Приложение

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Разработка программы планирования бюджета проекта

Код программы
КП.ПО-11. 1-40 01 01

Листов 34

Руководитель Якимук А.В.

Выполнил Юрашевич В.С.

Консультант по ЕСПД Якимук А.В.

2024

Program.cs

```
using SiteOfCompany.DAL;
using Microsoft.EntityFrameworkCore;
using SiteOfCompany.DAL.Repositories;
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Service.Interfaces;
using SiteOfCompany.Service.Implementations;
using Microsoft.AspNetCore.Authentication.Cookies;

var builder = WebApplication.CreateBuilder(args); /*С помощью класса WebApplication
создаём объект builder, благодаря которому можем всячески настраивать приложение
(установка конфигурации приложения, добавление сервисов, настройка логгирования в
приложении и т.д.), объявлять переменную app, которая представляет объект типа
WebApplication*/

builder.Services.AddControllersWithViews(); //Добавляет в коллекцию сервисов
сервисы, которые необходимы для работы контроллеров

var connection = builder.Configuration.GetConnectionString("DefaultConnection"); //В
переменную connection передаётся название сервера и как он должен подключаться
builder.Services.AddDbContext<ApplicationDbContext>(options =>
options.UseSqlServer(connection)); //Добавляем контекст данных ApplicationDbContext
в качестве сервиса в приложении для дальнейшего взаимодествия с базой данных

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme
)
    .AddCookie(options =>
    {
        options.LoginPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
        options.AccessDeniedPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
    });


builder.Services.AddScoped<IProjectRepository, ProjectRepository>();
builder.Services.AddScoped<IProjectService, ProjectService>();
builder.Services.AddScoped<IUserRepository, UserRepository>();
builder.Services.AddScoped<IUserService, UserService>();
builder.Services.AddScoped<IAccountService, AccountService>();

// Add services to the container.
var app = builder.Build();


// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

```
app.Run();
```

Domain:

Project.cs:

```
using SiteOfCompany.Domain.Enum;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace SiteOfCompany.Domain.Entity
{
    public class Project
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string ProjectName { get; set; }
        public ProjectType ProjectType { get; set; }
        public string Employee { get; set; }
        public int ProjectTime { get; set; }
        public decimal PriceForProjectTime { get; set; }
    }
}
```

User.cs:

```
using SiteOfCompany.Domain.Enum;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace SiteOfCompany.Domain.Entity
{
    public class User
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        public string Password { get; set; }
        public string Login { get; set; }
        public Role Role {  get; set; }
    }
}
```

ProjectType.cs:

```
using System.ComponentModel.DataAnnotations;

namespace SiteOfCompany.Domain.Enum
{
    //Перечисление, которое представляет вид работ, выполняемых на проекте
    public enum ProjectType
    {
        //Посмотреть, что делает Display
        [Display(Name = "Работа над требованиями")]
        Requirements = 0,
        [Display(Name = "Разработка архитектуры")]
        Architecture = 1,
        [Display(Name = "Реализация")]
        Realization = 2,
        [Display(Name = "Тестирование")]
        Testing = 3,
    }
}
```

Role.cs:

```
using System.ComponentModel.DataAnnotations;
```

```
namespace SiteOfCompany.Domain.Enum
{
    public enum Role
    {
        [Display(Name = "Пользователь")]
        User = 0,
        [Display(Name = "Админ")]
        Admin = 1,
    }
}
```

StatusCode.cs:

```
namespace SiteOfCompany.Domain.Enum
{
    /*Перечисление, которое представляет собой статус данных ("ответов"), полученных
из базы данных, например: получилось достать данные, выполнить удаление объекта,
    создать объект и засунуть в базу данных. И это перечисление служит для
определения состояния "ответов"*/
    public enum StatusCode
    {
        UserNotFound = 0,
        ProjectNotFound = 1,
        UserAlreadyExists = 10,
        OK = 200,
        InternalServerError = 500
    }
}
```

EnumExtension.cs:

```
using System.ComponentModel.DataAnnotations;
using System.Reflection;

namespace SiteOfCompany.Domain.Extensions
{
      public static class EnumExtension
      {
            public static string GetDisplayName(this System.Enum enumValue)
            {
                  return enumValue.GetType()
                        .GetMember(enumValue.ToString())
                        .First()
                        .GetCustomAttribute<DisplayAttribute>()
                        ?.GetName() ?? "Неопределённый";
            }
      }
}
```

HashPasswordHelper.cs:

```
using System.Security.Cryptography;
using System.Text;

namespace SiteOfCompany.Domain.Helpers
{
    public static class HashPasswordHelper
    {
        public static string HashPassword(string password)
        {
            using(var sha256 = SHA256.Create())
            {
                var hashedBytes =
sha256.ComputeHash(Encoding.UTF8.GetBytes(password));
```

```
                var hash = BitConverter.ToString(hashedBytes).Replace("-",
"").ToLower();

                return hash;
            }
        }
    }
}
```

BaseResponse.cs:

```
using SiteOfCompany.Domain.Enum;

namespace SiteOfCompany.Domain.Response
{
    /*Класс BaseResponse, представляющий собой "ответ", полученный из базы данных,
благодаря которому можем осуществить взаимодействие между базой данных, методами
работы
    с базой данных, полученными данными и самим приложением*/
    public class BaseResponse<T> : IBaseResponse<T>
    {
        public string Description { get; set; }
        public StatusCode StatusCode { get; set; }
        public T Data { get; set; }
    }

    public interface IBaseResponse<T>
    {
        T Data { get; set; }
        StatusCode StatusCode { get; }
        string Description { get; }
    }
}
```

LoginViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace SiteOfCompany.Domain.ViewModels.Account
{
    public class LoginViewModel
    {
        [Required(ErrorMessage = "Введите логин")]
        [MaxLength(20, ErrorMessage = "Имя должно иметь длину меньше 20 символов")]
        [EmailAddress(ErrorMessage = "Неправильно введён пароль")]
             public string Login { get; set; }

        [Required(ErrorMessage = "Введите пароль")]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }
    }
}
```

RegisterViewModel.cs:

```
using System.ComponentModel.DataAnnotations;

namespace SiteOfCompany.Domain.ViewModels.Account
{
    public class RegisterViewModel
    {
        [Required(ErrorMessage = "Укажите логин")]
             [MaxLength(20, ErrorMessage = "Имя должно иметь длину меньше 20
символов")]
             [EmailAddress(ErrorMessage = "Неправильно введён пароль")]
```

```
            public string Login { get; set; }

        [DataType(DataType.Password)]
        [Required(ErrorMessage = "Укажите пароль")]
        [MaxLength(16, ErrorMessage = "Пароль должен иметь длину меньше 16
символов")]
        [MinLength(6, ErrorMessage = "Пароль должен иметь длину больше 6 символов")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Required(ErrorMessage = "Подтвердите пароль")]
        [Compare("Password", ErrorMessage = "Пароли не совпадают")]
        public string PasswordConfirm { get; set; }
    }
}
```

ProjectViewModel.cs:

```
using SiteOfCompany.Domain.Enum;
using System.ComponentModel.DataAnnotations;

namespace SiteOfCompany.Domain.ViewModels.Project
{
    public class ProjectViewModel
    {
        public int Id { get; set; }
        [Display(Name = "Название проекта")]
        [Required(ErrorMessage = "Введите название проекта")]
        [MinLength(2, ErrorMessage = "Минимальная длина должна быть больше двух
символов")]
        public string ProjectName { get; set; }
        [Display(Name = "Вид работы")]
        [Required(ErrorMessage = "Выберите тип")]
        public ProjectType ProjectType { get; set; }
        [Display(Name = "Ф.И.О. сотрудника, отвечающего за проект")]
        [Required(ErrorMessage = "Укажите сотрудника")]
        [MinLength(6, ErrorMessage = "Минимальная длина должна быть больше шести
символов")]
        public string Employee { get; set; }
        [Display(Name = "Количество часов")]
        [Required(ErrorMessage = "Укажите количество часов")]
        public int ProjectTime { get; set; }
        [Display(Name = "Цена за час")]
        [Required(ErrorMessage = "Укажите стоимость одного часа")]
        public decimal PriceForProjectTime { get; set; }
    }
}
```

UserViewModel.cs:

```
using SiteOfCompany.Domain.Enum;
using System.ComponentModel.DataAnnotations;

namespace SiteOfCompany.Domain.ViewModels.User
{
    public class UserViewModel
    {
        public int Id { get; set; }
            [MaxLength(20, ErrorMessage = "Имя должно иметь длину меньше 20
символов")]
            [EmailAddress(ErrorMessage = "Неправильно введён пароль")]
            public string Login { get; set; }
            [MaxLength(16, ErrorMessage = "Пароль должен иметь длину меньше 16
символов")]
            [MinLength(6, ErrorMessage = "Пароль должен иметь длину больше 6
символов")]
```

```
            public string Password { get; set; }
        public Role Role { get; set; }
    }
}
```

DAL:

ApplicationDbContext.cs:

```
using Microsoft.EntityFrameworkCore;
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Helpers;

namespace SiteOfCompany.DAL
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext() { }
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
: base(options)
        {
            Database.EnsureCreated();
        }

        public DbSet<Project> Projects { get; set; }
        public DbSet<User> Users { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<User>(builder =>
            {
                builder.HasData(new User
                {
                    Id = 1,
                    Login = "yurashevich@tycomp.by",
                    Password = HashPasswordHelper.HashPassword("25080209"),
                    Role = Domain.Enum.Role.Admin
                });

                builder.ToTable("Users").HasKey(x => x.Id);

                builder.Property(x => x.Id).ValueGeneratedOnAdd();

                builder.Property(x => x.Password).IsRequired();
                builder.Property(x => x.Login).HasMaxLength(100).IsRequired();
            });
        }
    }
}
```

IBaseRepository.cs:

```
namespace SiteOfCompany.DAL.Interfaces
{
    public interface IBaseRepository<T>
    {
        Task Create(T enitity);
        Task<T> Get(int id);
        IEnumerable<T> Select();
        Task Delete(T entity);
        Task<T> Update(T entity);
    }
}
```

IProjectRepository.cs:

```
using SiteOfCompany.Domain.Entity;
namespace SiteOfCompany.DAL.Interfaces
{
    public interface IProjectRepository : IBaseRepository<Project>
    {
        Task<Project> GetByName(string name);
    }
}
```

IUserRepository.cs:

```
using SiteOfCompany.Domain.Entity;

namespace SiteOfCompany.DAL.Interfaces
{
    public interface IUserRepository : IBaseRepository<User>
    {
        Task<User> GetByLogin(string login);
    }
}
```

ProjectRepository.cs:

```
using Microsoft.EntityFrameworkCore;
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Domain.Entity;

namespace SiteOfCompany.DAL.Repositories
{
    public class ProjectRepository : IProjectRepository
    {
        private readonly ApplicationDbContext _db;
        public ProjectRepository(ApplicationDbContext db)
        {
            _db = db;
        }
        public async Task Create(Project entity)
        {
            await _db.Projects.AddAsync(entity);
            await _db.SaveChangesAsync();
        }

        public async Task Delete(Project entity)
        {
            _db.Projects.Remove(entity);
            await _db.SaveChangesAsync();
        }

        public async Task<Project> Get(int id)
        {
            return await _db.Projects.FirstOrDefaultAsync(x => x.Id == id);
        }

        public async Task<Project> GetByName(string name)
        {
                    return await _db.Projects.FirstOrDefaultAsync(x => x.ProjectName
== name);
        }

        public IEnumerable<Project> Select()
        {
            return _db.Projects.ToList();
        }
```

```
        public async Task<Project> Update(Project entity)
        {
            _db.Projects.Update(entity);
            await _db.SaveChangesAsync();

            return entity;
        }
    }
}
```

UserRepository.cs:

```
using Microsoft.EntityFrameworkCore;
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Domain.Entity;

namespace SiteOfCompany.DAL.Repositories
{
    public class UserRepository : IUserRepository
    {
        private readonly ApplicationDbContext _db;

        public UserRepository(ApplicationDbContext db)
        {
            _db = db;
        }

        public async Task Create(User enitity)
        {
            await _db.Users.AddAsync(enitity);
            await _db.SaveChangesAsync();
        }

        public async Task Delete(User entity)
        {
            _db.Users.Remove(entity);
            await _db.SaveChangesAsync();
        }

        public async Task<User> Get(int id)
        {
            return await _db.Users.FirstOrDefaultAsync(x => x.Id == id);
        }

        public async Task<User> GetByLogin(string login)
        {
            return await _db.Users.FirstOrDefaultAsync(x => x.Login == login);
        }

        public IEnumerable<User> Select()
        {
            return _db.Users.ToList();
        }

        public async Task<User> Update(User entity)
        {
            _db.Users.Update(entity);
            await _db.SaveChangesAsync();

            return entity;
        }
    }
}
```

Service:

IAccountService.cs:

```
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.Account;
using System.Security.Claims;

namespace SiteOfCompany.Service.Interfaces
{
    public interface IAccountService
    {
        Task<IBaseResponse<ClaimsIdentity>> Register(RegisterViewModel
registerViewModel);
        Task<IBaseResponse<ClaimsIdentity>> Login(LoginViewModel loginViewModel);
    }
}
```

IProjectService.cs:

```
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.Project;

namespace SiteOfCompany.Service.Interfaces
{
    public interface IProjectService
    {
        IBaseResponse<IEnumerable<Project>> GetProjects();
        Task<IBaseResponse<ProjectViewModel>> GetProject(int id);
        Task<IBaseResponse<ProjectViewModel>> GetProjectByName(string name);
        Task<IBaseResponse<Project>> CreateProject(ProjectViewModel
projectViewModel);
        Task<IBaseResponse<bool>> DeleteProject(int id);
        Task<IBaseResponse<Project>> Edit(int id, ProjectViewModel
projectViewModel);
    }
}
```

IUserService.cs:

```
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.User;

namespace SiteOfCompany.Service.Interfaces
{
    public interface IUserService
    {
        Task<IBaseResponse<User>> CreateUser(UserViewModel userViewModel);
        //Task<IBaseResponse<User>> GetUserByLogin(string login);
        IBaseResponse<IEnumerable<User>> GetUsers();
        Task<IBaseResponse<bool>> DeleteUser(int id);
        Task<IBaseResponse<User>> EditUser(int id, UserViewModel userViewModel);
    }
}
```

AccountService.cs:

```
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Enum;
using SiteOfCompany.Domain.Helpers;
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.Account;
using SiteOfCompany.Service.Interfaces;
using System.Security.Claims;
```

```
namespace SiteOfCompany.Service.Implementations
{
    public class AccountService : IAccountService
    {
        private readonly IUserRepository _userRepository;

        public AccountService(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }
        public async Task<IBaseResponse<ClaimsIdentity>> Login(LoginViewModel
loginViewModel)
        {
            try
            {
                var user = await _userRepository.GetByLogin(loginViewModel.Login);

                if (user == null)
                {
                    return new BaseResponse<ClaimsIdentity>()
                    {
                        Description = "Пользователь не найден"
                    };
                }

                if(user.Password !=
HashPasswordHelper.HashPassword(loginViewModel.Password))
                {
                    return new BaseResponse<ClaimsIdentity>()
                    {
                        Description = "Неверный пароль"
                    };
                }

                var result = Authenticate(user);

                return new BaseResponse<ClaimsIdentity>()
                {
                    Data = result,
                    StatusCode = StatusCode.OK
                };
            }
            catch (Exception ex)
            {
                return new BaseResponse<ClaimsIdentity>()
                {
                    Description = $"{Login} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }

        public async Task<IBaseResponse<ClaimsIdentity>> Register(RegisterViewModel
registerViewModel)
        {
            try
            {
                var user = await
_userRepository.GetByLogin(registerViewModel.Login);

                if (user != null)
                {
                    return new BaseResponse<ClaimsIdentity>()
                    {
```

```
                        Description = "Пользователь с таким логином уже есть"
                    };
                }

                user = new User()
                {
                    Login = registerViewModel.Login,
                    Role = Role.User,
                    Password =
HashPasswordHelper.HashPassword(registerViewModel.Password)
                };

                await _userRepository.Create(user);
                var result = Authenticate(user);

                return new BaseResponse<ClaimsIdentity>()
                {
                    Data = result,
                    Description = "Объект добавился",
                    StatusCode = StatusCode.OK
                };
            }
            catch (Exception ex)
            {
                return new BaseResponse<ClaimsIdentity>()
                {
                    Description = $"{Register} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }
        private ClaimsIdentity Authenticate(User user)
        {
            var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType, user.Login),
                new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.ToString())
            };
            return new ClaimsIdentity(claims, "ApplicationCookie",
                ClaimsIdentity.DefaultNameClaimType,
ClaimsIdentity.DefaultRoleClaimType);
        }
    }
}
```

ProjectService.cs:

```
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Enum;
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.Project;
using SiteOfCompany.Service.Interfaces;

namespace SiteOfCompany.Service.Implementations
{
      //Класс ProjectService, представляющий собой слой взаимодействия между базой
данных и самим приложением (в нём прописана реализация методов взаимодействия с
базой данных на более абстрактном уровне)
      public class ProjectService : IProjectService
      {
            private readonly IProjectRepository _projectRepository;

            public ProjectService(IProjectRepository projectRepository)
            {
                  _projectRepository = projectRepository;
```

```
        }

        public async Task<IBaseResponse<ProjectViewModel>> GetProject(int id)
        {
            try
            {
                var project = await _projectRepository.Get(id);

                if (project == null)
                {
                    return new BaseResponse<ProjectViewModel>()
                    {
                        Description = "Проект не найден",
                        StatusCode = StatusCode.ProjectNotFound
                    };
                }

                var data = new ProjectViewModel()
                {
                    ProjectName = project.ProjectName,
                    ProjectType = project.ProjectType,
                    Employee = project.Employee,
                    ProjectTime = project.ProjectTime,
                    PriceForProjectTime = project.PriceForProjectTime
                };

                return new BaseResponse<ProjectViewModel>()
                {
                    Data = data,
                    StatusCode = StatusCode.OK
                };
            }
            catch (Exception ex)
            {
                return new BaseResponse<ProjectViewModel>()
                {
                    Description = $"{GetProject} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }

        public async Task<IBaseResponse<ProjectViewModel>>
GetProjectByName(string name)
        {
            try
            {
                var project = await _projectRepository.GetByName(name);

                if (project == null)
                {
                    return new BaseResponse<ProjectViewModel>()
                    {
                        Description = "Проект не найден",
                        StatusCode = StatusCode.ProjectNotFound
                    };
                }

                var data = new ProjectViewModel()
                {
                    ProjectName = project.ProjectName,
                    ProjectType = project.ProjectType,
                    Employee = project.Employee,
                    ProjectTime = project.ProjectTime,
                    PriceForProjectTime = project.PriceForProjectTime
```

```
                        };

                        return new BaseResponse<ProjectViewModel>()
                        {
                                Data = data,
                                StatusCode = StatusCode.OK
                        };

                }
                catch (Exception ex)
                {
                        return new BaseResponse<ProjectViewModel>()
                        {
                                Description = $"{GetProjectByName} : {ex.Message}",
                                StatusCode = StatusCode.InternalServerError
                        };
                }
        }

        public IBaseResponse<IEnumerable<Project>> GetProjects()
        {
                var baseResponse = new BaseResponse<IEnumerable<Project>>();
                try
                {
                        var projects = _projectRepository.Select();
                        if (projects == null)
                        {
                                baseResponse.Description = "Найдено 0 элементов";
                                baseResponse.StatusCode =
StatusCode.ProjectNotFound; //Изменить StatusCode
                                return baseResponse;
                        }

                        baseResponse.Data = projects;
                        baseResponse.StatusCode = StatusCode.OK;

                        return baseResponse;
                }
                catch (Exception ex)
                {
                        return new BaseResponse<IEnumerable<Project>>()
                        {
                                Description = $"{GetProjects} : {ex.Message}",
                                StatusCode = StatusCode.InternalServerError
                        };
                }
        }

        public async Task<IBaseResponse<bool>> DeleteProject(int id)
        {
                try
                {
                        var project = await _projectRepository.Get(id);

                        if (project == null)
                        {
                                return new BaseResponse<bool>()
                                {
                                        Description = "Проект не найден",
                                        StatusCode = StatusCode.InternalServerError,
                                        Data = false
                                };
                        }

                        await _projectRepository.Delete(project);
```

```
                        return new BaseResponse<bool>()
                        {
                                Data = true,
                                StatusCode = StatusCode.OK
                        };
                }
                catch (Exception ex)
                {
                        return new BaseResponse<bool>()
                        {
                                Description = $"{DeleteProject} : {ex.Message}",
                                StatusCode = StatusCode.InternalServerError
                        };
                }
        }

        public async Task<IBaseResponse<Project>>
CreateProject(ProjectViewModel projectViewModel)
        {
                try
                {
                        var project = new Project()
                        {
                                ProjectName = projectViewModel.ProjectName,
                                ProjectType = projectViewModel.ProjectType,
                                Employee = projectViewModel.Employee,
                                ProjectTime = projectViewModel.ProjectTime,
                                PriceForProjectTime =
projectViewModel.PriceForProjectTime
                        };

                        await _projectRepository.Create(project);

                        return new BaseResponse<Project>()
                        {
                                Data = project,
                                StatusCode = StatusCode.OK
                        };
                }
                catch (Exception ex)
                {
                        return new BaseResponse<Project>()
                        {
                                Description = $"{CreateProject} : {ex.Message}",
                                StatusCode = StatusCode.InternalServerError
                        };
                }
        }

        public async Task<IBaseResponse<Project>> Edit(int id, ProjectViewModel
projectViewModel)
        {
                try
                {
                        var project = await _projectRepository.Get(id);

                        if (project == null)
                        {
                                return new BaseResponse<Project>()
                                {
                                        Description = "Проект не найден",
                                        StatusCode = StatusCode.ProjectNotFound
                                };
                        }
```

```
                        project.ProjectName = projectViewModel.ProjectName;
                        project.ProjectType = projectViewModel.ProjectType;
                        project.Employee = projectViewModel.Employee;
                        project.ProjectTime = projectViewModel.ProjectTime;
                        project.PriceForProjectTime =
projectViewModel.PriceForProjectTime;

                        await _projectRepository.Update(project);

                        return new BaseResponse<Project>()
                        {
                              Data = project,
                              StatusCode = StatusCode.OK
                        };
                  }
                  catch (Exception ex)
                  {
                        return new BaseResponse<Project>()
                        {
                              Description = $"{Edit} : {ex.Message}",
                              StatusCode = StatusCode.InternalServerError
                        };
                  }
            }
      }
}
```

UserService.cs:

```
using SiteOfCompany.DAL.Interfaces;
using SiteOfCompany.Domain.Entity;
using SiteOfCompany.Domain.Enum;
using SiteOfCompany.Domain.Helpers;
using SiteOfCompany.Domain.Response;
using SiteOfCompany.Domain.ViewModels.User;
using SiteOfCompany.Service.Interfaces;

namespace SiteOfCompany.Service.Implementations
{
    public class UserService : IUserService
    {
        private readonly IUserRepository _userRepository;

        public UserService(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        public async Task<IBaseResponse<User>> CreateUser(UserViewModel
userViewModel)
        {
            try
            {
                var user = await _userRepository.GetByLogin(userViewModel.Login);
                if (user != null)
                {
                    return new BaseResponse<User>()
                    {
                        Description = "Пользователь с таким логином уже существует",
                        StatusCode = Domain.Enum.StatusCode.UserAlreadyExists
                    };
                }

                user = new User()
```

```
                {
                    Login = userViewModel.Login,
                    Role = userViewModel.Role,
                    Password =
HashPasswordHelper.HashPassword(userViewModel.Password)
                };

                await _userRepository.Create(user);

                return new BaseResponse<User>()
                {
                    Data = user,
                    Description = "Пользователь добавлен",
                    StatusCode = Domain.Enum.StatusCode.OK
                };
            }
            catch (Exception ex)
            {
                return new BaseResponse<User>()
                {
                    Description = $"{CreateUser} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }

        public async Task<IBaseResponse<bool>> DeleteUser(int id)
        {
            try
            {
                var user = await _userRepository.Get(id);
                if (user == null)
                {
                    return new BaseResponse<bool>()
                    {
                        Description = "Пользователь не найден",
                        StatusCode = StatusCode.UserNotFound,
                        Data = false
                    };
                }

                await _userRepository.Delete(user);

                return new BaseResponse<bool>()
                {
                    Data = true,
                    StatusCode = StatusCode.OK
                };
            }
            catch(Exception ex)
            {
                return new BaseResponse<bool>()
                {
                    Description = $"{DeleteUser} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }

        public IBaseResponse<IEnumerable<User>> GetUsers()
        {
            var baseResponse = new BaseResponse<IEnumerable<User>>();
            try
            {
                var users = _userRepository.Select();
```

```
                if(users == null)
                {
                    baseResponse.Description = "Найдено 0 пользователей";
                    baseResponse.StatusCode = StatusCode.UserNotFound;
                    return baseResponse;
                }

                baseResponse.Data = users;
                baseResponse.StatusCode = Domain.Enum.StatusCode.OK;
                return baseResponse;
            }
            catch (Exception ex)
            {
                return new BaseResponse<IEnumerable<User>>()
                {
                    Description = $"{GetUsers} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }

        public async Task<IBaseResponse<User>> EditUser(int id, UserViewModel
userViewModel)
        {
            try
            {
                var user = await _userRepository.Get(id);
                if(user == null)
                {
                    return new BaseResponse<User>()
                    {
                        Description = "Пользователь не найден",
                        StatusCode = StatusCode.UserNotFound
                    };
                }

                user.Role = userViewModel.Role;

                return new BaseResponse<User>()
                {
                    Data = user,
                    StatusCode = StatusCode.OK
                };
            }
            catch(Exception ex)
            {
                return new BaseResponse<User>()
                {
                    Description = $"{EditUser} : {ex.Message}",
                    StatusCode = StatusCode.InternalServerError
                };
            }
        }
    }
}
```

ProjectController.cs:

```
using Microsoft.AspNetCore.Mvc;
using SiteOfCompany.Domain.ViewModels.Project;
using SiteOfCompany.Service.Interfaces;

namespace SiteOfCompany.Controllers
{
      public class ProjectController : Controller
```

```
    {
        private readonly IProjectService _projectService;
        public ProjectController(IProjectService projectService)
        {
            _projectService = projectService;
        }

        public IActionResult ProjectsInfo()
        {
            return View();
        }
        public IActionResult GetProjects()
        {
            var response = _projectService.GetProjects();

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return View(response.Data.ToList());
            }

            return RedirectToAction("Error");
        }

        public IActionResult ResultProjects()
        {
            var response = _projectService.GetProjects();

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return View(response.Data.ToList());
            }

            return RedirectToAction("Error");
        }

        public async Task<IActionResult> GetProject(int id)
        {
            var response = await _projectService.GetProject(id);

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return View(response.Data);
            }

            return RedirectToAction("Error");
        }

        public async Task<IActionResult> DeleteProject(int id)
        {
            var response = await _projectService.DeleteProject(id);

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return RedirectToAction("GetProjects");
            }

            return RedirectToAction("Error");
        }

        [HttpGet]
        public IActionResult CreateProject() => View();

        [HttpPost]
        public async Task<IActionResult> CreateProject(ProjectViewModel
projectViewModel)
```

```
            {
                  if (ModelState.IsValid)
                  {
                        var response = await
_projectService.CreateProject(projectViewModel);

                        if (response.StatusCode == Domain.Enum.StatusCode.OK)
                        {
                              return RedirectToAction("GetProjects", "Project");
                        }

                        ModelState.AddModelError("", response.Description);
                  }

                  return View(projectViewModel);
            }


            public async Task<IActionResult> Save(int id)
            {
                  if (id == 0)
                  {
                        return View();
                  }

                  var response = await _projectService.GetProject(id);

                  if (response.StatusCode == Domain.Enum.StatusCode.OK)
                  {
                        return View(response.Data);
                  }

                  return RedirectToAction("Error");
            }


            public async Task<IActionResult> Save(ProjectViewModel
projectViewModel)
            {
                  if (ModelState.IsValid)
                  {
                        if (projectViewModel.Id == 0)
                        {
                              await
_projectService.CreateProject(projectViewModel);
                        }
                        else
                        {
                              await _projectService.Edit(projectViewModel.Id,
projectViewModel);
                        }
                  }

                  return RedirectToAction("GetProjects");
            }
      }
}
```

UserController.cs:

```
using Microsoft.AspNetCore.Mvc;
using SiteOfCompany.Domain.ViewModels.User;
using SiteOfCompany.Service.Interfaces;

namespace SiteOfCompany.Controllers
```

```
{
    public class UserController : Controller
    {
        private readonly IUserService _userService;

        public UserController(IUserService userService)
        {
            _userService = userService;
        }
        public IActionResult GetUsers()
        {
            var response = _userService.GetUsers();

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return View(response.Data);
            }

            return RedirectToAction("Error");
        }

        public async Task<IActionResult> DeleteUser(int id)
        {
            var response = await _userService.DeleteUser(id);

            if (response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return RedirectToAction("GetUsers");
            }

            return RedirectToAction("Error");
        }

        public async Task<IActionResult> EditUser(int id, UserViewModel user)
        {
            var response = await _userService.EditUser(id, user);

            if(response.StatusCode == Domain.Enum.StatusCode.OK)
            {
                return RedirectToAction("GetUsers");
            }

            return RedirectToAction("Error");
        }

             [HttpGet]
             public IActionResult CreateUser() => View();

             [HttpPost]
             public async Task<IActionResult> CreateUser(UserViewModel
userViewModel)
             {
                    if (ModelState.IsValid)
                    {
                           var response = await
_userService.CreateUser(userViewModel);

                           if (response.StatusCode == Domain.Enum.StatusCode.OK)
                           {
                                  return RedirectToAction("GetUsers", "User");
                           }

                           ModelState.AddModelError("", response.Description);
                    }
```

```
                return View(userViewModel);
            }
        }
}
```

AccountController.cs:

```
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Mvc;
using SiteOfCompany.Domain.ViewModels.Account;
using SiteOfCompany.Service.Interfaces;
using System.Security.Claims;

namespace SiteOfCompany.Controllers
{
    public class AccountController : Controller
    {
        private readonly IAccountService _accountService;

        public AccountController(IAccountService accountService)
        {
            _accountService = accountService;
        }

        [HttpGet]
        public IActionResult Register() => View();

        [HttpPost]
        public async Task<IActionResult> RegisterAsync(RegisterViewModel
registerViewModel)
        {
            if(ModelState.IsValid)
            {
                var response = await _accountService.Register(registerViewModel);
                if(response.StatusCode == Domain.Enum.StatusCode.OK)
                {
                    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                        new ClaimsPrincipal(response.Data));

                    return RedirectToAction("Index", "Home");
                }

                ModelState.AddModelError("", response.Description);
            }

            return View(registerViewModel);
        }

        [HttpGet]
        public IActionResult Login() => View();

        [HttpPost]
        public async Task<IActionResult> Login(LoginViewModel loginViewModel)
        {
            if(ModelState.IsValid)
            {
                var response = await _accountService.Login(loginViewModel);
                if(response.StatusCode == Domain.Enum.StatusCode.OK)
                {
                    await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                        new ClaimsPrincipal(response.Data));
```

```
                    return RedirectToAction("Index", "Home");
                }

                ModelState.AddModelError("", response.Description);
            }

            return View(loginViewModel);
        }

        [HttpPost]
        public async Task<IActionResult> Authorization()
        {
            return View();
        }

        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Logout()
        {
            await
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
            return RedirectToAction("Index", "Home");
        }
    }
}
```

HomeController.cs:

```
using Microsoft.AspNetCore.Mvc;
using SiteOfCompany.Models;
using System.Diagnostics;

namespace SiteOfCompany.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore
= true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ??
HttpContext.TraceIdentifier });
        }
    }
}
```

Search.js:

```
$(document).ready(function () {
```

```
    var searchInput = $('#project-search-input');

    searchInput.on('input', function () {
        var searchText = $(this).val().toLowerCase();

        $('.mytable tbody tr').each(function () {
            var projectName = $(this).find('td:first-child').text().toLowerCase();

            if (projectName.includes(searchText)) {
                $(this).show();
            } else {
                $(this).hide();
            }
        });
    });
});
```

Sorting.js:

```
$(document).ready(function () {

    var tableHeaders = $('.mytable th');


    tableHeaders.click(function () {
        var table = $(this).closest('.mytable');
        var columnIndex = $(this).index();


        var sortOrder = $(this).hasClass('asc') ? 'desc' : 'asc';


        tableHeaders.removeClass('asc desc');


        $(this).addClass(sortOrder);


        var rows = table.find('tbody tr').get();


        rows.sort(function (a, b) {
            var cellA = $(a).find('td').eq(columnIndex).text().toUpperCase();
            var cellB = $(b).find('td').eq(columnIndex).text().toUpperCase();

            if (sortOrder === 'asc') {
                return cellA.localeCompare(cellB);
            } else {
                return cellB.localeCompare(cellA);
            }
        });

        $.each(rows, function (index, row) {
            table.children('tbody').append(row);
        });
    });
});
```

Html:

_Layout.cshtml:

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="utf-8" />
```

```
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>@ViewData["Title"] TY App</title>
        <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
        <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
        <link rel="stylesheet" href="~/SiteOfCompany.styles.css" asp-append-
version="true" />
        <link rel="stylesheet" href="~/css/style.css" asp-append-version="true"/>
        <link rel="stylesheet" href="~/css/tableStyle.css" asp-append-version="true"
/>
        <link rel="stylesheet" href="~/css/modal.css" asp-append-version="true" />
        <link rel="shortcut icon" href="~/icons/Logo.ico" type="image/x-icon" />
</head>
<body class="backgroundImage">
        <header>
                <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light
box-shadow mb-3" style="background-color: black; border-bottom: 1px solid #9d9d9d">
                        <div class="container-fluid" style="padding: 0 2% 0 2%">
                                <a class="navbar-brand logo" asp-area="" asp-
controller="Home" asp-action="Index"><img src="~/img/Logo.png" alt="TYCompany"
width="40px" , height="40px" /></a>
                                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
                                        aria-expanded="false" aria-label="Toggle
navigation">
                                        <span class="navbar-toggler-icon"></span>
                                </button>
                                <div class="navbar-collapse collapse d-sm-inline-flex
justify-content-between">
                                        <ul class="navbar-nav flex-grow-1">
                                                @if (User.IsInRole("Admin"))
                                                {
                                                        <li class="nav-item">
                                                                <a class="nav-link usual_text
text-white" asp-controller="User" asp-action="GetUsers">Управление учётными
записями</a>
                                                        </li>
                                                }
                                        </ul>
                                        @if (User.Identity.IsAuthenticated)
                                        {
                                                <form method="post" asp-controller="Account"
asp-action="Logout">
                                                        <input class="usual_text mybtn
mybtn_danger" type="submit" value="Выход" />
                                                </form>
                                        }
                                </div>
                        </div>
                </nav>
        </header>
        <main role="main" s>
                @RenderBody()
        </main>

        <footer class="footer text-muted" style="background-color: black; border-top:
1px solid #9d9d9d">
                <div class="container" style="text-align: center">
                        Проект TYCompany (студент: Юрашевич Виктор Сергеевич, группа:
ПО-11)
                </div>
        </footer>
        <script src="~/lib/jquery/dist/jquery.min.js"></script>
        <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
        <script src="~/js/search.js" asp-append-version="true"></script>
```

```
        <script src="~/js/sorting.js" asp-append-version="true"></script>
        <script src="~/js/site.js" asp-append-version="true"></script>
        @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Authorization.cshtml:

```
@{
        ViewData["Title"] = "Аутентификация";
}


<section class="authorization">
        <div class="container authorization__container">
                <div class="authorization__intro">
                        <div class="title authorization__title text-white">TY App</div>
                </div>
                <div class="authorization__widnow">
                        <div class="rectangle authorization__rectangle">
                                <div class="usual_text authorization__description">
                                        Строим будущее с помощью технологий
                                </div>
                                <div class="authorization__buttons">
                                        <div>
                                                <form asp-controller="Account" asp-
action="Login">
                                                        <input class="mybtn
authorization__mybtn usual_text" type="submit" value="Вход в аккаунт"/>
                                                </form>
                                        </div>
                                        <div>
                                                <form asp-controller="Account" asp-
action="Register">
                                                        <input class="mybtn
authorization__mybtn usual_text" type="submit" value="Создать аккаунт"/>
                                                </form>
                                        </div>
                                </div>
                        </div>
                </div>
        </div>
</section>
```

Login.cshtml:

```
@model SiteOfCompany.Domain.ViewModels.Account.LoginViewModel

@{
        Layout = "_Layout";
        ViewBag.Title = "Авторизация";
}


<section class="login">
        <div class="container">
                <form asp-controller="Account" asp-action="Login">
                        <div class="rectangle login__rectangle">
                                <div class="login__description">
                                        <div class="login__title usual_text">Вход в TY
App</div>
                                        <div class="line"></div>
                                </div>
                                <div class="login__buttons">
                                        <div class="login__login_input">
```

```
                                    <div class="usual_text login__input">Введите
логин</div>
                                    <input asp-for="Login" type="text"
                                           placeholder="Логин"
                                           class="login__field field
usual_text" />
                              </div>
                              <div class="login__password_input">
                                    <div class="usual_text login__input">Введите
пароль</div>
                                    <input asp-for="Password" type="password"
                                           placeholder="Пароль"
                                           class="login__field field
usual_text" />
                              </div>
                        </div>
                        <div>
                              <input class="mybtn login__mybtn usual_text"
type="submit" value="Войти"/>
                        </div>
                  </div>
            </form>
      </div>
</section>
```

Register.cshtml:

```
@model SiteOfCompany.Domain.ViewModels.Account.RegisterViewModel

@{
      Layout = "_Layout";
      ViewBag.Title = "Регистрация";
}


<section class="register">
      <div class="container">
            <form asp-controller="Account" asp-action="Register">
                  <div class="rectangle register__rectangle">
                        <div class="register__description">
                              <div class="register__title usual_text">Регистрация
в TY App</div>
                              <div class="line"></div>
                        </div>
                        <div class="register__buttons">
                              <div class="register__login_input">
                                    <div class="usual_text
register__input">Введите логин</div>
                                    <input asp-for="Login" type="text"
                                           placeholder="Логин"
                                           class="register__field field
usual_text" />
                              </div>
                              <div class="register__password_input">
                                    <div class="usual_text
register__input">Введите пароль</div>
                                    <input asp-for="Password" type="password"
                                           placeholder="Пароль"
                                           class="register__field field
usual_text" />
                              </div>
                              <div class="register__password_input">
                                    <div class="usual_text
register__input">Повторите пароль</div>
                                    <input asp-for="PasswordConfirm"
type="password"
```

```
                                                placeholder="Пароль"
                                                class="register__field field
usual_text" />
                                        </div>
                                </div>
                                <div>
                                        <input class="mybtn register__mybtn usual_text"
type="submit" value="Отправить" />
                                </div>
                                <div>
                                        <a asp-controller="Account" asp-action="Login"
class="usual_ref register__link">Уже есть аккаунт</a>
                                </div>
                        </div>
                </form>
        </div>
</section>
```

Index.cshtml:

```
@{
        ViewData["Title"] = "Домашняя страница";
}

@if (!User.Identity.IsAuthenticated)
{
        <section class="index">
                <div class="container">
                        <div class="index__row">
                                <div class="index__text_column">
                                        <h1 class="title index__title text-start text-
white">
                                                Меняй свою жизнь вместе с TY Company
                                        </h1>
                                        <div class="usual_text index__description text-
start">
                                                TY Company - компания по разработке
качественного ПО. Данный
                                                ресурс разработан для сотрудников компании с
целью облегчения
                                                процесса разработки.
                                        </div>
                                        <form asp-action="Authorization" asp-
controller="Account">
                                                <input class="mybtn index__mybtn usual_text
text-uppercase" type="submit" value="Начать" />
                                        </form>
                                </div>
                                <div class="index__img_column">
                                        <img src="./img/office.jpg"
                                                alt="Office"
                                                class="index__img" />
                                </div>
                        </div>
                </div>
        </section>
}
@if (User.Identity.IsAuthenticated)
{
        <section class="entrance">
                <div class="container">
                        <div class="entrance__container">
                                <div class="entrance__introduction">
                                        <div class="title entrance__title">Добро пожаловать
в TY App</div>
```

```
                                    <div class="usual_text entrance__description">
                                          Чтобы начать работу в TY App, нажмите кнопку
“Продолжить”
                                    </div>
                              </div>
                              <form asp-controller="Project" asp-action="ProjectsInfo">
                                    <input class="mybtn entrance__mybtn usual_text text-
uppercase"
                                                type="submit"
                                                value="Продолжить" />
                              </form>
                        </div>
                  </div>
            </section>
}
```

CreateProject.cshtml:

```
@model SiteOfCompany.Domain.ViewModels.Project.ProjectViewModel

@{
      Layout = "_Layout";
      ViewBag.Title = "Добавление проекта";
}

<section class="register">
      <div class="container">
            <form asp-controller="Project" asp-action="CreateProject">
                  <div class="rectangle project__rectangle">
                        <div class="register__description">
                              <div class="register__title usual_text">Добавление
проекта</div>
                              <div class="line"></div>
                        </div>
                        <div class="register__buttons">
                              <div class="register__login_input">
                                    <div class="usual_text
register__input">Введите наименование проекта</div>
                                    <input asp-for="@Model.ProjectName"
type="text"
                                                placeholder="Наименование"
                                                class="register__field field
usual_text" />
                              </div>
                              <div class="register__password_input">
                                    <div class="usual_text
register__input">Введите вид работ</div>
                                    <input asp-for="@Model.ProjectType"
type="text"
                                                placeholder="Вид работ"
                                                class="register__field field
usual_text" />
                              </div>
                              <div class="register__password_input">
                                    <div class="usual_text
register__input">Введите содрутника</div>
                                    <input asp-for="@Model.Employee" type="text"
                                                placeholder="Сотрудник"
                                                class="register__field field
usual_text" />
                              </div>
                              <div class="register__password_input">
                                    <div class="usual_text
register__input">Введите количество часов</div>
```

```
                                    <input asp-for="@Model.ProjectTime"
type="text"
                                           placeholder="Количество часов"
                                           class="register__field field
usual_text" />
                                </div>
                                <div class="register__password_input">
                                    <div class="usual_text
register__input">Введите стоимость часа</div>
                                    <input asp-for="@Model.PriceForProjectTime"
type="text"
                                           placeholder="Стоимость часа"
                                           class="register__field field
usual_text" />
                                </div>
                            </div>
                            <div>
                                <input class="mybtn register__mybtn usual_text"
type="submit" value="Отправить" />
                            </div>
                        </div>
                    </form>
                </div>
            </section>
```

GetProjects.cshtml:

```
@using SiteOfCompany.Domain.Extensions;
@model List<SiteOfCompany.Domain.Entity.Project>

@{
    ViewBag.Title = "Таблица проектов";
    Layout = "_Layout";
}

<section class="projects">
    <div class="container">
        <div class="title projects__title">Таблица проектов</div>
        <div class="search">
            <input class="projects__search" id="project-search-input"
type="text" placeholder="Поиск">
        </div>
        <div class="projects__table">
            <table class="mytable">
                <thead>
                    <tr>
                        <th>Наименование проекта</th>
                        <th>Вид работ</th>
                        <th>Ф.И.О. Сотрудника</th>
                        <th>Предполагаемое количество часов</th>
                        <th>Стоимость часов</th>
                        <th></th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var project in Model)
                    {
                        <tr>
                            <td>@project.ProjectName</td>

    <td>@EnumExtension.GetDisplayName(project.ProjectType)</td>
                            <td>@project.Employee</td>
                            <td>@project.ProjectTime</td>
                            <td>@project.PriceForProjectTime</td>
                            <td>
```

```
                                                <a asp-controller="Project" asp-
action="DeleteProject" asp-route-id="@project.Id">
                                                      <img
src="~/icons/crossing.svg"
                                                            alt="Crossing"
                                                            class="cross" />
                                                </a>
                                          </td>
                                          <td>
                                                <a>
                                                      <img
src="~/icons/edit.svg"
                                                           alt="Edit"
                                                            class="edit"/>
                                                </a>
                                          </td>
                                    </tr>
                              }
                        </tbody>
                  </table>
            </div>
            <div class="projects__buttons">
                  <div>
                        <form asp-controller="Project" asp-
action="ResultProjects">
                              <button class="mybtn usual_text
projects__mybtn">Провести анализ каждого проекта</button>
                        </form>
                  </div>
                  <div>
                        <form asp-controller="Project" asp-action="CreateProject">
                              <button class="mybtn usual_text
project__mybtn">Добавить проект</button>
                        </form>
                  </div>
            </div>
      </div>
</section>
```

ProjectsInfo.cshtml:

```
@{
      ViewData["Title"] = "Работа с проектами";
}

<section class="base">
      <div class="container">
            <div class="base__introduction">
                  <div class="title base__title">Работа над проектами</div>
                  <div class="base__line"></div>
            </div>
            <div class="base__projects">
                  <div class="usual_text base__description">
                        TY App служит хорошим инструментом для облегчения процесса
                        разработки ПО, сотрудники компании могут понять на каком
этапе
                        разработки находится проект и кто над ним работает, помимо
этого
                        данное приложение помогает распределить бюджет, выделенный
на
                        проект, узнать итоговую стоимость проекта и в зависимости
от этого
                        принимать определённые решения.
                  </div>
                  <div class="base__btn">
                        <form asp-controller="Project" asp-action="GetProjects">
```

```
                        <input type="submit"
                               class="mybtn base__mybtn usual_text"
                               value="Перейти к проектам" />
                    </form>
                </div>
            </div>
        </div>
</section>
```

ResultProjects.cshtml:

```
@using SiteOfCompany.Domain.Extensions

@model List<SiteOfCompany.Domain.Entity.Project>

@{
        ViewBag.Title = "title";
        Layout = "_Layout";
}
<section class="projects">
        <div class="container">
                <div class="title projects__title">Таблица проектов</div>
                <div class="search">
                        <input class="projects__search" id="project-search-input"
type="text" placeholder="Поиск">
                </div>
                <div class="projects__table">
                        <table class="mytable">
                                <thead>
                                        <tr>
                                                <th>Наименование проекта</th>
                                                <th>Вид работ</th>
                                                <th>Ф.И.О. Сотрудника, отвечающего за
проект</th>
                                                <th>Итоговая стоимость</th>
                                        </tr>
                                </thead>
                                <tbody>
                                        @foreach (var project in Model)
                                        {
                                                <tr>
                                                        <td>@project.ProjectName</td>

        <td>@EnumExtension.GetDisplayName(project.ProjectType)</td>
                                                        <td>@project.Employee</td>
                                                        @{
                                                                var price = project.ProjectTime *
project.PriceForProjectTime;
                                                                <td>@price</td>
                                                        }
                                                </tr>
                                        }
                                </tbody>
                        </table>
                </div>
                <div>
                        <form asp-controller="Project" asp-action="GetProjects">
                                <button class="mybtn usual_text projects__mybtn">Вернуться
назад</button>
                        </form>
                </div>
        </div>
</section>
```

CreateUser.cshtml:

```
@model SiteOfCompany.Domain.ViewModels.User.UserViewModel

@{
	Layout = "_Layout";
	ViewBag.Title = "Добавление пользователя";
}

<section class="login">
	<div class="container">
		<form asp-controller="User" asp-action="CreateUser">
			<div class="rectangle user__rectangle">
				<div class="login__description">
					<div class="login__title usual_text">Добавление
пользователя</div>
					<div class="line"></div>
				</div>
				<div class="login__buttons">
					<div class="login__login_input">
						<div class="usual_text login__input">Введите
логин</div>
						<input asp-for="Login" type="text"
							placeholder="Логин"
							class="login__field field
usual_text" />
					</div>
					<div class="login__password_input">
						<div class="usual_text login__input">Введите
пароль</div>
						<input asp-for="Password" type="text"
							placeholder="Пароль"
							class="login__field field
usual_text" />
					</div>
					<div class="login__password_input">
						<div class="usual_text login__input">Введите
роль</div>
						<input asp-for="Role" type="text"
							placeholder="Роль"
							class="login__field field
usual_text" />
					</div>
				</div>
				<div>
					<input class="mybtn login__mybtn usual_text"
type="submit" value="Отправить" />
				</div>
			</div>
		</form>
	</div>
</section>
```

GetUsers.cshtml:

```
@using SiteOfCompany.Domain.Extensions
@model List<SiteOfCompany.Domain.Entity.User>

@{
	ViewBag.Title = "Таблица учётных записей";
	Layout = "_Layout";
}

<section class="users">
	<div class="container">
		<div class="title users__title">Таблица учётных записей</div>
```

```
            <div class="search">
                  <input class="users__search" id="project-search-input"
type="text" placeholder="Поиск">
            </div>
            <div class="users__table">
                  <table class="mytable">
                        <thead>
                              <tr>
                                    <th>Логин</th>
                                    <th>Роль</th>
                                    <th></th>
                                    <th></th>
                              </tr>
                        </thead>
                        <tbody>
                              @foreach (var user in Model)
                              {
                                    <tr>
                                          <td>@user.Login</td>

      <td>@EnumExtension.GetDisplayName(user.Role)</td>
                                          <td>
                                                <a asp-controller="User" asp-
action="DeleteUser" asp-route-id="@user.Id">
                                                      <img
src="~/icons/crossing.svg"
                                                            alt="Crossing"
                                                            class="cross" />
                                                </a>
                                          </td>
                                          <td>
                                                <a>
                                                      <img
src="~/icons/edit.svg"
                                                      alt="Edit"
                                                      class="edit"/>
                                                </a>
                                          </td>
                                    </tr>
                              }
                        </tbody>
                  </table>
            </div>
            <div>
                  <form asp-controller="User" asp-action="CreateUser">
                        <button class="mybtn usual_text user__mybtn">Добавить
пользователя</button>
                  </form>
            </div>
      </div>
</section>
```