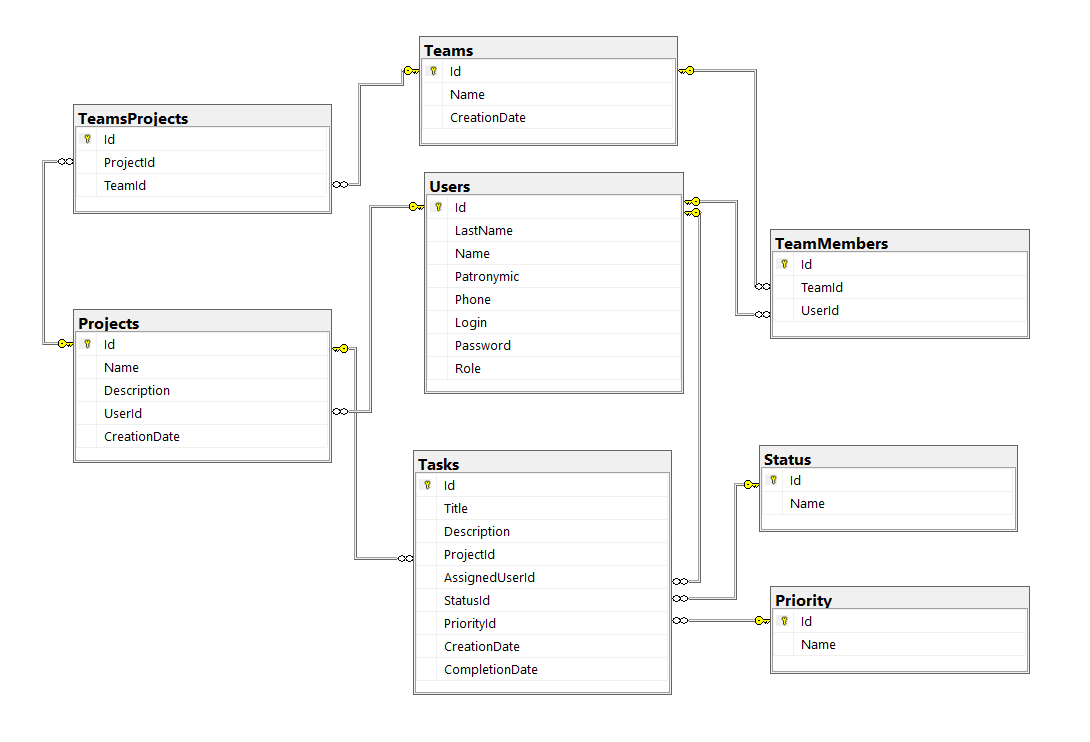
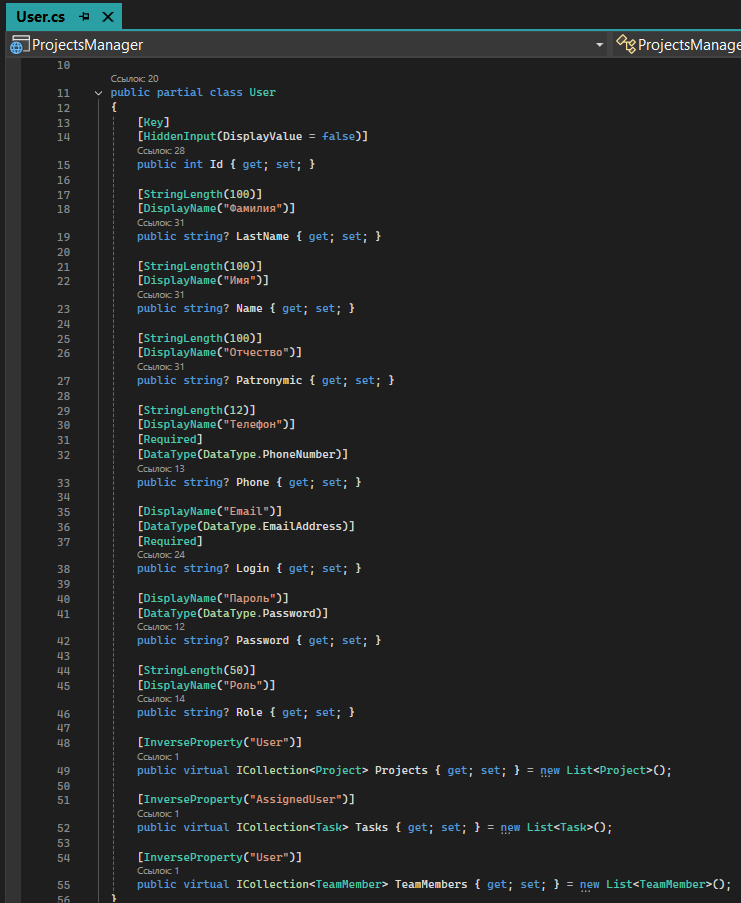
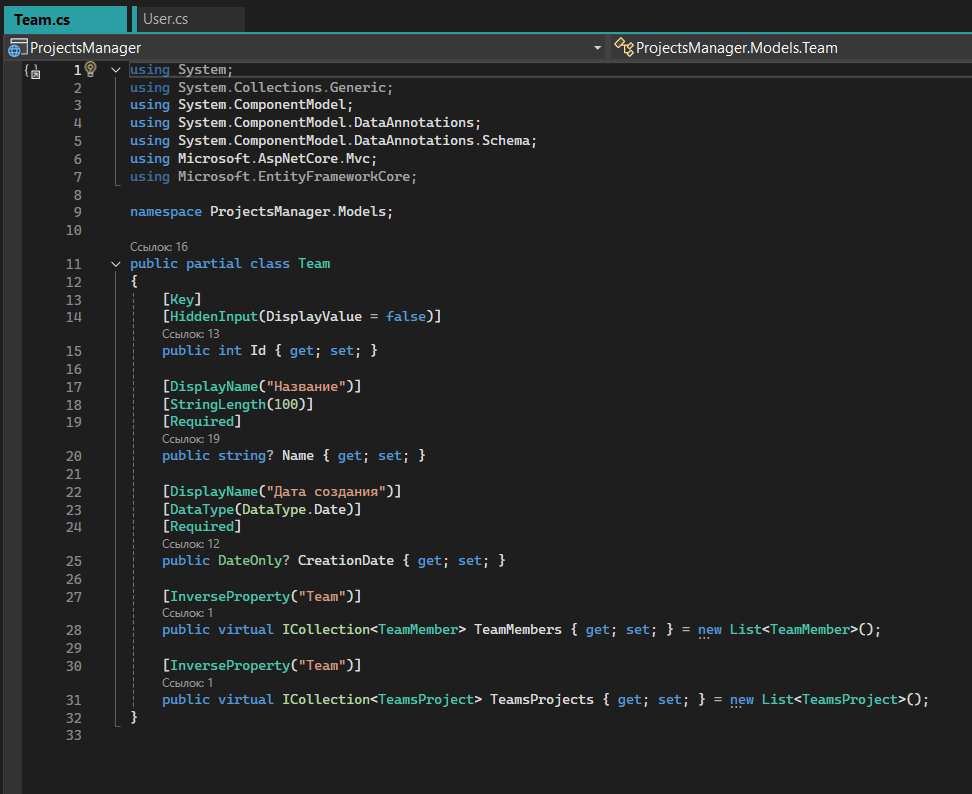
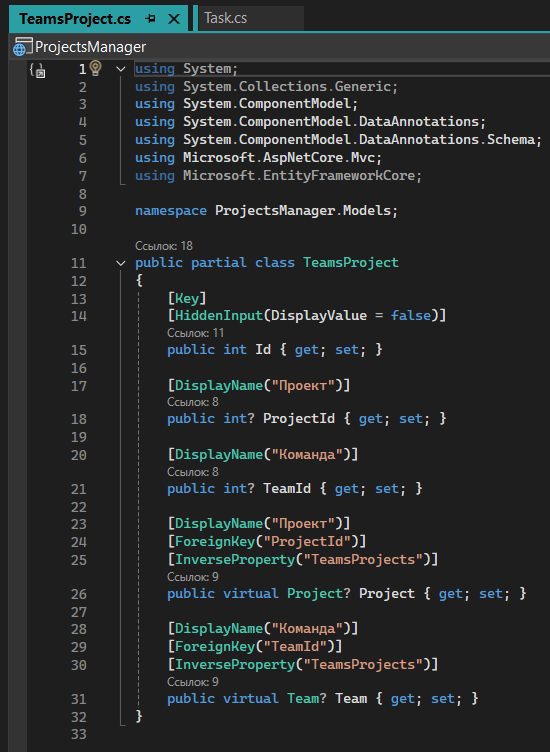
****

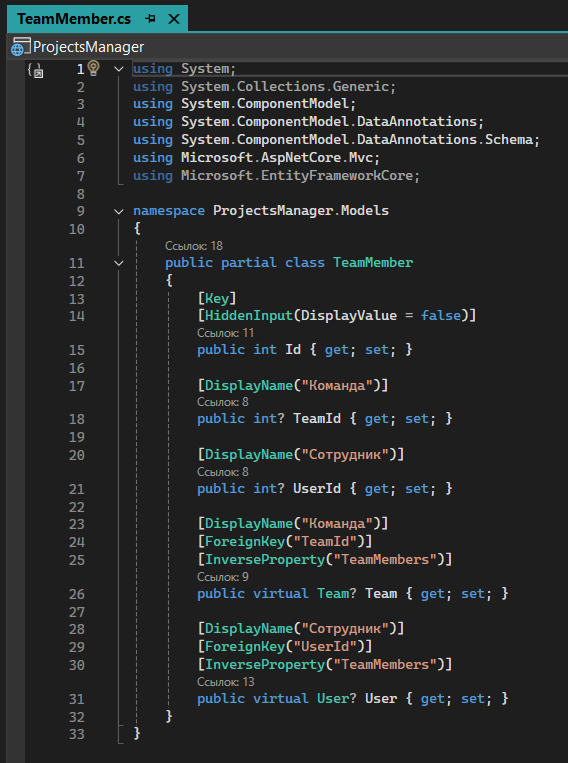
**Модели**

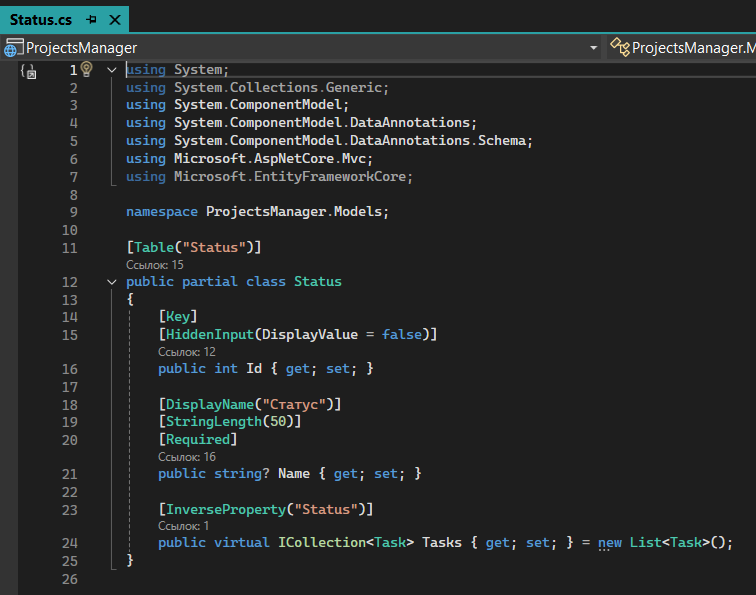


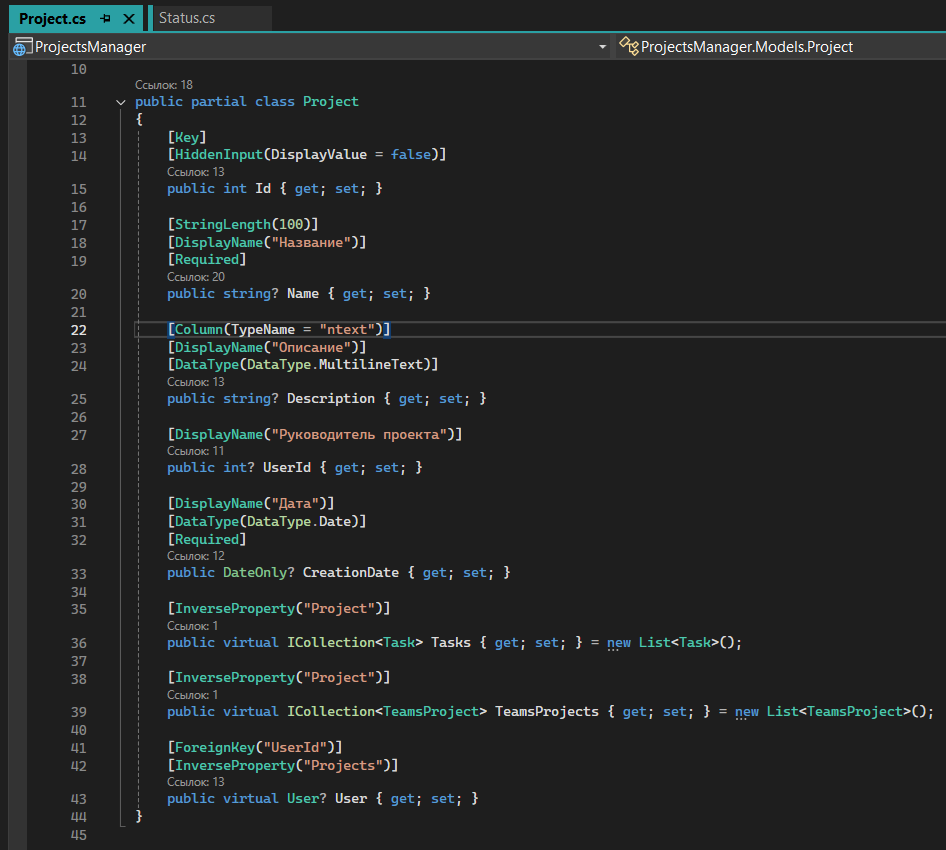


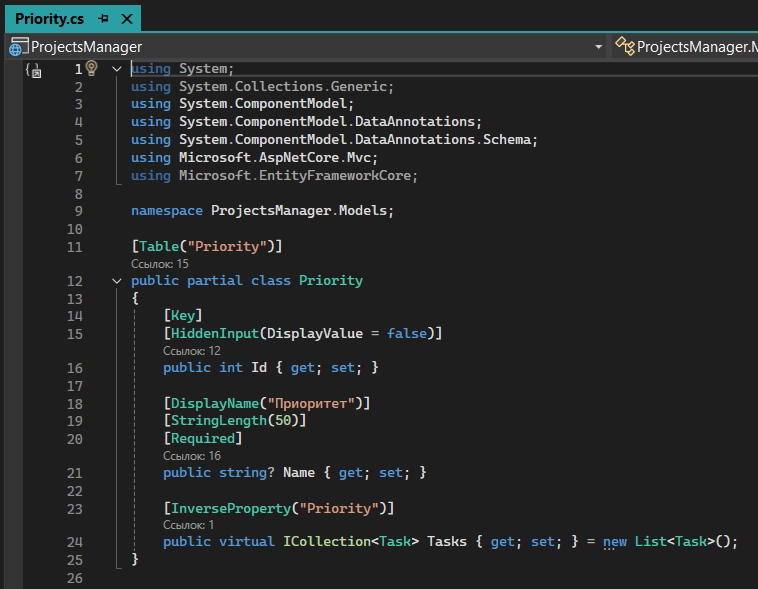












**ProjectsManagerContext.cs**

Объявление таблиц, установка связей между ними и строка подключения к бдpublic partial class ProjectsManagerContext : DbContext

{

public ProjectsManagerContext()

{

}

public ProjectsManagerContext(DbContextOptions<ProjectsManagerContext> options)

: base(options)

{

}

public virtual DbSet<Priority> Priorities { get; set; }

public virtual DbSet<Project> Projects { get; set; }

public virtual DbSet<Status> Statuses { get; set; }

public virtual DbSet<Task> Tasks { get; set; }

public virtual DbSet<Team> Teams { get; set; }

public virtual DbSet<TeamMember> TeamMembers { get; set; }

public virtual DbSet<TeamsProject> TeamsProjects { get; set; }

public virtual DbSet<User> Users { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

=> optionsBuilder.UseSqlServer("Data Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=C:\\College\\Дипломы\\ProjectsManager\\ProjectsManager\\App\_data\\ProjectsManager.mdf;Integrated Security=True");

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Priority>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Priority\_\_3214EC07C5140BF9");

entity.Property(e => e.Name).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

});

modelBuilder.Entity<Project>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_tmp\_ms\_x\_\_3214EC07DB4DBA73");

entity.Property(e => e.Description).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

entity.Property(e => e.Name).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

entity.HasOne(d => d.User).WithMany(p => p.Projects).HasConstraintName("FK\_Projects\_ToUsers");

});

modelBuilder.Entity<Status>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Status\_\_3214EC07C989301E");

entity.Property(e => e.Name).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

});

modelBuilder.Entity<Task>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_Table");

entity.Property(e => e.Description).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

entity.Property(e => e.Title).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

entity.HasOne(d => d.AssignedUser).WithMany(p => p.Tasks).HasConstraintName("FK\_Tasks\_ToUsers");

entity.HasOne(d => d.Priority).WithMany(p => p.Tasks).HasConstraintName("FK\_Tasks\_ToPriority");

entity.HasOne(d => d.Project).WithMany(p => p.Tasks).HasConstraintName("FK\_Tasks\_ToProjects");

entity.HasOne(d => d.Status).WithMany(p => p.Tasks).HasConstraintName("FK\_Tasks\_ToStatus");

});

modelBuilder.Entity<Team>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Teams\_\_3214EC078C4AADF2");

entity.Property(e => e.Name).UseCollation("SQL\_Latin1\_General\_CP1\_CI\_AS");

});

modelBuilder.Entity<TeamMember>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_TeamMember");

entity.HasOne(d => d.Team).WithMany(p => p.TeamMembers).HasConstraintName("FK\_TeamMembers\_ToTeams");

entity.HasOne(d => d.User).WithMany(p => p.TeamMembers).HasConstraintName("FK\_TeamMembers\_ToUsers");

});

modelBuilder.Entity<TeamsProject>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_TeamsPro\_\_3214EC079A8BFBAC");

entity.HasOne(d => d.Project).WithMany(p => p.TeamsProjects).HasConstraintName("FK\_TeamsProjects\_ToProjects");

entity.HasOne(d => d.Team).WithMany(p => p.TeamsProjects).HasConstraintName("FK\_TeamsProjects\_ToTeams");

});

modelBuilder.Entity<User>(entity =>

{

entity.HasKey(e => e.Id).HasName("PK\_\_Table\_\_3214EC0797687DAD");

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

**Program.cs**

Основной модуль программы. Создание приложения, подключение к БД, подключение уведомлений для пользователя, использование авторизации и аутентификации пользователя. Модуль с установкой аутентификационных куки при входе в систему, чтобы приложение помнило всегда с каким конкретным пользователем работает.

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ProjectsManagerContext>(options =>

{

options.UseSqlServer(builder.Configuration.GetConnectionString("ProjectsManagerContext"));

});

// Add services to the container.

builder.Services.AddControllersWithViews().AddRazorRuntimeCompilation();

// Add ToastNotification

builder.Services.AddNotyf(config =>

{

config.DurationInSeconds = 5;

config.IsDismissable = true;

config.Position = NotyfPosition.TopCenter;

});

// Подключаю куки

builder.Services.AddAuthentication("Cookies").AddCookie(options => options.LoginPath = "/Users/Login");

builder.Services.AddAuthorization();

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.UseNotyf();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapGet("/Users/SignIn/{id:int}", async (string? returnUrl, HttpContext context, int id, ProjectsManagerContext \_context) =>

{

var user = \_context.Users.FirstOrDefaultAsync(u => u.Id == id);

if (user.Result != null && user.Result.Role != null)

{

var claims = new List<Claim> { new Claim(ClaimTypes.Name, user.Result.Login), new Claim(ClaimTypes.Role, user.Result.Role), new Claim("ID", user.Result.Id.ToString()) };

// создаем объект ClaimsIdentity

ClaimsIdentity claimsIdentity = new ClaimsIdentity(claims, "Cookies");

// установка аутентификационных куки

await context.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));

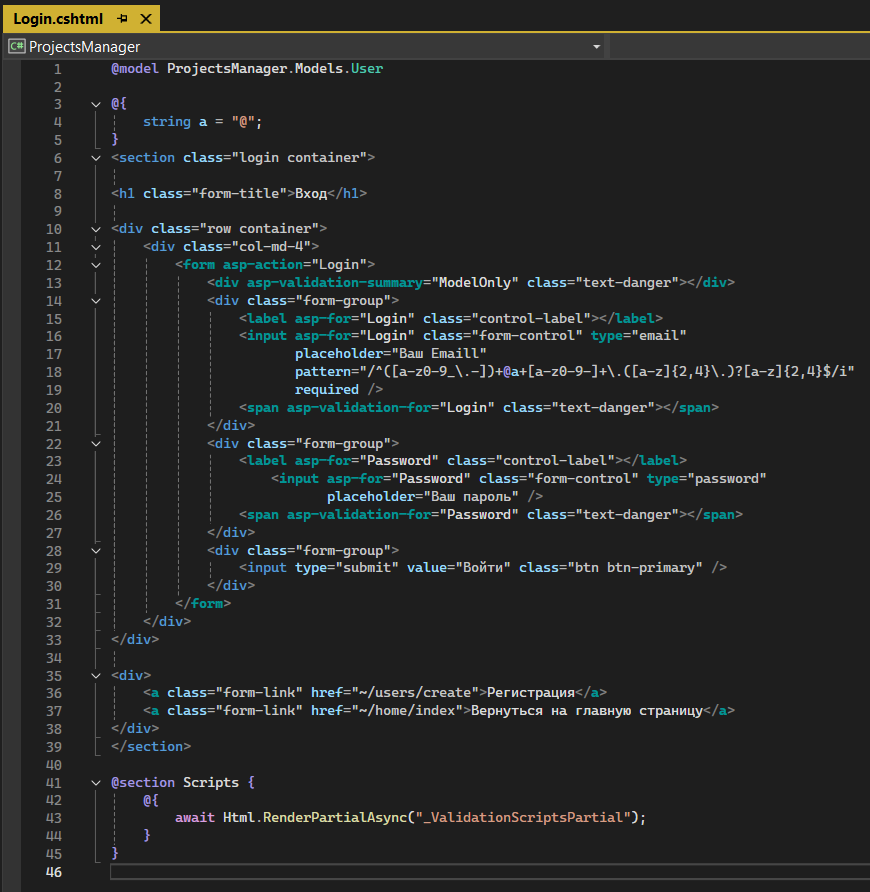
return Results.Redirect($"~/Users/Details/{user.Result.Id}");

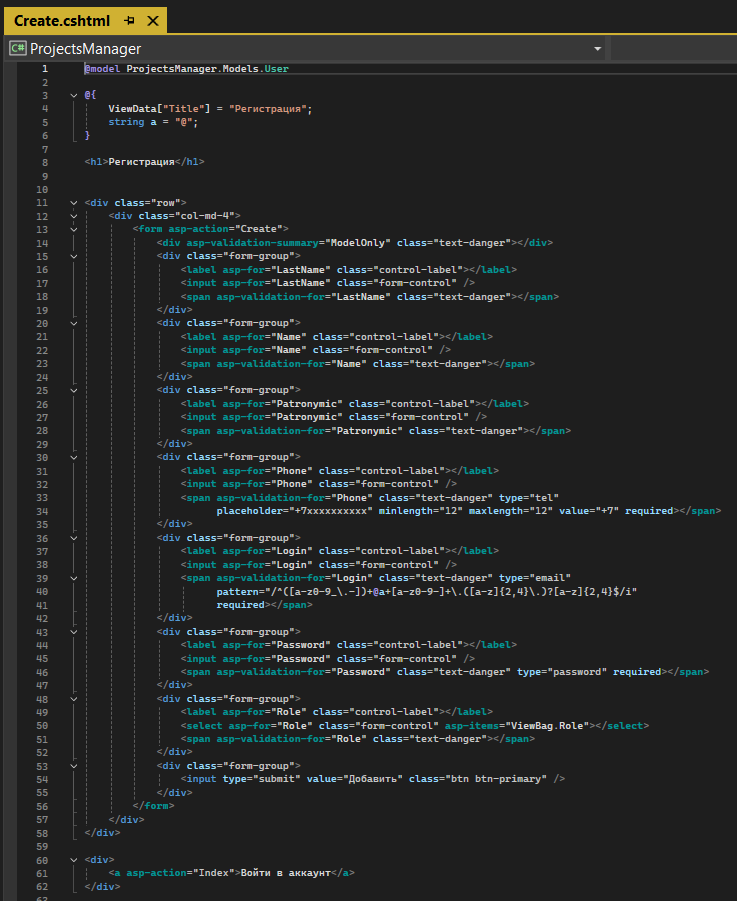
}

return Results.Unauthorized();

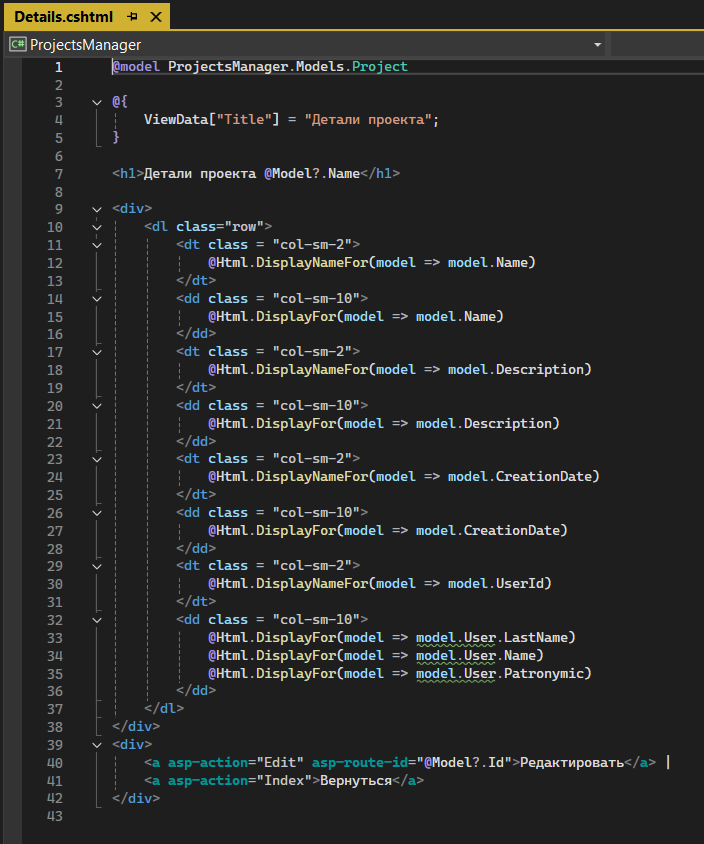
});

app.Run();

**Представление «Login.cshtml» контроллера «UsersController»**

**Представление «Create. cshtml» контроллера «UsersController»**

**Представление «Index. cshtml» контроллера «ProjectsController»** 

**Представление «Details. cshtml» контроллера «ProjectsController»**

**Методы контроллера «UsersController»**

// GET: Users/Create

public IActionResult Create()

{

List<SelectListItem> dropDownList = new List<SelectListItem>()

{

new SelectListItem { Text = "Admin", Value = "1" },

new SelectListItem { Text = "Manager", Value = "2" },

new SelectListItem { Text = "User", Value = "3" }

};

ViewData["Role"] = new SelectList(dropDownList, "Text", "Text", "User");

return View();

}

// POST: Users/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,LastName,Name,Patronymic,Phone,Login,Password,Role")] User user)

{

if (ModelState.IsValid)

{

if (user.Login != null && user.Password != null)

{

var count = \_context.Users.Where(u => u.Login == user.Login).Count();

if (count == 0)

{

try

{

user.Password = Crypto.Hash(user.Password.ToString(), "SHA-256");

\_context.Add(user);

await \_context.SaveChangesAsync();

\_toastNotification.Success("Вы успешно зарегистрированы!");

return Redirect($"~/Users/SignIn/{user.Id}");

}

catch (Exception ex)

{

\_toastNotification.Error("Ошибка регистрации!\n" + ex.Message);

}

}

else

{

\_toastNotification.Error("Аккаунт с таким логином уже существует!");

}

}

else

{

\_toastNotification.Error("Логин и пароль обязательны для заполнения!");

}

}

List<SelectListItem> dropDownList = new List<SelectListItem>()

{

new SelectListItem { Text = "Admin", Value = "1" },

new SelectListItem { Text = "Manager", Value = "2" },

new SelectListItem { Text = "User", Value = "3" }

};

ViewData["Role"] = new SelectList(dropDownList, "Text", "Text", user.Role);

return View(user);

}

// POST: Users/Login

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Login([Bind("Login,Password")] User model)

{

if (model.Login != null && model.Password != null)

{

var p = Crypto.Hash(model.Password, "SHA-256");

var user = \_context.Users.FirstOrDefaultAsync(u => u.Login == model.Login && u.Password == p);

if (user.Result != null && user.Result.Role != null)

{

int id = Convert.ToInt32(user.Result.Id);

model.Id = id;

\_toastNotification.Success("Вы успешно вошли в аккаунт!");

return Redirect($"~/Users/SignIn/{id}");

}

else

{

\_toastNotification.Error("Аккаунт не найден!");

return View(model);

}

}

else

{

\_toastNotification.Error("Введите логин и пароль!");

}

return View();

}

// POST: Users/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,LastName,Name,Patronymic,Phone,Login,Role")] User user)

{

if (id != user.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(user);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(user.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(user);

}

// POST: Users/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var user = await \_context.Users.FindAsync(id);

if (user != null)

{

\_context.Users.Remove(user);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

**Методы контроллера «UsersController»**

// GET: Projects/Create

public IActionResult Create()

{

var projectsManagerContext = \_context.Users;

var users = projectsManagerContext.ToList();

var dropDownList = users.Select(x=> new {

Id = x.Id,

Name = x.LastName.ToString() + " " + x.Name.ToString() + " " + x.Patronymic.ToString()

}).ToList();

ViewData["UserId"] = new SelectList(dropDownList,"Id","Name");

return View();

}

// POST: Projects/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Name,Description,UserId,CreationDate")] Project project)

{

if (ModelState.IsValid)

{

\_context.Add(project);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["UserId"] = new SelectList(\_context.Users, "Id", "Id", project.UserId);

return View(project);

}

// GET: Projects/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var project = await \_context.Projects.FindAsync(id);

if (project == null)

{

return NotFound();

}

var projectsManagerContext = \_context.Users;

var users = projectsManagerContext.ToList();

var dropDownList = users.Select(x => new

{

Id = x.Id,

Name = x.LastName.ToString() + " " + x.Name.ToString() + " " + x.Patronymic.ToString()

}).ToList();

ViewData["UserId"] = new SelectList(dropDownList, "Id", "Name", project.UserId);

return View(project);

}

// POST: Projects/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var project = await \_context.Projects.FindAsync(id);

if (project != null)

{

\_context.Projects.Remove(project);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}