Contents

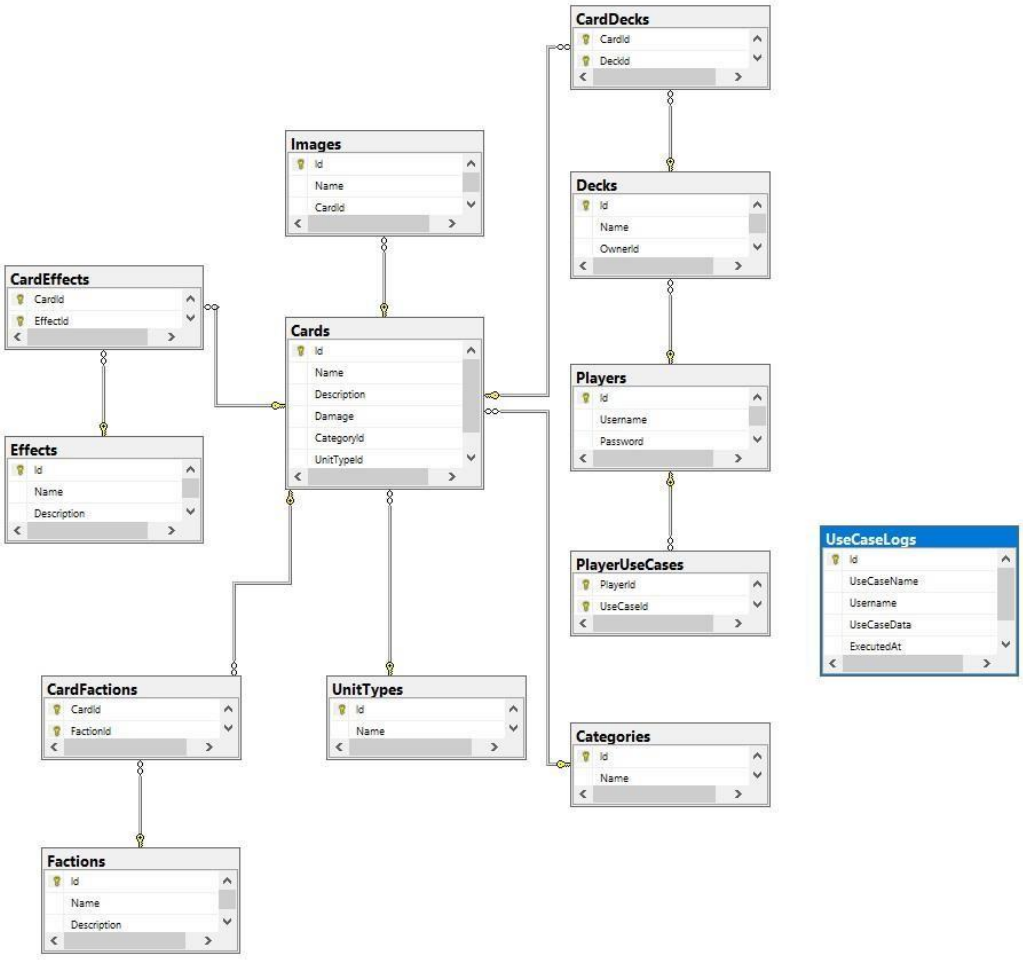
Opis Sistema:

Gwent je kartaška igra u kojoj postoje karte sa svojim:

- Fakcijama(svaka karta se može koristiti u više fakcija, ali postoje i specijalne karte koje pripadaju jednoj fakciji)
- Kategora(Npr Weather ili Vremenska karta koja upravlja napadom jedinica u odjeđenom tipu jedinice)
- Tip jedinice(postoje pešadija ili prva borbena linija, ranged jedinice ili druga borbena linija i siege jedinice ili treća borbena linija)
- Sistem je tako napravljen da postoji jedan korisnik koji može da upravlja celim sistemom dodavanjem novih entiteta
- Korisnici koji samo mogu da prave svoje deckove(špilove) i u njih da dodaju karte kao i da iz svojih špilova brišu karte(isključivo iz svojih što je regulisano validacijom i proverama)

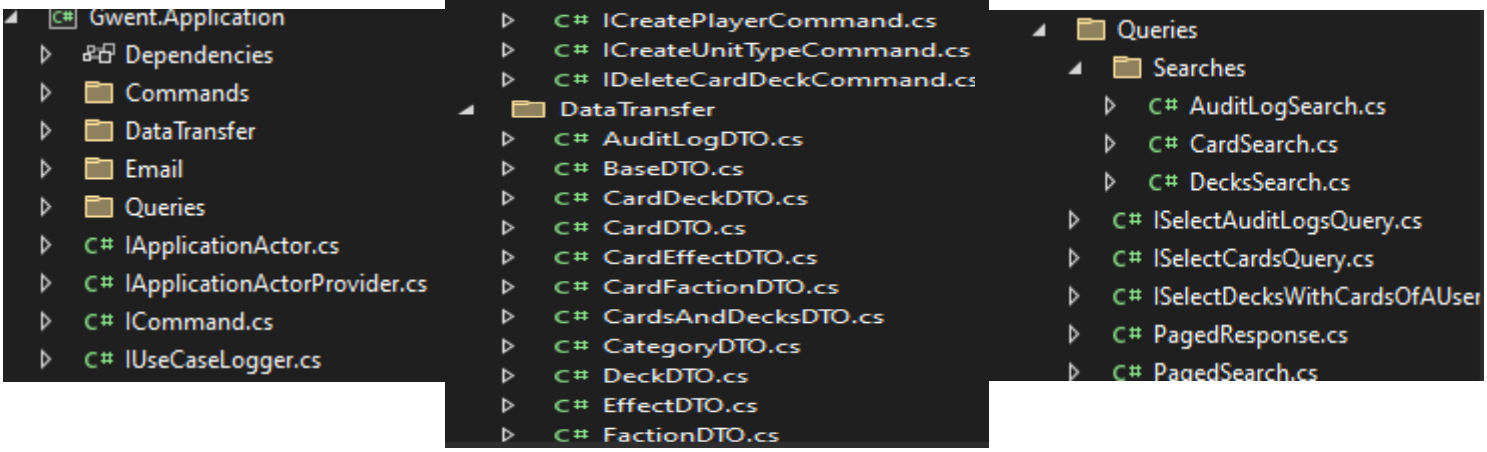
Sistem je osmišljen tako da predstavlja katalog za karte i da korisnici mogu da upravljaju svojim špilovima.

Dizajn baze:

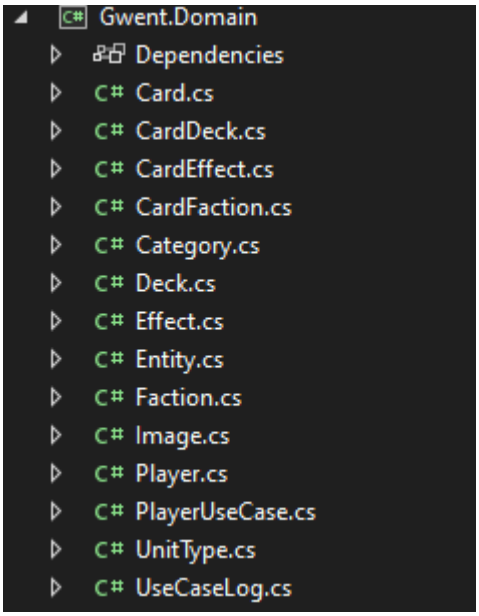

CardDecks
CardId
DeckId
Images
Id
Name
CardId
Decks
Id
Name
OwnerId
CardEffects
CardId
EffectId
Cards
Id
Name
Description
Damage
CategoryId
UnitTypeId
Players
Id
Username
Password
Effects
Id
Name
Description
PlayerUseCases
PlayerId
UseCaseId
UseCaseLogs
Id
UseCaseName
Username
UseCaseData
ExecutedAt
CardFactions
CardId
FactionId
UnitTypes
Id
Name
Categories
Id
Name
Factions
Id
Name
Description

Arhitektura sistema:

Sloj aplikacije:

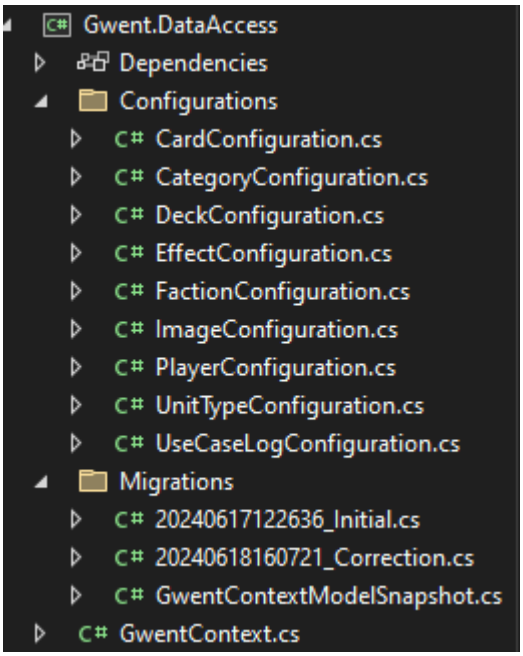


Domenski sloj:

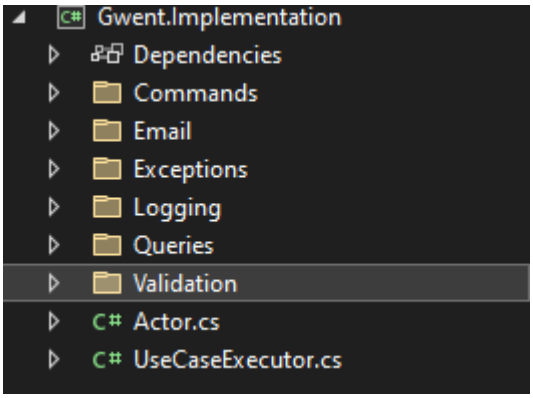


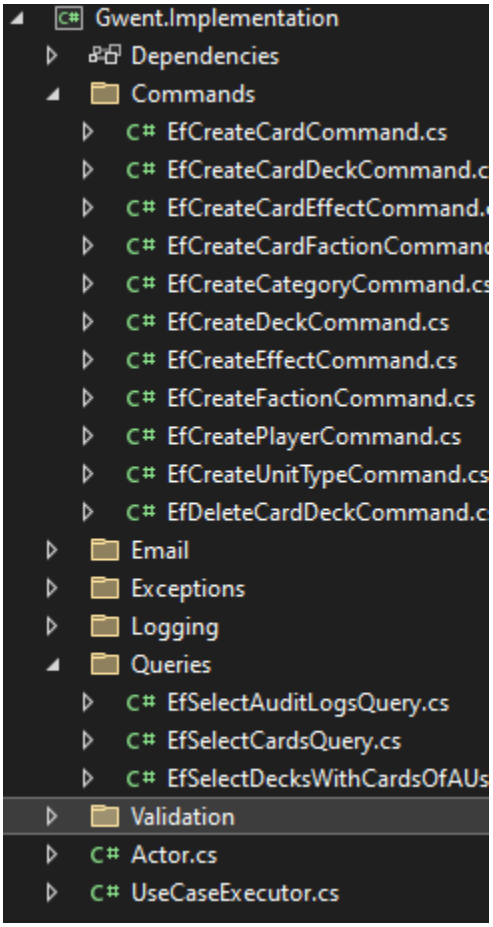
DataAccess sloj:

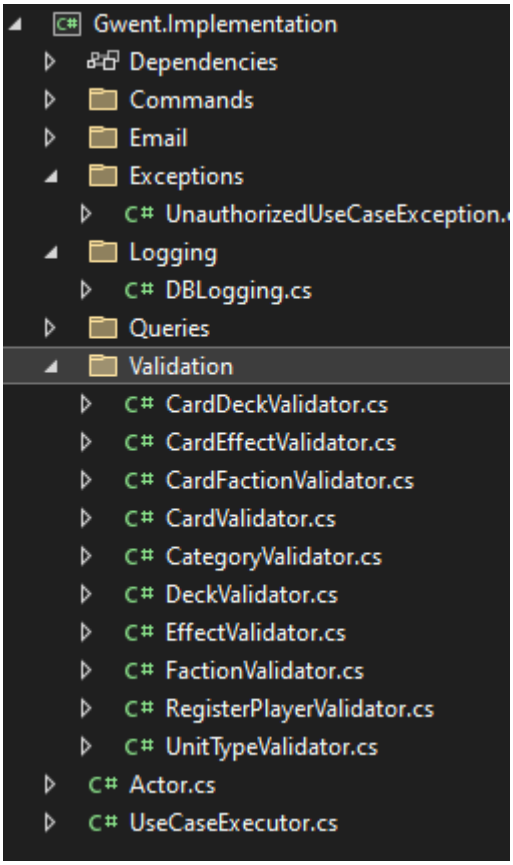
Baza je napravljena code-first pristupom



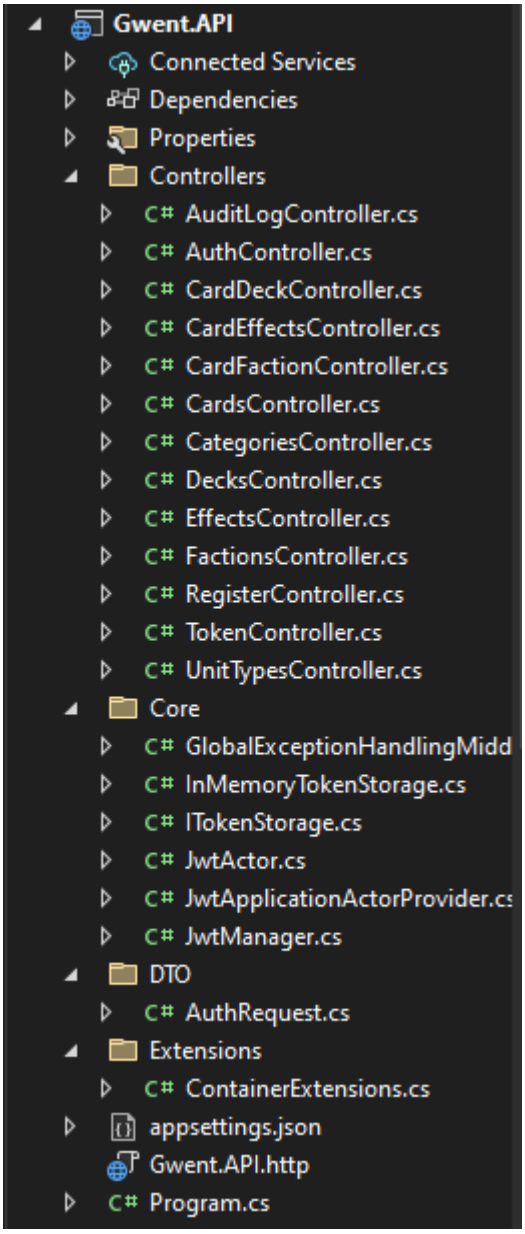
Sloj implementacije:







API sloj:

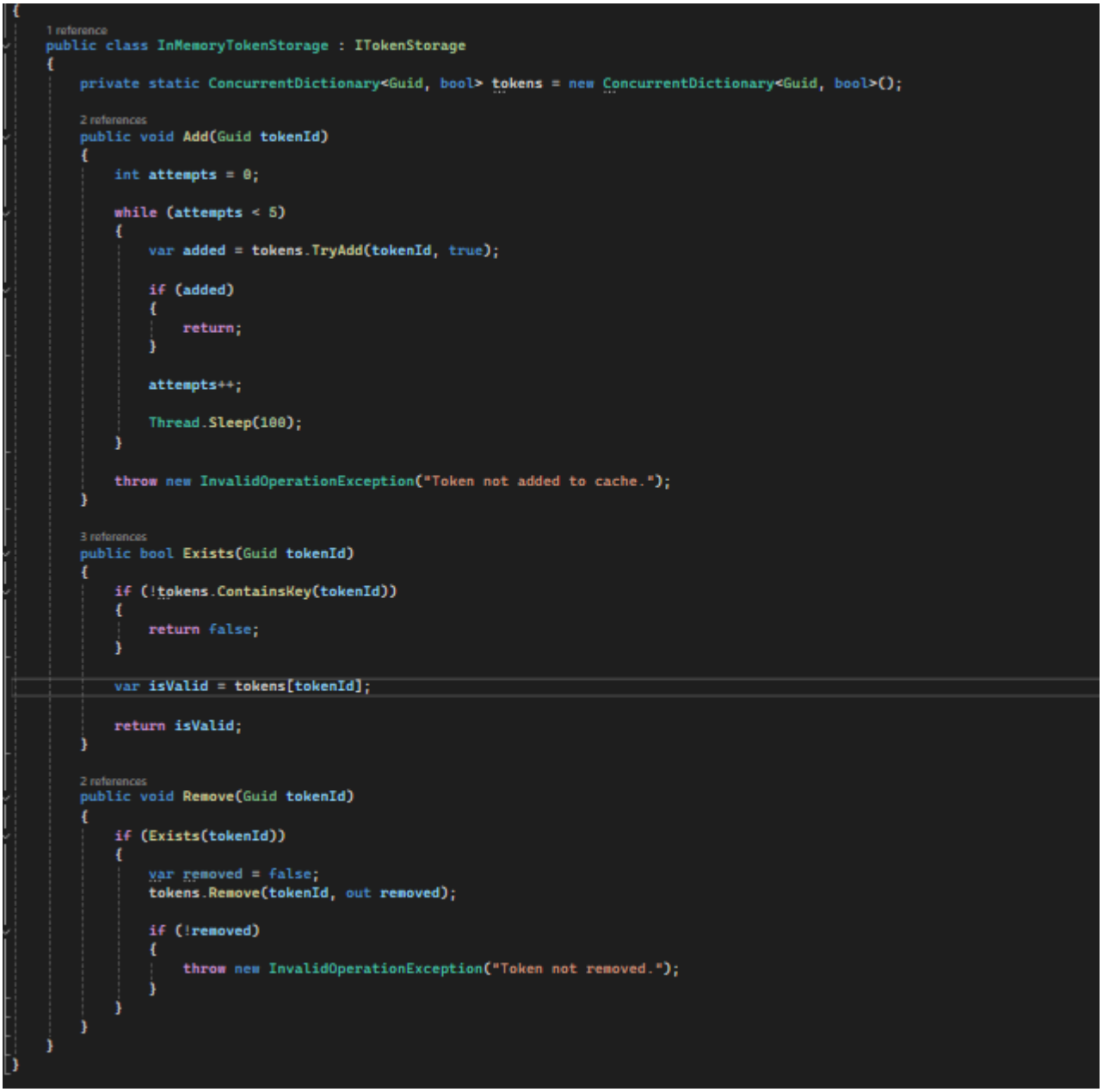

Gwent.API
Connected Services
Dependencies
Properties
Controllers
AuditLogController.cs
AuthController.cs
CardDeckController.cs
CardEffectsController.cs
CardFactionController.cs
CardsController.cs
CategoriesController.cs
DecksController.cs
EffectsController.cs
FactionsController.cs
RegisterController.cs
TokenController.cs
UnitTypesController.cs
Core
GlobalExceptionHandlingMidd
InMemoryTokenStorage.cs
ITokenStorage.cs
JwtActor.cs
JwtApplicationActorProvider.cs
JwtManager.cs
DTO
AuthRequest.cs
Extensions
ContainerExtensions.cs
appsettings.json
Gwent.API.http
Program.cs

Validacija JWT tokenom:

- controller za dohvatanje tokena koji prima objekat sa poljima Username i Password koji proverava korisnika u bazi, ako postoji vraća token sa njegovim podacima

takodje ovde postoji I metoda za logout koja brise korisnika iz token storage-a

```
public class AuthController : ControllerBase
{
    private readonly JwtManager _manager;
    0 references
    public AuthController(JwtManager _manager)
    {
        this._manager = _manager;
    }

    [HttpPost]
    0 references
    public IActionResult Post([FromBody] AuthRequest request)
    {
        string token = _manager.MakeToken(request.Username, request.Password);
        if(token == null)
        {
            return NotFound(new { error = "Invalid credentials, try again." });
        }
        return Ok(new AuthResponse { Token = token });
    }

    [Authorize]
    [HttpDelete]
    0 references
    public IActionResult Delete([FromServices] ITokenStorage storage)
    {
        storage.Remove(this.Request.GetTokenId().Value);
        return NoContent();
    }
}
```



```
{
    1 reference
    public class InMemoryTokenStorage : ITokenStorage
    {
        private static ConcurrentDictionary<Guid, bool> tokens = new ConcurrentDictionary<Guid, bool>();

        2 references
        public void Add(Guid tokenId)
        {
            int attempts = 0;

            while (attempts < 5)
            {
                var added = tokens.TryAdd(tokenId, true);

                if (added)
                {
                    return;
                }

                attempts++;

                Thread.Sleep(100);
            }

            throw new InvalidOperationException("Token not added to cache.");
        }

        3 references
        public bool Exists(Guid tokenId)
        {
            if (!tokens.ContainsKey(tokenId))
            {
                return false;
            }

            var isValid = tokens[tokenId];

            return isValid;
        }

        2 references
        public void Remove(Guid tokenId)
        {
            if (Exists(tokenId))
            {
                var removed = false;
                tokens.Remove(tokenId, out removed);

                if (!removed)
                {
                    throw new InvalidOperationException("Token not removed.");
                }
            }
        }
    }
}
```

```
public interface ITokenStorage
{
    3 references
    bool Exists(Guid tokenId);
    2 references
    void Add(Guid tokenId);
    2 references
    void Remove(Guid tokenId);
}
```

```
1 reference
public class Actor : IApplicationActor
{
    8 references
    public int Id { get; set; }

    4 references
    public string Identity { get; set; }

    4 references
    public IEnumerable<int> AllowedUseCases { get; set; }
}

2 references
public class UnauthorizedActor : IApplicationActor
{

    7 references
    public int Id => 0;
    3 references
    public string Identity => "unauthorized";
    3 references
    public IEnumerable<int> AllowedUseCases => new List<int> { 8 , 11};
}
```

```
public class JwtApplicationActorProvider : IApplicationActorProvider
{
    private string authorizationHeader;


    public JwtApplicationActorProvider(string authorizationHeader)
    {
        this.authorizationHeader = authorizationHeader;
    }

    public IApplicationActor GetActor()
    {
        if (authorizationHeader.Split("Bearer ").Length != 2)
        {
            return new UnauthorizedActor();
        }

        string token = authorizationHeader.Split("Bearer ")[1];

        var handler = new JwtSecurityTokenHandler();

        var tokenObj = handler.ReadJwtToken(token);

        var claims = tokenObj.Claims;

        var claim = claims.First(x => x.Type == "jti").Value;

        var actor = new Actor
        {
            Identity = claims.First(x => x.Type == "Identity").Value,
            Id = int.Parse(claims.First(x => x.Type == "Id").Value),
            AllowedUseCases = JsonConvert.DeserializeObject<List<int>>(claims.First(x => x.Type == "UseCases").Value)
        };

        return actor;
    }
}
```

```
public class JwtManager
{
    private readonly GwentContext _context;
    private readonly ITokenStorage _storage;

    0 references
    public JwtManager(GwentContext context, ITokenStorage storage)
    {
        _context = context;
        _storage = storage;
    }

    1 reference
    public string MakeToken(string username, string password)
    {
        var user = _context.Players.Where(x => x.Username == username).Select(x => new
        {
            x.Username,
            x.Password,
            x.Id,
            UseCases = x.UseCases.Select(x => x.UseCaseId)
        }).FirstOrDefault();

        if (user == null)
        {
            return null;
        }

        var issuer = "asp_api";
        var secretKey = "this is my custom Secret key for authentication";
        Guid token_guid = Guid.NewGuid();
        string token_id = token_guid.ToString();
        var claims = new List<Claim> // Jti : "",
        {
            new Claim(JwtRegisteredClaimNames.Jti, token_id, ClaimValueTypes.String),
            new Claim(JwtRegisteredClaimNames.Iss, "asp_api", ClaimValueTypes.String),
            new Claim(JwtRegisteredClaimNames.Iat, DateTimeOffset.UtcNow.ToUnixTimeSeconds().ToString(), ClaimValueTypes.Integer64),
            new Claim("Id", user.Id.ToString()),
            new Claim("Identity", user.Username.ToString()),
            new Claim("UseCases",JsonConvert.SerializeObject(user.UseCases)),
        };

        var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey));

        var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

        var now = DateTime.UtcNow;
        var token = new JwtSecurityToken(
            issuer: issuer,
            audience: "Any",
            claims: claims,
            notBefore: now,
            expires: now.AddSeconds(300),
            signingCredentials: credentials);

        _storage.Add(token_guid);

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
```

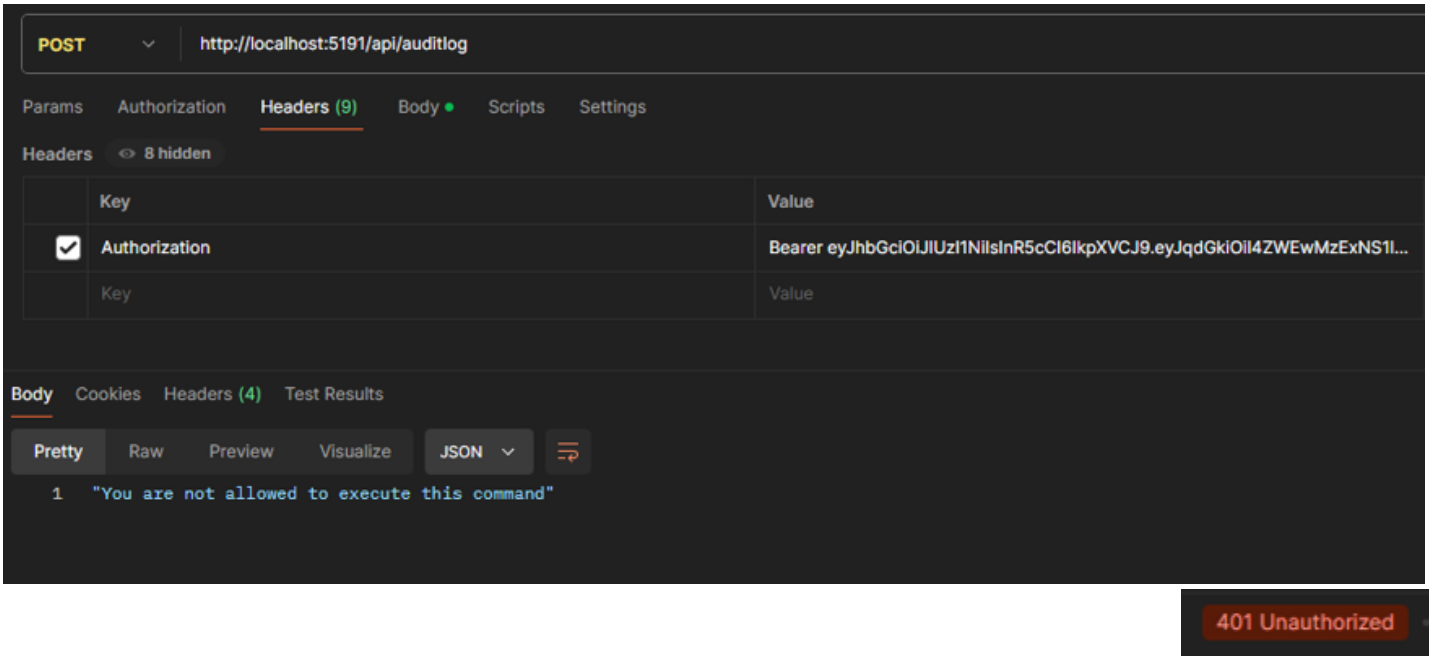
```
1 reference
public class AuthRequest
{
    1 reference
    public string Username { get; set; }
    1 reference
    public string Password { get; set; }
}

1 reference
public class AuthResponse
{
    1 reference
    public string Token { get; set; }
}
```

```
1 reference
public class RegisterController : ControllerBase
{
    public IApplicationActor actor;
    public UseCaseExecutor executor;
    public GwentContext context;

    0 references
    public RegisterController(IApplicationActor actor, UseCaseExecutor executor, GwentContext context)
    {
        this.actor = actor;
        this.executor = executor;
        this.context = context;
    }

    // POST api/<RegisterController>
    [HttpPost]
    0 references
    public IActionResult Post([FromBody] PlayerDTO dto, [FromServices] ICreatePlayerCommand command)
    {
        if (dto == null)
        {
            return UnprocessableEntity();
        }

        Player check = context.Players.Where(x => x.Username == dto.Username).FirstOrDefault();

        if (check != null)
        {
            return Conflict(new { error = "A player with the same username already exists! Try something else." });
        }

        executor.ExecuteCommand(command, dto);
        return Ok(new { message = "You have succesfully registered!" });
    }
}
}
```

Na ovom endpointu se korisnik registruje I automatski mu se dodeljuju use case-ovi koje moze da izvrsi.

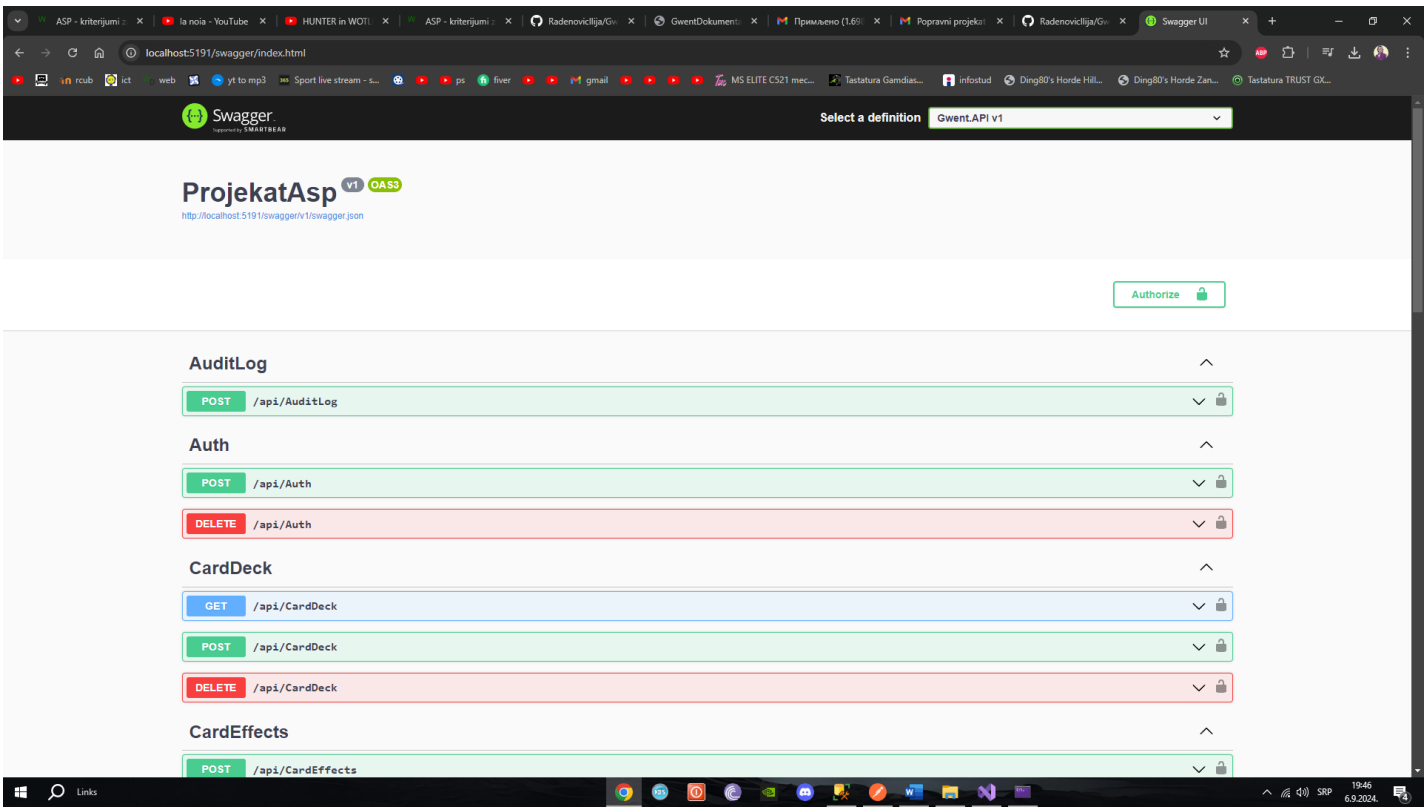
```
2 references
public void Execute(PlayerDTO request)
{
    validator.ValidateAndThrow(request);

    Player p = new Player
    {
        Username = request.Username,
        Password = request.Password,
    };

    context.Players.Add(p);
    context.SaveChanges();

    context.PlayerUseCases.Add(new PlayerUseCase
    {
        PlayerId = p.Id,
        UseCaseId = 9
    });
    context.PlayerUseCases.Add(new PlayerUseCase
    {
        PlayerId = p.Id,
        UseCaseId = 10
    });
    context.PlayerUseCases.Add(new PlayerUseCase
    {
        PlayerId = p.Id,
        UseCaseId = 11
    });
    context.PlayerUseCases.Add(new PlayerUseCase
    {
        PlayerId = p.Id,
        UseCaseId = 12
    });

    context.SaveChanges();
}
```

Autorizacija na nivou slucaja koriscenja(granulacija)

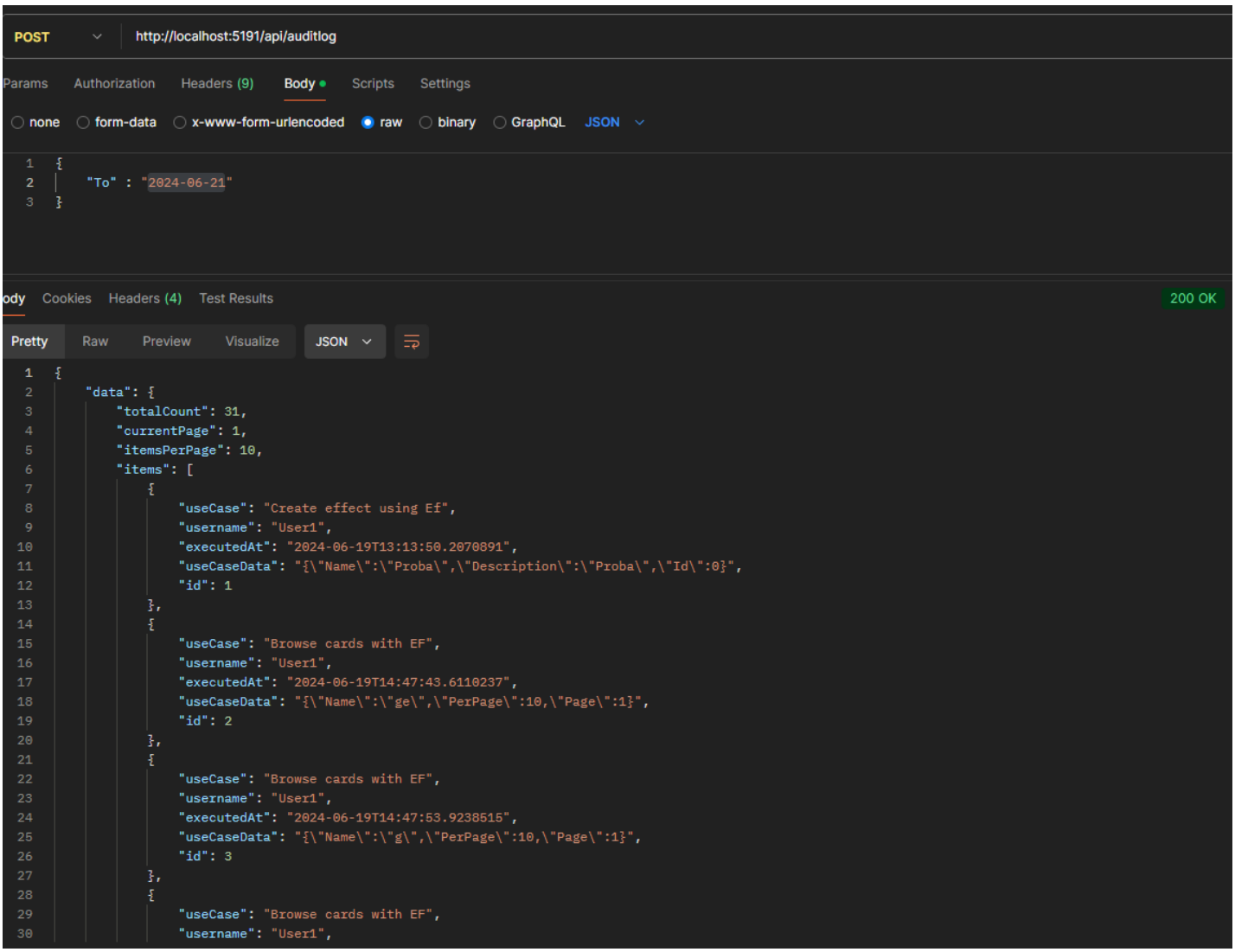


Korisnik koji je poslao zahtev na ovaj endpoint nema u svojim dozvoljenim use caseovima citanje auditlogova I time se ispisala poruka I statusni kod 401.

Swagger specifikacija:



Audit log sa pretragom I paginacijom:



Pretraga audit logova je moguca sa datumom od I do kada je izvrsen audit log I nazivom korisnika koji je izvrsio use case

Pretraga i paginacija

http://localhost:5191/api/cards?name=a

Na ovom endpointu dobija se rezultat pretrage karata sa opcijom paginacije


```
{
  "data": {
    "totalCount": 4,
    "currentPage": 1,
    "itemsPerPage": 10,
    "items": [
      {
        "name": "Geralt of Rivia",
        "description": "A seasoned witcher experienced in dealing with the most powerful monsters",
        "damage": 10,
        "unitTypeId": 1,
        "categoryId": 1,
        "id": 1
      },
      {
        "name": "Yennefer of Vangerberg",
        "description": "Sorceress with many skills and talents.",
        "damage": 7,
        "unitTypeId": 2,
        "categoryId": 1,
        "id": 3
      },
      {
        "name": "John Natalis",
        "description": "A strong warrior from the Northern Realms",
        "damage": 10,
        "unitTypeId": 1,
        "categoryId": 1,
        "id": 4
      },
      {
        "name": "Cirilla Fiona Ellen Riannon",
        "description": "Know when fairy tales cease to be tales? When people start believing in them",
        "damage": 15,
        "unitTypeId": 1,
        "categoryId": 1,
        "id": 5
      }
    ],
```

Granulacija privilegija na osnovu svakog korisnika i logovanje izvršavanja usecase-a u tabelu UseCaseLogs

Klasa UseCaseExecutor služi za obradu svakog usecase-a i takođe u sebi sadrži proveru da li korisnik koji je pokrenuo usecase može da izvrši isti time što proverava njegov niz int-ova koji se izvlači iz baze da li se u njemu nalazi Id tekućeg usecase-a,

```
public class UseCaseExecutor
{
    private readonly IApplicationActor actor;
    private readonly IUseCaseLogger logger;

    public UseCaseExecutor(IApplicationActor actor, IUseCaseLogger logger)
    {
        this.actor = actor;
        this.logger = logger;
    }

    public TResult ExecuteQuery<TSearch, TResult>(IQuery<TSearch, TResult> query, TSearch search)
    {
        logger.Log(query, actor, search);

        if (!actor.AllowedUseCases.Contains(query.Id))
        {
            throw new UnauthorizedUseCaseException(query, actor);
        }

        return query.Execute(search);
    }

    public void ExecuteCommand<TRequest>(
        ICommand<TRequest> command,
        TRequest request)
    {
        logger.Log(command, actor, request);
        if (!actor.AllowedUseCases.Contains(command.Id))
        {
            throw new UnauthorizedUseCaseException(command, actor);
        }

        command.Execute(request);
    }
}
```

```
public class DBLogging : IUseCaseLogger
{
    private readonly GwentContext context;

    public DBLogging(GwentContext context)
    {
        this.context = context;
    }


    public void Log(IUseCase useCase, IApplicationActor actor, object data)
    {
        UseCaseLog log = new UseCaseLog
        {
            UseCaseName = useCase.Name,
            Username = actor.Identity,
            UseCaseData = JsonConvert.SerializeObject(data),
            ExecutedAt = DateTime.Now,
        };

        context.UseCaseLogs.Add(log);
        context.SaveChanges();
    }
}
```

	Id	UseCaseName	Username	UseCaseData	ExecutedAt
1	1	Create effect using Ef	User1	{"Name":"Proba","Description":"Proba","Id":0}	2024-06-19 13:13:50.2070891
2	2	Browse cards with EF	User1	{"Name":"ge","PerPage":10,"Page":1}	2024-06-19 14:47:43.6110237
3	3	Browse cards with EF	User1	{"Name":"g","PerPage":10,"Page":1}	2024-06-19 14:47:53.9238515
4	4	Browse cards with EF	User1	{"Name":"a","PerPage":10,"Page":1}	2024-06-19 14:47:59.0547557
5	5	Create a deck using EF	User1	{"Name":"Deck1","OwnerId":1,"Id":0}	2024-06-19 16:26:47.8353171
6	6	Create a deck using EF	User1	{"Name":"Deck2","OwnerId":1,"Id":0}	2024-06-19 16:28:24.6783601
7	7	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:31:28.4865619
8	8	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:31:46.3448029
9	9	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:32:10.1398965
10	10	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:32:27.0933364
11	11	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:42:44.2868150
12	12	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:42:47.1287988
13	13	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:43:41.0564130
14	14	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:44:06.2331780
15	15	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:45:37.9497819
16	16	Create a deck using EF	User1	{"Name":"Deck3","OwnerId":1,"Id":0}	2024-06-19 16:53:20.5522054
17	17	Create a deck using EF	User1	{"Name":"Deck4","OwnerId":1,"Id":0}	2024-06-19 16:55:52.8034389
18	18	Create a deck using EF	User1	{"Name":"Deck5","OwnerId":1,"Id":0}	2024-06-19 16:55:55.9834030
19	19	Create a deck using EF	User1	{"Name":"Deck6","OwnerId":1,"Id":0}	2024-06-19 16:55:58.8548486