ДОДАТОК А

Графічний матеріал

магістерської

кваліфікаційної роботи

Міністерство освіти і науки України
Харківський Національний Університет Радіоелектроніки

«ЗАТВЕРДЖУЮ»
керівник магістерської
кваліфікаційної роботи
Мінухін С. В.

Розробка та дослідження компонентів системи управління документообігом з цифровими документами

Графічний матеріал

ЛИСТ ЗАТВЕРДЖЕННЯ

УЗГОДЖЕНО:

РОЗРОБИВ:
ст. гр. СПРм-20-1
Стаднік М.М.

2021 р

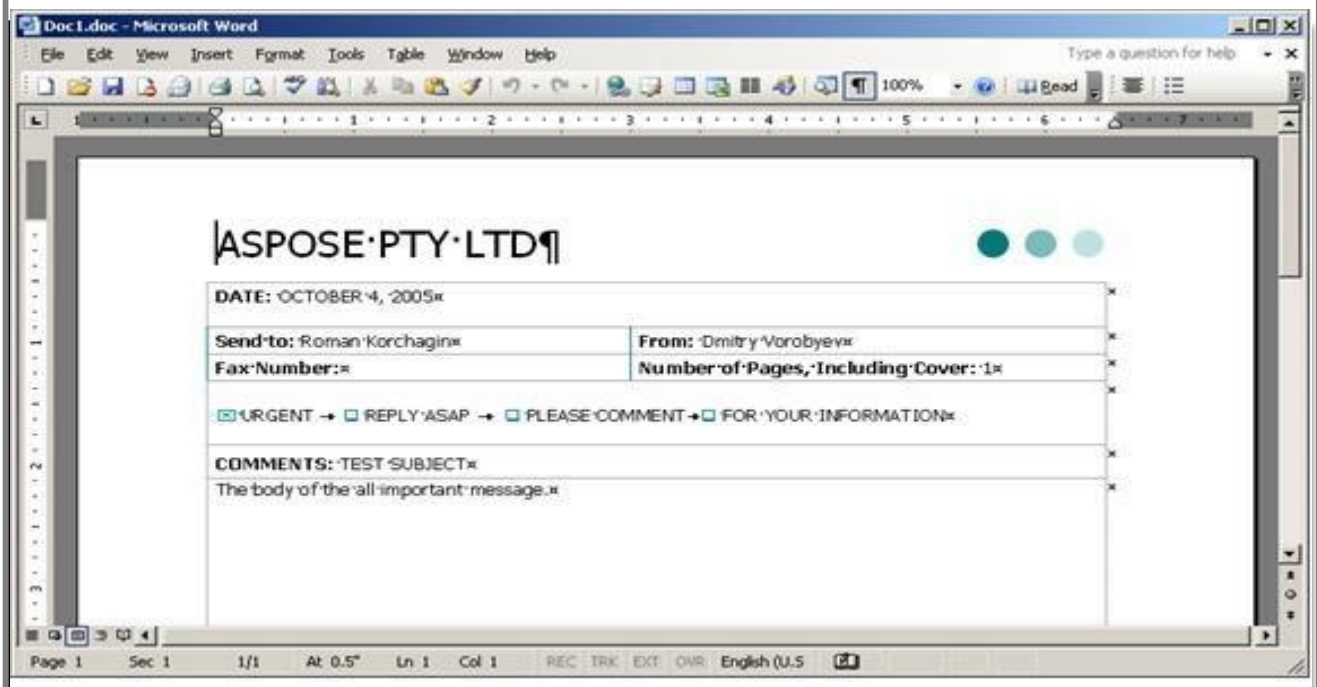
Розробка та дослідження компонентів системи управління документообігом з цифровими документами

Графічний матеріал

АРКУШІВ 6

2021 р

Результат роботи алгоритму генерування файлу



Розроб.	*Стаднік М.М.*			*Результат роботи алгоритму генерації файлу із засобами Aspose.Words*	
Перевір.	*Мінухін С. В.*				
Н. Контр.	*Мінухін С. В.*				
				СПРм-20-1	*Лист 1*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 6*

Результат роботи алгоритму парсингу файлу

- dbo.__EFMigrationsHistory
- dbo.ApiResourceClaims
- dbo.ApiResourceProperties
- dbo.ApiResources
- dbo.ApiResourceScopes
- dbo.ApiResourceSecrets
- dbo.ApiScopeClaims
- dbo.ApiScopeProperties
- dbo.ApiScopes
- dbo.AspNetRoleClaims
- dbo.AspNetRoles
- dbo.AspNetUserClaims
- dbo.AspNetUserLogins
- dbo.AspNetUserRoles
- dbo.AspNetUsers
- dbo.AspNetUserTokens
- dbo.ClientClaims
- dbo.ClientCorsOrigins
- dbo.ClientGrantTypes
- dbo.ClientIdPRestrictions
- dbo.ClientPostLogoutRedirectUris
- dbo.ClientProperties
- dbo.ClientRedirectUris
- dbo.Clients
- dbo.ClientScopes
- dbo.ClientSecrets
- dbo.DeviceCodes
- dbo.Groups
- dbo.IdentityResourceClaims

Результат створення токену із засобами IdentityServer

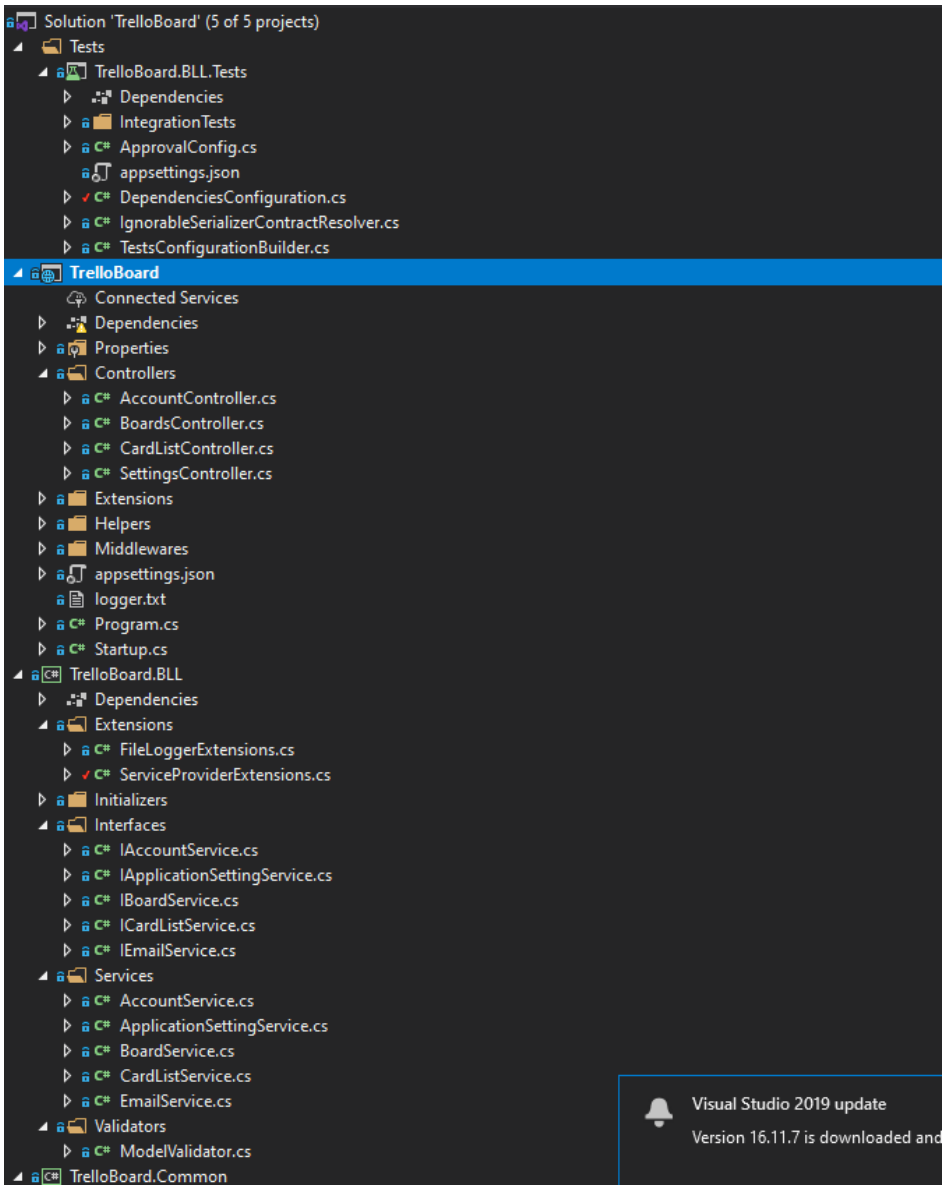
```
1 reference | 0 exceptions
private async Task<object> GenerateJwtTokenAsync(User user)
{
    var roles = await _userManager.GetRolesAsync(user);

    var claims = new List<Claim>
        {
            new Claim(JwtRegisteredClaimNames.Sub, user.Email),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            new Claim(JwtRegisteredClaimNames.Sid, user.Id.ToString()),
            new Claim(JwtRegisteredClaimNames.Typ, string.Join(", ", roles))
        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["JwtKey"]));
        var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
        var expires = DateTime.Now.AddDays(Convert.ToDouble(_configuration["JwtExpireDays"]));

        var token = new JwtSecurityToken(
            _configuration["JwtIssuer"],
            _configuration["JwtIssuer"],
            claims,
            expires: expires,
            signingCredentials: creds
        );

        return new { AccessToken = new JwtSecurityTokenHandler().WriteToken(token) };
}
```

Розроб.	*Стаднік М.М.*			*Результат створення токену із засобами IdentityServer*	
Перевір.	*Мінухін С. В.*				
Н. Контр.	*Мінухін С. В..*				
				СІПРм-20-1	*Лист 3*
Затверд.	*Гребеннік І.В.*			*СТ*	*Листів 6*

Ініціалізація сервісів із конфігурацією

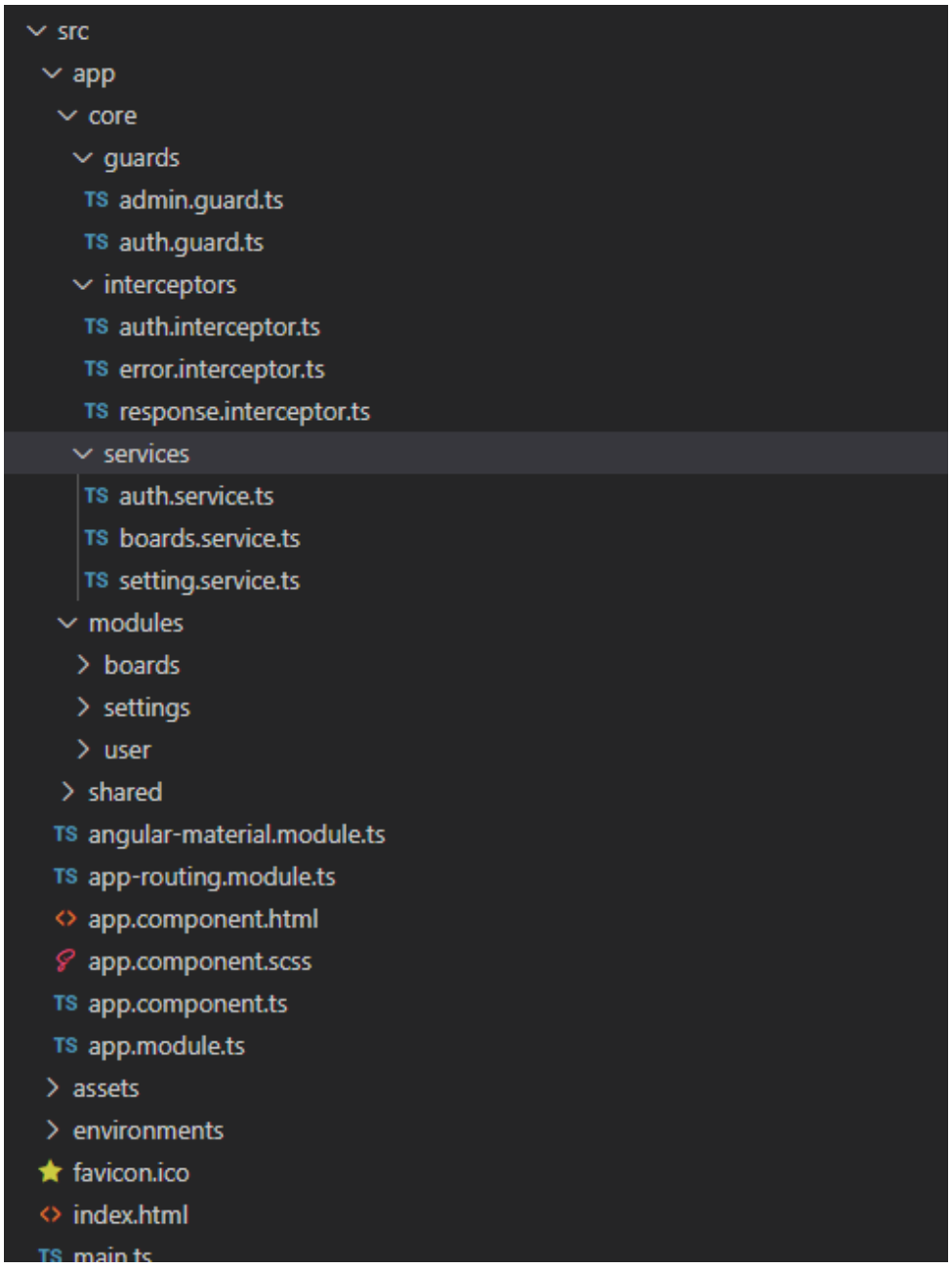
```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors(options =>
    {
        options.AddPolicy("EnableCORS", builder =>
        {
            builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod().AllowCredentials().Build();
        });
    });

    services.AddMvc(options => {
        options.EnableEndpointRouting = false;
    }).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);

    services.AddDependencies(Configuration);
    services.AddJwtAuthentication(Configuration);
}

0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseHsts();
    }

    loggerFactory.AddFile(Path.Combine(Directory.GetCurrentDirectory(), "logger.txt"));
    var logger = loggerFactory.CreateLogger("FileLogger");

    app.UseHangfireDashboard();
    app.UseCors("EnableCORS");
    app.UseAuthentication();
    app.UseRequestDurationMiddleware();
    app.UseMiddleware<ExceptionCatchMiddleware>(logger);
    app.UseMiddleware<ResponseMiddleware>();
    app.UseMvc();
    app.UseHttpsRedirection();

    DbInitializer.InitializeAsync(app.ApplicationServices.CreateScope().ServiceProvider).GetAwaiter();
}
}
```

Створення проекту на основі мікросервісів



Розроб.	Стаднік М.М.			*Створення проекту на основі мікросервісів*	
Перевір.	Мінухін С. В.				
Н. Контр.	Мінухін С. В.				
				СПРм-20-1	*Лист 5*
Затверд.	Гребеннік І.В.			*СТ*	*Листів 6*

Створення зовнішнього проекту



Розроб.	Стаднік М.М.			*Створення зовнішнього проекту*	
Перевір.	Мінухін С. В.				
Н. Контр.	Мінухін С. В.				
				СПРм-20-1	*Лист 6*
Затверд.	Гребеннік І.В.			*СТ*	*Листів 6*

ДОДАТОК Б

Текст програми

Міністерство освіти і науки України
Харківський Національний Університет Радіоелектроніки

«ЗАТВЕРДЖУЮ»
керівник магістерської
кваліфікаційної роботи
Мінухін С. В.

Розробка та дослідження компонентів системи управління документообігом з цифровими документами

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

УЗГОДЖЕНО:

РОЗРОБИВ:
ст. гр. СПРм-20-1
Стаднік М.М.

2021 р.

Розробка та дослідження компонентів системи управління документообігом з цифровими документами

Текст програми

АРКУШІВ 5

2021 р

```
public static void ExportExcel (DataTable dt, DataGridView dgv)
        {
            if (dt == null || dt.Rows.Count == 0)
            {
                MessageBox.Show(«Нет данных», «Подсказка», MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            else
            {
                List<string> listHeader = new List<string>();
                List<string> listDPName = new List<string>();
                for (int i = 0; i < dgv.Columns.Count; i++)
                {
                    if (dgv.Columns[i].Visible &&
dt.Columns.Contains(dgv.Columns[i].DataPropertyName)
                        && !listDPName.Contains(dgv.Columns[i].DataPropertyName))
                    {
                        listHeader.Add(dgv.Columns[i].HeaderText);
                        listDPName.Add(dgv.Columns[i].DataPropertyName);
                    }
                }
                DataTable exportDT = dt.DefaultView.ToTable(false, listDPName.ToArray());
                SaveFileDialog sfd = new SaveFileDialog();
                sfd.Filter = "файл xls | * .xls | файл xlsx | * .xlsx";
                if (sfd.ShowDialog() == DialogResult.OK)
                {
                    string result = Util.Pub.ExportExcel(exportDT, sfd.FileName, listHeader);
                    if (result == "")
                    {
                        MessageBox.Show(«Экспорт выполнен успешно», «Подсказка»,
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    else
                    {
                        MessageBox.Show(«Ошибка экспорта», «Подсказка»,
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }
        }
```

```
public async Task<object> LoginAsync(LoginDTO loginDTO)
    {
        var user = await _userManager.FindByEmailAsync(loginDTO.Email);

        if (user == null) throw new ValidationException("Invalid user");

        var result = await _signInManager.PasswordSignInAsync(loginDTO.Email,
loginDTO.Password, false, false);

        if (!result.Succeeded) throw new ValidationException("Invalid login attempt");

        return await GenerateJwtTokenAsync(user);
    }




    public async Task RegisterAsync(RegisterDTO registrationDTO)
    {
        ModelValidator.Validate(registrationDTO);

        User newUser = new User()
        {
            FirstName = registrationDTO.FirstName,
            LastName = registrationDTO.LastName,
            Email = registrationDTO.Email,
            UserName = registrationDTO.Email
        };
        string password = registrationDTO.Password;
        await CheckIfThePasswordIsValid(password);
        await CheckIfTheUserDoesNotExist(newUser);
        var isCreated = await _userManager.CreateAsync(newUser, password);

        if (!isCreated.Succeeded) throw new ValidationException(string.Join(", ",
isCreated.Errors.Select(p => p.Description)));

        var isAddedToRole = await _userManager.AddToRoleAsync(newUser, "user");

        if (!isAddedToRole.Succeeded) throw new ValidationException("Can't add user to
role");
    }
```

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from
'@angular/common/http';
import { Observable } from 'rxjs';
import { AuthService } from 'src/app/core/services/auth.service';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {
   constructor(private authService: AuthService) { }

   intercept(request: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
      if(this.authService.isLogged()) {
         request = request.clone({
            setHeaders: {
               Authorization: `Bearer ` + this.authService.getToken()
            }
         });
      }
      return next.handle(request);
   }
}

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environment';
import { map } from 'rxjs/operators';
import { TokenResponse } from 'src/app/shared/models/responses/token-response.model';
import { Login } from 'src/app/shared/models/auth/login.model';
import { JwtInformation } from 'src/app/shared/models/jwt-information.model';
import * as jwt_decode from "jwt-decode";
import { Register } from 'src/app/shared/models/auth/register.model';
import { Observable } from 'rxjs';




@Injectable({
  providedIn: 'root'
})
export class AuthService {
  jwtInfo: JwtInformation;
```

c

```
public static IdentityBuilder AddDependencies(this IServiceCollection services,
IConfiguration configuration)
        {
            TimeLimitConfiguration timeLimitConfiguration = new
TimeLimitConfiguration(configuration);
            EmailConfiguration emailConfiguration = new
EmailConfiguration(configuration);

            return services
                .AddScoped<IRepositoryBoard, BoardRepository>()
                .AddScoped<IRepositoryUser, UserRepository>()
                .AddScoped<IRepositoryCardList, CardListRepository>()
                .AddScoped<IRepositoryBoardDapper, BoardRepositoryDapper>(provider =>
new BoardRepositoryDapper(configuration.GetConnectionString("DefaultConnection")))
                .AddCache(configuration)
                .AddSingleton(timeLimitConfiguration)
                .AddSingleton(emailConfiguration)
                .AddHangFire(configuration)
                .AddTransient<IEmailService, EmailService>()
                .AddTransient<IAccountService, AccountService>()
                .AddTransient<IBoardService, BoardService>()
                .AddTransient<ICardListService, CardListService>()
                .AddIdentity(configuration);
        }

        private static IServiceCollection AddHangFire(this IServiceCollection services,
IConfiguration configuration)
        {
            return services.AddHangfire(x =>
x.UseSqlServerStorage(configuration.GetConnectionString("DefaultConnection")))
            .AddHangfireServer();
        }
```

```
onstructor(private httpClient: HttpClient) {
   if(this.isLogged()){
     this.jwtInfo = jwt_decode(this.getToken());
   }
  }

  isAdmin(): boolean{
```

```
    if(this.jwtInfo){
      return this.jwtInfo.typ.includes("admin");
    }
  }

  isLogged(): boolean{
    var token = JSON.parse(localStorage.getItem('accessToken'));
    return token !== null ? true : false;
  }

  getToken(): string{
    var token = JSON.parse(localStorage.getItem('accessToken'));
    if(token !== null)
      return token;
  }

  register(registerData: Register): Observable<any>{
    return this.httpClient.post<any>(environment.apiUrl + '/api/account/register',
registerData);
  }

  login(authData: Login) {
    return this.httpClient.post<TokenResponse>(environment.apiUrl + '/api/account/login',
authData)
      .pipe(map(tokenResponse => {
        var token = tokenResponse.accessToken;
        this.jwtInfo = jwt_decode(token
    private void ReturnPlatformToPool()
    {
        var tmp = activePlatforms[0];
        activePlatforms.RemoveAt(0);
        tmp.gameObject.SetActive(false);
        SetNextPlatform();
    }
```

ДОДАТОК В

«Специфікація»

Позначення	Найменування	Додаткові відомості
	Документація	
	Пояснювальна записка	71 стор.
	Текст програми	5 стор.

Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Стаднік М.М.			*Дослідження компонентів документообігу та генерації цифрових документів в веб додатках*	Арк.	Аркушів
Перевірив.		Мінухін С. В.				1	1
Н. Контр.		Мінухін С. В.				ХНУРЕ Кафедра СТ	
Затверд.		Гребеннік І.В.			Специфікація		

ДОДАТОК Г

«Відомість атестаційної роботи»

№	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1.		Пояснювальна записка	72 стор.
2.		Специфікація	1 стор
3.		Текст програми	5 стор.
		Графічні документи	
4.	С10	Результат роботи алгоритму генерування файлу	1 аркуш
5.		Результат роботи алгоритму парсингу файлу	1 аркуш
6.		Результат створення токену із засобами IdentityServer	1 аркуш
7.		Ініціалізація сервісів із конфігурацією	1 аркуш
8.		Створення проекту на основі мікросервісів	1 аркуш
9.		Створення зовнішнього проекту	1 аркуш
10.			
11.			
12.			
16.			
17.			

Змін.	Арк..	№ докум	Підп.	Дата			
Розроб.		*Стаднік М.М.*					
Перевір.		*Мінухін С.В.*			*Дослідження компонентів документообігу та генерації цифрових документів в веб додатках*	Аркуш	Аркушів
Н. Контр.		*Гребеннік І.В.*				*1*	*1*
Затв.						*ХНУРЕ Кафедра СТ*	