 **Технически Университет – София**

**Катедра „Компютърни системи”**

**Специалност „Компютърни Системи и Технологии”, магистър**

**индивидуално задание за проект**

**по Програмиране за Интернет**

**Студент: Жико Митов**

**ФАК. No: 121315014 Група: 222**

|  |
| --- |
| **Тема** |
| Разработване на уеб-базирано приложение за управление на файлове в облака. |

1. Да се проектира база от данни за това приложение.
2. Да се проектира и реализира приложение на C#, което работи със създадената база от данни.
3. От приложението се очаква да има графичен потребителски интерфейс.

София, Подпис на преподавателя:**…………………………………**

2015/2016 (гл. ас. д-р Антония Ташева)

Подпис на студента:**…………………………………**

Проект за управление на файлове в облака

Жико Митов

фак. № 121315014, група 222

e-mail: [zmitov@outlook.com](mailto:zmitov@outlook.com)

*Резюме: Облачно приложение, което позволява менажиране на файлове, включващо потребители, достъпност, съобщения и други*

Ключови думи: web, storage, html5, asp.net, mvc, .net, c#, entity, file, upload

Съдържание

[I. Описание на възможностите на реализирания проект 5](#_Toc440835583)

[A. Роли. 5](#_Toc440835584)

[1) Потребител 5](#_Toc440835585)

[2) Администратори 5](#_Toc440835586)

[B. Файлове мениджър 5](#_Toc440835587)

[1) Качване на файлове с асинхронен файлове мениджър – поддържат пауза, спиране, стартиране на множество файлове. 5](#_Toc440835588)

[2) Менажиране на пространството с директории като файлова система. 5](#_Toc440835589)

[3) Ъплоадване на файлове в текуща папка 5](#_Toc440835590)

[4) Сортировка на файлове 5](#_Toc440835591)

[5) Управление на директори 5](#_Toc440835592)

[6) Преименуване на файл/директория 5](#_Toc440835593)

[7) „Изтриване“ на файл/директория 6](#_Toc440835594)

[8) Сваляне 6](#_Toc440835595)

[9) Навигационно поле (бреадкум) с възможност преминаване в родителски папки. 6](#_Toc440835596)

[C. Търсене във всички директории и файлове 6](#_Toc440835597)

[D. Споделяне 6](#_Toc440835598)

[1) Избиране на достъпност за файл/директория 6](#_Toc440835599)

[2) Наследяване на достъпност от родителя (рекурсовно) 6](#_Toc440835600)

[3) Възможност за презаписване на родителските пермисии 6](#_Toc440835601)

[4) Публични, частни, по избор (по потребителско име) 6](#_Toc440835602)

[E. Администрация 6](#_Toc440835603)

[1) Потребителски файлове 6](#_Toc440835604)

[2) Заемано пространство 6](#_Toc440835605)

[3) Администриране на потребители ( смяна на роли) 6](#_Toc440835606)

[4) Съобщения 6](#_Toc440835607)

[5) Логове 6](#_Toc440835608)

[II. Asp.net 6](#_Toc440835609)

[A. Какво представлява 6](#_Toc440835610)

[B. Извършени оптимизации в приложението 6](#_Toc440835611)

[III. Microsoft Azure 7](#_Toc440835612)

[A. Azure blob storage 7](#_Toc440835613)

[1) Storage Services 7](#_Toc440835614)

[2) Сваляне на файл 7](#_Toc440835615)

[3) Качване на файл, 7](#_Toc440835616)

[IV. Entity Framework 7](#_Toc440835617)

[A. Моделите използване за създаването на базаста данни: 7](#_Toc440835618)

[1) Абстрактен клас за „изтриващ се“ модел 7](#_Toc440835619)

[2) Потребител 8](#_Toc440835620)

[3) Файл 8](#_Toc440835621)

[4) Лог 9](#_Toc440835622)

[5) Съобщение 9](#_Toc440835623)

[6) Достъпност 9](#_Toc440835624)

[B. Миграции 10](#_Toc440835625)

[C. Хранилище за изтрити модели 10](#_Toc440835626)

[V. Microsoft SQL Server 2014 11](#_Toc440835627)

[A. Схема на таблиците и релациите им в Базата данни ( генерирана чрез Code-First). 12](#_Toc440835628)

[VI. Asp.net MVC 12](#_Toc440835629)

[A. Модел 13](#_Toc440835630)

[B. Изглед модел (view model) 13](#_Toc440835631)

[C. MVC модели 13](#_Toc440835632)

[1) Директория (във файловия мениджър) 13](#_Toc440835633)

[2) Навигационна директория – за breadcum 13](#_Toc440835634)

[D. Контролери и действия 13](#_Toc440835635)

[1) Какво са контролери 13](#_Toc440835636)

[2) Основни контролери в приложението: 14](#_Toc440835637)

[E. Валидация 14](#_Toc440835638)

[1) Какво е валидиране 14](#_Toc440835639)

[2) Как работи валидирането 14](#_Toc440835640)

[3) Пример за валидация използвано чрез view model-а 15](#_Toc440835641)

[F. Изгледи 15](#_Toc440835642)

[1) Какво е изглед 15](#_Toc440835643)

[2) Начин на работа на изгледите 15](#_Toc440835644)

[3) Формата за качване на файлове (изглед) 15](#_Toc440835645)

[G. Рутиране 16](#_Toc440835646)

[1) Какво е рутиране 16](#_Toc440835647)

[2) Как се регистрира и дефинира рутирането 16](#_Toc440835648)

[H. Филтри на дейсвията 17](#_Toc440835649)

[1) Какво са филтрите 17](#_Toc440835650)

[2) Филтър за следене на потребителската дейсност 17](#_Toc440835651)

[3) Пример: 17](#_Toc440835652)

[VII. Automapper 17](#_Toc440835653)

[A. Създаване на mapping 17](#_Toc440835654)

[B. Как се използва 18](#_Toc440835655)

[VIII. Ninject 18](#_Toc440835656)

[A. Регистриране на базата данни за автоматично инжектиране при използване: 18](#_Toc440835657)

[B. Използване от контролер: 18](#_Toc440835658)

[IX. JavaScript 18](#_Toc440835659)

[A. Възможности 19](#_Toc440835660)

[1) Зареждане на данни чрез AJAX 19](#_Toc440835661)

[2) Ефекти с изображения и HTML елементи: скриване/показване, пренареждане, влачене, слайд шоу, анимация и много други. 19](#_Toc440835662)

[3) Управление на прозорци и рамки. 19](#_Toc440835663)

[4) Разпознаване на възможностите на браузъра. 19](#_Toc440835664)

[5) Използване на камерата и микрофона. 19](#_Toc440835665)

[6) Създаване на 3D графики WebGL. 19](#_Toc440835666)

[7) По-добър и гъвкав потребителски интерфейс 19](#_Toc440835667)

[B. Ограничения: 19](#_Toc440835668)

[1) Не може да се записва информация на потребителския компютър или отдалечения сървър. 19](#_Toc440835669)

[2) Не може да се запазва информация директно в отдалечена база данни. 19](#_Toc440835670)

[3) Не може да се стартират локални приложения. 19](#_Toc440835671)

[C. JavaScript библиотека 19](#_Toc440835672)

[1) Jquery - функционалност 19](#_Toc440835673)

[2) Селектиране на файл или директория 19](#_Toc440835674)

[3) KendoUI – grid 21](#_Toc440835675)

[X. Bootstrap 22](#_Toc440835676)

[A. Поддръжка на препроцесори 22](#_Toc440835677)

[B. Responsive сред всякакви устроиства и резолюции. 22](#_Toc440835678)

[C. Пример за модален диалог 23](#_Toc440835679)

[XI. Font-awesome 23](#_Toc440835680)

[A. 605 Икони 23](#_Toc440835681)

[B. Не изисква JavaScript 23](#_Toc440835682)

[C. Безкрайно скалиране 23](#_Toc440835683)

[D. Контрол чрез CSS 23](#_Toc440835684)

[E. Поддръжка от много фреймурки 23](#_Toc440835685)

[F. Пример 23](#_Toc440835686)

[XII. Microsoft .NET Framework 24](#_Toc440835687)

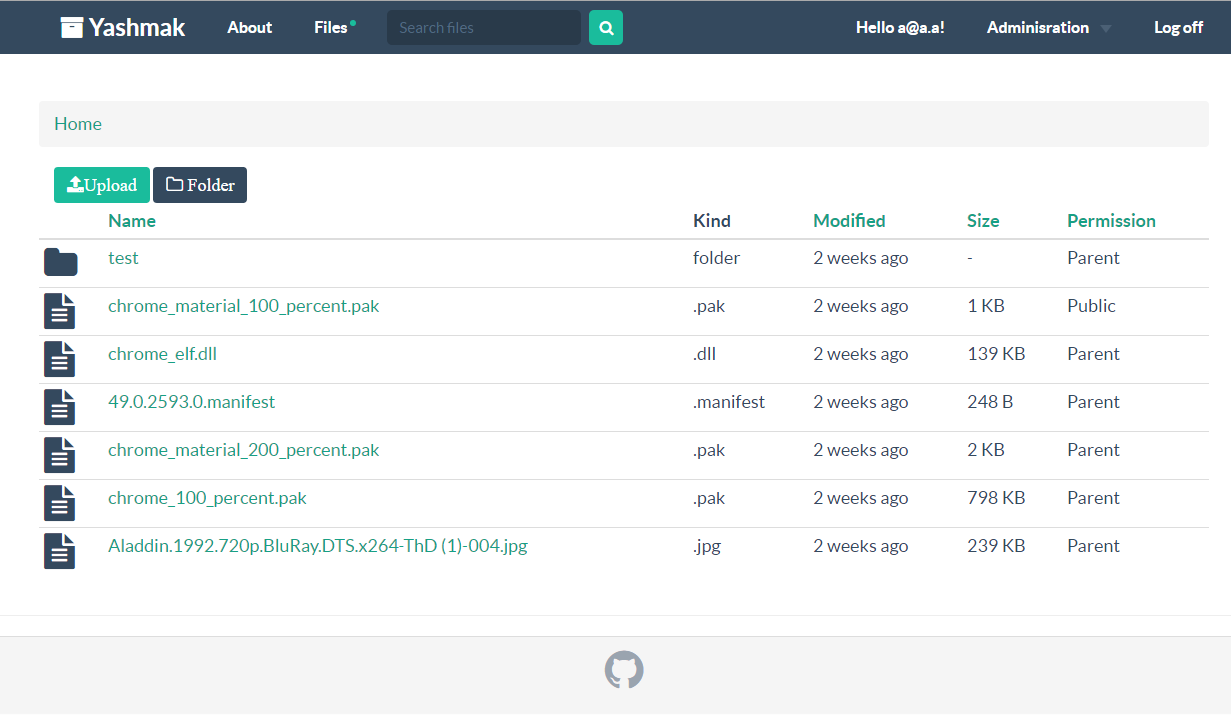
[A. Компоненти на .NET Framework 24](#_Toc440835688)

[1) Common Language Runtime (CLR) 24](#_Toc440835689)

[2) Framework Class Library (FCL) 24](#_Toc440835690)

[XIII. C# 24](#_Toc440835691)

# Описание на възможностите на реализирания проект

****

## Роли.

### Потребител

#### възможност за регистрация с емайл и парола

#### получаване на собствено виртуален диск

#### възможност за ползване на онлайн файлов мениджър

#### получаване на съобщения

### Администратори

#### Менажиране на потребители

#### Възможност за смяна на роли на потребителите

#### Следене на всички потребители

#### Разглеждане на логове

#### Следене на потребителската дейсност

#### Преглеждане на потребителски файлове

#### Изпращане на съобщения до потребители

#### Преглеждане на изтрити файлове

## Файлове мениджър

### *Качване на файлове с асинхронен файлове мениджър – поддържат пауза, спиране, стартиране на множество файлове.*

### Менажиране на пространството с директории като файлова система.

### Ъплоадване на файлове в текуща папка

### Сортировка на файлове

### Управление на директори

### Преименуване на файл/директория

### „Изтриване“ на файл/директория

### Сваляне

#### Сваляне на файл

#### Сваляне на директория като архив

### Навигационно поле (бреадкум) с възможност преминаване в родителски папки.

## Търсене във всички директории и файлове

## Споделяне

### Избиране на достъпност за файл/директория

### Наследяване на достъпност от родителя (рекурсовно)

### Възможност за презаписване на родителските пермисии

### Публични, частни, по избор (по потребителско име)

## Администрация

### Потребителски файлове

### Заемано пространство

### Администриране на потребители ( смяна на роли)

### Съобщения

### Логове

# Asp.net

## Какво представлява

ASP.NET е следващото поколение уеб рамка (framework), разработена от "[Майкрософт](https://bg.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82)". За първи път е публикуван през януари 2002 година с версия 1.0 на [.NET Framework](https://bg.wikipedia.org/wiki/.NET_Framework), и е наследник на Microsoft Active Server Pages ([ASP](https://bg.wikipedia.org/wiki/ASP)) технология, но да не се бърка, че не е подобрена версия на ASP. ASP.NET е изградена въз основа на Common Language Runtime ([CLR](https://bg.wikipedia.org/w/index.php?title=CLR&action=edit&redlink=1)), което позволява на програмистите да пишат ASP.NET код като използват .NET език по избор. За първи път е публикуван през януари 2002 година с версия 1.0 на. NET Framework, и е най-успешен наследник на Microsoft Active Server Pages (ASP) технология.

ASP.NET цели производителност спрямо останалите скрипт-базирани технологии (включително класическия ASP) като компилира сървърно кода в един или повече DLL файлове на Уеб сървъра. Тази компилация става автоматично когато страницата бива заредена за пръв път (което от своя страна означава, че програмистът не трябва да изпълнява отделни компилации за страниците). //Тази характеристика осигурява лекота на разработване предлагана от скриптовите езици с производителността на бинарните операции. Трябва да се има предвид обаче, че самата компилация може да причини забележимо забавяне при потребителя когато редактираната страница бива изискана за пръв път от Уеб сървъра, но това забавяне не би се появило отново преди следваща промяна.

## Извършени оптимизации в приложението

Asp.Net има два view engine, от които се използва само Razor. За да се оптимизира работата на приложението е премахнат view engine-a на Web Forms.

public static void RegisterEngines(ViewEngineCollection engines)

{

    engines.Clear();

    engines.Add(new RazorViewEngine());

}

# Microsoft Azure

Microsoft Azure е платформа и инфраструктора за облачни изчисления, създадена за изграждане, деплойване и менажиране на приложения и севизи през глобалната мрежа от манажирани от Microsft и партоньори – дейтацентрове. Осигурява едновременно  [PaaS](https://en.wikipedia.org/wiki/Platform_as_a_service) (platform as a service) и [IaaS](https://en.wikipedia.org/wiki/Infrastructure_as_a_service) (infrastructure as a service) сервизи.

## Azure blob storage

### Storage Services

Storage Services осигуряват [REST](https://en.wikipedia.org/wiki/REST) и [SDK](https://en.wikipedia.org/wiki/Software_development_kit) [APIs](https://en.wikipedia.org/wiki/API) за запазване и достъпване на информация в облака.

public AzureStorageProvider()

       {

           var connectionString = CloudConfigurationManager.GetSetting(ConnectionString);

           var storageAccount = CloudStorageAccount.Parse(connectionString);

           \_blobClient = storageAccount.CreateCloudBlobClient();

       }

#### Blob Service

Този сервиз позволя запазването на неструктуриран текст или бинарна информация като блоб, който може да бъде достъпен като url

### Сваляне на файл

public Stream StreamFile(string userId, string filename)

       {

           var container = \_blobClient.GetContainerReference(userId);

           return container.GetBlockBlobReference(filename).OpenRead();

       }

### Качване на файл,

public string UploadStream(string userId, string filename, Stream stream)

       {

           var container = this.blobClient.GetContainerReference(userId);

           container.CreateIfNotExistsAsync();

           var blockBlob = container.GetBlockBlobReference(filename);

           blockBlob.UploadFromStream(stream);

           return blockBlob.StorageUri.PrimaryUri.ToString();

       }

# Entity Framework

Entity Framework (EF) е обектно-релационна свързващ фрейморк с отворен код, част от ADO.NET

Използва се за генериране на базата данни чрез Code-First, което създава релационна база данни от вече съществуващи модели

## Моделите използване за създаването на базаста данни:

### Абстрактен клас за „изтриващ се“ модел

Чрез наследяване на този клас, моделите получават тези полета.

public abstract class DeletableEntity : AuditInfo, IDeletableEntity

  {

      [Display(Name = "Deleted?")]

      [Editable(false)]

      public bool IsDeleted { get; set; }

      [Display(Name = "Deletion date")]

      [Editable(false)]

      [DataType(DataType.DateTime)]

      public DateTime? DeletedOn { get; set; }

  }

### Потребител

public class AppUser : IdentityUser

    {

        private ICollection<File> files;

        private ICollection<Message> messages;

        public AppUser()

        {

            this.messages = new HashSet<Message>();

        }

        public virtual ICollection<Message> Messages

        {

            get { return this.messages; }

            set { this.messages = value; }

        }

        public virtual ICollection<File> Files

        {

            get { return this.files; }

            set { this.files = value; }

        }

        public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<AppUser> 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;

        }

### Файл

public class File : DeletableEntity

{

    [Key]

    public int Id { get; set; }

    [Required]

    public string FileName { get; set; }

    public string PathToFile { get; set; }

    public int Size { get; set; }

    public bool IsDirectory { get; set; }

    public int? ParentId { get; set; }

    [ForeignKey("ParentId")]

    public virtual File Parent { get; set; }

    public string UserId { get; set; }

    public virtual AppUser User { get; set; }

    public int? PermissionId { get; set; }

    [ForeignKey("PermissionId")]

    public virtual Permission Permission { get; set; }

}

### Лог

public class Log

   {

       [Key]

       public int Id { get; set; }

       public string Username { get; set; }

       public string Ip { get; set; }

       public string Action { get; set; }

       public DateTime DateTime { get; set; }

       public virtual AppUser User { get; set; }

   }

### Съобщение

public class Message : DeletableEntity

   {

       public int Id { get; set; }

       [Required]

       [MinLength(1)]

       public string Content { get; set; }

       public bool IsSeen { get; set; }

       public string ReceiverId { get; set; }

       public virtual AppUser Receiver { get; set; }

   }

### Достъпност

public class Permission

    {

        public int Id { get; set; }

        public AccessType AccessType { get; set; }

        public virtual ICollection<ShareName> AuthorizedUsers { get; set; }

    }

## Миграции

При промяна на моделите EF прави миграции, за да отговаря базата на последните корекции по моделите и генериране на ролите за таблицата на потребителите идваща от Identity.

internal sealed class Configuration : DbMigrationsConfiguration<YashmakDbContext>

   {

       private UserManager<AppUser> userManager;

       public Configuration()

       {

           this.AutomaticMigrationsEnabled = true;

           this.AutomaticMigrationDataLossAllowed = true;

       }

       protected override void Seed(YashmakDbContext context)

       {

           this.userManager = new UserManager<AppUser>(new UserStore<AppUser>(context));

           this.SeedRoles(context);

           this.SeedUsers(context);

       }

       private void SeedRoles(

           IdentityDbContext<AppUser, IdentityRole, string, IdentityUserLogin, IdentityUserRole, IdentityUserClaim> context)

       {

           context.Roles.AddOrUpdate(x => x.Name, new IdentityRole(Constants.AdminRole));

           context.Roles.AddOrUpdate(x => x.Name, new IdentityRole(Constants.NonPaidUser));

           context.SaveChanges();

       }

       private void SeedUsers(

           IdentityDbContext<AppUser, IdentityRole, string, IdentityUserLogin, IdentityUserRole, IdentityUserClaim> context)

       {

           if (context.Users.Any())

           {

               return;

           }

           var adminUser = new AppUser

               {

                   Email = "administrator ",

                   UserName = "admin"

               };

           this.userManager.Create(adminUser, "password");

           context.SaveChanges();

           this.userManager.AddToRole(adminUser.Id, Constants.AdminRole);

       }

   }

## Хранилище за изтрити модели

Чрез използването на този клас, извикването на заявки няма да се връщат модели, маркирани като изтрити, освен ако изрично не извикаме метода за изтрити.

public class DeletableEntityRepository<T> :

     GenericRepository<T>,

     IDeletableEntityRepository<T> where T : class, IDeletableEntity

 {

     public DeletableEntityRepository(IYashmakDbContext context)

         : base(context)

     {

     }

     public override IQueryable<T> All()

     {

         return base.All().Where(x => !x.IsDeleted);

     }

     public IQueryable<T> AllWithDeleted()

     {

         return base.All();

     }

     public override void Delete(T entity)

     {

         entity.DeletedOn = DateTime.Now;

         entity.IsDeleted = true;

         var entry = this.Context.Entry(entity);

         entry.State = EntityState.Modified;

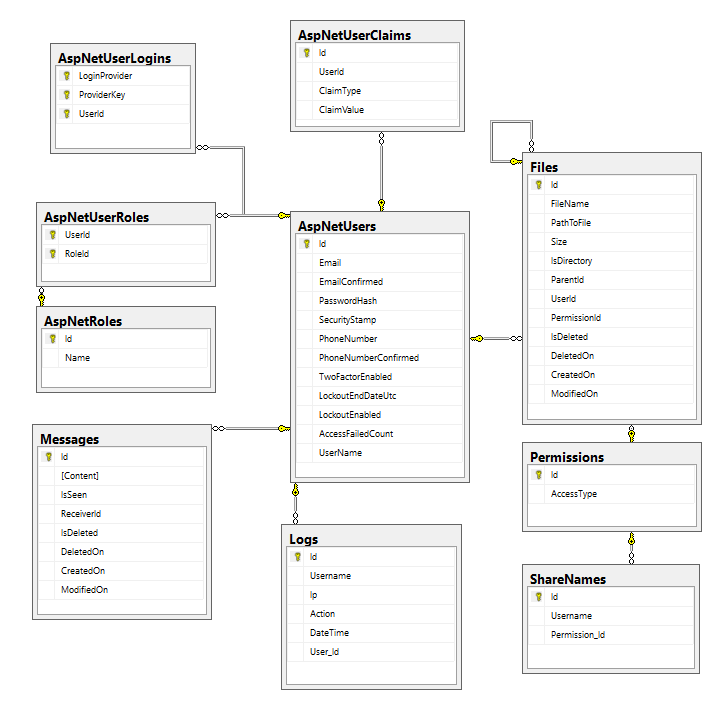
     }

 }

# Microsoft SQL Server 2014

е сървърна [система за управление на бази от данни](https://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%BE%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8) (и по-точно, на релационни бази от данни). Microsoft SQL Server е предназначена за управление на големи сървърно базирани БД.

## Схема на таблиците и релациите им в Базата данни ( генерирана чрез Code-First).



# Asp.net MVC

ASP.NET MVC е платформа, създадена от [Microsoft](https://bg.wikipedia.org/wiki/Microsoft), която служи за изработване на уеб приложения, използвайки модела [Model-View-Controller](https://bg.wikipedia.org/wiki/Model-View-Controller) (MVC). Платформата използва [C#](https://bg.wikipedia.org/wiki/C_Sharp), [HTML](https://bg.wikipedia.org/wiki/HTML), [CSS](https://bg.wikipedia.org/wiki/CSS), [JavaScript](https://bg.wikipedia.org/wiki/JavaScript) и [бази данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8). ASP.NET MVC е съвременно средство за изграждане на [уеб приложения](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), което не замества изцяло уеб формите. Платформата включва нови тенденции в разработката на уеб приложения, притежава много добър контрол върху [HTML](https://bg.wikipedia.org/wiki/HTML) и дава възможност за създаване на всякакви приложения. ASP.NET MVC може да бъде много лесно тествана и допълвана, защото е изградена от отделни модули, които са изцяло независими едни от други. Чрез платформата се създават цялостни приложения, които се стартират.

## Модел

Моделът представлява част от приложението, което реализира домейн логиката, също известна като бизнес логика. Домейн логиката обработва данните, които се предават между базата данни и потребителския интерфейс. Например, в една система за инвентаризация, моделът отговаря за това дали елемент от склада е наличен. Моделът може да бъде част от заявлението, което актуализира базата данни когато даден елемент е продаден или доставен в склада. Често моделът съхранява и извлича официална информация в базата данни.

## Изглед модел (view model)

Изглед модела позволява да се оформят няколко изгледа от един или повече модела от данни или източници в един обект. Този модел е оптимизиран за потребление и изпълнение.

## MVC модели

MVC (model — view — control) предоставя различни видове и начини за това как да бъде изграден даден модел. Практиката е доказала, че поради липсата на много информация, практики и препоръки, е възможно да се получат разминавания за това как да бъде направен модел. Всъщност, истината е, че няма еднотипно решение, което да задоволява всички възникнали проблеми. Тук ще разгледаме някой от основните модели, който се използват, с описание на конкретен пример. Трябва да се отбележи, че това са модели, използвани при решаването на реални практически проблеми.

Единствено интересен view-model е за директорията, тъй като реално нямаме реализирани директории и всеки един файл може да представлява директория, затова го представяме по следния начин:

### Директория (във файловия мениджър)

public class DirectoryViewModel

    {

        public int? FileNodeId { get; set; }

        public IQueryable<FileViewModel> Files { get; set; }

        public IEnumerable<NavigationDirectoryViewModel> NavigationModels { get; set; }

    }

### Навигационна директория – за breadcum

public class NavigationDirectoryViewModel : IMapFrom<File>

   {

       public string FileName { get; set; }

       public int? Id { get; set; }

   }

## Контролери и действия

### Какво са контролери

Контролерите са класове, които се създават в MVC приложението. Намират се в папка Controllers. Всеки един клас ,който е от този тип, трябва да има име завършващо с наставка "Controller". Контролерите обработват постъпващите заявки, въведени от потребителя и изпълняват подходящата логика за изпълнение на приложението. Класът контролер е отговорен за следните етапи на обработка:

#### Намиране и извикване на най-подходящия метод за действие (action method) и валидиране, че може да бъде извикан.

#### Взимането на стойности, които да се използват като аргументи в метода за действие.

#### Отстраняване на всички грешки, които могат да възникнат по време на изпълнението метода за действие.

#### Осигуряване на клас WebFormViewEngine по подразбиране за отваряне на страници с изглет от тип ASP.NET.

### Основни контролери в приложението:

#### Основния контролер, от който всички други се наследяват

public abstract class BaseController : Controller

   {

       protected BaseController(IYashmakData data)

       {

           this.Data = data;

       }

       public string UserId => this.HttpContext.User.Identity.GetUserId();

       protected IYashmakData Data { get; set; }

       protected AppUser CurrentUser => this.Data.Users.GetById(this.UserId);

       protected List<Message> GetUserMessages()

       {

           var messages = new List<Message>();

           if (this.UserId == null)

               return messages;

           messages = this.Data.Users

               .GetById(this.UserId)

               .Messages

               .Where(m => !m.IsSeen)

               .ToList();

           foreach (var message in messages)

           {

               message.IsSeen = true;

           }

           this.Data.SaveChanges();

           return messages;

       }

   }

## Валидация

### Какво е валидиране

**Валидирането** на входните данни, въведени от потребителя (user input), позволява да се уверим, че те съответстват на модела на данните в ASP.NET MVC. По този начин се защитава приложението от потребителски грешки или злонамерени потребители. Има много начини да се включи валидиране в едно MVC приложение. Например, може да бъде използвана публично достъпна библиотека за **валидиране**, като Microsoft Enterprise Library Validation Block.

### Как работи валидирането

Когато потребителят потвърди (submit) формуляр (form), данните от формуляра се предават на метод за действие (action method) с помощта на ViewDataDictionary колекция. ViewDataDictionary има ModelState свойство (property), което съдържа колекция от ModelState обекти. За всеки модел, който е дефиниран в MVC приложението, се създава съответния ModelState обект и добавя в колекцията. Действието, което получава данните от формуляра определя правилата за проверка, прилагани към формуляра с данни. Ако правилото е нарушено, действието използва ModelState свойство за да предаде информация за грешка обратно към изгледа.

### Пример за валидация използвано чрез view model-а

public class RegisterViewModel

   {

       [DataType(DataType.Password)]

       [Display(Name = "Confirm password")]

       [System.ComponentModel.DataAnnotations.Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]

       public string ConfirmPassword { get; set; }

       [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; }

   }

[[8]](https://bg.wikipedia.org/wiki/ASP.NET_MVC#cite_note-8)

## Изгледи

### Какво е изглед

Изгледите са тези, които определят как ще бъде визуализиран потребителският интерфейс (UI) на приложението. В ASP.NET MVC се поддържат средства (engines) за генериране на изгледи.

### Начин на работа на изгледите

Когато потребителт взаимодейства с изглед, данните се **рутират** от изгледа до метод за действие, който от своя страна може да създаде друг изглед. Едно MVC приложение може да има няколко контролери, всеки от които може да съдържа множество методи за действие, а всяко действие може да създаде различен изглед. Изгледите са организирани в папки, като името им се определя от това на свързания контролер (например изгледите от HomeController са в папката \Views\Home).

### Формата за качване на файлове (изглед)

@using MvcFileUploader.HtmlHelper

@using MvcFileUploader.Models

@using Yashmak.Common

@model int?

@{

    ViewBag.Title = "File Drag & Drop Upload";

}

@(Html.MvcFileUpload()

      .UploadAt(Url.Action("FileUpload"))

      .WithMaxFileSize(Constants.UploadFileMaxSize)

      .UIStyle(UploadUI.Bootstrap)

      .MaxNumberOfFiles(Constants.MaxСimultaneouslyUploads)

      .AddFormField("filenodeid", @Model.ToString())

      .RenderInline())

@Html.ActionLink("Go Back", "Index", "Files", new { filenodeid = Model }, new { @class = "btn btn-inverse pull-left" })

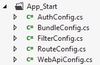
## Рутиране

### Какво е рутиране

**Рутирането** е метод за валидация и обработка на заявката от потребителя. След като рутирането получи заявката, то проверява дали тази заявка отговаря на рутиращите шаблони и ако бъдат намерени съответствия, първият физически поставен в кода шаблон, отговарящ на дадената заявка, изпълнява функции за подаване на данните от заявката към съответните **контролери**, **действия** и **параметри**.

### Как се регистрира и дефинира рутирането

При стартиране на **ASP.NET MVC** приложението първият файл, който се стартира, се нарича "**Global.asax**" и съответно в него първият метод, който се извиква, е с име "Application\_start()".Във този метод има фунции за регистрация на **Area**, **WebApi**,**Filters**, **Routing** и **Bundle**. Именно тук се регистрира и нашето рутиране (Routing) чрез извикване на класа **RoutingConfig**, през метода **RegisterRoutes** и през него се регистрира рутирането в глобалната таблица **RouteTable**.

[](https://bg.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:App_Start-RouteConfig.png)

Конфиг файловете в папка Апп Старт

За да дефинирате рутиранете е нужно да конфигурирате файла **RouteConfig.cs**, който се намира в папката **App\_Start** заедно с останалите Config файлове.

Тук вече се конфигурира метода **RegisterRoutes**, който се извиква от "**Global.asax**" файла. Този метод използва глобална таблица, която може да бъде извикана отвсякъде в нашето приложение и в нея има методи **IgnoreRoute** и **MapRoute**. **IgnoreRoute** методът се използва за да се пренебрегнат заявки, които не желаем да обработим. Чрез метода **MapRoute** ние регистрираме нашето рутиране като то си има **име**, **url-шаблон** и **стойности по подразбиране**.

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 },

        namespaces: new[] { "Yashmak.Web.Controllers" });

}

Шаблоните в рутирането са **case-insensitive**, което означава, че шаблонът ще разпознае данните без значение дали са с малки или главни букви. Ако при заявката никой от шаблоните не бъде обхванат, то тогава сървърът връща **error-404**.

## Филтри на дейсвията

### Какво са филтрите

// TODO: Translate

The goal of this tutorial is to explain action filters. An action filter is an attribute that you can apply to a controller action -- or an entire controller -- that modifies the way in which the action is executed.

### Филтър за следене на потребителската дейсност

Всяка стъпка на потребителите е следена и записвана в базата данни.

Записва се време, адрес, потребител и действието му. Може да послужи за проследяване на нередности.

public class LogAttribute : ActionFilterAttribute

   {

       [Inject]

       public IYashmakData Contex { get; set; }

       public override void OnActionExecuted(ActionExecutedContext filterContext)

       {

           if (HttpContext.Current.User != null)

           {

               var dateTime = DateTime.Now;

               var ip = HttpContext.Current.Request.UserHostAddress;

               var userId = HttpContext.Current.User.Identity.GetUserName();

               var action = HttpContext.Current.Request.RawUrl;

               var log = new Log { Action = action, DateTime = dateTime, Username = userId, Ip = ip };

               this.Contex.Logs.Add(log);

               this.Contex.SaveChanges();

           }

           base.OnActionExecuted(filterContext);

       }

   }

### Пример:

Използва се чрез анотация на клас или метод.

Контролерът, отговарящ за действията с файлове в приложението има анотация [Log].

[Log]

   [Authorize]

   public abstract class FileBaseController : BaseController

# Automapper

Е nugget пакет, който улеснява създаването на ViewModel от Model. За да се използва, първо трябва да се укаже как се мапва един обект към друг.

## Създаване на mapping

Описват се всички полета, отговарящи на новия обект ( поддържат се и трансформации).

Полетата с еднакви имена не е нужно да се оказват експлицитно.

public void CreateMappings(IConfiguration configuration)

       {

           configuration.CreateMap<AppUser, UserViewModel>()

               .ForMember(m => m.Id, opt => opt.MapFrom(t => t.Id))

               .ForMember(m => m.UserName, opt => opt.MapFrom(t => t.UserName))

               .ForMember(m => m.FilesCount, opt => opt.MapFrom(t => t.Files.Count))

               .ForMember(

                   m => m.Role,

                   opt =>

                   opt.MapFrom(

                       t =>

                       t.Roles.FirstOrDefault().UserId != null

                           ? new RoleViewModel { Name = Constants.AdminRole }

                           : new RoleViewModel { Name = Constants.NonPaidUser }))

               .ForMember(m => m.UsedSpace, opt => opt.MapFrom(t => t.Files.Sum(f => f.Size)))

               .ReverseMap();

       }

## Как се използва

protected override IEnumerable GetData()

        {

            var data = this.Data.Users.All().ProjectTo<UserViewModel>();

            return data;

        }

# Ninject

SOLID принципите са едни от основополагащите при разработката на едно приложение. Макар и на последно място “D” седи за Dependency Inversion. Именно тук се появява open-source библиотеката Ninject.

## Регистриране на базата данни за автоматично инжектиране при използване:

private static void RegisterServices(IBindingRoot kernel)

    {

        kernel.Bind<IYashmakDbContext>().To<YashmakDbContext>();

        kernel.Bind<IYashmakData>().To<YashmakData>();

    }

Може дори да байнднем към интерфейс ( “L” – Субституцията на Лисков)

## Използване от контролер:

Създаваме контролер с конструктор приемащ същия интерфейс, който регистрираме по-горе с Ninject. Това автоматично ще създаде нова инстанция от байндматия клас

public abstract class BaseController : Controller

   {

       protected BaseController(IYashmakData data)

       {

           this.Data = data;

       }

}

# JavaScript

JavaScript (чете се джаваскрипт) е интерпретируем [език за програмиране](https://bg.wikipedia.org/wiki/%D0%95%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), разпространяван с повечето [Уеб браузъри](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80). Поддържа [обектно-ориентиран](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD) и [функционален](https://bg.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) стил на програмиране.

 Javascript е стандартизиран под името [EcmaScript](https://bg.wikipedia.org/w/index.php?title=EcmaScript&action=edit&redlink=1).

JavaScript е програмен език, който позволява динамична промяна на поведението на [браузъра](https://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80) в рамките на дадена [HTML](https://bg.wikipedia.org/wiki/HTML) страницата. JavaScript се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до [Обектния модел на браузъра](https://bg.wikipedia.org/w/index.php?title=DOM&action=edit&redlink=1). JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). Javascript е най-широко разпространеният език за програмиране в Интернет. Прието е JavaScript програмите да се наричат [скриптове](https://bg.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82).

## Възможности

JavaScript може да влияе на почти всяка част от браузъра. Браузъра изпълнява JavaScript кода в [цикъла на събития](https://bg.wikipedia.org/w/index.php?title=Event_loop&action=edit&redlink=1) т.е. като резултат от действия на потребителя или събития в браузъра (например document.onLoad, виж [DOM](https://bg.wikipedia.org/w/index.php?title=DOM&action=edit&redlink=1)). Основни задачи в повечето JavaScript приложения са:

### Зареждане на данни чрез [AJAX](https://bg.wikipedia.org/wiki/AJAX)

### Ефекти с изображения и [HTML](https://bg.wikipedia.org/wiki/HTML) елементи: скриване/показване, пренареждане, влачене, слайд шоу, анимация и много други.

### Управление на прозорци и рамки.

### Разпознаване на възможностите на браузъра.

### Използване на камерата и микрофона.

### Създаване на 3D графики [WebGL](https://bg.wikipedia.org/w/index.php?title=WebGL&action=edit&redlink=1).

### По-добър и гъвкав [потребителски интерфейс](https://bg.wikipedia.org/w/index.php?title=UI&action=edit&redlink=1)

## Ограничения:

### Не може да се записва информация на потребителския компютър или отдалечения сървър.

### Не може да се запазва информация директно в отдалечена база данни.

### Не може да се стартират локални приложения.

## JavaScript библиотека

представлява предварително написан JavaScript код. Библиотеката може да се ползва, когато е необходимо да се включи някаква функционалност в уебсайта. Вместо да се изгражда от нулата, се ползва вече създадена библиотека, която предлага точно тази функционалност.

Библиотеките подобряват и намаляват времето за разработка на уеб проекта.

### Jquery - функционалност

Библиотеката е проектирана да улесни и опрости писането на JavaScript код, като едновременно с това е гъвкава и може да се надгражда с допълнителни плъгини.

Библиотеката се [поддържа от всички съвременни версии на уеб браузърите](http://jquery.com/browser-support).

 лесна за използване, постигане на повече функционалност с писане на по-малко код, голяма библиотека и изпълняване на голям брой функции, хиляди готови плъгини, богата помощна информация, поддръжка на Ajax и други.

#### [DOM](https://bg.wikipedia.org/wiki/Document_Object_Model) селекция, базирана на синтаксиса на [CSS](https://bg.wikipedia.org/wiki/Cascading_Style_Sheets) селектори + разширена функционалност.

#### DOM манипулация (с поддръжка за CSS 1-3), позволяваща създаване, манипулиране и премахване на елементи от уеб страницата.

#### Събития (events)

#### Ефекти и анимация

#### [Ajax](https://bg.wikipedia.org/wiki/AJAX)

#### парсване на JSON

#### Съвместимост с широк набор браузъри, в т.ч. стари версии

### Селектиране на файл или директория

// select file row in virtual disk

// and show menu for each row

$("body").on("click", ".file-row", function (ev) {

    var self = ($(this));

    if (self.hasClass("success")) {

        self.removeClass("success");

        self.prev().addClass("hidden");

    } else {

        self.addClass("success");

        self.prev().removeClass("hidden");

    }

});

// this script change button attribute fileNodeId

// to know which is the current folder

// to add files or create folder

function OnFolderChange(fileNodeId) {

    $(".dynamic-change-id").attr("href", function (i, a) {

        return a.replace(/(filenodeid=)[0-9]+/ig, '$1' + fileNodeId);

    });

}

//Modal dialog for delete in browse mode

$(function () {

    // Initialize numeric spinner input boxes

    //$(".numeric-spinner").spinedit();

    // Initialize modal dialog

    // attach modal-container bootstrap attributes to links with .modal-link class.

    // when a link is clicked with these attributes, bootstrap will display the href content in a modal dialog.

    $('body').on('click', '.modal-link', function (e) {

        e.preventDefault();

        $(this).attr('data-target', '#modal-container');

        $(this).attr('data-toggle', 'modal');

    });

    // Attach listener to .modal-close-btn's so that when the button is pressed the modal dialog disappears

    $('body').on('click', '.modal-close-btn', function () {

        $('#modal-container').modal('hide');

    });

    //clear modal cache, so that new content can be loaded

    $('#modal-container').on('hidden.bs.modal', function () {

        $(this).removeData('bs.modal');

    });

    $('#CancelModal').on('click', function () {

        return false;

    });

});

if ($("#access-id").val() != 1) {

    $("#custom-permission").hide();

}

$("#access-id").change(function(e) {

    if ($("#access-id").val() == 1) {

        $("#custom-permission").show();

    } else {

        $("#custom-permission").hide();

    }

});

### KendoUI – grid

Използвано в администрацията на потребителите

@using System.Collections

@using Kendo.Mvc.UI

@using Yashmak.Web.Areas.Admin.ViewModels.Users

@model UserViewModel

@{

    ViewBag.Title = "Users";

    const string ControllerName = "Users";

}

@{

    @Scripts.Render("~/bundles/kendo")

}

<h2>@ViewBag.Title</h2>

@(Html.Kendo()

      .Grid<UserViewModel>()

      .Name("grid")

      .Columns(cols =>

          {

              cols.Bound(pr => pr.Id).Hidden();

              cols.Bound(pr => pr.UserName);

              cols.ForeignKey(p => p.Role.Name, ViewData["Roles"] as IEnumerable, "Name", "Name")

                  .Title("Role").ClientTemplate("#=Role.Name#").Width(160);

              cols.Bound(pr => pr.FilesCount).Width(150);

              cols.Bound(pr => pr.UsedSpace);

              cols.Command(command => command.Custom("Send").Click("sendMessage")).Title("Message").Width(100);

              cols.Command(com => com.Destroy()).Title("Delete").Width(130);

          })

      .ToolBar(toolBar => toolBar.Save())

      .ColumnMenu()

      .Pageable(page => page.Refresh(true))

      .Sortable()

      .Groupable()

      .Filterable()

      .Editable(editable => editable.Mode(GridEditMode.InCell))

      .DataSource(data =>

                  data.Ajax()

                      .Batch(true)

                      .Model(m =>

                          {

                              m.Id(adv => adv.Id);

                              m.Field(u => u.Id).Editable(false);

                              m.Field(u => u.UserName).Editable(false);

                              m.Field(u => u.FilesCount).Editable(false);

                              m.Field(u => u.UsedSpace).Editable(false);

                              m.Field(p => p.Role.Name);

                          })

                      .Read(read => read.Action("Read", ControllerName))

                      .Update(update => update.Action("Update", ControllerName))

                      .Destroy(destroy => destroy.Action("Destroy", ControllerName))

      ))

@(Html.Kendo().Window().Name("Message")

      .Title("Send Message")

      .Visible(false)

      .Modal(true)

      .Draggable(true)

      .Width(400)

      )

<script type="text/x-kendo-template" id="template">

    <div id="details-container">

        @using (Ajax.BeginForm("SendMessage", "Users", new AjaxOptions { }, new { @class = "form-horizontal" }))

        {

            @Html.AntiForgeryToken()

            <input type="hidden" name="ReceiverId" value="#=Id#" />

            <h5>To: #= UserName #</h5>

            <textarea name="Content" placeholder="Enter your message" class="form-control" required="required"></textarea>

            <input type="submit" class="btn btn-primary" value="Send" />

        }

    </div>

</script>

<script type="text/javascript">

    var detailsTemplate = kendo.template($("#template").html());

    function sendMessage(e) {

        e.preventDefault();

        var dataItem = this.dataItem($(e.currentTarget).closest("tr"));

        var wnd = $("#Message").data("kendoWindow");

        wnd.content(detailsTemplate(dataItem));

        wnd.center().open();

    }

</script>

# Bootstrap

Bootstrap прави създването на front-end web разработката бърза и лесна. Не се изисква дългострочно отдаване от страна на разработчика за разучаване на неговите особености, защото такива липсват. Кривата на обучение е много ниска и е идеален за големи приложения – поддържа CSS стиловете в не много комплексен вид.

Bootstrap е с отворен код. Хоствано, разработвано и поддържано в GitHub.

## Поддръжка на препроцесори

Поддържа Sass и Less

Bootstrap идва с чист CSS, но неговия сорс код използва два много популярни CSS препроцесори.

## Responsive сред всякакви устроиства и резолюции.

Bootstrap лесно и ефективно скалира приложението измежду много устройства и резолюции с един единствен базисен код, от телефони до таблети и десктоп чрез CSS media queries.

## Пример за модален диалог

div class="modal-body">

    <div class="alert alert-success text-center">

        <span class="glyphicon glyphicon-file"></span>

        @Model.FileName

    </div>

    @using (Html.BeginForm("Download", "Files", FormMethod.Post))

    {

        @Html.AntiForgeryToken()

        <div class="row text-center">

            Share link:

        </div>

        <div class="row text-center">

            @Url.Action("Download", "Files", new { filenodeid = @Model.Id }, "http")

            &nbsp;

        </div>

        <input type="hidden" name="filenodeid" value="@Model.Id"/>

        <div class="row">

            <div class="row col-md-8 col-md-offset-4">

                <button type="submit" id="approve-btn"

                        class="btn btn-inverse">

                    Download

                </button>

                <button type="button" class="btn btn-default"

                        data-dismiss="modal">

                    Cancel

                </button>

            </div>

        </div>

    }

</div>

# Font-awesome

Font Awesome дава скалируема векторна графика с икони, които могат да бъдат направени според според предпочитанията — големина, цвят, сянка, всичко може да се направи с помощта на CSS.

## 605 Икони

В една единствена колекция

## Не изисква JavaScript

По-малко усложнения, което води по лесно възприемане и поддръжка.

## Безкрайно скалиране

Предимствата на векторната графика – скалируемост, означаващо, че всяка икона може да бъде разтеглена и смалена до всякакъв размер без загуба на кечество.

## Контрол чрез CSS

Цвят, размер, сянка, всичко, което може да се направи със CSS се поддържа.

## Поддръжка от много фреймурки

Оригинално създадено за Bootstrap.

## Пример

<footer class="panel-footer panel text-center">

   <p><a href="https://github.com/zmitov/" class="fa fa-github fa-3x"></a></p>

</footer>

Shared/\_Layout.cshtml, страницата се използва във footer-a, class="fa fa-github fa-3x”, създвайки векторна графика с 3-пъти размера на оригинала на gitgub.

<span class="fa fa-search">

Икона за търсене

# Microsoft .NET Framework

**Microsoft .NET Framework** е [платформа](https://bg.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%80%D0%B8)), създадена от [Microsoft](https://bg.wikipedia.org/wiki/Microsoft), която предоставя [програмен модел](https://bg.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B5%D0%BD_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB&action=edit&redlink=1), библиотека от класове ([FCL](https://bg.wikipedia.org/w/index.php?title=FCL&action=edit&redlink=1), *Framework Class Library*) и среда за изпълнение на написан специално за нея програмен код ([CLR](https://bg.wikipedia.org/w/index.php?title=CLR&action=edit&redlink=1), *Common Language Runtime*). Тя е ключов елемент от стратегията за развитие на [Microsoft](https://bg.wikipedia.org/wiki/Microsoft), чиято цел е повечето нови приложения за [Windows](https://bg.wikipedia.org/wiki/Windows) да бъдат базирани на .NET Framework.

.NET приложенията се пишат на езици от високо ниво ([C#](https://bg.wikipedia.org/wiki/C_Sharp) и други) и се компилират до платформено-независим междинен език, наречен [CIL](https://bg.wikipedia.org/w/index.php?title=CIL&action=edit&redlink=1) (*Common Intermediate Language*). По време на изпълнение CIL кодът (т. нар. „управляван код“) бива автоматично компилиран от CLR за конкретната хардуерна платформа и операционна система, с която работи потребителят.

## Компоненти на .NET Framework

### Common Language Runtime (CLR)

програмната среда, в която се изпълнява кодът на .NET приложенията. Представлява [виртуална машина](https://bg.wikipedia.org/wiki/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0), която компилира междинния код [CIL](https://bg.wikipedia.org/w/index.php?title=CIL&action=edit&redlink=1) (Common Intermediate Language) за конкретната хардуерна платформа и операционна система, с която работи потребителят. В допълнение средата осигурява редица услуги, като управление на паметта, вкл. автоматично "събиране на боклуци" ('garbage collection'), управление на прекъсванията, многонишковост ('multithreading') и други.

### Framework Class Library (FCL)

това е пълната колекция от класове, които .NET Framework предлага. Частта от FCL, известна като [BCL](https://bg.wikipedia.org/w/index.php?title=BCL&action=edit&redlink=1) (Base Class Library, 'стандартна библиотека от класове'), е разработената от Microsoft имплементация на стандарта [ECMA-335, Common Language Infrastructure](http://www.ecma-international.org/publications/standards/Ecma-335.htm). BCL осигурява голям набор базови функционалности, които програмистите могат да използват директно, например за [мрежова комуникация](https://bg.wikipedia.org/w/index.php?title=%D0%9C%D1%80%D0%B5%D0%B6%D0%BE%D0%B2%D0%B0_%D0%BA%D0%BE%D0%BC%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F&action=edit&redlink=1), [криптографски услуги](https://bg.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и други. В допълнение FCL предлага компоненти за изграждане на [потребителския интерфейс](https://bg.wikipedia.org/wiki/GUI) ([Windows.Forms](https://bg.wikipedia.org/w/index.php?title=Windows.Forms&action=edit&redlink=1) и [ASP.NET](https://bg.wikipedia.org/wiki/ASP.NET)), [уеб приложения](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [уеб услуги](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B0), достъп до [бази данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8) и др.

# C#

C# е [обектно-ориентиран](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) [език за програмиране](https://bg.wikipedia.org/wiki/%D0%95%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), разработен от [Microsoft](https://bg.wikipedia.org/wiki/Microsoft), като част от софтуерната платформа [.NET](https://bg.wikipedia.org/wiki/.NET). Стремежът още при създаването на C# езика е бил да се създаде един прост, модерен, обектно-ориентиран език с общо предназначение. Основа за C# са [C++](https://bg.wikipedia.org/wiki/C_Plus_Plus), [Java](https://bg.wikipedia.org/wiki/Java) и донякъде езици като [Delphi](https://bg.wikipedia.org/wiki/Delphi),[VB.NET](https://bg.wikipedia.org/w/index.php?title=VB.NET&action=edit&redlink=1) и [C](https://bg.wikipedia.org/wiki/C_(%D0%B5%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5)). Той е проектиран да балансира мощност ([C++](https://bg.wikipedia.org/wiki/C_Plus_Plus)) с възможност за бързо разработване ([Visual Basic](https://bg.wikipedia.org/wiki/Visual_Basic) и Java). Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика – инструкциите, които компютърът изпълнява. Програмите на C# представляват един или няколко файла с разширение .cs., в които се съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# (csc) до изпълним код и в резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll).

##### Литература

1. MSDN, <https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>
2. Microsft Azure, <https://msdn.microsoft.com/en-us/library/azure/dd163896.aspx>
3. MSDN Entity Framework Overview, <https://msdn.microsoft.com/en-us/library/vstudio/bb399567(v=vs.100).aspx>
4. Microsoft SQL Server Overview, <https://technet.microsoft.com/en-us/library/ms166352(v=sql.90).aspx>
5. ASP.NET MVC Overview, <https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx>
6. Automapper, <http://automapper.org/>
7. Ninject, <http://www.ninject.org/>
8. Javascript wikipedia (пълнеж), <https://bg.wikipedia.org/wiki/JavaScript>
9. jQuery, <https://jquery.com/>
10. Bootstrap, <http://getbootstrap.com/getting-started/>
11. Font Awesome, <https://fortawesome.github.io/Font-Awesome/>