

## ПРИЛОЖЕНИЕ 1

Веб-служба для доступа к электронной почте на основе двухфакторной аутентификации

Текст программы

ВС ДЭП

Москва, 2022

## СОДЕРЖАНИЕ

1	Контроллер работы с учетными записями.....	3
2	Контроллер авторизации .....	5
3	Контроллер работы с почтовыми серверами.....	9
4	Контроллер получения писем .....	11
5	Контроллер изменения писем .....	13
6	Контроллер регистрации пользователей.....	15

## 1 Контроллер работы с учетными записями

```
using Iris.Services.AccountsService;
using Iris.Services.ClaimsPrincipalHelperService;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Iris.Api.Controllers.AccountsControllers;
/// <summary>
///     Контроллер работы с учетными записями
/// </summary>
[Authorize]
public class AccountsController : Controller
{
    private readonly IAccountsService _accountsService;
    private readonly IClaimsPrincipalHelperService
_accountsPrincipalHelperService;
    /// <summary>
    ///     .ctor
    /// </summary>
    public AccountsController(IAccountsService accountsService,
        IClaimsPrincipalHelperService claimsPrincipalHelperService)
    {
        _accountsService = accountsService;
        _claimsPrincipalHelperService = claimsPrincipalHelperService;
    }
    /// <summary>
    ///     Добавить новую учетную запись
    /// </summary>
    /// <param name="contract">Запрос добавления новой учетной записи</param>
    [HttpPost("~/api/accounts/add")]
    [AllowAnonymous]
    [ProducesResponseType(typeof(void), 200)]
    public IActionResult AddNewAccount([FromBody] AccountRequestContract
contract)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        _accountsService.AddNewAccount(userId, contract);
    }
}
```

```

        return Ok();
    }
    /// <summary>
    ///     Удалить учетную запись
    /// </summary>
    /// <param name="accId">Id учетной записи</param>
    [HttpDelete("~/api/accounts/{accId}")]
    [AllowAnonymous]
    [ProducesResponseType(typeof(void), 200)]
    public IActionResult RemoveAccount(int accId)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        _accountsService.RemoveAccount(userId, accId);
        return Ok();
    }
}

```

## 2 Контроллер авторизации

```
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using Iris.Api.Results;
using Iris.Database;
using Iris.Services.AuthService;
using Iris.Services.ClaimsPrincipalHelperService;
using Iris.Stores.AuthRequestStore;
using Iris.Stores.TokensStore;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using ILogger = Serilog.ILogger;

namespace Iris.Api.Controllers.AuthControllers;

/// <summary>
///     Контроллер авторизации
/// </summary>
[Authorize]
public class AuthController : Controller
{
    private static readonly ILogger Log =
Serilog.Log.ForContext<AuthController>();
    private readonly IAuthRequestsStore _authRequestsStore;
    private readonly IAuthService _authService;
    private readonly IClaimsPrincipalHelperService
_authClaimsPrincipalHelperService;
    private readonly ITokensStore _tokensStore;

    /// <summary>
    ///     .ctor
    /// </summary>
    public AuthController(IAuthRequestsStore authRequestsStore,
        IAuthService authService, ITokensStore tokensStore,
        IClaimsPrincipalHelperService claimsPrincipalHelperService)
    {
```

```

        _authRequestsStore = authRequestsStore;
        _authService = authService;
        _tokensStore = tokensStore;
        _claimsPrincipalHelperService = claimsPrincipalHelperService;
    }

    /// <summary>
    ///     Инициализировать авторизацию
    /// </summary>
    /// <returns>Id запроса авторизации</returns>
    [HttpPost("~/api/authorize")]
    [AllowAnonymous]
    [ProducesResponseType(typeof(int), 201)]
    public IActionResult InitAuth()
    {
        var request = _authRequestsStore.CreateRequest();
        return Created(
            $"/api/authorize/{Uri.EscapeDataString(request.Id)}",
            request.Id
        );
    }

    /// <summary>
    ///     Выполнить авторизацию
    /// </summary>
    /// <param name="id">Id запроса авторизации</param>
    /// <param name="authRequest">Контракт запроса авторизации</param>
    /// <returns>Контракт ответа авторизации</returns>
    [HttpPut("~/api/authorize/{id}")]
    [AllowAnonymous]
    [ProducesResponseType(typeof(AuthResponseContract), 200)]
    public IActionResult ExecuteAuthorization(string id, [FromBody]
AuthRequestContract authRequest)
    {
        var operationRequest = _authRequestsStore.FindRequest(id);

        if (operationRequest == null) return new AuthErrorResult(message:
"Ошибка авторизации");
    }

```

```

ClaimsIdentity identity;
User user;

try
{
    (identity, user) = _authService.Authorize(operationRequest,
authRequest);
}
catch
{
    Log.Error($"Ошибка авторизации пользователя
{authRequest.Login}");
    throw;
}

if (identity == null)
{
    Log.Error($"Ошибка авторизации пользователя
{authRequest.Login}");
    return new AuthErrorResult();
}

var (token, expires) = _authService.GenerateToken(identity.Claims,
user);

var roles = identity.Claims.Where(_ => _.Type ==
identity.RoleClaimType)
    .Select(_ => _.Value).ToList();

var userIdVal = identity.Claims.First(c => c.Type ==
JwtRegisteredClaimNames.Sid).Value;
var userId = int.Parse(userIdVal);

_tokensStore.AddOrUpdate(userId.ToString(), token);

Log.Information($"Пользователь {user.Name} авторизован");

return Ok(new AuthResponseContract
{

```

```

        UserId = userIdVal,
        Login = identity.Name,
        Token = token,
        Roles = roles,
        TokenType = JwtBearerDefaults.AuthenticationScheme
    });
}

/// <summary>
///     Де-авторизация
/// </summary>
[HttpPost("~/api/authorize/deauth")]
[ProducesResponseType(typeof(OkResult), 200)]
public IActionResult DeAuth()
{
    var userId = _claimsPrincipalHelperService.GetUserId(User);

    _tokensStore.Remove(userId.ToString());
    Log.Information($"User {userId} is de-auth");

    Response?.Headers?.Add("Clear-Site-Data", "\"cache\"", "\"cookies\"",
        "\"storage\"");

    return Ok();
}

/// <summary>
///     Проверка авторизованности
/// </summary>
/// <returns></returns>
[HttpGet("~/api/authorize/isauth")]
[AllowAnonymous]
[ProducesResponseType(typeof(bool), 200)]
public IActionResult IsAuth()
{
    return Ok(User?.Identity?.IsAuthenticated ?? false);
}
}

```



### 3 Контроллер работы с почтовыми серверами

```
using Iris.Database;
using Iris.Services.ClaimsPrincipalHelperService;
using Iris.Services.MailServersService;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Iris.Api.Controllers.ConnectionsControllers;

/// <summary>
///     Контроллер работы с почтовыми серверами
/// </summary>
[Authorize]
public class MailServersController : Controller
{
    private readonly IClaimsPrincipalHelperService
        _claimsPrincipalHelperService;
    private readonly IMailServersService _mailServersService;

    /// <summary>
    ///     .ctor
    /// </summary>
    public MailServersController(IMailServersService mailServersService,
        IClaimsPrincipalHelperService claimsPrincipalHelperService)
    {
        _mailServersService = mailServersService;
        _claimsPrincipalHelperService = claimsPrincipalHelperService;
    }

    /// <summary>
    ///     Получить список аккаунтов к почтовым серверам пользователя
    /// </summary>
    [HttpGet("~/api/connections/mailservers/accounts/users")]
    [ProducesResponseType(typeof(IEnumerable<MailServerAccountContract>),
200)]
    public IActionResult GetMailServersAccounts()
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
```

```

        var serverAccounts =
_mailServersService.GetMailServerAccounts(userId);
        return Ok(serverAccounts);
    }
    /// <summary>
    ///     Получить список доступных почтовых серверов
    /// </summary>
    [HttpGet("~/api/connections/mailservers/accounts/availables")]
    [ProducesResponseType(typeof(IEnumerable<MailServer>), 200)]
    public IActionResult GetAvailableMailServers()
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        var serverAccounts =
_mailServersService.GetAvailableMailServers(userId);
        return Ok(serverAccounts);
    }
    /// <summary>
    ///     Добавить новый почтовый сервер
    /// </summary>
    /// <param name="mailServerContract">Контракт добавления сервера</param>
    [HttpPost("~/api/connections/mailservers")]
    [ProducesResponseType(typeof(void), 200)]
    public IActionResult AddUserMailService([FromBody] MailServerContract
mailServerContract)
    {
        _mailServersService.NewMailServer(mailServerContract);
        return Ok();
    }
}

```

## 4 Контроллер получения писем

```
using Iris.Services.ClaimsPrincipalHelperService;
using Iris.Services.FormatLettersService;
using Iris.Services.LettersService;
using Iris.Services.LettersService.Contracts;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Iris.Api.Controllers.LettersControllers
{
    /// <summary>
    /// Контроллер получения писем
    /// </summary>
    [Authorize]
    public class GetLettersController : Controller
    {
        private readonly ILetterService _letterService;
        private readonly IFormatLettersService _formatLettersService;
        private readonly IClaimsPrincipalHelperService
        _claimsPrincipalHelperService;

        /// <summary>
        /// .ctor
        /// </summary>
        public GetLettersController(ILetterService letterService,
        IFormatLettersService formatLettersService, IClaimsPrincipalHelperService
        claimsPrincipalHelperService)
        {
            _letterService = letterService;
            _formatLettersService = formatLettersService;
            _claimsPrincipalHelperService = claimsPrincipalHelperService;
        }

        /// <summary>
        /// Получить письма по запросу
        /// </summary>
        /// <param name="lettersRequest"></param>
```

```

    /// <returns></returns>
    [HttpGet("~/api/letters")]
    [ProducesResponseType(typeof(IEnumerable<LetterContract>), 200)]
    public IActionResult GetLetters([FromQuery] LettersRequest
lettersRequest)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        var letters = _letterService.GetLetters(userId, lettersRequest);

        return Ok(letters);
    }

    /// <summary>
    /// Получить письма по запросу
    /// </summary>
    /// <param name="lettersRequest">Запрос писем</param>
    /// <param name="format">Формат</param>
    /// <returns></returns>
    [HttpGet("~/api/{format}/letters")]
    [ProducesResponseType(typeof(IEnumerable<LetterContract>), 200)]
    public IActionResult GetLetters(string format, [FromQuery]
LettersRequest lettersRequest)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        var needFormat = _formatLettersSevice.GetFormat(format);
        var letters = _letterService.GetLetters(userId, lettersRequest);
        var formattedLetters =
_formatLettersSevice.FormatLetters(letters, needFormat);

        return Ok(formattedLetters);
    }
}
}

```

## 5 Контроллер изменения писем

```
using Iris.Services.ClaimsPrincipalHelperService;
using Iris.Services.LettersService;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
namespace Iris.Api.Controllers.LettersControllers;
/// <summary>
///     Контроллер изменения писем
/// </summary>
[Authorize]
public class UpdateLettersController : Controller
{
    private readonly IClaimsPrincipalHelperService
_claimsPrincipalHelperService;
    private readonly ILetterService _letterService;
    /// <summary>
    ///     .ctor
    /// </summary>
    public UpdateLettersController(ILetterService letterService,
        IClaimsPrincipalHelperService claimsPrincipalHelperService)
    {
        _letterService = letterService;
        _claimsPrincipalHelperService = claimsPrincipalHelperService;
    }
    /// <summary>
    ///     Установить флаг
    /// </summary>
    /// <param name="accId">Id учетной записи</param>
    /// <param name="letterId">Id письма</param>
    /// <param name="flag">Флаг</param>
    [HttpPost("~/api/letters/accaunt/{accId}/letter/{letterId}/flag/{flag}")]
    [ProducesResponseType(typeof(OkResult), 200)]
    public IActionResult ChangeFlag(int accId, string letterId, int flag)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        _letterService.ChangeFlag(userId, accId, letterId, flag);
        return Ok();
    }
}
```

```

    }
    /// <summary>
    ///     Удалить письмо
    /// </summary>
    /// <param name="accId">Id учетной записи</param>
    /// <param name="letterId">Id письма</param>
    [HttpDelete("~/api/letters/accaunt/{accId}/letter/{letterId}")]
    [ProducesResponseType(typeof(OkResult), 200)]
    public IActionResult RemoveLetter(int accId, string letterId)
    {
        var userId = _claimsPrincipalHelperService.GetUserId(User);
        _letterService.RemoveLetter(userId, accId, letterId);
        return Ok();
    }
}

```

## 6 Контроллер регистрации пользователей

```
using Iris.Api.Controllers.RegistrationControllers;
using Iris.Services.RegistrationService.cs;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Iris.Api.Controllers.RegistrationController;

/// <summary>
///     Контроллер регистрации пользователей
/// </summary>
[Authorize]
public class RegistrationController : Controller
{
    private readonly IRegistrationService _registrationService;

    /// <summary>
    ///     .ctor
    /// </summary>
    public RegistrationController(IRegistrationService registrationService)
    {
        _registrationService = registrationService;
    }

    /// <summary>
    ///     Зарегистрировать пользователя
    /// </summary>
    /// <param name="contract">Контракт создания пользователя</param>
    [HttpPost("~/api/registration")]
    [AllowAnonymous]
    [ProducesResponseType(typeof(RegistrationResponseContract), 200)]
    public IActionResult RegisterUser([FromBody] RegistrationRequestContract
contract)
    {
        return Ok(_registrationService.RegisterUser(contract));
    }
}
```