenCheques {
    public class EmailService : IIdentityMessageService {
        public Task SendAsync(IdentityMessage message) {
            return Task.FromResult(0);
        }
    }
    public class SmsService : IIdentityMessageService{
        public Task SendAsync(IdentityMessage message) {
            return Task.FromResult(0);
        }
    }
    public class ApplicationUserManager : UserManager<ApplicationUser> {
        public ApplicationUserManager(
            IUserStore<ApplicationUser> store
        ) : base(store) { }

        public static ApplicationUserManager
Create(IdentityFactoryOptions<ApplicationUserManager> options, IOwinContext
context) {
            var manager = new ApplicationUserManager(
                new UserStore<ApplicationUser>(
                    context.Get<DienstenChequesContext>()
                )
            );
            manager.UserValidator = new UserValidator<ApplicationUser>(
                manager
            ) {
                AllowOnlyAlphanumericUserNames = false,
                RequireUniqueEmail = true
            };
            manager.PasswordValidator = new PasswordValidator {
                RequiredLength = 6,
                RequireNonLetterOrDigit = true,
                RequireDigit = true,
                RequireLowercase = true,
                RequireUppercase = true,
            };
            manager.UserLockoutEnabledByDefault = true;
            manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
            manager.MaxFailedAccessAttemptsBeforeLockout = 5;
            manager.RegisterTwoFactorProvider(
                "Phone Code",
                new PhoneNumberTokenProvider<ApplicationUser> {
                    MessageFormat = "Your security code is {0}"
                }
            ); // Kan ook voor "Email Code" met een Subject & BodyFormat
            manager.SmsService = new SmsService(); // EmailService
            var dataProtectionProvider = options.DataProtectionProvider;
            if (dataProtectionProvider != null) {
                manager.UserTokenProvider =
                    new DataProtectorTokenProvider<ApplicationUser>(
                        dataProtectionProvider.Create("ASP.NET Identity")
                    );
            }
            return manager;
        }
    }
    // Configure the application sign-in manager
    public class ApplicationSignInManager : SignInManager<ApplicationUser,
string> {
        public ApplicationSignInManager(
            ApplicationUserManager userManager,
            IAuthenticationManager authenticationManager
        ) : base(userManager, authenticationManager) { }
        public override Task<ClaimsIdentity> CreateUserIdentityAsync(
            ApplicationUser user
        ) {
            return user.GenerateUserIdentityAsync(
                (ApplicationUserManager)UserManager
            );
        }
        public static ApplicationSignInManager Create(
            IdentityFactoryOptions<ApplicationSignInManager> options,
            IOwinContext context
        ) {
            return new ApplicationSignInManager(
                context.GetUserManager<ApplicationUserManager>(),
                context.Authentication
            );
        }
    }
    public class ApplicationRoleManager : RoleManager<IdentityRole> {
        public ApplicationRoleManager(
            IRoleStore<IdentityRole, string> roleStore
        ) : base(roleStore) { }
        public static ApplicationRoleManager Create(
            IdentityFactoryOptions<ApplicationRoleManager> options,
            IOwinContext context) {
            return new ApplicationRoleManager(
                new RoleStore<IdentityRole>(
                    context.Get<DienstenChequesContext>()
                )
            );
        }
    }
}
```

1.2.1.4 NinjectWebCommon.cs

```
namespace DienstenCheques.App_Start {
    public static class NinjectWebCommon {
        private static readonly Bootstrapper bootstrapper =
            new Bootstrapper();
        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start() {
            DynamicModuleUtility.RegisterModule(
                typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(
                typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }
        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop() {
            bootstrapper.ShutDown();
        }
        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel() {
            var kernel = new StandardKernel();
            try {
                kernel.Bind<Func<IKernel>>().ToMethod(
                    ctx => () => new Bootstrapper().Kernel);
                kernel.Bind<IHttpModule>()
                    .To<HttpApplicationInitializationHttpModule>();
                RegisterServices(kernel);
                return kernel;
            }
            catch { kernel.Dispose(); throw; }
        }
        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel) {
            kernel.Bind<DienstenChequesContext>().ToSelf().InRequestScope();
            kernel.Bind<IGebruikersRepository>().To<GebruikersRepository>
().InRequestScope();
        }
    }
}
```

1.2.1.5 RouteConfig.cs

```
namespace DienstenCheques {
    public class RouteConfig {
        public static void RegisterRoutes(RouteCollection routes) {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new {
                    controller = "Home",
                    action = "Index",
                    id = UrlParameter.Optional
                }
            );
        }
    }
}
```

1.2.1.6 Startup.Auth.cs

```
namespace DienstenCheques {
    public partial class Startup {
        public void ConfigureAuth(IAppBuilder app) {
            app.CreatePerOwinContext(DienstenChequesContext.Create);
            app.CreatePerOwinContext<ApplicationUserManager>
(ApplicationUserManager.Create);
            app.CreatePerOwinContext<ApplicationSignInManager>
(ApplicationSignInManager.Create);
            app.CreatePerOwinContext<ApplicationRoleManager>
(ApplicationRoleManager.Create);

            app.UseCookieAuthentication(new CookieAuthenticationOptions {
                AuthenticationType =
                    DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login"),
                Provider = new CookieAuthenticationProvider {
                    OnValidateIdentity =
SecurityStampValidator.OnValidateIdentity<ApplicationUserManager,
ApplicationUser>(validateInterval: TimeSpan.FromMinutes(30),
regenerateIdentity: (manager, user) =>
user.GenerateUserIdentityAsync(manager))
                }
            });

app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie,
TimeSpan.FromMinutes(5));
            app.UseTwoFactorRememberBrowserCookie(
                DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);
            // Third party login providers
            //app.UseMicrosoftAccountAuthentication(
            //    clientId: "",
            //    clientSecret: "");
        }
    }
}
```

1.2.2 Content

Dit zijn de css files.

1.2.3 Controllers

1.2.3.1 AccountController.cs

```
namespace DienstenCheques.Controllers {
    [Authorize]
    public class AccountController : Controller {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;
        public AccountController() { }
        public AccountController(ApplicationUserManager userManager,
            ApplicationSignInManager signInManager) {
            UserManager = userManager;
            SignInManager = signInManager;
        }
        // Zelfde voor ApplicationUserManager
        public ApplicationSignInManager SignInManager {
            get {
                return _signInManager
                    ?? HttpContext.GetOwinContext().Get<
                        ApplicationSignInManager>();
            }
            private set { _signInManager = value; }
        }

        // GET: /Account/Login
        [AllowAnonymous]
        public ActionResult Login(string returnUrl) {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }
        // POST: /Account/Login
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(
            LoginViewModel model, string returnUrl) {
            if ( ! ModelState.IsValid) { // Zeer belangrijke validatie stap
                return View(model);
            }
            // To enable password failures change to shouldLockout: true
            var result = await
SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
            switch (result) {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new {
                        ReturnUrl = returnUrl,
                        RememberMe = model.RememberMe
                    });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid login attempt.");
                    return View(model);
            }
        }
        // GET: /Account/VerifyCode
        [AllowAnonymous]
        public async Task<ActionResult> VerifyCode(
            string provider, string returnUrl, bool rememberMe) {
            // Require that the user has already logged in
            // via username/password or external login
            if ( ! await SignInManager.HasBeenVerifiedAsync())
                return View("Error");
            return View(new VerifyCodeViewModel {
                Provider = provider,
                ReturnUrl = returnUrl,
                RememberMe = rememberMe
            });
        }
        // POST: /Account/VerifyCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> VerifyCode(VerifyCodeViewModel
model) {
            if (!ModelState.IsValid) { return View(model); }
            // The following code protects for brute force attacks against
the two factor codes.
            // If a user enters incorrect codes for a specified amount of
time then the user account
            // will be locked out for a specified amount of time.
            // You can configure the account lockout settings in
IdentityConfig
            var result = await
SignInManager.TwoFactorSignInAsync(model.Provider, model.Code, isPersistent:
model.RememberMe, rememberBrowser: model.RememberBrowser);
            switch (result) {
                case SignInStatus.Success:
                    return RedirectToLocal(model.ReturnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid code.");
                    return View(model);
            }
        }
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register() { return View(); }
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Register(RegisterViewModel model) {
            if (ModelState.IsValid) {
                var user = new ApplicationUser { UserName = model.Email,
Email = model.Email };
                var result = await UserManager.CreateAsync(user,
model.Password);
                if (result.Succeeded) {
                    await SignInManager.SignInAsync(user,
isPersistent:false, rememberBrowser:false);
                    // For more information on how to enable account
confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkID=320771
                    // Send an email with this link
                    // string code = await
UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
                    // var callbackUrl = Url.Action("ConfirmEmail",
"Account", new { userId = user.Id, code = code }, protocol:
Request.Url.Scheme);
                    // await UserManager.SendEmailAsync(user.Id, "Confirm
your account", "Please confirm your account by clicking <a href=\"" +
callbackUrl + "\">here</a>");
                    return RedirectToAction("Index", "Home");
                }
                AddErrors(result);
            }
            // If we got this far, something failed, redisplay form
            return View(model);
        }
        // GET: /Account/ConfirmEmail
        [AllowAnonymous]
        public async Task<ActionResult> ConfirmEmail(string userId, string
code) {
            if (userId == null || code == null) { return View("Error"); }
            var result = await UserManager.ConfirmEmailAsync(userId, code);
            return View(result.Succeeded ? "ConfirmEmail" : "Error");
        }
        // GET: /Account/ForgotPassword
        [AllowAnonymous]
        public ActionResult ForgotPassword() { return View(); }
        // POST: /Account/ForgotPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ForgotPassword(ForgotPasswordViewModel model) {
            if (ModelState.IsValid) {
                var user = await UserManager.FindByNameAsync(model.Email);
                if (user == null || !(await
UserManager.IsEmailConfirmedAsync(user.Id))) {
                    // Don't reveal that the user does not exist or is not
confirmed
```

```
                return View("ForgotPasswordConfirmation");
            }
            // For more information on how to enable account
confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkID=320771
            // Send an email with this link
            // string code = await
UserManager.GeneratePasswordResetTokenAsync(user.Id);
            // var callbackUrl = Url.Action("ResetPassword", "Account",
new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);
            // await UserManager.SendEmailAsync(user.Id, "Reset
Password", "Please reset your password by clicking <a href=\"" + callbackUrl
+ "\">here</a>");
            // return RedirectToAction("ForgotPasswordConfirmation",
"Account");
        }
        // If we got this far, something failed, redisplay form
        return View(model);
    }
    // GET: /Account/ForgotPasswordConfirmation
    [AllowAnonymous]
    public ActionResult ForgotPasswordConfirmation() { return View(); }
    // GET: /Account/ResetPassword
    [AllowAnonymous]
    public ActionResult ResetPassword(string code) {
        return code == null ? View("Error") : View();
    }
    // POST: /Account/ResetPassword
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> ResetPassword(ResetPasswordViewModel
model) {
        if ( ! ModelState.IsValid) {
            return View(model);
        }
        var user = await UserManager.FindByNameAsync(model.Email);
        if (user == null) {
            // Don't reveal that the user does not exist
            return RedirectToAction("ResetPasswordConfirmation",
"Account");
        }
        var result = await UserManager.ResetPasswordAsync(user.Id,
model.Code, model.Password);
        if (result.Succeeded) {
            return RedirectToAction("ResetPasswordConfirmation",
"Account");
        }
        AddErrors(result);
        return View();
    }
    // GET: /Account/ResetPasswordConfirmation
    [AllowAnonymous]
    public ActionResult ResetPasswordConfirmation() {
        return View();
    }
    // POST: /Account/ExternalLogin
```

```
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
{
            // Request a redirect to the external login provider
            return new ChallengeResult(provider,
Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl
}));
        }
        // GET: /Account/SendCode
        [AllowAnonymous]
        public async Task<ActionResult> SendCode(string returnUrl, bool
rememberMe) {
            var userId = await SignInManager.GetVerifiedUserIdAsync();
            if (userId == null) {
                return View("Error");
            }
            var userFactors = await
UserManager.GetValidTwoFactorProvidersAsync(userId);
            var factorOptions = userFactors.Select(purpose => new
SelectListItem { Text = purpose, Value = purpose }).ToList();
            return View(new SendCodeViewModel { Providers = factorOptions,
ReturnUrl = returnUrl, RememberMe = rememberMe });
        }
        // POST: /Account/SendCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SendCode(SendCodeViewModel model) {
            if ( ! ModelState.IsValid) { return View(); }
            // Generate the token and send it
            if (!await
SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider)) {
                return View("Error");
            }
            return RedirectToAction("VerifyCode", new { Provider =
model.SelectedProvider, ReturnUrl = model.ReturnUrl, RememberMe =
model.RememberMe });
        }
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        public async Task<ActionResult> ExternalLoginCallback(string
returnUrl) {
            var loginInfo = await
AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null) {
                return RedirectToAction("Login");
            }
            // Sign in the user with this external login provider if the
user already has a login
            var result = await SignInManager.ExternalSignInAsync(loginInfo,
isPersistent: false);
            switch (result) {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl =
returnUrl, RememberMe = false });
                case SignInStatus.Failure:
                default:
                    // If the user does not have an account, then prompt the
user to create an account
                    ViewBag.ReturnUrl = returnUrl;
                    ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                    return View("ExternalLoginConfirmation", new
ExternalLoginConfirmationViewModel { Email = loginInfo.Email });
            }
        }
        // POST: /Account/ExternalLoginConfirmation
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string
returnUrl) {
            if (User.Identity.IsAuthenticated) {
                return RedirectToAction("Index", "Manage");
            }
            if (ModelState.IsValid) {
                // Get the information about the user from the external
login provider
                var info = await
AuthenticationManager.GetExternalLoginInfoAsync();
                if (info == null) { return View("ExternalLoginFailure"); }
                var user = new ApplicationUser { UserName = model.Email,
Email = model.Email };
                var result = await UserManager.CreateAsync(user);
                if (result.Succeeded) {
                    result = await UserManager.AddLoginAsync(user.Id,
info.Login);
                    if (result.Succeeded) {
                        await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                        return RedirectToLocal(returnUrl);
                    }
                }
                AddErrors(result);
            }
            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }
        // POST: /Account/LogOff
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff() {
            AuthenticationManager.SignOut();
            return RedirectToAction("Index", "Home");
        }
        // GET: /Account/ExternalLoginFailure
        [AllowAnonymous]
        public ActionResult ExternalLoginFailure() {
            return View();
        }
        protected override void Dispose(bool disposing) {
            if (disposing) {
                if (_userManager != null) {
                    _userManager.Dispose();
                    _userManager = null;
                }
                if (_signInManager != null) {
                    _signInManager.Dispose();
                    _signInManager = null;
                }
            }
            base.Dispose(disposing);
        }
        #region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";
        private IAuthenticationManager AuthenticationManage) {
            get {
                return HttpContext.GetOwinContext().Authentication;
            }
        }
        private void AddErrors(IdentityResult result) {
            foreach (var error in result.Errors) {
                ModelState.AddModelError("", error);
            }
        }
        private ActionResult RedirectToLocal(string returnUrl) {
            if (Url.IsLocalUrl(returnUrl)) {
                return Redirect(returnUrl);
            }
            return RedirectToAction("Index", "Home");
        }
        internal class ChallengeResult : HttpUnauthorizedResult {
            public ChallengeResult(string provider, string redirectUri)
                : this(provider, redirectUri, null) { }
            public ChallengeResult(string provider, string redirectUri,
string userId) {
                LoginProvider = provider;
                RedirectUri = redirectUri;
                UserId = userId;
            }
            public string LoginProvider { get; set; }
            public string RedirectUri { get; set; }
            public string UserId { get; set; }
            public override void ExecuteResult(ControllerContext context) {
                var properties = new AuthenticationProperties { RedirectUri
= RedirectUri };
                if (UserId != null) {
                    properties.Dictionary[XsrfKey] = UserId;
                }

context.HttpContext.GetOwinContext().Authentication.Challenge(properties,
LoginProvider);
            }
        }
        #endregion
    }
}
```

1.2.3.2 BestellingenController.cs

```
namespace DienstenCheques.Controllers {
    [Authorize(Roles = "customer")]
    public class BestellingenController : Controller {
        private IGebruikersRepository gebruikersRepository;
        public BestellingenController(IGebruikersRepository
gebruikersRepository) {
            this.gebruikersRepository = gebruikersRepository;
        }
        // GET: Bestellingen
        public ActionResult Index(Gebruiker gebruiker, int aantalMaanden=6)
{
            BestellingenViewModel vm = new BestellingenViewModel() {
                Bestellingen = gebruiker.GetBestellingen(aantalMaanden)
                    .Select(b => new BestellingViewModel(b)),
                AantalBeschikbareCheques =
gebruiker.AantalBeschikbareElektronischeCheques,
                AantalOpenstaandePrestatieUren =
gebruiker.AantalOpenstaandePrestatieUren,
                AantalMaanden = aantalMaanden
            };
            if (Request.IsAjaxRequest())
                return PartialView("Bestelling", vm.Bestellingen);
            return View(vm);
        }
        public ActionResult Nieuw(Gebruiker gebruiker) {
            NieuweBestellingViewModel vm = new
NieuweBestellingViewModel(Bestelling.BEDRAGCHEQUE);
            return View(vm);
        }
        [HttpPost]
        public ActionResult Nieuw(Gebruiker gebruiker,
NieuweBestellingViewModel model) {
            if (ModelState.IsValid) {
                try{
                    Bestelling b =
gebruiker.AddBestelling(model.AantalCheques, model.Elektronisch,
model.DebiteerDatum);
                    gebruikersRepository.SaveChanges();
                    TempData["message"] = $"Uw bestelling voor een
totaalbedrag van {b.TotaalBedrag:C} werd gecreëerd";
                    return  RedirectToAction("Index");
                } catch (Exception ex) {
                    ModelState.AddModelError("", ex.Message);
                }
            }
            model.Zichtwaarde = Bestelling.BEDRAGCHEQUE;
            return View(model);
        }
    }
}
```

1.2.3.3 HomeController.cs

```
namespace DienstenCheques.Controllers {
    public class HomeController : Controller {
        public ActionResult Index() { return View(); }
        public ActionResult About() {
            ViewBag.Message = "Your application description page.";
            return View();
        }
        public ActionResult Contact() {
            ViewBag.Message = "Your contact page.";
            return View();
        }
    }
}
```

1.2.3.4 ManageController.cs

```
namespace DienstenCheques.Controllers {
    [Authorize]
    public class ManageController : Controller {
        private ApplicationSignInManager _signInManager;
        private ApplicationUserManager _userManager;
        public ManageController() { }
        public ManageController(ApplicationUserManager userManager,
ApplicationSignInManager signInManager) {
            UserManager = userManager;
            SignInManager = signInManager;
        }
        public ApplicationSignInManager SignInManager {
            get {
                return _signInManager ??
HttpContext.GetOwinContext().Get<ApplicationSignInManager>();
            }
            private set {
                _signInManager = value;
            }
        }
        public ApplicationUserManager UserManager {
            get {
                return _userManager ??
HttpContext.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set {
                _userManager = value;
            }
        }
        // GET: /Manage/Index
        public async Task<ActionResult> Index(ManageMessageId? message) {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Your
password has been changed."
                : message == ManageMessageId.SetPasswordSuccess ? "Your
password has been set."
                : message == ManageMessageId.SetTwoFactorSuccess ? "Your
two-factor authentication provider has been set."
                : message == ManageMessageId.Error ? "An error has
occurred."
                : message == ManageMessageId.AddPhoneSuccess ? "Your phone
number was added."
                : message == ManageMessageId.RemovePhoneSuccess ? "Your
phone number was removed."
                : "";
            var userId = User.Identity.GetUserId();
            var model = new IndexViewModel {
                HasPassword = HasPassword(),
                PhoneNumber = await UserManager.GetPhoneNumberAsync(userId),
                TwoFactor = await
UserManager.GetTwoFactorEnabledAsync(userId),
                Logins = await UserManager.GetLoginsAsync(userId),
                BrowserRemembered = await
AuthenticationManager.TwoFactorBrowserRememberedAsync(userId)
            };
            return View(model);
        }
        // POST: /Manage/RemoveLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> RemoveLogin(string loginProvider,
string providerKey) {
            ManageMessageId? message;
            var result = await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new
UserLoginInfo(loginProvider, providerKey));
            if (result.Succeeded) {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null) {
                    await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                }
                message = ManageMessageId.RemoveLoginSuccess;
            } else {
                message = ManageMessageId.Error;
            }
            return RedirectToAction("ManageLogins", new { Message = message
});
        }
        // GET: /Manage/AddPhoneNumber
        public ActionResult AddPhoneNumber() { return View(); }
        // POST: /Manage/AddPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
AddPhoneNumber(AddPhoneNumberViewModel model) {
            if ( ! ModelState.IsValid) {
                return View(model);
            }
            // Generate the token and send it
            var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
model.Number);
            if (UserManager.SmsService != null) {
                var message = new IdentityMessage {
                    Destination = model.Number,
                    Body = "Your security code is: " + code
                };
                await UserManager.SmsService.SendAsync(message);
            }
            return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber =
model.Number });
        }
        // POST: /Manage/EnableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> EnableTwoFactorAuthentication() {
            await
UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null) {
                await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }
        // POST: /Manage/DisableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> DisableTwoFactorAuthentication() {
            await
UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null) {
                await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }
        // GET: /Manage/VerifyPhoneNumber
        public async Task<ActionResult> VerifyPhoneNumber(string
phoneNumber) {
            var code = await
UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
phoneNumber);
            // Send an SMS through the SMS provider to verify the phone
number
            return phoneNumber == null ? View("Error") : View(new
VerifyPhoneNumberViewModel { PhoneNumber = phoneNumber });
```

```
        }
        // POST: /Manage/VerifyPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
VerifyPhoneNumber(VerifyPhoneNumberViewModel model) {
            if ( ! ModelState.IsValid) {
                return View(model);
            }
            var result = await
UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(),
model.PhoneNumber, model.Code);
            if (result.Succeeded) {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null) {
                    await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message =
ManageMessageId.AddPhoneSuccess });
            }
            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "Failed to verify phone");
            return View(model);
        }
        // GET: /Manage/RemovePhoneNumber
        public async Task<ActionResult> RemovePhoneNumber() {
            var result = await
UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);
            if (!result.Succeeded) {
                return RedirectToAction("Index", new { Message =
ManageMessageId.Error });
            }
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null) {
                await SignInManager.SignInAsync(user, isPersistent: false,
rememberBrowser: false);
            }
            return RedirectToAction("Index", new { Message =
ManageMessageId.RemovePhoneSuccess });
        }
        // GET: /Manage/ChangePassword
        public ActionResult ChangePassword() { return View(); }
        // POST: /Manage/ChangePassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ChangePassword(ChangePasswordViewModel model) {
            if (!ModelState.IsValid) {
                return View(model);
            }
            var result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(),
model.OldPassword, model.NewPassword);
            if (result.Succeeded) {
                var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null) {
                    await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message =
ManageMessageId.ChangePasswordSuccess });
            }
            AddErrors(result);
            return View(model);
        }
        // GET: /Manage/SetPassword
        public ActionResult SetPassword() { return View(); }
        // POST: /Manage/SetPassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SetPassword(SetPasswordViewModel
model) {
            if (ModelState.IsValid) {
                var result = await
UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);
                if (result.Succeeded) {
                    var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
                    if (user != null) {
                        await SignInManager.SignInAsync(user, isPersistent:
false, rememberBrowser: false);
                    }
                    return RedirectToAction("Index", new { Message =
ManageMessageId.SetPasswordSuccess });
                }
                AddErrors(result);
            }
            // If we got this far, something failed, redisplay form
            return View(model);
        }
        // GET: /Manage/ManageLogins
        public async Task<ActionResult> ManageLogins(ManageMessageId?
message) {
            ViewBag.StatusMessage =
                message == ManageMessageId.RemoveLoginSuccess ? "The
external login was removed."
                : message == ManageMessageId.Error ? "An error has
occurred."
                : "";
            var user = await
UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user == null) {
                return View("Error");
            }
            var userLogins = await
UserManager.GetLoginsAsync(User.Identity.GetUserId());
            var otherLogins =
AuthenticationManager.GetExternalAuthenticationTypes().Where(auth =>
userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();
            ViewBag.ShowRemoveButton = user.PasswordHash != null ||
userLogins.Count > 1;
            return View(new ManageLoginsViewModel {
                CurrentLogins = userLogins,
                OtherLogins = otherLogins
            });
        }
        // POST: /Manage/LinkLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LinkLogin(string provider) {
            // Request a redirect to the external login provider to link a
login for the current user
            return new AccountController.ChallengeResult(provider,
Url.Action("LinkLoginCallback", "Manage"), User.Identity.GetUserId());
        }
        // GET: /Manage/LinkLoginCallback
        public async Task<ActionResult> LinkLoginCallback() {
            var loginInfo = await
AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
            if (loginInfo == null) {
                return RedirectToAction("ManageLogins", new { Message =
ManageMessageId.Error });
            }
            var result = await
UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
            return result.Succeeded ? RedirectToAction("ManageLogins") :
RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
        }
        protected override void Dispose(bool disposing) {
            if (disposing && _userManager != null) {
                _userManager.Dispose();
                _userManager = null;
            }
            base.Dispose(disposing);
        }
        #region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";
        private IAuthenticationManager AuthenticationManager {
            get {
                return HttpContext.GetOwinContext().Authentication;
            }
        }
        private void AddErrors(IdentityResult result) {
            foreach (var error in result.Errors) {
                ModelState.AddModelError("", error);
            }
        }
        private bool HasPassword() {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null) {
                return user.PasswordHash != null;
            }
            return false;
        }
        private bool HasPhoneNumber() {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null) {
                return user.PhoneNumber != null;
            }
            return false;
        }
        public enum ManageMessageId {
            AddPhoneSuccess,
            ChangePasswordSuccess,
            SetTwoFactorSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
            RemovePhoneSuccess,
            Error
        }
        #endregion
    }
}
```

1.2.4 Global.asax.cs

```
namespace DienstenCheques {
    public class MvcApplication : System.Web.HttpApplication {
        protected void Application_Start() {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            ModelBinders.Binders.Add(typeof(Gebruiker), new
GebruikerModelBinder());
            DienstenChequesContext db = new DienstenChequesContext();
            db.Database.Initialize(true);
        }
    }
}
```

1.2.5 Infrastructure

1.2.5.1 GebruikerModelBinder.cs

```
namespace DienstenCheques.Infrastructure {
    public class GebruikerModelBinder : IModelBinder {
        public object  BindModel(ControllerContext controllerContext,
ModelBindingContext bindingContext) {
            if (controllerContext.HttpContext.User.Identity.IsAuthenticated)
{
                IGebruikersRepository repos =
(IGebruikersRepository)DependencyResolver.Current.GetService(typeof(IGebruikersRepository));
```

```
                // return repos.FindBy("1000000000");
                return
repos.FindByEmail(controllerContext.HttpContext.User.Identity.Name);
            }
            return null;
        }
    }
}
```

1.2.6 Models

1.2.6.1 DAL

1.2.6.1.1 Mapper

1.2.6.1.1 BestellingMapper.cs

```
namespace DienstenCheques.Models.DAL.Mapper {
    public class BestellingMapper : EntityTypeConfiguration<Bestelling> {
        public BestellingMapper() {
            //Table
            ToTable("Bestelling");
        }
    }
}
```

1.2.6.1.2 DienstenChequeMapper.cs

```
namespace DienstenCheques.Models.DAL.Mapper {
    public class DienstenChequeMapper :
EntityTypeConfiguration<DienstenCheque> {
        public DienstenChequeMapper() {
            //Table
            ToTable("DienstenCheque");
            HasKey(t => t.DienstenChequeNummer);
            //relationships
            HasOptional(t => t.Prestatie).WithMany().Map(t =>
t.MapKey("PrestatieId")).WillCascadeOnDelete(false);
        }
    }
}
```

1.2.6.1.3 GebruikerMapper.cs

```
namespace DienstenCheques.Models.DAL.Mapper {
    public class GebruikerMapper : EntityTypeConfiguration<Gebruiker> {
        public GebruikerMapper() {
            ToTable("Gebruiker");
            //Properties
            HasKey(t => t.GebruikersNummer);
            Property(t =>
t.GebruikersNummer).IsFixedLength().HasMaxLength(10);
            Property(t => t.Naam).IsRequired().HasMaxLength(100);
            Property(t => t.Voornaam).IsRequired().HasMaxLength(100);
            Property(t=>t.Email).IsRequired().HasMaxLength(100);
            //Relationships
            HasMany(t => t.Bestellingen)
                .WithRequired()
                .Map(m => m.MapKey("GebruikersNummer"))
                .WillCascadeOnDelete(false);
             HasMany(t => t.Prestaties)
                .WithRequired()
                .Map(m => m.MapKey("GebruikersNummer"))
                .WillCascadeOnDelete(false);
             HasMany(t => t.Portefeuille)
                .WithRequired()
                .Map(m => m.MapKey("GebruikersNummer"))
            .WillCascadeOnDelete(false);
        }
    }
}
```

1.2.6.1.4 PrestatieMapper.cs

```
namespace DienstenCheques.Models.DAL.Mapper {
    public class PrestatieMapper : EntityTypeConfiguration<Prestatie> {
        public PrestatieMapper() {
            //Table
            ToTable("Prestatie");
            //Relationships
            HasRequired(t => t.Onderneming)
               .WithMany()
               .Map(m => m.MapKey("OndernemingId"))
               .WillCascadeOnDelete(false);
        }
    }
}
```

1.2.6.1.2 DienstenChequesContext.cs

```
namespace DienstenCheques.Models.DAL {
    public class DienstenChequesContext : IdentityDbContext<ApplicationUser>
{
        public DienstenChequesContext() : base("DienstenCheques") { }
        public DbSet<Gebruiker> Gebruikers { get; set; }
        public DbSet<Onderneming> Ondernemingen { get; set; }
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
            base.OnModelCreating(modelBuilder);
            modelBuilder.Conventions.Remove<PluralizingTableNameConvention>
();

modelBuilder.Configurations.AddFromAssembly(Assembly.GetExecutingAssembly());

        }
        public static DienstenChequesContext Create() {
            return
DependencyResolver.Current.GetService<DienstenChequesContext>();
        }
        //static DienstenChequesContext()
        //{
        //    Database.SetInitializer<DienstenChequesContext>(new
DienstenChequesInitializer());
        //}
    }
}
```

1.2.6.1.3 DienstenChequesInitializer.cs

```
namespace DienstenCheques.Models.DAL {
    public class DienstenChequesInitializer :
DropCreateDatabaseAlways<DienstenChequesContext> {
        protected override void Seed(DienstenChequesContext context) {
            try {
                CreateRoles(context);
                Onderneming onderneming = new Onderneming() { Naam =
"Hogeschool Gent" };
                context.Ondernemingen.Add(onderneming);
                //onvoldoende cheques beschikbaar, nog 2 openstaande
prestatie waarvoor 7 cheques nodig
                Gebruiker jan = new Gebruiker() {
                    GebruikersNummer = "1000000000",
                    Naam = "Peeters",
                    Voornaam = "Jan",
                    Email = "jan.peeters@hogent.be"
                };
                for (int i = 12; i >= 0; i--) {
                    Prestatie pres = new Prestatie() {
                        AantalUren = 4,
                        DatumPrestatie = DateTime.Today.AddMonths(-i),
                        PrestatieType = PrestatieType.Schoonmaken,
                        Onderneming = onderneming,
                        Betaald = true
                    };
                    jan.Prestaties.Add(pres);
                }
                jan.GetPrestatie(11).Betaald = false;
                jan.GetPrestatie(12).Betaald = false;
                int p = 0;
                int pi = 0;
                for (int i = 3; i > 0; i--) {
                    Bestelling b = new Bestelling() {
                        AantalAangekochteCheques = 15,
                        Elektronisch = true
                    };
                    b.StelDatumsIn(DateTime.Today.AddMonths(-4 * i),
DateTime.Today.AddMonths(-4 * i));
                    jan.Bestellingen.Add(b);
                    for (int j = 1; j <= 15; j++) {
                        DienstenCheque d = new DienstenCheque(true,
DateTime.Today.AddMonths(-4 * i));
                        if (p < 11) {
                            d.Prestatie = jan.GetPrestatie(p);
                            d.GebruiksDatum = d.Prestatie.DatumPrestatie;
                        }
                        jan.Portefeuille.Add(d);
                        if (pi < 3)
                            pi++;
                        else {
                            pi = 0;
                            p++;
                        }
                    }
                }
                context.Gebruikers.Add(jan);
                ApplicationUser user1 = new ApplicationUser { UserName =
jan.Email, Email = jan.Email };
                CreateAccount(user1, context);
                //alle cheques zijn toegewezen aan prestaties, geen
openstaande prestatie meer
                Gebruiker an = new Gebruiker() { GebruikersNummer =
"1000000001", Naam = "Pieters", Voornaam = "An", Email =
"an.pieters@hogent.be" };
                Bestelling anBestelling = new Bestelling() {
AantalAangekochteCheques = 20, Elektronisch = true };
                anBestelling.StelDatumsIn(DateTime.Today.AddMonths(-1),
DateTime.Today.AddMonths(-1));
                an.Bestellingen.Add(anBestelling);
                for (int i = 4; i > 0; i--)
                    an.Prestaties.Add(new Prestatie() { AantalUren = 5,
DatumPrestatie = DateTime.Today.AddMonths(-i), PrestatieType =
PrestatieType.Schoonmaken, Onderneming = onderneming, Betaald = true });
                for (int j = 0; j <= 19; j++) {
                    DienstenCheque d = new DienstenCheque(true,
DateTime.Today.AddMonths(-1));
                    d.Prestatie = an.GetPrestatie(j / 5);
                    d.GebruiksDatum = d.Prestatie.DatumPrestatie;
                    an.Portefeuille.Add(d);
                }
                context.Gebruikers.Add(an);
                ApplicationUser user2 = new ApplicationUser { UserName =
an.Email, Email = an.Email };
                CreateAccount(user2, context);
                //nog 2 cheques niet gebruikt, geen openstaande prestaties
                Gebruiker tine = new Gebruiker() { GebruikersNummer =
"1000000002", Naam = "Pieters", Voornaam = "Tine", Email =
"tine.pieters@hogent.be" };
                Bestelling tineBestelling = new Bestelling() {
AantalAangekochteCheques = 6, Elektronisch = true };
                tineBestelling.StelDatumsIn(DateTime.Today.AddMonths(-1),
DateTime.Today.AddMonths(-1));
                tine.Bestellingen.Add(tineBestelling);
                tine.Prestaties.Add(new Prestatie() { AantalUren = 4,
DatumPrestatie = DateTime.Today.AddDays(-10), PrestatieType =
PrestatieType.Schoonmaken, Onderneming = onderneming, Betaald = true });
                for (int j = 1; j <= 6; j++) {
                    DienstenCheque d = new DienstenCheque(true,
DateTime.Today.AddMonths(-1));
                    if (j < 5) {
                        d.Prestatie = tine.GetPrestatie(0);
                        d.GebruiksDatum = d.Prestatie.DatumPrestatie;
                    }
                    tine.Portefeuille.Add(d);
                }
                context.Gebruikers.Add(tine);
                ApplicationUser user3 = new ApplicationUser { UserName =
tine.Email, Email = tine.Email };
                CreateAccount(user3, context);
                // Belangrijk!
                context.SaveChanges();
                //    InitializeIdentityAndRolesForEF(context, "user",
context.Gebruikers.Select(g => g.Email));
            } catch (DbEntityValidationException e) {
                string s = "Fout creatie database ";
                foreach (var eve in e.EntityValidationErrors) {
                    s += String.Format("Entity of type \"{0}\" in state \"
{1}\" has the following validation errors:",
                         eve.Entry.Entity.GetType().Name,
eve.Entry.GetValidationResult());
                    foreach (var ve in eve.ValidationErrors) {
                        s += String.Format("- Property: \"{0}\", Error: \"
{1}\"",
                            ve.PropertyName, ve.ErrorMessage);
                    }
                }
                throw new Exception(s);
            }
        }
        private void CreateAccount(ApplicationUser user,
DienstenChequesContext context) {
            var userStore = new UserStore<ApplicationUser>(context);
            var userManager = new UserManager<ApplicationUser>(userStore);
            var roleStore = new RoleStore<IdentityRole>(context);
            var roleManager = new RoleManager<IdentityRole>(roleStore);
            var role = roleManager.FindByName("customer");
            userManager.Create(user, "P@ssword1");
            userManager.SetLockoutEnabled(user.Id, false);
            userManager.AddToRole(user.Id, role.Name);
        }
        private void CreateRoles(DienstenChequesContext context) {
            var roleStore = new RoleStore<IdentityRole>(context);
            var roleManager = new RoleManager<IdentityRole>(roleStore);
            var role = new IdentityRole("customer");
            roleManager.Create(role);
        }
    }
}
```

1.2.6.1.4 GebruikersRepository.cs

```
namespace DienstenCheques.Models.DAL {
    public class GebruikersRepository : IGebruikersRepository {
        private DienstenChequesContext context;
        private DbSet<Gebruiker> gebruikers;
        public GebruikersRepository(DienstenChequesContext context) {
            this.context = context;
            gebruikers = context.Gebruikers;
        }
        public Gebruiker FindBy(string gebruikersNummer) {
            return gebruikers.Find(gebruikersNummer);
        }
        public Gebruiker FindByEmail(string email) {
            return gebruikers.FirstOrDefault(g => g.Email == email);
        }
        public IQueryable<Gebruiker> FindAll() {
            return gebruikers;
        }
        public void SaveChanges() {
             context.SaveChanges();
        }
    }
}
```

1.2.6.2 Domain

1.2.6.2.1 ApplicationUser.cs

```
namespace DienstenCheques.Models.Domain {
    public class ApplicationUser : IdentityUser {
        public async Task<ClaimsIdentity>
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager) {
            // Note the authenticationType must match the one defined in
CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            return userIdentity;
        }
    }
}
```

1.2.6.2.2 Bestelling.cs

```
namespace DienstenCheques.Models.Domain {
    public class Bestelling {
        #region "Attributes and Properties"
        private int aantalAangekochteCheques;
        public static decimal BEDRAGCHEQUE = 9.0M;
        public int BestellingId { get; private set; }
        public DateTime CreatieDatum { get; private set; }
        public DateTime DebiteerDatum { get; private set; }
        public int AantalAangekochteCheques {
            get { return aantalAangekochteCheques; }
            set {
                if (value <= 0 || value > 50)
                    throw new ArgumentException("U kan maximaal 50 
dienstencheques bestellen");
                aantalAangekochteCheques = value;
            }
        }
        public bool Elektronisch { get; set; }
        public decimal TotaalBedrag {
            get { return AantalAangekochteCheques * BEDRAGCHEQUE; }
        }
        #endregion
        #region "Constructors"
        public Bestelling() {
            StelDatumsIn(DateTime.Today, DateTime.Today);
        }
        public Bestelling(int aantal, bool elektronisch, DateTime 
debiteerDatum) : this() {
            AantalAangekochteCheques = aantal;
            Elektronisch = elektronisch;
            StelDatumsIn(DateTime.Today, debiteerDatum);
        }
        #endregion
        #region "methods"
        public void StelDatumsIn(DateTime creatieDatum, DateTime 
debiteerDatum) {
            if ( creatieDatum.Date > debiteerDatum.Date  || 
debiteerDatum.Date > creatieDatum.AddMonths(1).Date)
                throw new ArgumentException("Debiteerdatum maximaal 1 maand 
na bestelling");
            CreatieDatum = creatieDatum;
            DebiteerDatum = debiteerDatum;
        }
        #endregion
    }
}
```

1.2.6.2.3 DienstenCheque.cs

```
namespace DienstenCheques.Models.Domain {
    public class DienstenCheque {
        #region "Properties"
        public int DienstenChequeNummer { get; set; }
        public virtual Prestatie Prestatie { get; set; }
        public DateTime? GebruiksDatum { get; set; }
        public DateTime CreatieDatum { get; private set; }
        public bool Elektronisch { get; private set; }
        #endregion
        #region "Constructors"
        public DienstenCheque() { }
        public DienstenCheque(bool elektronisch, DateTime creatieDatum) :
this() {
            Elektronisch = elektronisch;
            CreatieDatum = creatieDatum;
        }
        public DienstenCheque(bool elektronisch) : this(elektronisch,
DateTime.Today) { }
        #endregion
    }
}
```

1.2.6.2.4 Gebruiker.cs

```
namespace DienstenCheques.Models.Domain {
    public class Gebruiker {
        #region Properties
        public string GebruikersNummer { get; set; }
        public String Naam { get; set; }
        public String Voornaam { get; set; }
        public String Email { get; set; }
        public virtual ICollection<Bestelling> Bestellingen { get; set; }
        public virtual ICollection<DienstenCheque> Portefeuille { get; set;
}
        public virtual ICollection<Prestatie> Prestaties { get; set; }
        public int AantalOpenstaandePrestatieUren {
            get { return GetOpenstaandePrestaties().Sum(p=>p.AantalUren); }
        }
        public int AantalBeschikbareElektronischeCheques {
            get { return
GetBeschikbareElektronischeDienstenCheques().Count(); }
        }
        #endregion
        #region Constructors
        public Gebruiker() {
            Bestellingen = new List<Bestelling>();
            Prestaties = new List<Prestatie>();
            Portefeuille = new List<DienstenCheque>();
        }
        #endregion
        #region Methods
        private IEnumerable<Prestatie> GetOpenstaandePrestaties() {
            return Prestaties.Where(p => !p.Betaald);
        }
        private IEnumerable<DienstenCheque>
GetBeschikbareElektronischeDienstenCheques() {
            return Portefeuille
                .Where(c => c.Elektronisch && !c.GebruiksDatum.HasValue)
                //.Where(c=>c.Elektronisch && c.Prestatie==null)
                .OrderBy(c => c.CreatieDatum);
        }
        private int GetAantalBesteldeCheques(int jaar) {
            return Bestellingen.Where(b => b.CreatieDatum.Year == jaar)
                .Sum(b => b.AantalAangekochteCheques);
        }
        public IEnumerable<Bestelling> GetBestellingen(int aantalMaanden) {
            return Bestellingen.Where(b => (DateTime.Today.AddMonths(-
aantalMaanden) <= b.CreatieDatum)).OrderByDescending(b=>b.CreatieDatum);
        }
        private void BetaalPrestatie(Prestatie p) {
            if (!p.Betaald && AantalBeschikbareElektronischeCheques >=
p.AantalUren) {
                IList<DienstenCheque> openstaandeCheques =
                    GetBeschikbareElektronischeDienstenCheques().ToList();
                for (int i = 0; i < p.AantalUren; i++) {
                    openstaandeCheques[i].Prestatie = p;
                    openstaandeCheques[i].GebruiksDatum = DateTime.Today;
                }
                p.Betaald = true;
            }
        }
        public Bestelling AddBestelling(int aantalCheques, bool
elektronisch, DateTime debiteerDatum) {
            Bestelling b = new Bestelling(aantalCheques, elektronisch,
debiteerDatum);
            if (GetAantalBesteldeCheques(b.CreatieDatum.Year) +
aantalCheques > 500)
                throw new ArgumentException("Je hebt de grens van 500
checques bereikt");
            Bestellingen.Add(b);
            if (elektronisch) {
                for (int i = 0; i < aantalCheques; i++)
                    Portefeuille.Add(new DienstenCheque(elektronisch));
                IEnumerable<Prestatie> nietBetaaldePrestaties =
GetOpenstaandePrestaties();
                foreach (Prestatie p in nietBetaaldePrestaties) {
                    BetaalPrestatie(p);
                }
            }
            return b;
        }
        public Prestatie GetPrestatie(int index) {
            return Prestaties.ToList()[index];
        }
        #endregion
    }
}
```

1.2.6.2.5 IGebruikersRepository.cs

```
namespace DienstenCheques.Models.Domain {
    public interface IGebruikersRepository {
        Gebruiker FindBy(string gebruikersNummer);
        Gebruiker FindByEmail(string email);
        IQueryable<Gebruiker> FindAll();
        void SaveChanges();
    }

}
```

1.2.6.2.6 Onderneming.cs

```
namespace DienstenCheques.Models.Domain {
    public class Onderneming {
        #region "Properties"
        public int OndernemingId { get; set; }
        public string Naam { get; set; }
        #endregion
    }
}
```

1.2.6.2.7 Prestatie.cs

```
namespace DienstenCheques.Models.Domain {
   public class Prestatie {
       #region "Properties"
       public int PrestatieId { get; set; }
       public int AantalUren { get; set; }
       public PrestatieType PrestatieType { get; set; }
       public virtual Onderneming Onderneming { get; set; }
       public bool Betaald { get; set; }
       public DateTime DatumPrestatie { get; set; }
       #endregion
    }
}
```

1.2.6.2.8 PrestatieType.cs

```
namespace DienstenCheques.Models.Domain {
    public enum PrestatieType {
        Schoonmaken,
        WassenEnStrijken,
        BereidenMaaltijden
    }
}
```

1.2.6.3 AccountViewModels.cs

```
namespace DienstenCheques.Models {
    public class ExternalLoginConfirmationViewModel {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }
    public class ExternalLoginListViewModel {
        public string ReturnUrl { get; set; }
    }
    public class SendCodeViewModel {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get;
set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }
    public class VerifyCodeViewModel {
        [Required]
        public string Provider { get; set; }
        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }
        [Display(Name = "Remember this browser?")]
        public bool RememberBrowser { get; set; }
        public bool RememberMe { get; set; }
    }
    public class ForgotViewModel {
        [Required]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }
    public class LoginViewModel {
        [Required]
        [Display(Name = "Email")]
        [EmailAddress]
        public string Email { get; set; }
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Wachtwoord")]
        public string Password { get; set; }
        [Display(Name = "Onthou me?")]
        public bool RememberMe { get; set; }
    }
    public class RegisterViewModel {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }
        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }
    public class ResetPasswordViewModel {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }
        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
        public string Code { get; set; }
    }
    public class ForgotPasswordViewModel {
        [Required]
        [EmailAddress]
        [Display(Name = "Email")]
        public string Email { get; set; }
    }
}
```

1.2.6.4 ManageViewModels.cs

```
namespace DienstenCheques.Models {
    public class IndexViewModel {
        public bool HasPassword { get; set; }
        public IList<UserLoginInfo> Logins { get; set; }
        public string PhoneNumber { get; set; }
        public bool TwoFactor { get; set; }
        public bool BrowserRemembered { get; set; }
    }
    public class ManageLoginsViewModel {
        public IList<UserLoginInfo> CurrentLogins { get; set; }
        public IList<AuthenticationDescription> OtherLogins { get; set; }
    }
    public class FactorViewModel {
        public string Purpose { get; set; }
    }
    public class SetPasswordViewModel {
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }
        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and
confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }
    public class ChangePasswordViewModel {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }
        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2}
characters long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }
        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and
confirmation password do not match.")]
        public string ConfirmPassword { get; set; }
    }
    public class AddPhoneNumberViewModel {
        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string Number { get; set; }
    }
    public class VerifyPhoneNumberViewModel {
        [Required]
        [Display(Name = "Code")]
        public string Code { get; set; }
        [Required]
        [Phone]
        [Display(Name = "Phone Number")]
        public string PhoneNumber { get; set; }
    }
    public class ConfigureTwoFactorViewModel {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get;
set; }
    }
}
```

1.2.7 Scripts

1.2.7.1 Bestellingen.js

```
var bestellingenView = {
    init: function() {
        $("#AantalMaanden").change(function () {
            $.post(this.action, $(this).serialize(), function(data) {
                $("#bestellingen").html(data);
            });
        });
    }
}

$(function() { bestellingenView.init(); });
```

1.2.8 Startup.cs

```
[assembly: OwinStartupAttribute(typeof(DienstenCheques.Startup))]
namespace DienstenCheques {
    public partial class Startup {
        public void Configuration(IAppBuilder app) {
            ConfigureAuth(app);
        }
    }
}
```

1.2.9 ViewModels

1.2.9.1 BestellingenViewModels.cs

```
namespace DienstenCheques.ViewModels.BestellingenViewModels {
    public class BestellingenViewModel {
        public IEnumerable<BestellingViewModel> Bestellingen { get;  set; }
        [Display(Name="Aantal beschikbare cheques")]
        public int AantalBeschikbareCheques { get; set; }
        [Display(Name = "Aantal niet betaalde prestaties (uren)")]
        public int AantalOpenstaandePrestatieUren { get; set; }
        public SelectList AantalMaandenKeuzeList { get; private set; }
        [Display(Name="Periode overzicht bestellingen")]
        public int AantalMaanden { get; set; }
        public BestellingenViewModel() {
            int[] maanden = new int[] {1, 2, 3, 6, 12, 18, 24};
            List<SelectListItem> items  = new List<SelectListItem>();
            foreach (int maand in maanden)
                items.Add(new SelectListItem(){Text = maand + " " + ((maand
== 1)? "maand":"maanden"), Value=maand.ToString()});
            AantalMaandenKeuzeList = new SelectList(items, "Value", "Text");
        }
    }
    public class BestellingViewModel
    {
        [Display(Name = "Referentie")]
        public int BestellingId { get; set; }
        [Display(Name = "Datum bestelling")]
        public DateTime CreatieDatum { get; set; }
        public bool Elektronisch { get; set; }
        [Display(Name = "Hoeveelheid")]
        public int AantalCheques { get; set; }
        [Display(Name = "Zichtwaarde")]
        public decimal Zichtwaarde { get; set; }
        [Display(Name = "Totaal bedrag")]
        public decimal TotaalBedrag { get; set; }
        public BestellingViewModel(Bestelling bestelling) {
            BestellingId = bestelling.BestellingId;
            CreatieDatum = bestelling.CreatieDatum;
            Elektronisch = bestelling.Elektronisch;
            AantalCheques = bestelling.AantalAangekochteCheques;
            Zichtwaarde = Bestelling.BEDRAGCHEQUE;
            TotaalBedrag = bestelling.TotaalBedrag;
        }
    }
    public class NieuweBestellingViewModel {
       public bool Elektronisch { get;  set; }
        [Required(ErrorMessage = "{0} is verplicht")]
        [Display(Name = "Aantal dienstencheques")]
        [Range(1, 50, ErrorMessage = "U kan maximaal 50 dienstencheques
bestellen")]
        public int AantalCheques { get;  set; }
        [Required(ErrorMessage = "{0} is verplicht")]
        [DataType(DataType.Date)]
        [Display(Name = "Datum debitering")]
        [DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}",
ApplyFormatInEditMode = true)]
        public DateTime DebiteerDatum { get; set; }
        public decimal Zichtwaarde { get; set; }
        public NieuweBestellingViewModel(decimal zichtwaarde) {
            DebiteerDatum = DateTime.Today;
            Elektronisch = true;
            AantalCheques = 20;
            Zichtwaarde = zichtwaarde;
        }
        public NieuweBestellingViewModel() : this(0){ }
    }
}
```

1.2.10 Views

1.2.10.1 Account

1.2.10.1.1 ConfirmEmail.cshtml

```
@{
    ViewBag.Title = "Confirm Email";
}

<h2>@ViewBag.Title.</h2>
<div>
    <p>
        Thank you for confirming your email. Please @Html.ActionLink("Click 
here to Log in", "Login", "Account", routeValues: null, htmlAttributes: new 
{ id = "loginLink" })
    </p>
</div>
```

1.2.10.1.2 ExternalLoginConfirmation.cshtml

```
@model DienstenCheques.Models.ExternalLoginConfirmationViewModel
@{
    ViewBag.Title = "Register";
}
<h2>@ViewBag.Title.</h2>
<h3>Associate your @ViewBag.LoginProvider account.</h3>

@using (Html.BeginForm("ExternalLoginConfirmation", "Account", new { 
ReturnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()

    <h4>Association Form</h4>
    <hr />
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    <p class="text-info">
        You've successfully authenticated with 
<strong>@ViewBag.LoginProvider</strong>.
        Please enter a user name for this site below and click the Register 
button to finish
        logging in.
    </p>
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" 
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.Email, "", new { @class = 
"text-danger" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}
```

```
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.3 ExternalLoginFailure.cshtml

```
@{
    ViewBag.Title = "Login Failure";
}

<hgroup>
    <h2>@ViewBag.Title.</h2>
    <h3 class="text-danger">Unsuccessful login with service.</h3>
</hgroup>
```

1.2.10.1.4 ForgotPassword.cshtml

```
@model DienstenCheques.Models.ForgotPasswordViewModel
@{
    ViewBag.Title = "Forgot your password?";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ForgotPassword", "Account", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Enter your email.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Email Link"
/>
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.5 ForgotPasswordConfirmation.cshtml

```
@{
    ViewBag.Title = "Forgot Password Confirmation";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
</hgroup>
<div>
    <p>
        Please check your email to reset your password.
    </p>
</div>
```

1.2.10.1.6 Login.cshtml

```
@using DienstenCheques.Models
@model LoginViewModel
@{
    ViewBag.Title = "Log in";
}

<h2>@ViewBag.Title.</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">
            @using (Html.BeginForm("Login", "Account", new { ReturnUrl =
ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role
= "form" }))
            {
                @Html.AntiForgeryToken()

                <hr />
                @Html.ValidationSummary(true, "", new { @class = "text-
danger" })
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, new { @class = "col-md-2
control-label" })
                    <div class="col-md-10">
                        @Html.TextBoxFor(m => m.Email, new { @class = "form-
control" })
                        @Html.ValidationMessageFor(m => m.Email, "", new {
@class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, new { @class = "col-md-2
control-label" })
                    <div class="col-md-10">
                        @Html.PasswordFor(m => m.Password, new { @class =
"form-control" })
```

```
                        @Html.ValidationMessageFor(m => m.Password, "", new
{ @class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <div class="checkbox">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Log in" class="btn btn-
default" />
                    </div>
                </div>
                <p>
                    @*@Html.ActionLink("Register as a new user",
"Register")*@
                </p>
                @* Enable this once you have account confirmation enabled
for password reset functionality
                    <p>
                        @Html.ActionLink("Forgot your password?",
"ForgotPassword")
                    </p>*@
            }
        </section>
    </div>
   <!-- <div class="col-md-4">
        <section id="socialLoginForm">
            @Html.Partial("_ExternalLoginsListPartial", new
ExternalLoginListViewModel { ReturnUrl = ViewBag.ReturnUrl })
        </section>
    </div>-->
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.7 Register.cshtml

```
@model DienstenCheques.Models.RegisterViewModel
@{
    ViewBag.Title = "Register";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class
= "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2
control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Register" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.8 ResetPassword.cshtml

```
@model DienstenCheques.Models.ResetPasswordViewModel
@{
    ViewBag.Title = "Reset password";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ResetPassword", "Account", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Reset your password.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    @Html.HiddenFor(model => model.Code)
    <div class="form-group">
        @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control"
})
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2
control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Reset" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.9 ResetPasswordConfirmation.cshtml

```
@{
    ViewBag.Title = "Reset password confirmation";
}

<hgroup class="title">
    <h1>@ViewBag.Title.</h1>
</hgroup>
<div>
    <p>
        Your password has been reset. Please @Html.ActionLink("click here to
log in", "Login", "Account", routeValues: null, htmlAttributes: new { id =
"loginLink" })
    </p>
```

```
</div>
```

1.2.10.1.10 SendCode.cshtml

```
@model DienstenCheques.Models.SendCodeViewModel
@{
    ViewBag.Title = "Send";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("SendCode", "Account", new { ReturnUrl =
Model.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role =
"form" })) {
    @Html.AntiForgeryToken()
    @Html.Hidden("rememberMe", @Model.RememberMe)
    <h4>Send verification code</h4>
    <hr />
    <div class="row">
        <div class="col-md-8">
            Select Two-Factor Authentication Provider:
            @Html.DropDownListFor(model => model.SelectedProvider,
Model.Providers)
            <input type="submit" value="Submit" class="btn btn-default" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.11 VerifyCode.cshtml

```
@model DienstenCheques.Models.VerifyCodeViewModel
@{
    ViewBag.Title = "Verify";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyCode", "Account", new { ReturnUrl =
Model.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role =
"form" })) {
    @Html.AntiForgeryToken()
    @Html.Hidden("provider", @Model.Provider)
    @Html.Hidden("rememberMe", @Model.RememberMe)
    <h4>Enter verification code</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <div class="checkbox">
                @Html.CheckBoxFor(m => m.RememberBrowser)
                @Html.LabelFor(m => m.RememberBrowser)
            </div>
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.1.12 _ExternalLoginsListPartial.cshtml

```
@model DienstenCheques.Models.ExternalLoginListViewModel
@using Microsoft.Owin.Security

<h4>Use another service to log in.</h4>
<hr />
@{
    var loginProviders =
Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();
    if (loginProviders.Count() == 0) {
        <div>
            <p>
                There are no external authentication services configured.
See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>
                for details on setting up this ASP.NET application to
support logging in via external services.
            </p>
        </div>
    }
    else {
        using (Html.BeginForm("ExternalLogin", "Account", new { ReturnUrl =
Model.ReturnUrl })) {
            @Html.AntiForgeryToken()
            <div id="socialLoginList">
                <p>
                    @foreach (AuthenticationDescription p in loginProviders)
{
                        <button type="submit" class="btn btn-default"
id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType"
title="Log in using your @p.Caption account">@p.AuthenticationType</button>
                    }
                </p>
            </div>
        }
    }
}
```

1.2.10.2 Bestellingen

1.2.10.2.1 Bestelling.cshtml

```
@model
IEnumerable<DienstenCheques.ViewModels.BestellingenViewModels.BestellingViewModel>


<p>
    @Html.ActionLink("Plaats een nieuwe bestelling", "Nieuw")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.BestellingId)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.CreatieDatum)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Elektronisch)
        </th>
        <th class="text-right">
            @Html.DisplayNameFor(model => model.AantalCheques)
        </th>
        <th class="text-right">
            @Html.DisplayNameFor(model => model.Zichtwaarde)
        </th>
        <th class="text-right">
            @Html.DisplayNameFor(model => model.TotaalBedrag)
        </th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.BestellingId)
        </td>
        <td>
            @item.CreatieDatum.ToShortDateString()
        </td>
        <td>
```

```
                @Html.DisplayFor(modelItem => item.Elektronisch)
            </td>
            <td class="text-right">
                @Html.DisplayFor(modelItem => item.AantalCheques)
            </td>
            <td class="text-right">
                @Html.DisplayFor(modelItem => item.Zichtwaarde) €
            </td>
            <td class="text-right">
                @Html.DisplayFor(modelItem => item.TotaalBedrag) €
            </td>
        </tr>
    }

    </table>
```

1.2.10.2.2 Index.cshtml

```
@model
DienstenCheques.ViewModels.BestellingenViewModels.BestellingenViewModel

@{
    ViewBag.Title = "Opvolging van bestellingen";
}

<h2>Opvolging van bestellingen</h2>
<p> </p>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">

        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.AantalBeschikbareCheques,
htmlAttributes: new { @class = "control-label col-md-3" })
            <div class="col-md-9">
                <p class="form-control-static"> @Html.DisplayFor(model =>
model.AantalBeschikbareCheques)</p>
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.AantalOpenstaandePrestatieUren,
htmlAttributes: new { @class = "control-label col-md-3" })
            <div class="col-md-9">
                <p class="form-control-static">@Html.DisplayFor(model =>
model.AantalOpenstaandePrestatieUren) u</p>
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.AantalMaanden, htmlAttributes: new
{ @class = "control-label col-md-3" })
```

```
            <div class="col-md-9">
                @Html.DropDownListFor(model => model.AantalMaanden,
Model.AantalMaandenKeuzeList) <input type="submit" value="Tonen" class="btn
btn-default" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-3 col-md-9">

            </div>
        </div>
    </div>
}

<div id="bestellingen">
    @Html.Partial("Bestelling", Model.Bestellingen)
</div>

@section Scripts{
    @Scripts.Render("~/bundles/scripts")
}
```

1.2.10.2.3 Nieuw.cshtml

```
@model
DienstenCheques.ViewModels.BestellingenViewModels.NieuweBestellingViewModel

@{
    ViewBag.Title = "Plaats een nieuwe bestelling";
}

<h2>Plaats een nieuwe bestelling voor dienstencheques met zichtwaarde
@Model.Zichtwaarde.ToString("C")</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Elektronisch, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                <div class="checkbox">
                    @Html.EditorFor(model => model.Elektronisch)
                    @Html.ValidationMessageFor(model => model.Elektronisch,
"", new { @class = "text-danger" })
                </div>
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.AantalCheques, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.AantalCheques, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.AantalCheques, "",
new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DebiteerDatum, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DebiteerDatum, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DebiteerDatum, "",
new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Plaats bestelling" class="btn
btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Terug naar overzicht", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.3 Home

1.2.10.3.1 About.cshtml

```
@{
    ViewBag.Title = "About";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<p>Use this area to provide additional information.</p>
```

1.2.10.3.2 Contact.cshtml

```
@{
    ViewBag.Title = "Contact";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<address>
    One Microsoft Way<br />
    Redmond, WA 98052-6399<br />
    <abbr title="Phone">P:</abbr>
    425.555.0100
</address>

<address>
    <strong>Support:</strong>   <a
href="mailto:Support@example.com">Support@example.com</a><br />
    <strong>Marketing:</strong> <a
href="mailto:Marketing@example.com">Marketing@example.com</a>
</address>
```

1.2.10.3.3 Index.cshtml

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>Dienstencheques</h1>
    <p>
        Een betaalmiddel waarvoor de overheid een financiële bijdrage
levert. Iedere meerderjarige privépersoon die in België gedomicilieerd is,
kan hiermee tegen een gunstige prijs gebruikmaken van prestaties op het
gebied van huishoudhulp.

        <ul>
            <li>Een gesubsidieerd betaalmiddel.</li>
            <li>Hulp bij huishoudelijk werk bij u thuis en buiten uw woning.
</li>
            <li>Talrijke voordelen.</li>
            <li>Een tarief van €9 per uur en een belastingvermindering</li>
            <li>Een vast salaris voor de huishoudhulpen.</li>
        </ul>

    <p><a href="http//www.dienstencheques-rva.be/" class="btn btn-primary
btn-lg">Meer &raquo;</a></p>
</div>
```

1.2.10.4 Manage

1.2.10.4.1 AddPhoneNumber.cshtml

```
@model DienstenCheques.Models.AddPhoneNumberViewModel
@{
    ViewBag.Title = "Phone Number";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("AddPhoneNumber", "Manage", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Add a phone number</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Number, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Number, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.4.2 ChangePassword.cshtml

```
@model DienstenCheques.Models.ChangePasswordViewModel
@{
    ViewBag.Title = "Change Password";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("ChangePassword", "Manage", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>Change Password Form</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.OldPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.OldPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.NewPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.NewPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2
control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Change password" class="btn btn-
default" />
        </div>
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.4.3 Index.cshtml

```
@model DienstenCheques.Models.IndexViewModel
@{
    ViewBag.Title = "Manage";
}

<h2>@ViewBag.Title.</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
<div>
    <h4>Change your account settings</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>Password:</dt>
        <dd>
            [
            @if (Model.HasPassword)
            {
                @Html.ActionLink("Change your password", "ChangePassword")
            }
            else
            {
                @Html.ActionLink("Create", "SetPassword")
            }
            ]
        </dd>
        <dt>External Logins:</dt>
        <dd>
            @Model.Logins.Count [
            @Html.ActionLink("Manage", "ManageLogins") ]
        </dd>
        @*
            Phone Numbers can used as a second factor of verification in a
two-factor authentication system.

             See <a href="http://go.microsoft.com/fwlink/?
LinkId=403804">this article</a>
                for details on setting up this ASP.NET application to
support two-factor authentication using SMS.

             Uncomment the following block after you have set up two-factor
authentication
        *@
        @*
            <dt>Phone Number:</dt>
            <dd>
                @(Model.PhoneNumber ?? "None") [
                @if (Model.PhoneNumber != null)
                {
                    @Html.ActionLink("Change", "AddPhoneNumber")
                    @: &nbsp;|&nbsp;
                    @Html.ActionLink("Remove", "RemovePhoneNumber")
                }
                else
                {
                    @Html.ActionLink("Add", "AddPhoneNumber")
                }
                ]
            </dd>
        *@
        <dt>Two-Factor Authentication:</dt>
        <dd>
            <p>
                There are no two-factor authentication providers configured.
See <a href="http://go.microsoft.com/fwlink/?LinkId=403804">this article</a>
                for details on setting up this ASP.NET application to
support two-factor authentication.
            </p>
            @*@if (Model.TwoFactor)
                {
                    using (Html.BeginForm("DisableTwoFactorAuthentication",
"Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form"
}))
                    {
                        @Html.AntiForgeryToken()
                        <text>Enabled
                        <input type="submit" value="Disable" class="btn btn-
link" />
                        </text>
                    }
                }
                else
                {
                    using (Html.BeginForm("EnableTwoFactorAuthentication",
"Manage", FormMethod.Post, new { @class = "form-horizontal", role = "form"
}))
                    {
                        @Html.AntiForgeryToken()
                        <text>Disabled
                        <input type="submit" value="Enable" class="btn btn-
link" />
                        </text>
                    }
                }*@
        </dd>
    </dl>
</div>
```

1.2.10.4.4 ManageLogins.cshtml

```
@model DienstenCheques.Models.ManageLoginsViewModel
@using Microsoft.Owin.Security
@{
    ViewBag.Title = "Manage your external logins";
}

<h2>@ViewBag.Title.</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
@{
    var loginProviders =
Context.GetOwinContext().Authentication.GetExternalAuthenticationTypes();
    if (loginProviders.Count() == 0) {
        <div>
            <p>
                There are no external authentication services configured.
See <a href="http://go.microsoft.com/fwlink/?LinkId=313242">this article</a>
                for details on setting up this ASP.NET application to
support logging in via external services.
            </p>
        </div>
    }
    else
    {
        if (Model.CurrentLogins.Count > 0)
        {
            <h4>Registered Logins</h4>
            <table class="table">
                <tbody>
```

```
                        @foreach (var account in Model.CurrentLogins)
                        {
                            <tr>
                                <td>@account.LoginProvider</td>
                                <td>
                                    @if (ViewBag.ShowRemoveButton)
                                    {
                                        using (Html.BeginForm("RemoveLogin",
"Manage"))
                                        {
                                            @Html.AntiForgeryToken()
                                            <div>
                                                @Html.Hidden("loginProvider",
account.LoginProvider)
                                                @Html.Hidden("providerKey",
account.ProviderKey)
                                                <input type="submit" class="btn
btn-default" value="Remove" title="Remove this @account.LoginProvider login
from your account" />
                                            </div>
                                        }
                                    }
                                    else
                                    {
                                        @: &nbsp;
                                    }
                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            }
            if (Model.OtherLogins.Count > 0)
            {
                using (Html.BeginForm("LinkLogin", "Manage"))
                {
                    @Html.AntiForgeryToken()
                    <div id="socialLoginList">
                    <p>
                        @foreach (AuthenticationDescription p in
Model.OtherLogins)
                        {
                            <button type="submit" class="btn btn-default"
id="@p.AuthenticationType" name="provider" value="@p.AuthenticationType"
title="Log in using your @p.Caption account">@p.AuthenticationType</button>
                        }
                    </p>
                    </div>
                }
            }
        }
    }
```

1.2.10.4.5 SetPassword.cshtml

```
@model DienstenCheques.Models.SetPasswordViewModel
@{
    ViewBag.Title = "Create Password";
}

<h2>@ViewBag.Title.</h2>
<p class="text-info">
    You do not have a local username/password for this site. Add a local
    account so you can log in without an external login.
</p>

@using (Html.BeginForm("SetPassword", "Manage", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()

    <h4>Create Local Login</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.NewPassword, new { @class = "col-md-2 control-
label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.NewPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2
control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-
control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="Set password" class="btn btn-
default" />
        </div>
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.4.6 VerifyPhoneNumber.cshtml

```
@model DienstenCheques.Models.VerifyPhoneNumberViewModel
@{
    ViewBag.Title = "Verify Phone Number";
}
```

```
<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyPhoneNumber", "Manage", FormMethod.Post, new {
@class = "form-horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    @Html.Hidden("phoneNumber", @Model.PhoneNumber)
    <h4>Enter verification code</h4>
    <h5>@ViewBag.Status</h5>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label"
})
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-default" value="Submit" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

1.2.10.5 Shared

1.2.10.5.1 Error.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.
</h2>
```

1.2.10.5.2 Lockout.cshtml

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Locked Out";
}

<hgroup>
    <h1 class="text-danger">Locked out.</h1>
    <h2 class="text-danger">This account has been locked out, please try
again later.</h2>
</hgroup>
```

1.2.10.5.3 _Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("DIENSTENCHEQUES", "Index", "Home", new {
area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("Opvolgen Bestellingen", "Index",
"Bestellingen")</li>
                    <li>@Html.ActionLink("Over ons", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
<div class="container body-content">
    @if (TempData["message"] != null)
    {
        <div class="alert alert-success">@TempData["message"]</div>
    }
    @if (TempData["error"] != null)
    {
        <div class="alert alert-warning">@TempData["message"]</div>
    }
    @RenderBody()
    <hr/>
    <footer>
        <p>&copy; @DateTime.Now.Year - Hogeschool Gent - Examen Web III</p>
    </footer>
</div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

1.2.10.5.4 _LoginPartial.cshtml

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account", FormMethod.Post, new { id =
"logoutForm", @class = "navbar-right" }))
    {
    @Html.AntiForgeryToken()

    <ul class="nav navbar-nav navbar-right">
        <li>
            @Html.ActionLink("Hello " + User.Identity.GetUserName() + "!",
"Index", "Manage", routeValues: null, htmlAttributes: new { title = "Manage"
})
        </li>
        <li><a
href="javascript:document.getElementById('logoutForm').submit()">Log off</a>
</li>
    </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
      <!--  <li>@Html.ActionLink("Register", "Register", "Account",
routeValues: null, htmlAttributes: new { id = "registerLink" })</li>-->
        <li>@Html.ActionLink("Log in", "Login", "Account", routeValues:
null, htmlAttributes: new { id = "loginLink" })</li>
    </ul>
}
```

1.2.10.6 Web.config

```
<?xml version="1.0"?>
```

```
<configuration>
  <configSections>
    <sectionGroup name="system.web.webPages.razor"
type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
      <section name="host"
type="System.Web.WebPages.Razor.Configuration.HostSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
      <section name="pages"
type="System.Web.WebPages.Razor.Configuration.RazorPagesSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
    </sectionGroup>
  </configSections>

  <system.web.webPages.razor>
    <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory,
System.Web.Mvc, Version=5.2.2.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" />
    <pages pageBaseType="System.Web.Mvc.WebViewPage">
      <namespaces>
        <add namespace="System.Web.Mvc" />
        <add namespace="System.Web.Mvc.Ajax" />
        <add namespace="System.Web.Mvc.Html" />
        <add namespace="System.Web.Optimization"/>
        <add namespace="System.Web.Routing" />
        <add namespace="DienstenCheques" />
      </namespaces>
    </pages>
  </system.web.webPages.razor>

  <appSettings>
    <add key="webpages:Enabled" value="false" />
  </appSettings>

  <system.webServer>
    <handlers>
      <remove name="BlockViewHandler"/>
      <add name="BlockViewHandler" path="*" verb="*"
preCondition="integratedMode" type="System.Web.HttpNotFoundHandler" />
    </handlers>
  </system.webServer>
</configuration>
```

1.2.10.7 _ViewStart.cshtml

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

1.2.11 Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please
visit
  http://go.microsoft.com/fwlink/?LinkId=301880
  -->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <connectionStrings>
     <add name="DefaultConnection"
connectionString="Server=localhost\sqlexpress;
Database=DienstenChequesUsers;Integrated Security=true"
providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>
    <authentication mode="None" />
    <compilation debug="true" targetFramework="4.5.1" />
    <httpRuntime targetFramework="4.5.1" />
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthentication" />
    </modules>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.OAuth"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security.Cookies"
```

```
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" culture="neutral"
publicKeyToken="30ad4fe6b2a6aeed" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-5.2.2.0" newVersion="5.2.2.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Optimization"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/>
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35"
/>
        <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234"
newVersion="1.5.2.14234" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <entityFramework>
        <contexts>
    <context type="DienstenCheques.Models.DAL.DienstenChequesContext,
DienstenCheques">
      <databaseInitializer
type="DienstenCheques.Models.DAL.DienstenChequesInitializer,
DienstenCheques" />
    </context>
    </contexts>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.SqlConnectionFactory,
EntityFramework" />
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices,
EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

1.3 DienstenCheques.Tests

1.3.1 Controllers

1.3.1.1 BestellingenControllerTest.cs

```
namespace DienstenCheques.Tests.Controllers {
    [TestClass]
    public class BestellingenControllerTest {
        private BestellingenController controller;
        private Gebruiker jan;
        private Mock<IGebruikersRepository> mockGebruikersRepository;
        private NieuweBestellingViewModel model;
        private NieuweBestellingViewModel modelMetFout;
        [TestInitialize]
        public void SetUpContext() {
            DummyContext context = new DummyContext();
            mockGebruikersRepository = new Mock<IGebruikersRepository>();
            jan = context.Jan;
            controller = new BestellingenController(
                mockGebruikersRepository.Object
            );
            model = new NieuweBestellingViewModel(9.0M) {
                Elektronisch = true,
                AantalCheques = 20,
                DebiteerDatum = DateTime.Today
            };
            modelMetFout = new NieuweBestellingViewModel(9.0M) {
                Elektronisch = true,
                AantalCheques = 70,
                DebiteerDatum = DateTime.Today
            };
        }
        #region "Index"
        [TestMethod]
        public void Index6MaandenRetourneertModel() {
            ViewResult result = controller.Index(jan) as ViewResult;
            BestellingenViewModel bestellingenViewModel = (
                (BestellingenViewModel) result.Model
            );
            Assert.AreEqual(
                1,
                bestellingenViewModel.AantalBeschikbareCheques
            );
            Assert.AreEqual(
                8,
                bestellingenViewModel.AantalOpenstaandePrestatieUren
            );
            Assert.AreEqual(
                1,
                bestellingenViewModel.Bestellingen.Count()
            );
            Assert.AreEqual(
                DateTime.Today.AddMonths(-4),
                bestellingenViewModel.Bestellingen
                                     .ToArray()[0]
                                     .CreatieDatum
            );
        }
        [TestMethod]
        public void Index12MaandenRetourneertModel() {
            ViewResult result = controller.Index(jan, 12) as ViewResult;
            BestellingenViewModel bestellingenViewModel = (
                (BestellingenViewModel) result.Model
            );
            Assert.AreEqual(
                1,
                bestellingenViewModel.AantalBeschikbareCheques
            );
            Assert.AreEqual(
                8,
                bestellingenViewModel.AantalOpenstaandePrestatieUren
            );
            Assert.AreEqual(
                3,
                bestellingenViewModel.Bestellingen.Count()
            );
            Assert.AreEqual(
                DateTime.Today.AddMonths(-12),
                bestellingenViewModel.Bestellingen
                                     .Last()
                                     .CreatieDatum
            );
        }
        #endregion
        #region Nieuw
        [TestMethod]
        public void NieuwReturnsNieuweBestellingViewModel() {
            ViewResult result = controller.Nieuw(jan) as ViewResult;
            NieuweBestellingViewModel nieuweBestellingViewModel =
                result.Model as NieuweBestellingViewModel;
            Assert.AreEqual(20, nieuweBestellingViewModel.AantalCheques);
            Assert.AreEqual(9.0M, nieuweBestellingViewModel.Zichtwaarde);
            Assert.AreEqual(
                DateTime.Today,
                nieuweBestellingViewModel.DebiteerDatum
            );
            Assert.IsTrue(nieuweBestellingViewModel.Elektronisch);
        }
        #endregion
        #region HttpPost Nieuw
        [TestMethod]
        public void NieuwPostReturnsToIndexWhenUpdateSuccessfull() {
            RedirectToRouteResult result =
                controller.Nieuw(jan, model) as RedirectToRouteResult;
            Assert.AreEqual("Index", result.RouteValues["action"]);
        }
        [TestMethod]
        public void NieuwPostVoegtBestellingToe() {
            int aantal = jan.Bestellingen.Count;
            controller.Nieuw(jan, model);
            Assert.AreEqual(aantal+1, jan.Bestellingen.Count);
            mockGebruikersRepository.Verify(
                m => m.SaveChanges(),
                Times.Once()
            );
        }
        [TestMethod]
        public void NieuwPostVoegtBestellingNietToeBijFout() {
            controller.Nieuw(jan, modelMetFout);
            Assert.AreEqual(3, jan.Bestellingen.Count);
            mockGebruikersRepository.Verify(
                m => m.SaveChanges(),
                Times.Never
            );
        }
        [TestMethod]
        public void NieuwPostRetourneertViewBijFout() {
            ViewResult result =  controller.Nieuw(
                jan, modelMetFout
            ) as ViewResult;
            NieuweBestellingViewModel nieuweBestellingViewModel =
                ((NieuweBestellingViewModel)result.Model);
            Assert.AreEqual(70, nieuweBestellingViewModel.AantalCheques);
            Assert.AreEqual(
                DateTime.Today,
                nieuweBestellingViewModel.DebiteerDatum
            );
            Assert.AreEqual(true, nieuweBestellingViewModel.Elektronisch);
            Assert.AreEqual(9.0M, nieuweBestellingViewModel.Zichtwaarde);
        }
        #endregion
    }
}
```

1.3.1.2 DummyContext.cs

```
namespace DienstenCheques.Tests.Controllers {
    class DummyContext {
        public Gebruiker Jan { get; set; }
        public Gebruiker An { get; set; }
        public Gebruiker Tine { get; set; }
        public DummyContext() {
            Onderneming onderneming = new Onderneming() {
                Naam = "Hogeschool Gent"
            };
            //Nog 2 openstaande prestaties, 1 cheque over
            Jan = new Gebruiker() {
                GebruikersNummer = "1000000000",
                Naam = "Peeters",
                Voornaam = "Jan",
                Email = "jan.peeters@hogent.be"
            };
            for (int i = 12; i >= 0; i--) {
                Prestatie pres = new Prestatie() {
                    AantalUren = 4,
                    DatumPrestatie = DateTime.Today.AddMonths(-i),
                    PrestatieType = PrestatieType.Schoonmaken,
                    Onderneming = onderneming,
                    Betaald = true
                };
                Jan.Prestaties.Add(pres);
            }
            Jan.GetPrestatie(11).Betaald = false;
            Jan.GetPrestatie(12).Betaald = false;
            int p = 0;
            int pi = 0;
            for (int i = 3; i > 0; i--) {
                Bestelling b = new Bestelling() {
                    AantalAangekochteCheques = 15,
                    Elektronisch = true
                };
                b.StelDatumsIn(
                    DateTime.Today.AddMonths(-4 * i),
                    DateTime.Today.AddMonths(-4 * i)
                );
                Jan.Bestellingen.Add(b);
                for (int j = 1; j <= 15; j++) {
                    DienstenCheque d = new DienstenCheque(
                        true,
                        DateTime.Today.AddMonths(-4 * i)
                    );
                    if (p < 11) {
                        d.Prestatie = Jan.GetPrestatie(p);
                        d.GebruiksDatum = d.Prestatie.DatumPrestatie;
                    }
                    Jan.Portefeuille.Add(d);
                    if (pi < 3) {
                        pi++;
                    } else {
                        pi = 0;
                        p++;
                    }
                }
            }
            // Alle cheques zijn toegewezen aan prestaties,
            // geen openstaande prestaties meer
            An = new Gebruiker() {
```

```
        GebruikersNummer = "1000000001",
        Naam = "Pieters",
        Voornaam = "An",
        Email = "an.pieters@hogent.be"
    };
    Bestelling anBestelling = new Bestelling() {
        AantalAangekochteCheques = 20,
        Elektronisch = true
    };
    anBestelling.StelDatumsIn(
        DateTime.Today.AddMonths(-1),
        DateTime.Today.AddMonths(-1)
    );
    An.Bestellingen.Add(anBestelling);
    for (int i = 4; i > 0; i--) {
        An.Prestaties.Add(new Prestatie() {
            AantalUren = 5,
            DatumPrestatie = DateTime.Today.AddMonths(-i),
            PrestatieType = PrestatieType.Schoonmaken,
            Onderneming = onderneming,
            Betaald = true
        });
    }
    for (int j = 0; j <= 19; j++) {
        DienstenCheque d = new DienstenCheque(
            true,
            DateTime.Today.AddMonths(-1)
        );
        d.Prestatie = An.GetPrestatie(j / 5);
        d.GebruiksDatum = d.Prestatie.DatumPrestatie;
        An.Portefeuille.Add(d);
    }
    //nog 2 cheques niet gebruikt, geen openstaande prestaties
    Tine = new Gebruiker() {
        GebruikersNummer = "1000000002",
        Naam = "Pieters",
        Voornaam = "Tine",
        Email = "tine.pieters@hogent.be"
    };
    Bestelling tineBestelling = new Bestelling() {
        AantalAangekochteCheques = 6,
        Elektronisch = true
    };
    tineBestelling.StelDatumsIn(
        DateTime.Today.AddMonths(-1),
        DateTime.Today.AddMonths(-1)
    );
    Tine.Bestellingen.Add(tineBestelling);
    Tine.Prestaties.Add(new Prestatie() {
        AantalUren = 4,
        DatumPrestatie = DateTime.Today.AddDays(-10),
        PrestatieType = PrestatieType.Schoonmaken,
        Onderneming = onderneming,
        Betaald = true
    });
    for (int j = 1; j <= 6; j++) {
        DienstenCheque d = new DienstenCheque(
                    true,
                    DateTime.Today.AddMonths(-1)
                );
                if (j < 5) {
                    d.Prestatie = Tine.GetPrestatie(0);
                    d.GebruiksDatum = d.Prestatie.DatumPrestatie;
                }
                Tine.Portefeuille.Add(d);
            }
        }
    }
}
```

1.3.1.3 HomeControllerTest.cs

```
namespace DienstenCheques.Tests.Controllers {
    [TestClass]
    public class HomeControllerTest {
        [TestMethod]
        public void Index() {
            // Arrange
            HomeController controller = new HomeController();
            // Act
            ViewResult result = controller.Index() as ViewResult;
            // Assert
            Assert.IsNotNull(result);
        }
        [TestMethod]
        public void About() {
            // Arrange
            HomeController controller = new HomeController();
            // Act
            ViewResult result = controller.About() as ViewResult;
            // Assert
            Assert.AreEqual(
                "Your application description page.",
                result.ViewBag.Message
            );
        }
        [TestMethod]
        public void Contact() {
            // Arrange
            HomeController controller = new HomeController();
            // Act
            ViewResult result = controller.Contact() as ViewResult;
            // Assert
            Assert.IsNotNull(result);
        }
    }
}
```

1.3.2 Models

1.3.2.1 BestellingTest.cs

```
namespace DienstenCheques.Tests.Models {
    [TestClass]
    public class BestellingTest {
        [TestMethod]
        public void BestellingAanmakenSlaagt() {
            Bestelling b = new Bestelling(10, true, DateTime.Today);
            Assert.AreEqual(10, b.AantalAangekochteCheques);
            Assert.IsTrue(b.Elektronisch);
            Assert.AreEqual(DateTime.Today, b.CreatieDatum);
        }
        [ExpectedException(typeof(ArgumentException))]
        [TestMethod]
        public void BestellingAanmakenAantalFoutief() {
            Bestelling b = new Bestelling(70, true, DateTime.Today);
        }
        [ExpectedException(typeof(ArgumentException))]
        [TestMethod]
        public void BestellingAanmakenDebiteerDatumFoutief() {
            Bestelling b = new Bestelling(
                -10,
                true,
                DateTime.Today.AddMonths(2)
            );
        }
        [TestMethod]
        public void TotaalBedrag() {
            Bestelling b = new Bestelling(10, true, DateTime.Today);
            Assert.AreEqual(90, b.TotaalBedrag);
        }
    }
}
```

1.3.2.2 DienstenChequeTest.cs

```
namespace DienstenCheques.Tests.Models {
    [TestClass]
    public class DienstenChequeTest {
        [TestMethod]
        public void DienstenChequeAanmakenSlaagt() {
            DienstenCheque dc = new DienstenCheque(true);
            Assert.IsTrue(dc.Elektronisch);
            Assert.AreEqual(DateTime.Today, dc.CreatieDatum);
            Assert.IsNull(dc.Prestatie);
            Assert.IsNull(dc.GebruiksDatum);
        }
    }
}
```

1.3.2.3 GebruikerTest.cs

```
namespace DienstenCheques.Tests.Models {
    [TestClass]
    public class GebruikerTest {
        DummyContext context;
        [TestInitialize]
        public void Initialize() {
            context = new DummyContext();
        }
        [TestMethod]
        public void AddBestellingVoegtBestellingToe() {
            context.Jan.AddBestelling(20, true, DateTime.Today);
            Assert.AreEqual(4, context.Jan.Bestellingen.Count);
        }
        [TestMethod]
        public void AddBestellingVoegtDienstenChequesToeAanPortefeuille() {
            int aantal = context.Jan.Portefeuille.Count;
            context.Jan.AddBestelling(20, true, DateTime.Today);
            Assert.AreEqual(aantal+20,context.Jan.Portefeuille.Count);
        }
        [TestMethod]
        public void AddBestellingPastOpenstaandePrestatiesAan() {
            context.Jan.AddBestelling(20, true, DateTime.Today);
            Assert.AreEqual(0, context.Jan.AantalOpenstaandePrestatieUren);
        }
        [TestMethod]
        public void AddBestellingPastBeschikbareDienstenChequesAan() {
            context.Jan.AddBestelling(20, true, DateTime.Today);
            Assert.AreEqual(
                13,
                context.Jan.AantalBeschikbareElektronischeCheques
            );
        }
    }
}
```