*Код программы*

```
namespace Backend_DiplomProject
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) {
}

        public DbSet<Role> Roles { get; set; }
        public DbSet<Username> Usernames { get; set; }
        public DbSet<MonetizationCourse> MonetizationCourses { get; set; }
        public DbSet<LevelKnowledge> LevelKnowledges { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<AgePeople> AgePeoples { get; set; }
        public DbSet<Course> Courses { get; set; }
        public DbSet<Pages> Pages { get; set; }
        public DbSet<FavoritesAndHistory> FavoritesAndHistories { get; set; }
        public DbSet<Pay> Pays { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Username>()
                .HasOne(u => u.Role)
                .WithMany()
                .HasForeignKey(u => u.Idrole)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Course>()
                .HasOne(c => c.User)
                .WithMany()
                .HasForeignKey(c => c.Idusername)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Course>()
                .HasOne(c => c.Monetization)
                .WithMany()
                .HasForeignKey(c => c.Idmonetizationcourse)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Course>()
                .HasOne(c => c.Level)
                .WithMany()
                .HasForeignKey(c => c.Idlevelknowledge)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Course>()
                .HasOne(c => c.Category)
                .WithMany()
                .HasForeignKey(c => c.Idcategory)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Course>()
                .HasOne(c => c.Age)
                .WithMany()
                .HasForeignKey(c => c.Idagepeople)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Pages>()
                .HasOne(p => p.Course)
                .WithMany(c => c.Pages)
                .HasForeignKey(p => p.Idcourse)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<FavoritesAndHistory>()
                .HasOne(f => f.Course)
                .WithMany()
```

```
                .HasForeignKey(f => f.Idcourse)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<FavoritesAndHistory>()
                .HasOne(f => f.User)
                .WithMany()
                .HasForeignKey(f => f.Idusername)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Pay>()
                .HasOne(p => p.Course)
                .WithMany()
                .HasForeignKey(p => p.Idcourse)
                .OnDelete(DeleteBehavior.Cascade);

            modelBuilder.Entity<Pay>()
                .HasOne(p => p.User)
                .WithMany()
                .HasForeignKey(p => p.Idusername)
                .OnDelete(DeleteBehavior.Cascade);
        }
    }
}


namespace Backend_DiplomProject
{
    public class Program
    {
        public static async Task Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);

            Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("#########################################################");
            Console.WriteLine("#   Запуск API ASP.NET веб-платформы Knowledge+
#");

Console.WriteLine("#########################################################");
            Console.ResetColor();

            // Конфигурация сервисов
            builder.Services.AddControllers();
            ConfigureSwagger(builder.Services);
            ConfigureCors(builder.Services);
            ConfigureDatabase(builder, builder.Configuration);

            var app = builder.Build();

            // Настройка middleware
            app.UseHttpsRedirection();
            app.UseRouting();
            app.UseCors("ViteReactPolicy");
            app.UseAuthorization();
            app.MapControllers();

            // Применение миграций и проверка БД
            await ApplyMigrationsAndCheckDb(app);

            // Запуск Swagger только в Development
            if (app.Environment.IsDevelopment())
            {
                app.UseSwagger();
                app.UseSwaggerUI(c =>
```

```
            {
                c.SwaggerEndpoint("/swagger/v1/swagger.json", "Diplom API
v1");
                c.ConfigObject.DisplayRequestDuration = true;
            });
        }

        app.Run();

        // Настройка Swagger
        void ConfigureSwagger(IServiceCollection services)
        {
            services.AddEndpointsApiExplorer();
            services.AddSwaggerGen(c =>
            {
                c.SwaggerDoc("v1", new OpenApiInfo
                {
                    Title = "Diplom API",
                    Version = "v1",
                    Description = "API для дипломного проекта"
                });

                // Фильтрация по namespace для Swagger-контроллеров
                c.DocInclusionPredicate((docName, apiDesc) =>
                {
                    var controllerActionDescriptor = apiDesc.ActionDescriptor
as Microsoft.AspNetCore.Mvc.Controllers.ControllerActionDescriptor;
                    if (controllerActionDescriptor == null) return false;

                    var controllerNamespace =
controllerActionDescriptor.ControllerTypeInfo.Namespace;


                    return controllerNamespace != null &&

controllerNamespace.StartsWith("Backend_DiplomProject.Controllers.Swagger");
                });
            });
        }

        // Настройка CORS
        void ConfigureCors(IServiceCollection services)
        {
            services.AddCors(options =>
            {
                options.AddPolicy("ViteReactPolicy", policy =>
                {
                    policy.WithOrigins("http://localhost:5173")
                          .AllowAnyHeader()
                          .AllowAnyMethod()
                          .AllowCredentials();
                });
            });
        }

        // Подключение и настройка базы данных
        void ConfigureDatabase(WebApplicationBuilder builder, IConfiguration
configuration)
        {
            builder.Services.AddDbContext<AppDbContext>(options =>

options.UseNpgsql(configuration.GetConnectionString("PostgreSQLConnection"))
                    .LogTo(Console.WriteLine)
                    .EnableSensitiveDataLogging());
```

```
            }

            // Метод для применения миграций и проверки БД
            async Task ApplyMigrationsAndCheckDb(WebApplication app)
            {
                using var scope = app.Services.CreateScope();
                var dbContext =
scope.ServiceProvider.GetRequiredService<AppDbContext>();

                try
                {
                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.WriteLine("\nПроверка подключения к БД...\n");
                    Console.ResetColor();

                    await dbContext.Database.OpenConnectionAsync();
                    await dbContext.Database.CloseConnectionAsync();

                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.WriteLine("\nПодключение к БД успешно
установлено\n");
                    Console.ResetColor();

                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.WriteLine("Применение миграций...\n");
                    Console.ResetColor();

                    await dbContext.Database.MigrateAsync();

                    Console.ForegroundColor = ConsoleColor.Green;
                    Console.WriteLine("\nМиграции успешно применены\n");
                    Console.ResetColor();
                }
                catch (Exception ex)
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.WriteLine($"\nОшибка подключения к БД:
{ex.Message}");
                    Console.WriteLine("Приложение продолжит работу, но
функциональность БД недоступна\n");
                    Console.ResetColor();
                }

                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("Успешный запуск API ASP.NET веб-платформы
Knowledge+\n");
                Console.ResetColor();

                Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("#######################################################");
                Console.WriteLine("#   Журнал API ASP.NET веб-платформы
Knowledge+   #");

Console.WriteLine("#######################################################");
                Console.WriteLine("");
                Console.ResetColor();
            }
        }
    }
}
```

```
namespace Backend_DiplomProject.Controllers
{
    [Route("api/ping")]
    [ApiController]
    public class PingController : ControllerBase
    {
        [HttpGet]
        public IActionResult Ping()
        {
            return Ok(new { status = "ok" });
        }
    }
}



namespace Backend_DiplomProject.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CoursesListController : ControllerBase
    {
        private readonly AppDbContext _context;

        public CoursesListController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        public async Task<IActionResult> GetFilteredCourses(
            [FromQuery] string? searchQuery,
            [FromQuery] int? selectedCategory,
            [FromQuery] int? selectedAge,
            [FromQuery] int? selectedLevel,
            [FromQuery] int? selectedMonetization,
            [FromQuery] int? priceMin,
            [FromQuery] int? priceMax
        )
        {
            var query = _context.Courses
                .Include(c => c.Monetization)
                .Include(c => c.Level)
                .Include(c => c.Category)
                .Include(c => c.Age)
                .AsQueryable();

            if (!string.IsNullOrWhiteSpace(searchQuery))
            {
                var trimmed = searchQuery.Trim();
                query = query.Where(c => c.Title.Contains(trimmed));
            }

            if (selectedCategory.HasValue)
            {
                query = query.Where(c => c.Idcategory == selectedCategory.Value);
            }

            if (selectedAge.HasValue)
            {
                query = query.Where(c => c.Idagepeople == selectedAge.Value);
            }

            if (selectedLevel.HasValue)
```

```
            {
                query = query.Where(c => c.Idlevelknowledge ==
selectedLevel.Value);
            }

            if (selectedMonetization.HasValue)
            {
                query = query.Where(c => c.Idmonetizationcourse ==
selectedMonetization.Value);
            }

            if (selectedMonetization == 2 && priceMin.HasValue &&
priceMax.HasValue)
            {
                query = query.Where(c =>
                    c.Price.HasValue &&
                    c.Price.Value >= priceMin.Value &&
                    c.Price.Value <= priceMax.Value
                );
            }

            var result = await query
                .Select(c => new CourseListDto
                {
                    Idcourse = c.Idcourse,
                    Title = c.Title,
                    Description = c.Description,

                    Idmonetizationcourse = c.Idmonetizationcourse,
                    MonetizationType = c.Monetization.type,

                    Price = c.Price,

                    Level = c.Level.Type,
                    Category = c.Category.Type,
                    Age = c.Age.Type,

                    IconBase64 = c.Icon != null
                        ? "" +
Convert.ToBase64String(c.Icon)
                        : null
                })
                .ToListAsync();

            return Ok(new
            {
                success = true,
                courses = result
            });
        }
    }
}



namespace Backend_DiplomProject.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CoursesControllerEditList : ControllerBase
    {
        private readonly AppDbContext _dbContext;
        private readonly AppDbContext _context;
```

```
        public CoursesControllerEditList(AppDbContext dbContext, AppDbContext
context)
        {
            _dbContext = dbContext;
            _context = context;
        }

        /// <summary>
        /// Загрузка списка "Редактор курсов"
        /// </summary>
        /// <param name="userId"></param>
        /// <param name="search"></param>
        /// <returns></returns>
        [HttpGet("user/{userId}")]
        public async Task<IActionResult> GetCoursesByUser(long userId,
[FromQuery] string? search = null)
        {
            bool userExists = await _dbContext.Usernames
                .AsNoTracking()
                .AnyAsync(u => u.Idusername == userId);

            if (!userExists)
            {
                return NotFound(new { message = $"Пользователь с id={userId} не
найден." });
            }

            var query = _dbContext.Courses
                .AsNoTracking()
                .Where(c => c.Idusername == userId);

            if (!string.IsNullOrWhiteSpace(search))
            {
                string lowered = search.Trim().ToLower();
                query = query.Where(c => c.Title.ToLower().Contains(lowered));
            }

            var result = await query
                .OrderByDescending(c => c.Dateadd)
                .Select(c => new CourseDto
                {
                    IdCourse = c.Idcourse,
                    Title = c.Title,
                    DateAdd = c.Dateadd
                })
                .ToListAsync();

            return Ok(result);
        }

        /// <summary>
        /// Удаление личного курса из "Редактор курсов"
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteCourse(long id)
        {
            var course = await _context.Courses.FindAsync(id);
            if (course == null)
                return NotFound();

            _context.Courses.Remove(course);
            await _context.SaveChangesAsync();
```

```
            return NoContent();
        }
    }
}



namespace Backend_DiplomProject.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class CoursesControllerCreateAndEdit : ControllerBase
    {
        private readonly AppDbContext _context;

        public CoursesControllerCreateAndEdit(AppDbContext context)
        {
            _context = context;
        }

        /// <summary>
        /// Создание курса
        /// </summary>
        /// <param name="dto"></param>
        /// <returns></returns>
        [HttpPost]
        public async Task<IActionResult> CreateCourse([FromForm] CourseFormDto
dto)
        {
            if (!ModelState.IsValid)
                return BadRequest(ModelState);

            byte[] iconBytes = null;
            if (dto.Icon != null)
            {
                using var ms = new System.IO.MemoryStream();
                await dto.Icon.CopyToAsync(ms);
                iconBytes = ms.ToArray();
            }

            // Десериализуем JSON-строку pages
            List<PageDto> pagesList;
            try
            {
                pagesList =
JsonSerializer.Deserialize<List<PageDto>>(dto.PagesJson, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
            }
            catch
            {
                return BadRequest(new { error = "Невалидный JSON в поле pages."
});
            }

            if (pagesList == null || pagesList.Count == 0)
                return BadRequest(new { error = "Должна быть хотя бы одна
страница." });

            var courseEntity = new Course
            {
                Title = dto.Title,
                Description = dto.Description,
                Icon = iconBytes,
                Dateadd = DateTime.UtcNow.Date,
```

```
                Idusername = dto.Idusername,
                Idmonetizationcourse = dto.Idmonetizationcourse,

                Price = (dto.Idmonetizationcourse == 2 && dto.Price.HasValue)
                            ? dto.Price.Value
                            : null,

                Idlevelknowledge = dto.Idlevelknowledge,
                Idcategory = dto.Idcategory,
                Idagepeople = dto.Idagepeople
            };

            await _context.Courses.AddAsync(courseEntity);
            await _context.SaveChangesAsync();

            foreach (var p in pagesList)
            {
                if (string.IsNullOrWhiteSpace(p.Content))
                    return BadRequest(new { error = $"Содержимое страницы
order={p.Order} не может быть пустым." });

                var pageEntity = new Pages
                {
                    Numberpage = p.Order,
                    File = Encoding.UTF8.GetBytes(p.Content),
                    Idcourse = courseEntity.Idcourse
                };
                await _context.Pages.AddAsync(pageEntity);
            }
            await _context.SaveChangesAsync();

            if (dto.Idmonetizationcourse == 2)
            {
                bool existsPay = await _context.Pays
                    .AnyAsync(x => x.Idcourse == courseEntity.Idcourse &&
x.Idusername == dto.Idusername);
                if (!existsPay)
                {
                    var payEntity = new Pay
                    {
                        Idcourse = courseEntity.Idcourse,
                        Idusername = dto.Idusername
                    };
                    await _context.Pays.AddAsync(payEntity);
                    await _context.SaveChangesAsync();
                }
            }

            return Ok(new
            {
                message = "Курс успешно создан",
                idcourse = courseEntity.Idcourse
            });
        }

        /// <summary>
        /// Редактирование курса
        /// </summary>
        /// <param name="id"></param>
        /// <param name="dto"></param>
        /// <returns></returns>
        [HttpPut("{id:long}")]
        public async Task<IActionResult> UpdateCourse(long id, [FromForm]
CourseFormDto dto)
```

```
        {
            if (dto.Icon == null || dto.Icon.Length == 0)
            {
                return BadRequest(new { error = "Файл Icon не передан или
пустой." });
            }

            if (!ModelState.IsValid)
                return BadRequest(ModelState);

            var courseEntity = await _context.Courses
                .Include(c => c.Pages)
                .FirstOrDefaultAsync(c => c.Idcourse == id);
            if (courseEntity == null)
                return NotFound(new { error = $"Курс с id={id} не найден." });

            courseEntity.Title = dto.Title;
            courseEntity.Description = dto.Description;

            courseEntity.Dateadd = DateTime.UtcNow.Date;

            if (dto.Icon != null)
            {
                using var ms = new System.IO.MemoryStream();
                await dto.Icon.CopyToAsync(ms);
                courseEntity.Icon = ms.ToArray();
            }

            courseEntity.Idmonetizationcourse = dto.Idmonetizationcourse;

            courseEntity.Price = (dto.Idmonetizationcourse == 2 &&
dto.Price.HasValue)
                                    ? dto.Price.Value
                                    : null;

            courseEntity.Idlevelknowledge = dto.Idlevelknowledge;
            courseEntity.Idcategory = dto.Idcategory;
            courseEntity.Idagepeople = dto.Idagepeople;

            _context.Courses.Update(courseEntity);
            await _context.SaveChangesAsync();

            _context.Pages.RemoveRange(courseEntity.Pages);
            await _context.SaveChangesAsync();

            // Сохраняем новые страницы из JSON
            List<PageDto> pagesList;
            try
            {
                pagesList =
JsonSerializer.Deserialize<List<PageDto>>(dto.PagesJson, new
JsonSerializerOptions { PropertyNameCaseInsensitive = true });
            }
            catch
            {
                return BadRequest(new { error = "Невалидный JSON в поле pages."
});
            }

            foreach (var p in pagesList)
            {
                if (string.IsNullOrWhiteSpace(p.Content))
                    return BadRequest(new { error = $"Содержимое страницы
order={p.Order} не может быть пустым." });
```

```
                var pageEntity = new Pages
                {
                    Numberpage = p.Order,
                    File = Encoding.UTF8.GetBytes(p.Content),
                    Idcourse = courseEntity.Idcourse
                };
                await _context.Pages.AddAsync(pageEntity);
            }
            await _context.SaveChangesAsync();

            var existingPay = await _context.Pays
                .FirstOrDefaultAsync(x => x.Idcourse == courseEntity.Idcourse &&
x.Idusername == dto.Idusername);

            if (dto.Idmonetizationcourse == 2)
            {
                if (existingPay == null)
                {
                    var payEntity = new Pay
                    {
                        Idcourse = courseEntity.Idcourse,
                        Idusername = dto.Idusername
                    };
                    await _context.Pays.AddAsync(payEntity);
                    await _context.SaveChangesAsync();
                }
            }
            else
            {
                if (existingPay != null)
                {
                    _context.Pays.Remove(existingPay);
                    await _context.SaveChangesAsync();
                }
            }

            return Ok(new { message = "Курс успешно обновлён", idcourse =
courseEntity.Idcourse });
        }

        /// <summary>
        /// Загрузка для редактирования курса
        /// </summary>
        /// <param name="id"></param>
        /// <returns></returns>
        [HttpGet("{id}")]
        public async Task<IActionResult> GetCourse(long id)
        {
            var courseEntity = await _context.Courses
                .AsNoTracking()
                .Include(c => c.Pages)
                .FirstOrDefaultAsync(c => c.Idcourse == id);

            if (courseEntity == null)
                return NotFound(new { message = $"Курс с id={id} не найден." });

            // Конвертация иконки (byte[]) в base64 data URI (если есть)
            string? iconUri = null;
            if (courseEntity.Icon != null && courseEntity.Icon.Length > 0)
            {
                iconUri =
$"{Convert.ToBase64String(courseEntity.Icon)}";
            }
```

```
            var dto = new CourseDetailDto
            {
                IdCourse = courseEntity.Idcourse,
                Title = courseEntity.Title,
                Description = courseEntity.Description,
                MonetizationType = courseEntity.Idmonetizationcourse,
                Price = courseEntity.Price,
                Category = courseEntity.Idcategory,
                AgeRestriction = courseEntity.Idagepeople,
                Level = courseEntity.Idlevelknowledge,
                IconBase64 = iconUri,
                Pages = courseEntity.Pages
                    .OrderBy(p => p.Numberpage)
                    .Select(p => new PageDetailDto
                    {
                        Order = p.Numberpage,
                        Content = Encoding.UTF8.GetString(p.File)
                    })
                    .ToList()
            };

            return Ok(dto);
        }
    }
}



namespace Backend_DiplomProject.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class CategoryController : Controller
    {
        private readonly AppDbContext _context;

        public CategoryController(AppDbContext context)
        {
            _context = context;
        }

        [HttpGet("monetization-types")]
        public async Task<IActionResult> GetMonetizationTypes()
        {
            var monetizationTypes = await _context.MonetizationCourses
                .Select(m => new { m.idmonetizationcourse, m.type })
                .ToListAsync();

            return Ok(new { success = true, monetizationTypes });
        }

        [HttpGet("levels")]
        public async Task<IActionResult> GetLevels()
        {
            var levels = await _context.LevelKnowledges
                .Select(l => new { l.Idlevelknowledge, l.Type })
                .ToListAsync();

            return Ok(new { success = true, levels });
        }

        [HttpGet("categories")]
        public async Task<IActionResult> GetCategories()
        {
```

```
            var categories = await _context.Categories
                .Select(c => new { c.Idcategory, c.Type })
                .ToListAsync();

            return Ok(new { success = true, categories });
        }

        [HttpGet("age-restrictions")]
        public async Task<IActionResult> GetAgeRestrictions()
        {
            var ageRestrictions = await _context.AgePeoples
                .Select(a => new { a.Idagepeople, a.Type })
                .ToListAsync();

            return Ok(new { success = true, ageRestrictions });
        }
    }
}


namespace Backend_DiplomProject.Controllers;

[Route("api/[controller]")]
[ApiController]
public class AuthController : ControllerBase
{
    private readonly AppDbContext _context;
    private readonly ILogger<AuthController> _logger;

    public AuthController(
        AppDbContext context,
        ILogger<AuthController> logger)
    {
        _context = context;
        _logger = logger;
    }

    [HttpPost("login")]
    public async Task<ActionResult<object>> Login(LoginDto dto)
    {
        try
        {
            _logger.LogInformation($"Login attempt for user: {dto.Login}");

            var user = await _context.Usernames
                .Include(u => u.Role)
                .FirstOrDefaultAsync(u => u.Login == dto.Login);

            if (user == null)
            {
                _logger.LogWarning($"User not found: {dto.Login}");
                return Unauthorized(new { message = "Invalid credentials" });
            }

            if (!VerifyPassword(dto.Password, user.Password))
            {
                _logger.LogWarning($"Password mismatch for user: {dto.Login}");
                return Unauthorized(new { message = "Invalid credentials" });
            }

            _logger.LogInformation($"Login successful for user: {dto.Login}");
            return Ok(new
            {
                idusername = user.Idusername,
```

```
            login = user.Login,
            idrole = user.Idrole
        });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error during login");
        return StatusCode(500, new { message = "Internal server error" });
    }
}

/// <summary>
/// Преобразовывает пароль из хеша
/// </summary>
/// <param name="inputPassword"></param>
/// <param name="storedHash"></param>
/// <returns></returns>
private bool VerifyPassword(string inputPassword, string storedHash)
{
    using var sha256 = SHA256.Create();
    var inputHash = Convert.ToHexString(
        sha256.ComputeHash(Encoding.UTF8.GetBytes(inputPassword)));

    return inputHash.Equals(storedHash, StringComparison.OrdinalIgnoreCase);
}


[HttpPost("register")]
public async Task<ActionResult<object>> Register(RegisterDto dto)
{
    if (await _context.Usernames.AnyAsync(u => u.Login == dto.Login))
        return BadRequest(new { message = "Username already exists" });

    var newUser = new Username
    {
        Login = dto.Login,
        Password = HashPassword(dto.Password),
        Dateaddaccount = DateTime.UtcNow.Date,
        Idrole = 2
    };

    _context.Usernames.Add(newUser);
    await _context.SaveChangesAsync();

    return Ok(new
    {
        idusername = newUser.Idusername,
        login = newUser.Login,
        idrole = newUser.Idrole
    });
}

/// <summary>
/// Преобразовывает пароль в хеш
/// </summary>
/// <param name="password"></param>
/// <returns></returns>
private string HashPassword(string password)
{
    using var sha256 = SHA256.Create();
    return Convert.ToHexString(
        sha256.ComputeHash(Encoding.UTF8.GetBytes(password)));
}
```

```
    [HttpPost("logout")]
    public IActionResult Logout()
    {
        try
        {
            foreach (var cookie in Request.Cookies.Keys)
            {
                Response.Cookies.Delete(cookie);
            }

            return Ok(new { message = "Logged out successfully" });
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Logout failed");
            return StatusCode(500, new { message = "Logout failed" });
        }
    }

    [HttpPut("change-password")]
    public async Task<IActionResult> ChangePassword(ChangePasswordDto dto)
    {
        try
        {
            _logger.LogInformation($"Password change request for user:
{dto.UserId}");

            var user = await _context.Usernames.FindAsync(dto.UserId);
            if (user == null)
            {
                _logger.LogWarning($"User not found: {dto.UserId}");
                return NotFound(new { message = "User not found" });
            }

            user.Password = HashPassword(dto.NewPassword);
            await _context.SaveChangesAsync();

            _logger.LogInformation($"Password changed successfully for user:
{dto.UserId}");
            return Ok(new { message = "Password changed successfully" });
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, "Password change failed");
            return StatusCode(500, new { message = "Internal server error" });
        }
    }

    /// <summary>
    /// Проверка верификации
    /// </summary>
    /// <param name="dto"></param>
    /// <returns></returns>
    [HttpPost("verify-password")]
    public async Task<IActionResult> VerifyPassword([FromBody] VerifyPasswordDto
dto)
    {
        // Попытаемся найти пользователя по Idusername
        var user = await _context.Usernames
            .AsNoTracking()
            .FirstOrDefaultAsync(u => u.Idusername == dto.Idusername);

        if (user == null)
        {
```

```
            // Если вдруг пользователь не найден — считаем, что неверные данные
            return NotFound(new { success = false, message = "User not found" });
        }

        // Проверим хеш пароля
        bool passwordMatches = false;
        using (var sha256 = SHA256.Create())
        {
            var inputHash = Convert.ToHexString(
                sha256.ComputeHash(Encoding.UTF8.GetBytes(dto.Password))
            );
            passwordMatches = inputHash.Equals(user.Password,
StringComparison.OrdinalIgnoreCase);
        }

        return Ok(new { success = passwordMatches });
    }

}

function MainRouter() {

  return null;

}


function App() {


  const [isServerAvailable, setIsServerAvailable] = useState<boolean | null>(null);

  {

    useEffect(() => {

      const checkServer = async () => {

        try {

          const response = await api.get('/ping'); // Пинг-запрос

          if (response.status === 200 && response.data.status === 'ok') {

            setIsServerAvailable(true);

          } else {

            setIsServerAvailable(false);

          }

        } catch(error) {

          setIsServerAvailable(false);

          console.error('Ошибка подключения к серверу', error)

        }

      };


      checkServer();

      const interval = setInterval(checkServer, 5000); // Повторяем каждые 5 сек

      return () => clearInterval(interval);
```

```
  }, []);

  if (isServerAvailable === null || isServerAvailable === false) {
    return <LoadingScreen />;
  }
};

return (
    <Router>
      <AuthProvider>
        <ToastProvider>
          <Layout>
            <Routes>
              <Route path="/" element={<HomePage />} />
              <Route path="/login" element={<LoginPage />} />
              <Route path="/register" element={<RegisterPage />} />
              <Route path="/courses" element={<CourseListPage />} />
              <Route path="/courses/:id" element={<CourseViewPage />} />

              {/* Protected routes */}
              <Route path="/profile" element={
                <ProtectedRoute>
                  <ProfilePage />
                </ProtectedRoute>
              } />
              <Route path="/courses/editor" element={
                <ProtectedRoute>
                  <CourseEditorPage />
                </ProtectedRoute>
              } />
              <Route path="/courses/editor/:id" element={
                <ProtectedRoute>
                  <CourseFormPage />
                </ProtectedRoute>
              } />
              <Route path="/courses/history" element={
                <ProtectedRoute>

                  <CourseHistoryPage />
```

```
                </ProtectedRoute>
              } />
            </Routes>
          </Layout>
        </ToastProvider>
      </AuthProvider>
    </Router>
  );
}

export default App;

import axios from 'axios';
// Создание api
const api = axios.create({
  baseURL: '/api',
  timeout: 10000
});

// Вывод ошибок от API
api.interceptors.response.use(
    response => response,
    error => {
        // Логирование деталей ошибки
        console.error('API ошибка:', {
            message: error.message,
            status: error.response?.status,
            data: error.response?.data,
           url: error.config.url
        });

        // Обработка ошибки 401
        if (error.response?.status === 401) {
            localStorage.removeItem('token');
            localStorage.removeItem('user');
            window.location.href = '/login';
        }
        return Promise.reject(error);
```

```
    }
);

// Перехватчик для добавления токена к запросам
api.interceptors.request.use(config => {
    const token = localStorage.getItem('token');
    if (token) {
        config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
});

// API методы
export const getMonetizationTypes = () =>
    api.get('/category/monetization-types').then(res => res.data);
export const getLevelKnowledgeTypes = () =>
    api.get('/category/levels').then(res => res.data);
export const getCategories = () =>
    api.get('/category/categories').then(res => res.data);

export const getAgeGroups = () =>
    api.get('/category/age-restrictions').then(res => res.data);
export { api };

import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': '/src', // Алиас для удобных импортов
    },
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:5226',
```

```
        changeOrigin: true,



        secure: false,
        rewrite: (path) => path.replace(/^\/api/, '/api'),
        headers: {
          Connection: 'Keep-Alive'
        }
      },
    },
    port: 5173,
    open: false,
  },
  build: {
    outDir: 'dist',
    sourcemap: true,
  },
  optimizeDeps: {
    exclude: ['lucide-react'],
    include: ['react-router-dom'],
  }
});

import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  resolve: {
    alias: {
      '@': '/src', // Алиас для удобных импортов
    },
  },
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:5226',
        changeOrigin: true,
```

```
        secure: false,
        rewrite: (path) => path.replace(/^\/api/, '/api'),
        headers: {

          Connection: 'Keep-Alive'

        }
      },
    },
    port: 5173,
    open: false,
  },
  build: {
    outDir: 'dist',
    sourcemap: true,
  },
  optimizeDeps: {
    exclude: ['lucide-react'],
    include: ['react-router-dom'],
  }
});

import { useEffect } from 'react';
import { Link } from 'react-router-dom';
import { BookOpen, FileText, Users, DollarSign } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';

const HomePage = () => {

  const { user } = useAuth();

  return (
    <div className="bg-gray-50">
      {/* Основная секция */}
      <section className="bg-gradient-to-r from-gray-900 to-gray-800 text-white">
        <div className="container mx-auto px-4 py-16 md:py-24">
          <div className="max-w-4xl mx-auto text-center">
```

```
            <span className="inline-block px-3 py-1 text-sm bg-orange-600 rounded-full mb-
4">
              Универсальная платформа образования
            </span>
            <h1 className="text-3xl md:text-5xl font-bold mb-6">
              Создавайте и изучайте курсы на <span className="text-orange-
500">Knowledge+</span>


            </h1>
            <p className="text-lg md:text-xl text-gray-300 mb-8">


              Наша платформа объединяет экспертов и учащихся из различных областей.
              Создавайте, делитесь и изучайте материалы в удобном формате.
            </p>
            <div className="flex flex-wrap justify-center gap-4">
              <Link
                to="/courses"
                className="px-6 py-3 bg-orange-600 hover:bg-orange-700 rounded-md font-
medium transition-colors"
              >
                Начать обучение
              </Link>
              {user && (
                <Link
                  to="/courses/editor"
                  className="px-6 py-3 bg-white text-gray-900 hover:bg-gray-100 rounded-md
font-medium transition-colors"
                >
                  Создать курс
                </Link>
              )}
            </div>
          </div>
        </div>
      </section>


      {/* Блочная секция */}
      <section className="py-16">
        <div className="container mx-auto px-4">
```

```
          <h2 className="text-3xl font-bold text-center mb-12">Преимущества платформы</h2>
          <p className="text-center text-gray-600 max-w-3xl mx-auto mb-16">
            Knowledge+ предоставляет широкие возможности для создания и изучения
            образовательных материалов
          </p>

          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-8">
            {/* Блок 1 */}

            <div className="bg-white p-6 rounded-lg shadow-md transition-transform
duration-300 hover:translate-y-[-5px]">
              <div className="bg-orange-100 w-12 h-12 rounded-lg flex items-center justify-
center mb-4">

                <BookOpen className="text-orange-600" size={24} />
              </div>
              <h3 className="text-xl font-semibold mb-3">Богатый формат</h3>
              <p className="text-gray-600">
                Создавайте материалы с использованием Markdown, добавляйте изображения и
видео
              </p>
            </div>

            {/* Блок 2 */}
            <div className="bg-white p-6 rounded-lg shadow-md transition-transform
duration-300 hover:translate-y-[-5px]">
              <div className="bg-blue-100 w-12 h-12 rounded-lg flex items-center justify-
center mb-4">
                <FileText className="text-blue-600" size={24} />
              </div>
              <h3 className="text-xl font-semibold mb-3">Современный редактор</h3>
              <p className="text-gray-600">
                Удобный интерфейс для создания контента с предпросмотром в реальном времени
              </p>
            </div>

            {/* Блок 3 */}
            <div className="bg-white p-6 rounded-lg shadow-md transition-transform
duration-300 hover:translate-y-[-5px]">
              <div className="bg-green-100 w-12 h-12 rounded-lg flex items-center justify-
center mb-4">
```

```
                <Users className="text-green-600" size={24} />
              </div>
              <h3 className="text-xl font-semibold mb-3">Учитесь в своем темпе</h3>
              <p className="text-gray-600">
                Доступ к материалам 24/7, изучайте в удобное для вас время
              </p>
            </div>

            {/* Блок 4 */}

            <div className="bg-white p-6 rounded-lg shadow-md transition-transform
duration-300 hover:translate-y-[-5px]">
              <div className="bg-purple-100 w-12 h-12 rounded-lg flex items-center justify-
center mb-4">
                <DollarSign className="text-purple-600" size={24} />
              </div>

              <h3 className="text-xl font-semibold mb-3">Монетизация</h3>
              <p className="text-gray-600">
                Создавайте бесплатные или платные курсы и делитесь своими знаниями
              </p>
            </div>
          </div>
        </div>
      </section>

      {/* Решительный вопрос */}
      <section className="bg-orange-600 text-white py-16">
        <div className="container mx-auto px-4 text-center">
          <h2 className="text-3xl font-bold mb-6">Готовы начать обучение?</h2>
          <p className="text-xl mb-8 max-w-2xl mx-auto">
            Присоединяйтесь к нашему сообществу и получите доступ к качественным
образовательным материалам
          </p>
          <div className="flex flex-wrap justify-center gap-4">
            <Link
              to="/courses"
              className="px-6 py-3 bg-white text-orange-600 hover:bg-gray-100 rounded-md
font-medium transition-colors"
```

```
            >
              Смотреть курсы
            </Link>
            {!user && (
              <Link
                to="/register"
                className="px-6 py-3 border-2 border-white hover:bg-orange-700 rounded-md
font-medium transition-colors"
              >
                Зарегистрироваться
              </Link>
            )}

          </div>
        </div>
      </section>

      {/* Секция популярных курсов */}
      <section className="py-16">

        <div className="container mx-auto px-4">
          <h2 className="text-3xl font-bold text-center mb-12">Популярные курсы</h2>

          <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
            {/* Курс 1 */}
            <div className="bg-white rounded-lg shadow-md overflow-hidden transition-
transform duration-300 hover:translate-y-[-5px]">
              <div className="h-48 bg-gray-200 flex items-center justify-center">
                <BookOpen size={48} className="text-gray-400" />
              </div>
              <div className="p-6">
                <div className="flex justify-between items-center mb-2">
                  <span className="px-2 py-1 bg-green-100 text-green-800 text-xs rounded-
full">
                    Бесплатно
                  </span>
                  <span className="text-sm text-gray-500">Начинающий</span>
                </div>
                <h3 className="text-xl font-semibold mb-2">Введение в учебную платформу
Knowledge+</h3>
```

```
                <p className="text-gray-600 mb-4">
                  Описание платформы Knowledge+
                </p>
                <Link
                  to="/courses/1"
                  className="block text-center w-full py-2 bg-orange-600 text-white
rounded-md hover:bg-orange-700 transition-colors"
                >
                  Открыть курс
                </Link>
              </div>
            </div>

            {/* Курс 2 */}

            <div className="bg-white rounded-lg shadow-md overflow-hidden transition-
transform duration-300 hover:translate-y-[-5px]">
              <div className="h-48 bg-gray-200 flex items-center justify-center">
                <BookOpen size={48} className="text-gray-400" />
              </div>
              <div className="p-6">
                <div className="flex justify-between items-center mb-2">

                  <span className="px-2 py-1 bg-green-100 text-green-800 text-xs rounded-
full">
                    Бесплатно
                  </span>
                  <span className="text-sm text-gray-500">Средний</span>
                </div>
                <h3 className="text-xl font-semibold mb-2">Просмотр курсов</h3>
                <p className="text-gray-600 mb-4">
                  Основы просмотра курсов на платформе Knowledge+
                </p>
                <Link
                  to="/courses/2"
                  className="block text-center w-full py-2 bg-orange-600 text-white
rounded-md hover:bg-orange-700 transition-colors"
                >
                  Открыть курс
```

```
                </Link>
              </div>
            </div>

            {/* Курс 3 */}
            <div className="bg-white rounded-lg shadow-md overflow-hidden transition-
transform duration-300 hover:translate-y-[-5px]">
              <div className="h-48 bg-gray-200 flex items-center justify-center">
                <BookOpen size={48} className="text-gray-400" />
              </div>
              <div className="p-6">
                <div className="flex justify-between items-center mb-2">
                  <span className="px-2 py-1 bg-green-100 text-green-800 text-xs rounded-
full">
                    Бесплатно
                  </span>

                  <span className="text-sm text-gray-500">Продвинутый</span>
                </div>
                <h3 className="text-xl font-semibold mb-2">Создание курсов</h3>
                <p className="text-gray-600 mb-4">
                  Основы создания курсов на платформе Knowledge+
                </p>
                <Link
                  to="/login"
                  className="block text-center w-full py-2 bg-orange-600 text-white
rounded-md hover:bg-orange-700 transition-colors"
                >
                  Открыть курс
                </Link>
              </div>
            </div>
          </div>

          <div className="text-center mt-12">
            <Link
              to="/courses"
              className="inline-flex items-center text-orange-600 hover:text-orange-700
font-medium"
```

```
            >
              Смотреть все курсы
              <svg className="ml-2 w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0
24 24" xmlns="http://www.w3.org/2000/svg">
                <path strokeLinecap="round" strokeLinejoin="round" strokeWidth="2" d="M14
5l7 7m0 0l-7 7m7-7H3"></path>
              </svg>
            </Link>
          </div>
        </div>
      </section>
    </div>
  );
};

export default HomePage;

const Footer = () => {

  return (
    <footer className="bg-gray-900 text-gray-300 mt-12">
      <div className="container mx-auto px-4 py-8">
        <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
          <div>
            <Link to="/" className="flex items-center text-xl font-bold mb-4">
              <BookOpen className="mr-2" size={24} />
              <span className="text-white">Knowledge</span>

              <span className="text-orange-500">+</span>
            </Link>
            <p className="text-sm">
              Универсальная платформа образования, объединяющая экспертов и учащихся
              из различных областей
            </p>
          </div>

          <div></div>

          <div>
```

```
            <h3 className="text-lg font-semibold mb-4">Контакты</h3>
            <ul className="space-y-2">
              <li className="flex items-center">
                <Mail size={16} className="mr-2" />
                <a href="mailto:nikoir@nikitabuyanovr.ru" className="hover:text-orange-400
transition-colors">
                  nikoir@nikitabuyanovr.ru
                </a>
              </li>
              <li className="flex items-center">
                <Phone size={16} className="mr-2" />
                <a href="tel:+79145054052" className="hover:text-orange-400 transition-
colors">
                  +7 (914) 505-4052
                </a>
              </li>
            </ul>
          </div>
        </div>



        <div className="border-t border-gray-800 mt-8 pt-6 text-sm text-center">
          <p>© {new Date().getFullYear()} Knowledge+. Все права защищены.</p>
          <p>Дипломный проект Буянова Никиты Родионовича ИСиП-22-4к</p>
        </div>
      </div>
    </footer>
  );
};

export default Footer;

import { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { BookOpen, Search, LogIn, UserCircle, LogOut, Menu, X } from 'lucide-react';
import { useAuth } from '../../contexts/AuthContext';

const Header = () => {
  const { user, logout } = useAuth();
```

```
const navigate = useNavigate();
const [isMenuOpen, setIsMenuOpen] = useState(false);
const [showLogoutConfirm, setShowLogoutConfirm] = useState(false);

const handleLogout = () => {
  setShowLogoutConfirm(true);
};

const confirmLogout = () => {
  logout();
  setShowLogoutConfirm(false);
  navigate('/');
};

const cancelLogout = () => {
  setShowLogoutConfirm(false);
};

const toggleMenu = () => {
  setIsMenuOpen(!isMenuOpen);
};



return (
  <header className="bg-gray-900 text-white sticky top-0 z-50">
    <div className="container mx-auto px-4 py-3 flex justify-between items-center">
      <div className="flex items-center">
        <Link to="/" className="flex items-center text-xl font-bold">
          <BookOpen className="mr-2" size={24} />
          <span className="text-white">Knowledge</span>
          <span className="text-orange-500">+</span>
        </Link>
      </div>

      {/* Desktop Navigation */}
      <nav className="hidden md:flex items-center space-x-6">
        {user ? (
          <>
```

```
              <Link to="/courses" className="hover:text-orange-400 transition-colors">
                Каталог курсов
              </Link>
              <Link to="/courses/editor" className="hover:text-orange-400 transition-
colors">
                Редактор курсов
              </Link>
              <Link to="/courses/history" className="hover:text-orange-400 transition-
colors">
                История курсов
              </Link>
            </>
          ) : null}
        </nav>
        <div className="hidden md:flex items-center space-x-3">
          {user ? (
              <div className="relative group flex items-center gap-2">
                <button
                    className="flex items-center bg-gray-800 rounded-full px-3 py-1
hover:bg-gray-700 transition-colors"
                    onClick={() => navigate('/profile')}
                >
                  <UserCircle className="mr-2" size={20} />
                  <span className="truncate max-w-[100px]">{user.login}</span>

                </button>

                <button
                    onClick={handleLogout}
                    className="flex items-center justify-center p-2 rounded-full bg-gray-
800 hover:bg-gray-700 transition-colors"
                >
                  <LogOut size={18} />
                </button>
              </div>
          ) : (
            <div className="flex space-x-2">
              <Link
                to="/login"
```

```
                className="px-4 py-1 border border-gray-600 rounded-md hover:bg-gray-800
transition-colors"
              >
                Войти
              </Link>
              <Link
                to="/register"
                className="px-4 py-1 bg-orange-600 rounded-md hover:bg-orange-700
transition-colors"
              >
                Регистрация
              </Link>
            </div>
          )}
        </div>

        {/* Mobile menu button */}
        <button
          className="md:hidden flex items-center"
          onClick={toggleMenu}
        >
          {isMenuOpen ? <X size={24} /> : <Menu size={24} />}
        </button>
      </div>

      {/* Mobile Navigation */}

      {isMenuOpen && (
        <div className="md:hidden bg-gray-800 p-4">
          <nav className="flex flex-col space-y-3">
            <Link
              to="/courses"
              className="hover:text-orange-400 transition-colors py-2"
              onClick={() => setIsMenuOpen(false)}
            >
              Курсы
            </Link>
            {user ? (
              <>
```

```
                <Link
                  to="/courses/editor"
                  className="hover:text-orange-400 transition-colors py-2"
                  onClick={() => setIsMenuOpen(false)}
                >
                  Редактор курсов
                </Link>
                <Link
                  to="/courses/history"
                  className="hover:text-orange-400 transition-colors py-2"
                  onClick={() => setIsMenuOpen(false)}
                >
                  История курсов
                </Link>
                <Link
                  to="/profile"
                  className="hover:text-orange-400 transition-colors py-2"
                  onClick={() => setIsMenuOpen(false)}
                >
                  Профиль
                </Link>
                <button
                  onClick={handleLogout}
                  className="text-left hover:text-orange-400 transition-colors py-2"
                >
                  Выйти
                </button>
              </>
            ) : (
              <div className="flex flex-col space-y-2 pt-2">
                <Link
                  to="/login"
                  className="px-4 py-2 text-center border border-gray-600 rounded-md
hover:bg-gray-700 transition-colors"
                  onClick={() => setIsMenuOpen(false)}
                >
                  Войти
                </Link>
```

```
                <Link
                  to="/register"
                  className="px-4 py-2 text-center bg-orange-600 rounded-md hover:bg-
orange-700 transition-colors"
                  onClick={() => setIsMenuOpen(false)}
                >
                  Регистрация
                </Link>
              </div>
            )}
          </nav>
        </div>
      )}


      {/* Logout Confirmation Modal */}
      {showLogoutConfirm && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-
center z-50">
          <div className="bg-white rounded-lg p-6 max-w-sm mx-4">
            <h3 className="text-lg font-medium text-gray-900 mb-4">Выход из профиля</h3>
            <p className="text-gray-600 mb-6">Хотите ли вы выйти из профиля?</p>
            <div className="flex justify-end space-x-3">
              <button
                onClick={cancelLogout}
                className="px-4 py-2 border border-gray-300 rounded-md text-gray-700
hover:bg-gray-100 transition-colors"
              >
                Отмена
              </button>

              <button
                onClick={confirmLogout}
                className="px-4 py-2 bg-orange-600 text-white rounded-md hover:bg-orange-
700 transition-colors"
              >
                Выйти
              </button>
            </div>
          </div>
```

```
        </div>
      )}
    </header>
  );
};

export default Header;

interface LayoutProps {
  children: ReactNode;
}

const Layout = ({ children }: LayoutProps) => {
  return (
    <div className="flex flex-col min-h-screen">
      <Header />
      <main className="flex-grow">
        {children}
      </main>
      <Footer />
      <Toast />
    </div>
  );
};

export default Layout;
```