ПРИЛОЖЕНИЕ

Листинг программы

Модуль Ziro.Web:

```
namespace Ziro.Web
{
  public class Startup
  {
		public Startup(IHostingEnvironment env)
		{
			var builder = new ConfigurationBuilder()
				.SetBasePath(env.ContentRootPath)
				.AddJsonFile("system-settings.json");

			Configuration = builder.Build();
		}

		public IConfiguration Configuration { get; }

		public void ConfigureServices(IServiceCollection services)
		{
			// add configuration
			services.AddOptions();
			services.Configure<SystemSettings>(Configuration);
			services.AddSingleton<ISystemSettings>(x =>
x.GetService<IOptions<SystemSettings>>().Value);

			//nhibernate
			services.AddSingleton<ISessionFactory>(x =>
			{
				var connectionString = x.GetService<ISystemSettings>().ConnectionString;
				var nhConfiguration = new Configuration().DataBaseIntegration(db =>
				{
					db.ConnectionString = connectionString;
					db.Dialect<MsSql2012Dialect>();
					db.Driver<Sql2008ClientDriver>();

				})
				.SetNamingStrategy(new ZiroNamingStrategy());
				var s = new ConventionModelMapper();
				var mapper = new ModelMapper();

  mapper.AddMappings(Assembly.GetAssembly(typeof(UserMap)).GetExportedTypes());
				var mapping = mapper.CompileMappingForAllExplicitlyAddedEntities();
				nhConfiguration.AddMapping(mapping);
				return nhConfiguration.BuildSessionFactory();
			});
			services.AddScoped<NHibernate.ISession>(x =>
x.GetService<ISessionFactory>().OpenSession());

			//authentication
			services.AddAuthentication("/Account/Login", "/Account/AccessDenied");

			//react
			services.AddJsEngineSwitcher(options => options.DefaultEngineName =
ChakraCoreJsEngine.EngineName)
				.AddChakraCore();
			services.AddReact();

			//mvc
			services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();
			services.AddMvc()
				.SetCompatibilityVersion(CompatibilityVersion.Version_2_1)
				.AddFluentValidation(fv =>
fv.RegisterValidatorsFromAssemblyContaining<LoginValidator>())
				.ConfigureApiBehaviorOptions(options =>
				{
					options.SuppressConsumesConstraintForFormFileParameters = true;
					options.SuppressInferBindingSourcesForParameters = true;
					options.SuppressModelStateInvalidFilter = true;
					options.SuppressMapClientErrors = true;

	options.SuppressUseValidationProblemDetailsForInvalidModelStateResponses = false;
				});
			#region Other
			// Services
			services.AddTransient<IAvatarService, AvatarService>();
			services.AddTransient<IUserService, UserService>();
			services.AddTransient<ITaskService, TaskService>();
			services.AddTransient<IProjectService, ProjectService>();
			// Repositories
			services.AddTransient<IAvatarRepository, AvatarRepository>();
			services.AddTransient<IUserRepository, UserRepository>();
			services.AddTransient<ITaskRepository, TaskRepository>();
			services.AddTransient<IProjectRepository, ProjectRepository>();
			services.AddTransient<IProjectViewRepository, ProjectViewRepository>();
			services.AddTransient<ICommentRepository, CommentRepository>();
			services.AddTransient<ILogWorkRepository, LogWorkRepository>();
			services.AddTransient<IProjectDocumentRepository, ProjectDocumentRepository>();
			#endregion
			services.AddSingleton<IResourceProvider, ResourceProvider>();
		}

		public void Configure(IApplicationBuilder app, IHostingEnvironment env)
		{
			app.UseStatusCodePages(async context =>
			{
				var statusCode = context.HttpContext.Response.StatusCode;

				if (context.HttpContext.IsApiRequest())
				{
					string responseContent = null;

					if (statusCode == (int)HttpStatusCode.NotFound)
						responseContent =
JsonConvert.SerializeObject(ErrorModel.CreateNotFound());
					else if (statusCode == (int)HttpStatusCode.Unauthorized)
						responseContent =
JsonConvert.SerializeObject(ErrorModel.CreateNotAuthenicated());
					else if (statusCode == (int)HttpStatusCode.Forbidden)
						responseContent =
JsonConvert.SerializeObject(ErrorModel.CreateForbidden());

					if (responseContent != null)
					{
						context.HttpContext.Response.Clear();
```

```
                                                        await
context.HttpContext.Response.WriteAsync(responseContent);
                                                    }
                                            }
                                    });

                                    app.UseExceptionHandling(@"/Error/InternalServerError");

                                    app.UseStaticFiles();
                                    app.UseReact(config =>
                                    {
                                            config
                                                    .SetReuseJavaScriptEngines(true)
                                                    .SetLoadBabel(false)
                                                    .SetLoadReact(false)
                                                    .AddScriptWithoutTransform("~/dist/bundle.js")
                                                    .AddScriptWithoutTransform("~/dist/vendor.bundle.js")
                                                    .AddScriptWithoutTransform("~/dist/main.bundle.js");
                                    });
                                    app.UseCookiePolicy();

                                    app.UseMiddleware<DataAccessMiddleware>();
                                    app.UseAuthentication();
                                    app.UseMvc(routes =>
                                    {
                                            routes.MapRoute("authorize",
                                                    "authorization",
                                                    new { controller = "Home", action = "Index" }
                                            );

                                            routes.MapRoute("default",
                                                    "{controller}/{action}/{id?}",
                                                    new { controller = "Home", action = "Index" }
                                            );
                                    });
                            }
                    }
            }


            namespace Ziro.Web.Controllers
            {
              [AllowAnonymous]
              public class AccountController : BaseController
              {
                      private readonly IUserService _userService;
                      private readonly IAuthenticationProvider _authenticationProvider;

                      public AccountController(IUserService userService, IAuthenticationProvider authenticationProvider)
                      {
                              _userService = userService;
                              _authenticationProvider = authenticationProvider;
                      }

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

                      [HttpPost]
                      public async Task<IActionResult> Login(LoginVm vm)
                      {
                        if (!ModelState.IsValid) return View(vm);

                        var user = _userService.GetUser(vm.Email, vm.Password);

                        if (user != null)
                        {
                                await _authenticationProvider.SignInAsync(HttpContext, vm.Email,
user.Role.ToString(), user.Id);

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

                        ModelState.AddModelError("", "Некорректные логин и(или) пароль");

                        return View();
                }

                public async Task<IActionResult> Logout()
                {
                        await _authenticationProvider.LogoutAsync(HttpContext);
                        return RedirectToAction("Login", "Account");
                }

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


        using Microsoft.AspNetCore.Authorization;
        using Microsoft.AspNetCore.Mvc;
        using System.Collections.Generic;
        using System.Linq;
        using Ziro.Core.Enums;

        namespace Ziro.Web.Controllers
        {
          [Authorize(Roles = nameof(Roles.User))]
          public class BaseController : Controller
          {

                public IList<string> ExtractErrorrs()
                {
                        if (ModelState.IsValid) return new List<string>();

                        var result = ModelState.Values.Select(x => string.Join(";", x.Errors.Select(e =>
e.ErrorMessage))).ToList();
                        return result;
                }

          }
        }

        namespace Ziro.Web.Controllers.api
        {
          public class ProjectController : BaseApiController
          {
                private readonly IProjectService _projectService;

                public ProjectController(IProjectService projectService, IAuthenticatedUserProvider authProvider) :
base(authProvider)
```

```
		{
			_projectService = projectService;
		}

		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult GetCurrentProjects()
		{
			var userId = CurrentUser.Id;
			var projects = _projectService.GetProjects(userId);
			var result = new GetCurrentProjectResponse
			{
				Projects = projects.Select(x=>x.ToProjectResponse()).ToList()
			};

			return SuccessResult(result);
		}

		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult GetCurrentProjectsInfos()
		{
			var userId = CurrentUser.Id;
			var projects = _projectService.GetProjectsInfos(userId);
			var result = new GetCurrentProjectsInfosResponse
			{
				Projects = projects.Select(x => x.ToProjectInfoResponse()).ToList()
			};

			return SuccessResult(result);
		}


		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult CreateDoc()
		{
			var res =
createDoc(@"E:\Education\zaochka\DP\dev\ziro\src\Ziro\Ziro.Web\wwwroot\dist\Facebook - структура.docx", new
Guid("15F09976-6A3C-4AF8-A9CC-D0921741CE87"));


			return File(res.Content, res.ContentType, res.FileName);
			//return SuccessResult();
		}

		private ProjectDocumentDTO createDoc(string path, Guid projectId)
		{
			var doc = System.IO.File.OpenRead(path);
			var contentType = doc;
			var name = Path.GetFileName(doc.Name);
			byte[] data = new byte[(int)doc.Length];
			var stream = doc.Read(data, 0, (int)doc.Length);
			var dto = new ProjectDocumentDTO {
				ContentType = "application/vnd.openxmlformats-
officedocument.wordprocessingml.document",
				Content = data,
				ProjectId = projectId,
				Description = "Общая структура системы и ее отдельных компонентов",
				FileName = name,
				UploadDate = DateTime.Now
			};
			_projectService.SaveProjectDocument(dto);
			return dto;
		}
```

```
}


namespace Ziro.Web.Controllers.api
{
  public class TaskController : BaseApiController
  {
		private readonly ITaskService _taskService;
		private readonly IResourceProvider _resourceProvider;

		public TaskController(ITaskService taskService, IResourceProvider resourceProvider,
IAuthenticatedUserProvider authenticatedUserProvider) : base(authenticatedUserProvider)
		{
			_resourceProvider = resourceProvider;
			_taskService = taskService;
		}

		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult GetAvailableStatuses()
		{
			var result = new Dictionary<int, string>
			{
				{(int)TaskStatuses.Open, _resourceProvider.GetLocalizedEnum(TaskStatuses.Open)
},
				{(int)TaskStatuses.InProgress,
_resourceProvider.GetLocalizedEnum(TaskStatuses.InProgress) },
				{(int)TaskStatuses.InTest,
_resourceProvider.GetLocalizedEnum(TaskStatuses.InTest) },
				{(int)TaskStatuses.Done, _resourceProvider.GetLocalizedEnum(TaskStatuses.Done)
},
				{(int)TaskStatuses.Review,
_resourceProvider.GetLocalizedEnum(TaskStatuses.Review) }
			};

			return SuccessResult(result);
		}

		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult GetAvailablePriorities()
		{
			var result = new Dictionary<int, string>
			{
				{(int)Priorities.Trivial, _resourceProvider.GetLocalizedEnum(Priorities.Trivial) },
				{(int)Priorities.Minor, _resourceProvider.GetLocalizedEnum(Priorities.Minor) },
				{(int)Priorities.Major, _resourceProvider.GetLocalizedEnum(Priorities.Major) },
				{(int)Priorities.Critical, _resourceProvider.GetLocalizedEnum(Priorities.Critical) },
				{(int)Priorities.Blocker, _resourceProvider.GetLocalizedEnum(Priorities.Blocker) }
			};

			return SuccessResult(result);
		}

		[Authorize(Roles = nameof(Roles.User))]
		public IActionResult GetAvailableTypes()
		{
			var result = new Dictionary<int, string>
			{
				{(int)TaskTypes.Task, _resourceProvider.GetLocalizedEnum(TaskTypes.Task) },
				{(int)TaskTypes.SubTask,
_resourceProvider.GetLocalizedEnum(TaskTypes.SubTask) },
                                {(int)TaskTypes.Feature, _resourceProvider.GetLocalizedEnum(TaskTypes.Feature)
},
                                {(int)TaskTypes.Bug, _resourceProvider.GetLocalizedEnum(TaskTypes.Bug) }
                        };

                        return SuccessResult(result);
                }

                [Authorize(Roles = nameof(Roles.User))]
                public IActionResult GetCurrentTasks()
                {
                        var userId = CurrentUser.Id;
                        var tasks = _taskService.GetShort(userId);
                        var response = new CurrentTasksResponse {
                                Tasks = tasks.Select(x => x.ToShortTask(_resourceProvider)).ToList()
                        };
                        return SuccessResult(response);
                }

                [Authorize(Roles = nameof(Roles.User))]
                [HttpPost]
                public IActionResult GetTaskDetails([FromBody]GetTaskDetailsRequest request)
                {
                        var taskId = request.TaskId;
                        var task = _taskService.GetDetails(taskId);
                        var response = task.ToTaskDetails(_resourceProvider);
                        response.Comments = new List<TaskDetailsComment>();
                        return SuccessResult(response);
                }

                [Authorize(Roles = nameof(Roles.User))]
                [HttpPost]
                public IActionResult GetTaskDetailsByNumber([FromBody]GetTaskDetailsRequestByNumber
request)
                {
                        var taskId = request.TaskNumber;
                        var task = _taskService.GetDetails(taskId);
                        var response = task.ToTaskDetails(_resourceProvider);

                        return SuccessResult(response);
                }

             [Authorize(Roles = nameof(Roles.User))]
             [HttpPost]
             public IActionResult AddComment([FromBody]AddCommentRequest request)
             {
                var userId = CurrentUser.Id;
                var taskId = request.TaskId;
                var commentText = request.Text;
                var savedComment = _taskService.AddComment(userId, taskId, commentText);
                        var response = savedComment.ToCommentResponse();

                        return SuccessResult(response);
             }

             [Authorize(Roles = nameof(Roles.User))]
             [HttpPost]
             public IActionResult AddLogWork([FromBody]AddLogWorkRequest request)
             {
                var userId = CurrentUser.Id;
                var taskId = request.TaskId;
                var savedLogWork = _taskService.AddLogWork(userId,
             taskId,
            request.Text,
            request.SpentHours);
                        var response = savedLogWork.ToLogWorkResponse();

                        return SuccessResult(response);

    }
}
```

Модуль Ziro.Business:

```
namespace Ziro.Business.Services
{
  public class ProjectService : IProjectService
  {
            private readonly IProjectRepository _projectRepository;
            private readonly IProjectDocumentRepository _projectDocumentRepository;
            private readonly IProjectViewRepository _projectViewRepository;

            public ProjectService(IProjectRepository projectRepository,
                        IProjectViewRepository projectViewRepository,
                        IProjectDocumentRepository projectDocumentRepository)
            {
                        _projectRepository = projectRepository;
                        _projectViewRepository = projectViewRepository;
                        _projectDocumentRepository = projectDocumentRepository;
            }

            public IList<ProjectViewDTO> GetProjects(Guid userId)
            {
                        var result = _projectViewRepository.Get(userId).ToList();
                        return result;
            }

            public IList<ProjectInfoDTO> GetProjectsInfos(Guid userId)
            {
                        var userProjectsIds = _projectRepository.GetIds(userId).ToArray();
                        var projeInfos = _projectRepository.GetProjectInfos(userProjectsIds).ToList();

                        return projeInfos;
            }

            public void SaveProjectDocument(ProjectDocumentDTO document)
            {
                        var project = _projectRepository.Get(document.ProjectId);
                        _projectDocumentRepository.Save(document, project);
            }
  }
}
using System;
using Ziro.Core.Business.Services;
using Ziro.Core.DataAccess.Repositories;
using Ziro.Core.DTO;
using Ziro.Core.Mappers;
using System.Linq;
using System.Collections.Generic;

namespace Ziro.Business.Services
{
  public class UserService: IUserService
  {
	private readonly IUserRepository _userRepository;
	private readonly IProjectRepository _projectRepository;

	public UserService(IUserRepository userRepository, IProjectRepository projectRepository)
	{
		_userRepository = userRepository;
		_projectRepository = projectRepository;
	}

	public UserDTO GetUser(Guid id)
	{
		var result = _userRepository.Get(id);
		return result?.ToDTO();
	}

	public UserDTO GetUser(string email, string password)
	{
		var result = _userRepository.Get(email, password);
		return result?.ToDTO();
	}

	public IList<UserInfoDTO> GetTeamMembersInfos(Guid userId)
	{
		var userProjectsIds = _projectRepository.GetIds(userId).ToArray();
		var result = _userRepository.GetColleguasInfos(userId, userProjectsIds).ToList();
		return result;
	}
  }
}
```

Модуль Ziro.Persistence:

```
using NHibernate.Mapping.ByCode;
using NHibernate.Mapping.ByCode.Conformist;
using Ziro.Domain.Entities;

namespace Ziro.Persistence.Mappings
{
  public class UserMap : BaseEntityMap<User>
  {
	public UserMap()
	{
		Id(x => x.Id, m => m.Generator(Generators.Guid));

		Property(x => x.Email, m =>
		{
			m.Length(50);
			m.NotNullable(notnull: true);
		});

		Property(x => x.PasswordHash, m =>
		{
			m.Length(255);
			m.NotNullable(notnull: true);
		});
		Property(x => x.Role, m => m.NotNullable(notnull: true));
		Property(x => x.Name, m =>
		{
			m.Length(250);
			m.NotNullable(notnull: false);
```

```
                    });
                    Property(x => x.LastName, m =>
                    {
                              m.Length(250);
                              m.NotNullable(notnull: false);
                    });
                    Property(x => x.Skype, m =>
                    {
                              m.Length(150);
                              m.NotNullable(notnull: false);
                    });
                    Property(x => x.PhoneNumber, m =>
                    {
                              m.Length(20);
                              m.NotNullable(notnull: false);
                    });
                    Property(x => x.DateOfBirth, m => { m.NotNullable(notnull: false); });
                    ManyToOne(x => x.Position, c => {
                              c.Cascade(Cascade.None);
                              c.Column(FKColumnName(nameof(Position)));
                              c.NotNullable(notnull: false);
                    });
                    Set(x => x.Avatars,
                              c => {
                                        c.Key(k => k.Column(FKColumnName(nameof(User))));
                                        c.Inverse(true);
                              },
                              r => r.OneToMany());
                    Set(a => a.Projects,
                    c => {
                              c.Cascade(Cascade.Persist);
                              c.Key(k => k.Column(FKColumnName(nameof(User))));
                              c.Table(MToMTableName(nameof(Project), nameof(User)));
                    },
                    r => r.ManyToMany(m => m.Column(FKColumnName(nameof(Project)))));
          }
  }
}

using NHibernate;
using System;
using System.Linq;
using System.Collections.Generic;
using Ziro.Core.DataAccess.Repositories;
using Ziro.Domain.Entities;
using NHibernate.Transform;
using NHibernate.Criterion;
using Ziro.Core.DTO;

namespace Ziro.Persistence.Repositories
{
  public class ProjectRepository : IProjectRepository
  {
          private readonly ISession _session;

          public ProjectRepository(ISession session)
          {
                    _session = session;
          }

          public Project Get(Guid id)
          {
```

```
			var query = _session.QueryOver<Project>().Where(x => x.Id == id);

			var result = query.SingleOrDefault();
			return result;
		}

		public IEnumerable<Project> GetAll(Guid userId)
		{
			User userAlias = null;
			var query = _session.QueryOver<Project>()
				.JoinAlias(x => x.Users, () => userAlias)
				.Where(x => userAlias.Id == userId);

			var result = query.List();
			return result;
		}

		public IEnumerable<Guid> GetIds(Guid userId)
		{
			User userAlias = null;
			var query = _session.QueryOver<Project>()
				.JoinAlias(x => x.Users, () => userAlias)
				.Where(x => userAlias.Id == userId)
				.Select(x => x.Id);

			var result = query.List<Guid>();
			return result;
		}

		public IEnumerable<ProjectInfoDTO> GetProjectInfos(Guid[] projectIds)
		{
			if (projectIds == null || projectIds.Length == 0) return new List<ProjectInfoDTO>();

			var query = _session.QueryOver<ProjectInfoView>()
				.Where(x => x.ProjectId.IsIn(projectIds));

			query = mapOnProjectInfoDTO(query);
			var result = query.List<ProjectInfoDTO>();
			return result;
		}

		private IQueryOver<ProjectInfoView, ProjectInfoView>
mapOnProjectInfoDTO(IQueryOver<ProjectInfoView, ProjectInfoView> query)
		{
			ProjectInfoDTO resultDTO = null;

			return query.SelectList(list => list
				.Select(x => x.ProjectId).WithAlias(() => resultDTO.ProjectId)
				.Select(x => x.ProjectName).WithAlias(() => resultDTO.ProjectName)
				.Select(x => x.ProjectShortName).WithAlias(() => resultDTO.ProjectShortName)
				.Select(x => x.ProjectDescription).WithAlias(() => resultDTO.ProjectDescription)
				.Select(x => x.NonClosedTasksCount).WithAlias(() =>
resultDTO.NonClosedTasksCount)
				.Select(x => x.TotalUsersCount).WithAlias(() => resultDTO.TotalUsersCount)
			)
			.TransformUsing(Transformers.AliasToBean<ProjectInfoDTO>());
		}
	}
}

using NHibernate;
using NHibernate.Transform;
using System;
using System.Collections.Generic;
using Ziro.Core.DataAccess.Repositories;
using Ziro.Core.DTO;
using Ziro.Domain.Entities;

namespace Ziro.Persistence.Repositories
{
  public class ProjectViewRepository : IProjectViewRepository
  {
          private readonly ISession _session;

          public ProjectViewRepository(ISession session)
          {
                  _session = session;
          }

          public IEnumerable<ProjectViewDTO> Get(Guid userId)
          {
                  var query = _session.QueryOver<ProjectView>()
                          .Where(x => x.UserId == userId);
                  query = mapOnDTO(query);
                  var result = query.List<ProjectViewDTO>();
                  return result;
          }

          private IQueryOver<ProjectView, ProjectView> mapOnDTO(IQueryOver<ProjectView,
ProjectView> query)
          {
                  ProjectViewDTO resultDTO = null;

                  return query.SelectList(list => list
                          .Select(x => x.UserId).WithAlias(() => resultDTO.UserId)
                          .Select(x => x.ProjectId).WithAlias(() => resultDTO.ProjectId)
                          .Select(x => x.ProjectName).WithAlias(() => resultDTO.ProjectName)
                          .Select(x => x.ProjectShortName).WithAlias(() => resultDTO.ProjectShortName)
                          .Select(x => x.ProjectDescription).WithAlias(() => resultDTO.ProjectDescription)
                          .Select(x => x.TasksInProgressCount).WithAlias(() =>
resultDTO.TasksInProgressCount)
                          .Select(x => x.OpenTasksCount).WithAlias(() => resultDTO.OpenTasksCount)
                          .Select(x => x.TotalTasksCount).WithAlias(() => resultDTO.TotalTasksCount)
                  )
                  .TransformUsing(Transformers.AliasToBean<ProjectViewDTO>());
          }
    }
}

using NHibernate.Cfg;

namespace Ziro.Persistence
{
  public class ZiroNamingStrategy : INamingStrategy
  {
          private const string TableNameMToMTemplate = "Ziro_{0}_{1}";
          private const string TableNameTemplate = "Ziro_{0}";
          private const string ForeignKeyColumnTemplate = "{0}Id";

          private readonly INamingStrategy _defaultStrategy = DefaultNamingStrategy.Instance;

          public string ClassToTableName(string className)
          {
                  var defaultTableName = _defaultStrategy.ClassToTableName(className);
```

```
            var tableName = string.Format(TableNameTemplate, defaultTableName);
            return tableName;
        }

        public string ColumnName(string columnName)
        {
            return _defaultStrategy.ClassToTableName(columnName);
        }

        public string LogicalColumnName(string columnName, string propertyName)
        {
            return _defaultStrategy.LogicalColumnName(columnName, propertyName);
        }

        public string PropertyToColumnName(string propertyName)
        {
            return _defaultStrategy.PropertyToColumnName(propertyName);
        }

        public string PropertyToTableName(string className, string propertyName)
        {
            return _defaultStrategy.PropertyToTableName(className, propertyName);
        }

        public string TableName(string tableName)
        {
            return _defaultStrategy.TableName(tableName);
        }

        public string TableNameMToM(string entityFirst, string entitySecond)
        {
            return string.Format(TableNameMToMTemplate, entityFirst, entitySecond);
        }

        public string ForeignKeyColumn(string propertyName)
        {
            return string.Format(ForeignKeyColumnTemplate, propertyName);
        }
  }
}
```

Модуль Ziro.Core:

```
namespace Ziro.Core.DTO
{
  public class UserDTO
  {
        public Guid Id { get; set; }
        public string Email { get; set; }
        public Roles Role { get; set; }
  }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace Ziro.Core.DTO
{
  public class ProjectDocumentDTO
  {
```

```
            public virtual Guid Id { get; set; }
            public virtual string FileName { get; set; }
            public virtual string Description { get; set; }
            public virtual string ContentType { get; set; }
            public virtual byte[] Content { get; set; }
            public virtual DateTime UploadDate { get; set; }
            public virtual Guid ProjectId { get; set; }
  }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace Ziro.Core.DTO
{
  public class ProjectDTO
  {
            public Guid Id { get; set; }
            public string Name { get; set; }
            public string ShortName { get; set; }
            public string Description { get; set; }
  }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace Ziro.Core.DTO
{
  public class TaskDetailsDTO
  {
            public Guid Id { get; set; }
            public int Number { get; set; }
            public byte Type { get; set; }
            public byte Status { get; set; }
            public string Title { get; set; }
            public string Description { get; set; }
            public byte Priority { get; set; }
            public double EstimatedTime { get; set; }
            public double SpentTime { get; set; }
            public DateTime CreationDate { get; set; }
            public virtual DateTime LastUpdateDate { get; set; }
            public Guid ProjectId { get; set; }
            public string ProjectName { get; set; }
            public string ShortProjectName { get; set; }
            public Guid AssigneeId { get; set; }
            public string AssigneeName { get; set; }
            public string AssigneeLastName { get; set; }
            public Guid OwnerId { get; set; }
            public string OwnerName { get; set; }
            public string OwnerLastName { get; set; }
            public IList<LogWorkDTO> LogWorks { get; set; }
            public IList<CommentDTO> Comments { get; set; }
            public string FullNumber => string.Format(Consts.TaskNumberTemplate, this.ShortProjectName, 
this.Number);
  }
}
namespace Ziro.Core.Business.Services
{
  public interface IProjectService
  {
            IList<ProjectViewDTO> GetProjects(Guid userId);
            IList<ProjectInfoDTO> GetProjectsInfos(Guid userId);
            void SaveProjectDocument(ProjectDocumentDTO document);
  }
}

namespace Ziro.Core.Business.Services
{
  public interface ITaskService
  {
            IList<ShortTaskDTO> GetShort(Guid userId);
            TaskDetailsDTO GetDetails(Guid id);
            TaskDetailsDTO GetDetails(string taskNumber);
      CommentDTO AddComment(Guid userId, Guid taskId, string commentText);
            LogWorkDTO AddLogWork(Guid userId, Guid taskId, string text, double spentHours);
   }
}
namespace Ziro.Core.Business.Services
{
  public interface IUserService
  {
            UserDTO GetUser(Guid id);
            UserDTO GetUser(string email, string password);
            UserProfileDTO GetUserProfile(Guid id);
            IList<UserInfoDTO> GetTeamMembersInfos(Guid userId);
  }
}

namespace Ziro.Core.DataAccess.Repositories
{
  public interface ICommentRepository
  {
            IEnumerable<CommentDTO> GetAll(Guid taskId);
            Comment Save(User user, Task task, string commentText);

   }
}

namespace Ziro.Core.DataAccess.Repositories
{
  public interface ITaskRepository
  {
            Task Get(Guid Id);
            IEnumerable<ShortTaskDTO> GetShort(Guid userId);
            TaskDetailsDTO GetDetails(Guid id);
            TaskDetailsDTO GetDetails(int number, string projectShortName);
  }
}

using System;
namespace Ziro.Core.DataAccess.Repositories
{
  public interface IUserRepository
  {
            UserProfileDTO GetProfile(Guid id);
            User Get(Guid Id);
            User Get(string email, string password);
            IEnumerable<User> GetUsers();
            IEnumerable<UserInfoDTO> GetColleguasInfos(Guid userId, Guid[] userProjectsIds);
  }
```

```
}
```

Модуль Ziro.Domain:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Ziro.Domain.Entities
{
  public class Avatar
  {
            public virtual Guid Id { get; set; }
            public virtual string ContentType { get; set; }
            public virtual byte[] ImageData { get; set; }
            public virtual User User{ get; set; }
  }
}

namespace Ziro.Domain.Entities
{
  public class User
  {
            public virtual Guid Id { get; set; }
            public virtual string Email { get; set; }
            public virtual string PasswordHash { get; set; }
            public virtual byte Role { get; set; }
            public virtual string Name { get;set;}
            public virtual string LastName { get; set; }
            public virtual string Skype { get; set; }
            public virtual string PhoneNumber { get; set; }
            public virtual DateTime? DateOfBirth { get; set; }
            public virtual Position Position{ get; set; }
            public virtual ISet<Avatar> Avatars { get; set; }
            public virtual ISet<Project> Projects { get; set; }
  }
}

namespace Ziro.Domain.Entities
{
  public class Task
  {
            public virtual Guid Id { get; set; }
            public virtual int Number { get; set; }
            public virtual byte Type { get; set; }
            public virtual byte Status { get; set; }
            public virtual string Title { get; set; }
            public virtual string Description { get; set; }
            public virtual byte Priority { get; set; }
            public virtual double EstimatedTime { get; set; }
            public virtual double SpentTime { get; set; }
            public virtual DateTime CreationDate { get; set; }
            public virtual DateTime LastUpdateDate { get; set; }
            public virtual Project Project { get; set; }
            public virtual User Assignee { get; set; }
            public virtual User Owner { get; set; }

  }
}
```

```
namespace Ziro.Domain.Entities
{
  public class Position
  {
            public virtual Guid Id { get; set; }
            public virtual string Name { get; set; }
            public virtual ISet<User> Users { get; set; }
  }
}
```