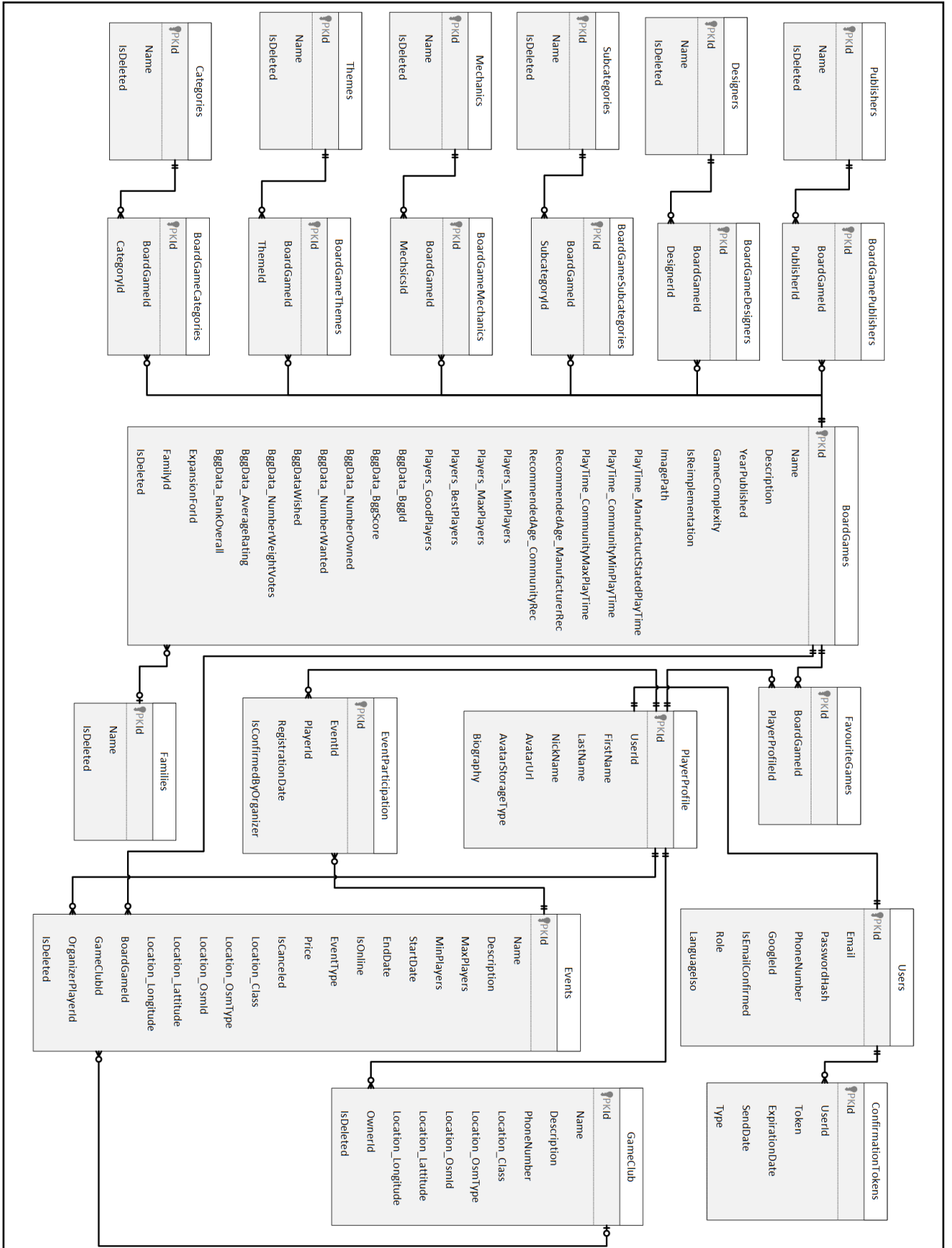


# ДОДАТОК А

## ER-діаграма у нотації Crow's Foot



## ДОДАТОК Б

### Фрагменти програмного коду серверної частини

#### Б.1 Код сервісу для імпорту даних про настільні ігри

```

internal class BoardGamesCsvImportService :
IBoardGamesCsvImportService
{
    private readonly CsvConfiguration csvReaderConfig =
new(CultureInfo.InvariantCulture)
    {
        Delimiter = ";",
        MissingFieldFound = null
    };
    private const string bggIdHeaderColumnName = "BGGId";
    private const string categoryHeaderColumnPrefix = "Cat:";
    private const string rankHeaderColumnName = "Rank:";
    private const string themeHeaderColumnPrefix = "Theme_";

    public BoardGamesCsvResultDto GetBoardGamesImportedData(Stream
csvStream)
    {
        csvStream.Position = 0;
        if (csvStream == null || csvStream.Length == 0)
            throw new ArgumentException("CSV stream is empty.",
nameof(csvStream));

        using var reader = new StreamReader(csvStream);
        using var csv = new CsvReader(reader, csvReaderConfig);
        var records = new List<BoardGameCsvInputDto>();

        var categoriesProperties = typeof(BoardGameCsvInputDto)
            .GetProperties()
            .Where(prop => {
                var attribute =
prop.GetCustomAttribute<NameAttribute>();
                return attribute != null && attribute.Names.Any(n =>
n.StartsWith(categoryHeaderColumnPrefix,
StringComparison.OrdinalIgnoreCase));
            })
            .ToList();

        var rankProperties = typeof(BoardGameCsvInputDto)
            .GetProperties()
            .Where(prop => {
                var attribute =
prop.GetCustomAttribute<NameAttribute>();
                return attribute != null && attribute.Names.Any(n =>
n.StartsWith(rankHeaderColumnName, StringComparison.OrdinalIgnoreCase));
            });

        csv.Read();

        if (!csv.ReadHeader() || csv.HeaderRecord == null)

```

```

        throw new InvalidOperationException("CSV header is
null.");

        var categories =
GetCategoriesFromImportedData(csv.HeaderRecord!);

        foreach (var record in
csv.GetRecords<BoardGameCsvInputDto>())
        {
            records.Add(record);
        }

        var families = records
            .Select(r => r.Family)
            .Distinct()
            .Where(f => !string.IsNullOrEmpty(f))
            .Select(f => new ClassifierCsvReturnDto(f!))
            .ToList();

        var boardGames = records.Select(r => new
BoardGameCsvReturnDto(
            r.BggId,
            r.Name,
            r.Description,
            r.YearPublished,
            r.GameWeight <= 1 ? 1 : r.GameWeight,
            r.BayesAvgRating,
            r.AvgRating,
            r.MinPlayers,
            r.MaxPlayers,
            r.ComAgeRec is not null?
(int)Math.Round(r.ComAgeRec.Value) : null,
            r.LanguageEase.HasValue &&
                (int)Math.Round(r.LanguageEase.Value) >= 1 &&
                (int)Math.Round(r.LanguageEase.Value) <= 5
                ?
(LanguageDependence?) ((int)Math.Round(r.LanguageEase.Value))
                : null,
            r.BestPlayers,
            r.GoodPlayers,
            r.NumOwned,
            r.NumWant,
            r.NumWish,
            r.NumWeightVotes,
            r.MfgPlayTime,
            r.ComMinPlaytime,
            r.ComMaxPlaytime,
            r.MfgAgeRec,
            r.IsReimplementation,
            r.RankBoardGame,
            families.FirstOrDefault(f => f.Name == r.Family)?.Name,
            r.ImagePath)).OrderBy(r => r.BggId).ToList();

        var boardGameCategories = records
            .SelectMany(r =>
            {
                return categoriesProperties
                    .Where(p =>

```

```

        {
            var gameInCategory = (bool)p.GetValue(r)!;
            return gameInCategory;
        })
        .Select(p => new
BoardGameCategoryRelationCsvReturnDto(
            r.BggId,
            (categories.FirstOrDefault(
                c => $"{categoryHeaderColumnPrefix}{c.Name}"
== (
                    p.GetCustomAttribute<NameAttribute>()?
                    .Names
                    .FirstOrDefault(n =>
n.StartsWith(categoryHeaderColumnPrefix,
StringComparison.OrdinalIgnoreCase)))?.Name) ?? "",
                    rankProperties.FirstOrDefault(
                        rankProp =>
rankProp.GetCustomAttribute<NameAttribute>()?
                            .Names.FirstOrDefault(n =>
n.StartsWith(rankHeaderColumnName,
StringComparison.OrdinalIgnoreCase))?.Replace(rankHeaderColumnName, "")
==
p.GetCustomAttribute<NameAttribute>()?.Names.FirstOrDefault(n =>
n.StartsWith(categoryHeaderColumnPrefix,
StringComparison.OrdinalIgnoreCase))?.Replace(categoryHeaderColumnPrefix,
""))
                            ?.GetValue(r) as int? ?? 0));
                ))
                .ToList();

            return new BoardGamesCsvResultDto(boardGames, families,
categories, boardGameCategories);
        }

        public ClassifiersWithRelationsCsvResultDto
GetClassifiersWithRelationsImportedDataDefault(Stream csvStream)
        {
            return GetClassifiersWithRelationsImportedData(csvStream,
null);
        }

        public ClassifiersWithRelationsCsvResultDto
GetThemesImportedData(Stream csvStream)
        {
            return GetClassifiersWithRelationsImportedData(csvStream,
GetThemeNameFromHeader);
        }

        private ClassifiersWithRelationsCsvResultDto
GetClassifiersWithRelationsImportedData(
            Stream csvStream,
            Func<string, string>? classifierNameTransformSelector)
        {
            using var reader = new StreamReader(csvStream);
            using var csv = new CsvReader(reader, csvReaderConfig);

            csv.Read();
            csv.ReadHeader();

```

```

        if (csv.HeaderRecord == null)
            throw new InvalidOperationException("CSV header is
null.");

        var classifierRecords = csv.HeaderRecord.Where(r => r !=
bggIdHeaderColumnName).ToList();
        var classifiersDictionary = csv.HeaderRecord
            .Select((name, index) => new { name, index })
            .Where(r => r.name != bggIdHeaderColumnName)
            .ToDictionary(x => x.index, x => x.name);

        var classifierNames = classifierRecords;
        if (classifierNameTransformSelector != null)
        {
            classifierNames = classifierNames.Select(classifierName
=> classifierNameTransformSelector(classifierName)).ToList();
        }
        var classifiers = classifierNames.Select(p => new
ClassifierCsvReturnDto(p)).ToList();
        var boardGameWithClassifiersRelations = new
List<BoardGameClassifierRelationCsvReturnDto>();

        while (csv.Read())
        {
            int gameIdFieldIndex = csv.GetFieldIndex("BGGId");
            int gameId = csv.GetField<int>(gameIdFieldIndex);

            for (int i = 0; i < csv.HeaderRecord.Length; i++)
            {
                if (i != gameIdFieldIndex && csv.GetField<int>(i) ==
1)
                {
                    boardGameWithClassifiersRelations.Add(new(gameId,
classifiersDictionary[i]));
                }
            }
            return new(classifiers, boardGameWithClassifiersRelations);
        }

        private static List<ClassifierCsvReturnDto>
GetCategoriesFromImportedData(string[] headerRecords)
        {
            var categories = headerRecords.Where(headerRecords =>
headerRecords.StartsWith(categoryHeaderColumnNamePrefix)).ToList();
            return categories.Select(c => new
ClassifierCsvReturnDto(c.Split(":")[1])).ToList();
        }

        private static string GetThemeNameFromHeader(string header)
        {
            if (header.StartsWith(themeHeaderColumnNamePrefix))
                return header.Replace(themeHeaderColumnNamePrefix, "");

            return header;
        }
    }
}

```

## Б.2 Код методів для алгоритму рекомендації настільних ігор

```

public List<BoardGameRecommendationResultDto>
RecommendGames(List<int> favoriteGameIds, int topN = 10)
{
    var favorites = _games.Where(g =>
favoriteGameIds.Contains(g.Id)).ToList();
    if (!favorites.Any()) return [];

    var userVector = AverageVector(favorites);

    var favouriteProfile = BuildFavoriteProfile(favorites);

    return _games
        .Where(g => !favoriteGameIds.Contains(g.Id))
        .Select(g =>
        {
            var vector = GameToVector(g);
            var similarity = CosineSimilarity(userVector, vector);

            var explanation = BuildExplanation(g, favouriteProfile);
            return new BoardGameRecommendationResultDto(
                g.Id,
                new BoardGameRecommendationExplanationDto(
                    similarity,
                    explanation.Categories,
                    explanation.Mechanics,
                    explanation.Themes,
                    explanation.Subcategories,
                    explanation.SharedFamily));
        })
        .OrderByDescending(r => r.Explanation.SimilarityScore)
        .Take(topN)
        .ToList();
}

private Vector<double> AverageVector(List<BoardGame> games) =>
    games.Select(GameToVector).Aggregate((a, b) => a + b) /
games.Count;

private Vector<double> GameToVector(BoardGame g)
{
    var v = new List<double>();

    double weightTags = 0.31;
    var tagVector = new List<double>();
    tagVector.AddRange(_allCategories.Select(id =>
g.BoardGameCategories.Any(c => c.CategoryId == id) ? 1.0 : 0.0));
    tagVector.AddRange(_allMechanics.Select(id =>
g.BoardGameMechanics.Any(m => m.MechanicsId == id) ? 1.0 : 0.0));
    tagVector.AddRange(_allThemes.Select(id =>
g.BoardGameThemes.Any(t => t.ThemeId == id) ? 1.0 : 0.0));
    tagVector.AddRange(_allSubcategories.Select(id =>
g.BoardGameSubcategories.Any(s => s.SubcategoryId == id) ? 1.0 : 0.0));
    var normalizedTagVector = NormalizeGroup(tagVector);
    v.AddRange(ScaleGroup(normalizedTagVector, weightTags));
}

```

```

double weightAttributes = 0.12;
var attributes = new List<double>
{
    Normalize(g.GameComplexity, 1, 5),
    Normalize(g.RecommendedAge?.ManufacturerRecomended ?? 0, 3,
21),
    Normalize(g.PlayTime?.ManufacturerStatedPlayTime ?? 0, 5,
300),
    Normalize(g.Players?.MinPlayers ?? 1, 1, 20),
    Normalize(g.Players?.MaxPlayers ?? 1, 1, 20)
};
v.AddRange(ScaleGroup(NormalizeGroup(attributes),
weightAttributes));

double weightRatings = 0.15;
var ratings = new List<double>
{
    Normalize(g.BggData?.BggScore ?? 0, 0, 10),
    Normalize(g.BggData?.AverageRating ?? 0, 0, 10)
};
v.AddRange(ScaleGroup(NormalizeGroup(ratings), weightRatings));

double weightRank = 0.12;
var rank = new List<double> {
NormalizeReverse((g.BggData?.RankOverall == null || g.BggData.RankOverall <
1) ? 30000 : g.BggData.RankOverall.Value, 1, 30000) };
v.AddRange(ScaleGroup(rank, weightRank));

double weightPopularity = 0.10;
var popularity = new List<double> {
Normalize(g.BggData?.NumberOwned ?? 0, 0, 100000) };
v.AddRange(ScaleGroup(popularity, weightPopularity));

double weightYear = 0.12;
var year = new List<double> { Normalize(g.YearPublished, 1950,
2025) };
v.AddRange(ScaleGroup(year, weightYear));

double weightFamily = 0.03;
var family = new List<double>
{
    g.FamilyId != null ? 1.0 : 0.0,
    g.IsReimplementation ? 1.0 : 0.0
};
v.AddRange(ScaleGroup(family, weightFamily));

double weightLang = 0.05;
var lang = new List<double> {
Normalize((double)g.LanguageDependence, 0, 5) };
v.AddRange(ScaleGroup(lang, weightLang));

return DenseVector.OfEnumerable(v);
}

```

### Б.3 Код сервісу для релевантного сортування подій

```

public class EventsRecommendaionService : IEventsRecommendaionService
{
    private const double W_ONLINE = 0.7;
    private const double W_DISTANCE = 1.7;
    private const double W_FAVORITE = 2.0;
    private const double W_CLUB = 1.5;
    private const double W_REGISTERED = 1.0;
    private const double W_NOGAME = 0.3;

    private readonly IEventsRepository _eventsRepository;

    public EventsRecommendaionService(IEventsRepository
eventsRepository)
    {
        _eventsRepository = eventsRepository;
    }

    public async Task<List<EventRecommendationReturnDto>>
GetRelevantEventsAsync(
        List<int> favouriteGames, LocationCoordinatesDto
playerLocation, int page, int take, CancellationToken cancellation)
    {
        var events = await
_eventsRepository.GetActiveEventsWithDistancesAsync(
        playerLocation.Longitude, playerLocation.Latitude,
cancellation);
        var scoredEvents = events.Select(e => new
        {
            Details = e,
            Score = CalculateEventRecommendationScore(e,
favouriteGames)
        });

        return scoredEvents
            .Select(e =>
                new EventRecommendationReturnDto(
                    e.Details.Event,
                    e.Details.BoardGameName,
                    e.Details.GameClubName,
                    e.Details.OrganizerFullName,
                    e.Details.OrganizerNickname,
                    e.Details.Distance,
                    e.Details.RegisteredPlayers,
                    e.Score))
            .OrderByDescending(e => e.RecommendationScore)
            .Skip((page - 1) * take)
            .Take(take)
            .ToList();
    }

    private double
CalculateEventRecommendationScore(EventWithDetailsDto details,
List<int> favouriteGames)
    {
        double score = 0.0;
    }
}

```

```

    if (details.Event.IsOnline)
    {
        score += W_ONLINE;
    }

    if (details.Distance != null)
    {
        // f(dist) = 1 / (1 + dist), множимо на wDistance
        // Якщо dist = 0 => f(dist) = 1, чим більше dist, тим
менше вага.
        score += W_DISTANCE * (1.0 / (1 +
details.Distance.Value));
    }

    if (details.Event.BoardGameId != null)
    {
        var isFavourite =
favouriteGames.Contains(details.Event.BoardGameId.Value);

        if (isFavourite)
        {
            score += W_FAVORITE;
        }
    }
    else
    {
        score += W_NOGAME;
    }

    if (details.Event.GameClubId != null)
    {
        score += W_CLUB;
    }

    // g(RegisteredPlayers) = ln(1 + RegisteredPlayers)
    var playersFactor = Math.Log(1 + details.RegisteredPlayers);
    score += W_REGISTERED * playersFactor;

    return score;
}
}

```

## ДОДАТОК В

## Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

Дата звіту 6/3/2025

Дата редагування ---

Звіт не був оцінений

---

## Звіт подібності

### метадані

---

Назва організації  
**Kharkiv National University of Radio Electronics**

Заголовок  
**2025\_Б\_ПІ\_ПЗПІ-21-6\_Цвик\_В\_І\_скорочений**

Автор Науковий керівник / Експерт  
**Цвик Владислав Іванович Олена Олійник**

підрозділ  
**каф. ПІ**

### Обсяг знайдених подібностей

---

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.86%

1.86%

КП 1

2.54%

2.54%

КЦ

25

Довжина фрази для коефіцієнта подібності 2

12359

Кількість слів

105028

Кількість символів

### Тривога

---

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		1
Інтервали		0
Мікропробіли		0
Білі знаки		0
Парафрази (SmartMarks)	<u>a</u>	19

### Подібності за списком джерел

---



Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.


### 10 найдовших фраз

		Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ (ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ))
1	<a href="https://stackoverflow.com/questions/4181198/how-to-hash-a-password">https://stackoverflow.com/questions/4181198/how-to-hash-a-password</a>	22 0.18 %
2	<a href="https://www.yoqihosting.com/value-converters-comparers-entity-framework-core/">https://www.yoqihosting.com/value-converters-comparers-entity-framework-core/</a>	13 0.11 %
3	<a href="https://www.yoqihosting.com/value-converters-comparers-entity-framework-core/">https://www.yoqihosting.com/value-converters-comparers-entity-framework-core/</a>	13 0.11 %


## ДОДАТОК Г

### Слайди презентації




 МІНІСТЕРСТВО  
ОСВІТИ І НАУКИ  
УКРАЇНИ


 ХАРКІВСЬКИЙ  
НАЦІОНАЛЬНИЙ  
УНІВЕРСИТЕТ  
РАДІОЕЛЕКТРОНІКИ

**Програмна система для  
персоналізації досвіду  
гравців у настільні ігри.  
Серверна частина**


**Tabletop  
Connect**

Підготував:  
ст. гр. ПЗПІ-21-6 Цвик В.І.  
Керівник: проф. Дудар З. В.




12 червня 2025

Рисунок Г.1 – Слайд №1 (рисунок виконано самостійно)

## Мета роботи

- ❖ Створення повнофункціональної системи для задоволення потреб гравців у настільні ігри
- ❖ Впровадження рекомендацій подій та настільних ігор у систему
- ❖ Створення просунутого профілю гравця
- ❖ Реалізація тісного взаємозв'язку між усіма функціями системи
- ❖ Інтеграція з зовнішніми API (Google OAuth2, BGG, Nominatim, GeoNames)





**Tabletop  
Connect**

Рисунок Г.2 – Слайд №2 (рисунок виконано самостійно)

# Аналіз проблеми

- ❖ Підвищення популярності настільних ігор в Україні та світі
- ❖ Відсутність єдиної системи, що поєднує усі необхідні функції на гравців
- ❖ Обмеженість рекомендацій щодо вибору настільних ігор та подій на основі штучного інтелекту

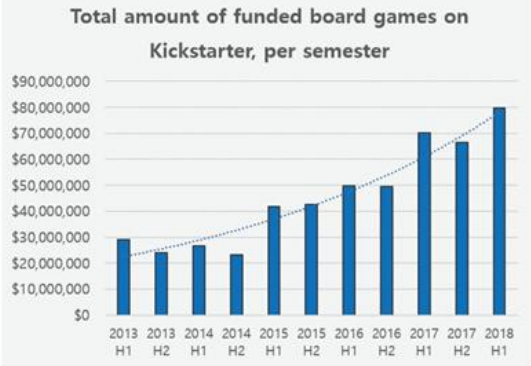
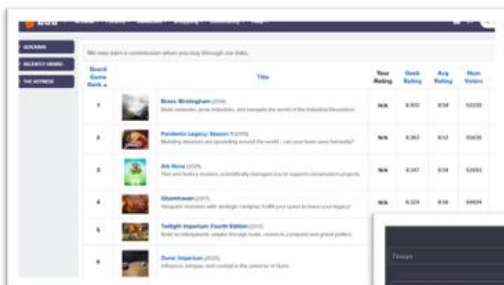
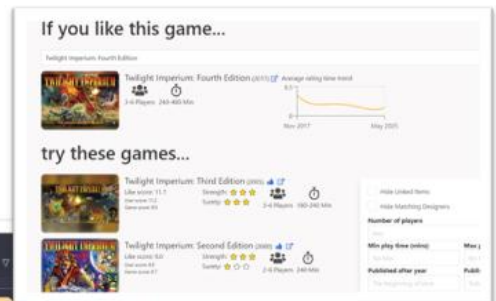


Рисунок Г.3 – Слайд №3 (рисунок виконано самостійно)

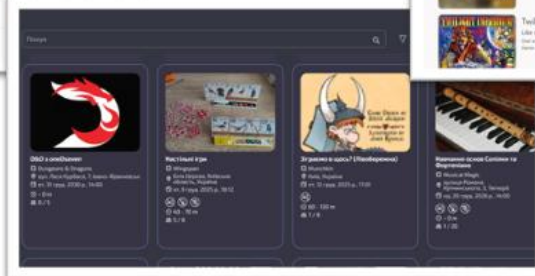
# Аналіз існуючих рішень



Board Game Geek



Try These Games



Geeker



Рисунок Г.4 – Слайд №4 (рисунок виконано самостійно)

# Аналіз існуючих рішень

Аналізований показник	Наша платформа	Board Game Geek	Try These Games	Geeker
Інтерактивність	Висока	Середня	Низька	Середня
Кількість ігор	Велика	Дуже велика	Середня	Середня
Персоналізація рекомендацій	Висока	Низька	Низька	Низька
Спільнота	Активна	Дуже активна	Не активна	Не активна
Огляди та аналізи ігор	Так	Так	Ні	Ні
Доступність багатомовності	Так	Так	Ні	Частково



Рисунок Г.5 – Слайд №5 (рисунок виконано самостійно)

# Опис системи

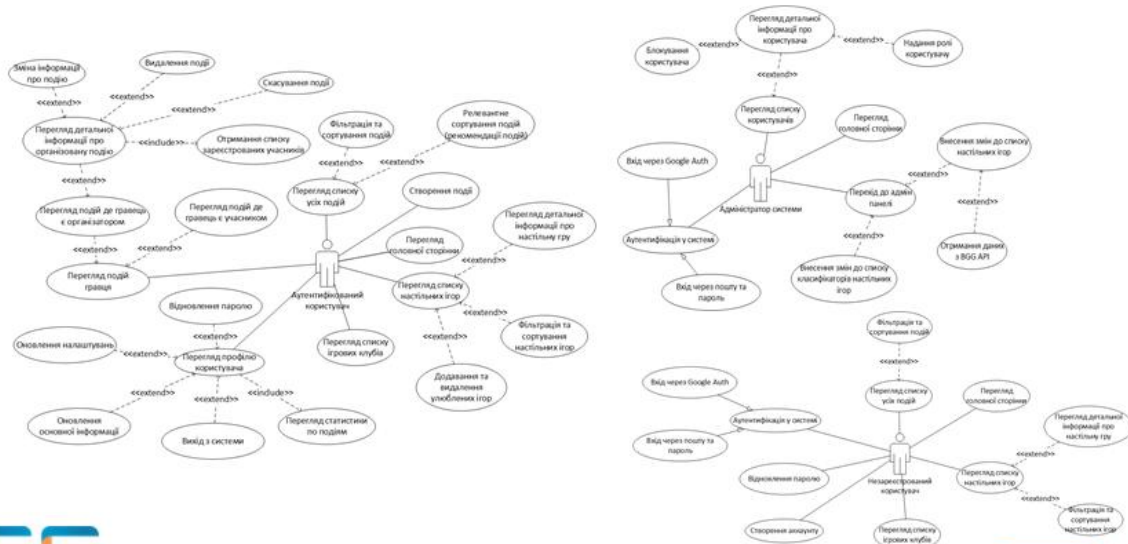


Рисунок Г.6 – Слайд №6 (рисунок виконано самостійно)

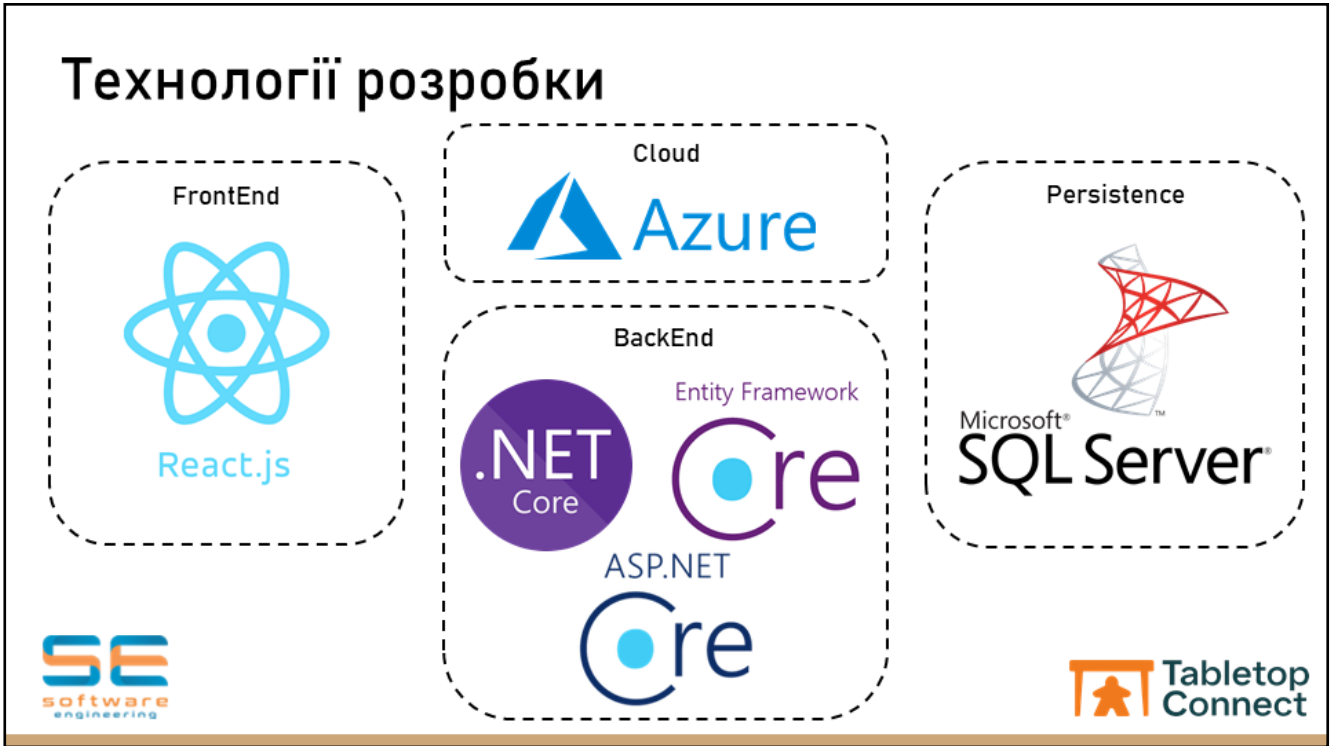


Рисунок Г.7 – Слайд №7 (рисунок виконано самостійно)



Рисунок Г.8 – Слайд №8 (рисунок виконано самостійно)

# Архітектура серверної частини

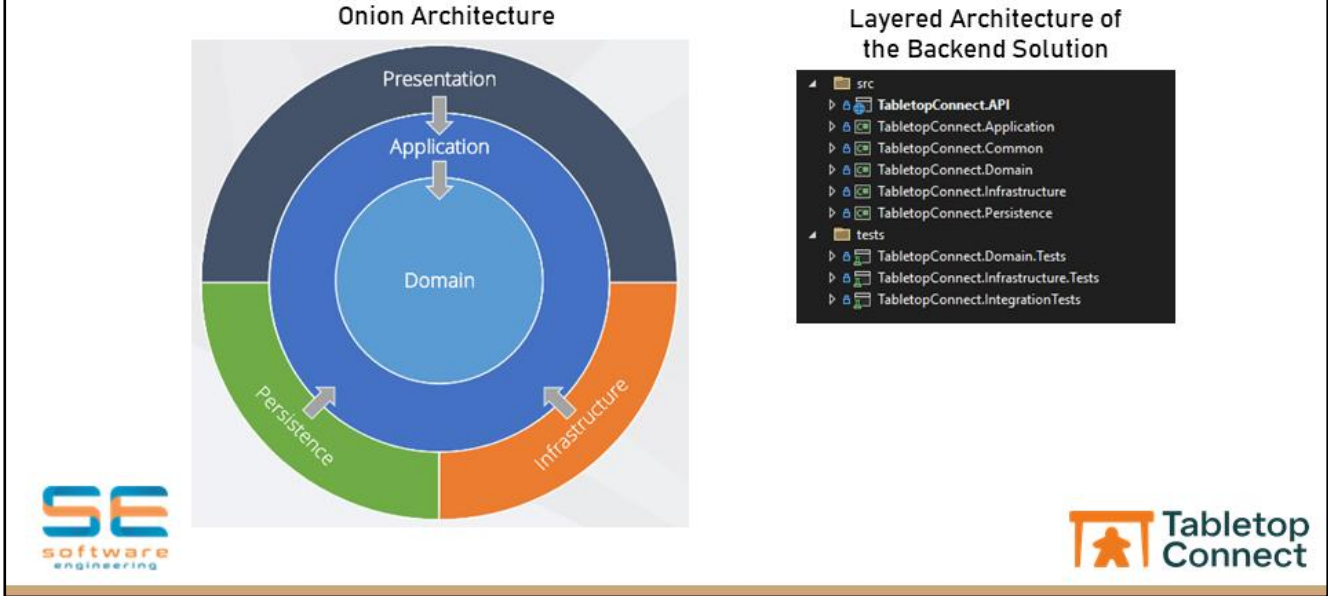


Рисунок Г.9 – Слайд №9 (рисунок виконано самостійно)

# Domain Driven Design

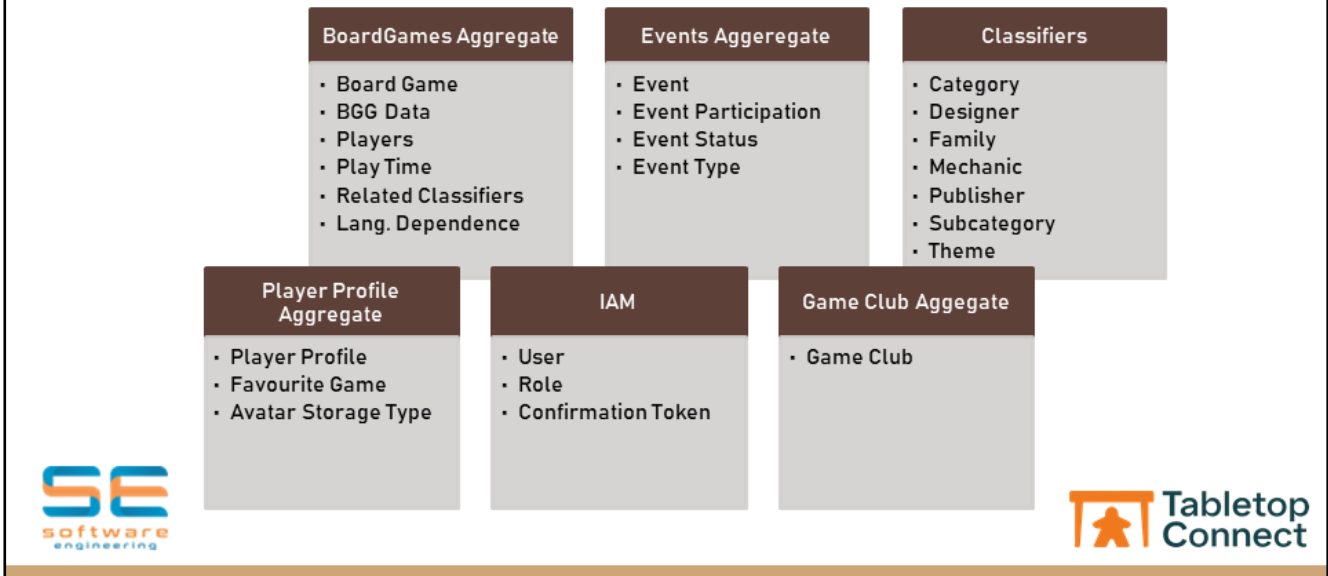


Рисунок Г.10 – Слайд №10 (рисунок виконано самостійно)



Рисунок Г.11 – Слайд №11 (рисунок виконано самостійно)



Рисунок Г.12 – Слайд №12 (рисунок виконано самостійно)

# Алгоритм релевантного сортування подій

## Events relevant soring method

Фактор	Позначення	Вага	Розрахунок	Призначення
Онлайн подія	$W_{online}$	0.7	Додається, якщо подія онлайн	Онлайн-події гнучкіші у доступі.
Відстань до події	$W_{distance}$	1.7	$1 / (1 + distance)$	Чим ближча подія, тим вищий бал.
Улюблена настільна гра	$W_{fav}$	2.0	Якщо гра є в списку улюблених	Перевага для улюблених ігор.
Без гри (відкрита подія)	$W_{nogame}$	0.3	Якщо гра не вказана	Низький бал, бо невідомо, чи буде цікаво.
Приязка до ігрового клубу	$W_{club}$	1.5	Додається, якщо є клуб	Клубні події вважаються якіснішими.
Кількість зареєстрованих гравців	$W_{reg}$	1.0	$\ln(1 + N)$	Більше людей — потенційно цікавіше.

```

1  private double CalculateEventRecommendationScore(EventWithDetails details, List<int> favouriteGames)
2  {
3      double score = 0.0;
4
5      if (details.Event.IsOnline)
6      {
7          score += W_ONLINE;
8      }
9
10     if (details.Distance != null)
11     {
12         // f(dist) = 1 / (1 + dist), monotonically decreasing
13         // Range dist = 0 to f(dist) = 1, non-linear dist, too much near.
14         score += W_DISTANCE * (1.0 / (1 + details.Distance.Value));
15     }
16
17     if (details.Event.BoardGameId != null)
18     {
19         var isFavourite = favouriteGames.Contains(details.Event.BoardGameId.Value);
20         if (isFavourite)
21         {
22             score += W_FAVORITE;
23         }
24     }
25     else
26     {
27         score += W_NOGAME;
28     }
29
30     if (details.Event.GameClubId != null)
31     {
32         score += W_CLUB;
33     }
34
35     // g(registeredPlayers) = ln(1 + registeredPlayers)
36     var playerFactor = Math.Log(1 + details.RegisteredPlayers);
37     score += W_REGISTERED * playerFactor;
38
39     return score;
40 }
    
```



$$score = W_{online} + W_{distance} + W_{fav} + W_{nogame} + W_{club} + W_{reg}$$



Рисунок Г.13 – Слайд №13 (рисунок виконано самостійно)

# Інтеграції: BoardGameGeekAPI

**Add / Update Board Game**

Search Board Game

Property	Existing	Updated
Name	Mare Mediterraneum	Mare Mediterraneum
Description	ancient land mediterranean player attempt satisfy unique victory condition trade war construction lavishly produce game contain wooden game component beautiful robust vinyl map player produce score different commodity trade city hope create income fit capital building produce attack fit warehouse good	in the ancient lands along the Mediterranean, players attempt to satisfy their unique victory conditions via trade, war and construction. This lavishly produced game contains lots of wooden game components and a beautiful roll-out vinyl map. Players produce a score of different commodities to trade with other cities in the hope of creating enough income to fit their capitals with buildings, produce artwork, and fit warehouses with goods.&#160;&#160;&#160;
Year Published	1989	1989
Complexity	3	3
Min Players	2	2
Max Players	6	6
Good Papers		4, 5, 6
Manufacturer Stated Play Time	240	240
Community Min Play Time	240	240
Community Max Play Time	240	240
Manufacturer Recommended Age	12	12
Community Recommended Age		13
BGG Score	5.548408793009	5.52888
Average Rating	6.35369997024360	6.4523
Base Overall	12166	16040
Language Dependence	Unknown	Some necessary in-game text
Categories		
Designers	Jean du Poff	Jean du Poff
Mechanics	Dice Rolling	Dice Rolling
Publishers	Historien Spiele Galerie (Historien Spielgalerie)	Historien Spiele Galerie (Historien Spielgalerie)
Themes	Civilization, Neutral	Civilization, Neutral
Family		

## BGG XML HTTP Client


```

1  public class BggHttpClient : IBggHttpClient
2  {
3      private readonly HttpClient _httpClient;
4      private readonly string _baseUrl = "https://boardgamegeek.com/api/v2/";
5
6      public BggHttpClient(HttpClient httpClient)
7      {
8          _httpClient = httpClient;
9      }
10
11     public async Task<int> SearchByManufacturerID(int manufacturerID, CancellationToken cancellationToken)
12     {
13         var url = $"{_baseUrl}/search?query={manufacturerID}&boardgame=";
14         var response = await _httpClient.GetAsync(url, cancellationToken);
15         response.EnsureSuccessStatusCode();
16
17         var url = $"{_baseUrl}/search?query={manufacturerID}&boardgame=";
18         var serializer = new JsonSerializer<BggGameSearchResponse>();
19         var gameResults = JsonSerializer.Deserialize<BggGameSearchResponse>(response.Content);
20         return gameResults.Items ?? [];
21     }
22
23     public async Task<BggGameDetails> GetBoardGameDetails(int BggID, CancellationToken cancellationToken)
24     {
25         var url = $"{_baseUrl}/thing/{BggID}?entity=";
26         var response = await _httpClient.GetAsync(url, cancellationToken);
27         response.EnsureSuccessStatusCode();
28
29         var url = $"{_baseUrl}/thing/{BggID}/game/{BggID}";
30         var serializer = new JsonSerializer<BggGameDetails>();
31         var gameDetails = JsonSerializer.Deserialize<BggGameDetails>(response.Content);
32         return gameDetails ?? BggGameDetails.Default;
33     }
34
35     public async Task<BggGameDetails> GetBoardGameDetails(int BggID, CancellationToken cancellationToken)
36     {
37         return null;
38     }
39 }
    
```



Рисунок Г.14 – Слайд №14 (рисунок виконано самостійно)

# Інтеграції: Електронна пошта



**Hello, vlcvik!**

We received a request to reset your password. Click the button below to set a new password for your account.


[Reset Your Password](#)

If the button above does not work, copy and paste the following link into your browser:

<https://localhost:5173/reset-password?confirmToken=MH15TAwC4C22bwt65GyvuUGR9SmsC206zK9CwU8u8e9k4>

If you did not request a password reset, please ignore this email. For security, do not share this link with anyone.

If you have any questions, contact our support team at [tabletopconnect@gmail.com](mailto:tabletopconnect@gmail.com)  
© 2025 Tabletop Connect. All rights reserved.



**Hello, vlcvik!**


Thank you for registering. To complete your sign-up, please confirm your email address by clicking the button below.

[Confirm Your Email](#)

If the button above does not work, copy and paste the following link into your browser:

<https://localhost:5001/confirm-email?confirmToken=nl-fa8fY0Y0wIR99o4OC23J06R0VWkZ1tRfbA8eBy>

If you did not create an account, no further action is required.  
Need help? Contact our support team at [tabletopconnect@gmail.com](mailto:tabletopconnect@gmail.com)  
© 2025 Tabletop Connect. All rights reserved.



**Hello!**

We want to let you know that the information about an event you are involved in has been updated. Please see the details below:

**Event Name:** Twilight impetum play upd  
**Board Game:** Krieg und Frieden  
**Event Dates:** 22-19-22:04.2025 – 22-19-24.05.2025  
**Price:** 10,00  
**Location:** Lviv Urban Hirnada, Lviv Raion, Lviv Oblast, Ukraine  
**Status:** Ongoing

[View Event Details](#)

If the button above does not work, copy and paste the following link into your browser:

<https://localhost:5001/>





Рисунок Г.15 – Слайд №15 (рисунок виконано самостійно)

# Інтеграції: Геопошук

### Geo Names

```
200 Response body
{
  "id": "1668234",
  "code": "TZ",
  "name": "Talism"
},
{
  "id": "1662289",
  "code": "TZ",
  "name": "Tanzania"
},
{
  "id": "686792",
  "code": "UA",
  "name": "Ukraine"
}
},
{
  "id": "703442",
  "name": "Lviv",
  "countryId": "686792",
  "country": "Ukraine",
  "countryCode": "UA"
},
{
  "id": "711386",
  "name": "Brvory",
  "countryId": "686792",
  "country": "Ukraine",
  "countryCode": "UA"
}
},
```

### Nominatim (Open Street Maps)

```
Code Details
200 Response body
{
  "osmid": "2832289",
  "osstype": "relation",
  "class": null,
  "longitude": 24.8115925,
  "latitude": 49.841595,
  "shortName": "Lviv",
  "fullName": "Lviv, Lviv Urban Hirnada, Lviv Raion, Lviv Oblast, Ukraine"
}
},
```

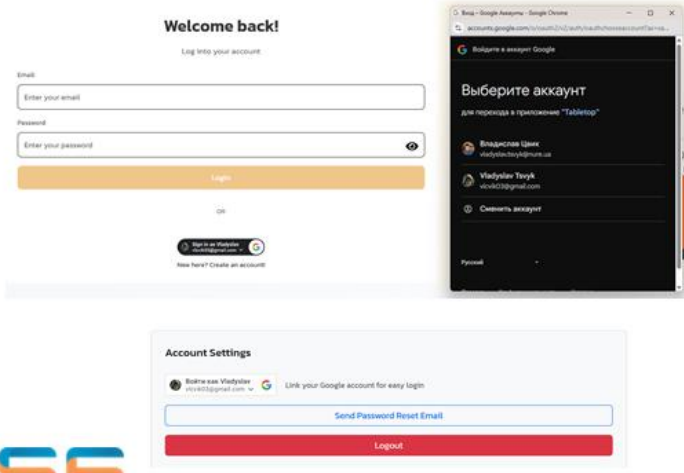
### Nominatim (Open Street Maps)

```
Code Details
200 Response body
{
  "osmid": "2832289",
  "osstype": "relation",
  "class": null,
  "longitude": 24.8115925,
  "latitude": 49.841595,
  "shortName": "Lviv",
  "fullName": "Lviv, Lviv Urban Hirnada, Lviv Raion, Lviv Oblast, Ukraine"
}
},
```



Рисунок Г.16 – Слайд №16 (рисунок виконано самостійно)

# Інтеграції: Google OAuth2



## Google Auth Service

```
7 internal class GoogleAuthService : IGoogleAuthService
8
9
10 public async Task<GoogleAuthDto?> ValidateGoogleTokenAsync(string token)
11 {
12     try
13     {
14         var payload = await GoogleJsonWebSignature.ValidateAsync(token);
15         return new GoogleAuthDto(
16             payload.Subject,
17             payload.Email,
18             payload.EmailVerified,
19             payload.GivenName,
20             payload.FamilyName,
21             payload.Locale?.Split('-')[0],
22             payload.Picture);
23     }
24     catch (InvalidOperationException)
25     {
26         return null;
27     }
28 }
```



Рисунок Г.17 – Слайд №17 (рисунок виконано самостійно)

# Генерація звітів

The screenshot shows a report titled 'Twilight imperium play upd' with the following data:

#	Full Name	Phone	Email	Registered
1	Vladyslav Tryk	Unknown Phone	vivuk03@gmail.com	22.04.2025 22:30
2	Irina Posukan	Unknown Phone	posukan@gmail.com	22.04.2025 20:00

The screenshot shows a 'Player Events Report' for 'Player: Vladyslav Tryk'. It lists organized and participated events:

- Organized Events:** NewEventOnline (20.05.2025 20:15 - 21.05.2025 20:15), Game: Federation & Empire, Participants: 0, Status: Ended; Twilight imperium play upd (22.04.2025 22:19 - 24.05.2025 22:19), Game: Krieg und Frieden, Participants: 1, Status: Cancelled; EventMyIv (20.04.2025 20:15 - 19.05.2025 20:15), Game: Krieg und Frieden, Participants: 0, Status: Ended.
- Participated Events:** Twilight imperium play upd (22.04.2025 22:19 - 24.05.2025 22:19), Game: Krieg und Frieden, Participants: 1, Status: Cancelled.
- Most Popular Games:** Krieg und Frieden - 2 events, Federation & Empire - 1 events.



Рисунок Г.18 – Слайд №18 (рисунок виконано самостійно)

# Імпорт даних про настільні ігри

EggImportService

- ❖ Настільних ігор: > 20,000
- ❖ Класифікаторів (категорій, тематики, жанри...): > 5,000
- ❖ Відповідностей між іграми та класифікаторами: > 200,000

```

18 internal class BaseBoardGameImportService : BaseBoardGameImportService
19 {
20     private readonly IConfiguration configuration = new ConfigurationBuilder()
21         .AddJsonFile("appsettings.json", optional: false)
22         .Build();
23     private readonly string boardGameColumnName = "BoardGame";
24     private readonly string categoryHeaderColumnName = "Category";
25     private readonly string boardGamePrefix = "BoardGame";
26     private readonly string boardGameSuffix = "BoardGame";
27 }
28
29 public BoardGameImportData GetBoardGameImportData(Stream csvStream)
30 {
31     var streamPosition = 0;
32     if (csvStream == null || csvStream.Length == 0)
33         throw new ArgumentException("CSV stream is empty.", nameof(csvStream));
34
35     using var reader = new StreamReader(csvStream);
36     using var csv = new CsvReader(reader, configuration);
37     var records = new List<BoardGameImportData>();
38     var categoryProperties = typeof(BoardGameImportData)
39         .GetProperties()
40         .Where(p => p.Name.StartsWith("Category"));
41     var attributes = prep.GetCustomAttributes(typeof(Attribute), true)
42         .Where(a => a.Name.EndsWith("CategoryHeaderColumnName", StringComparison.OrdinalIgnoreCase));
43     var record = new BoardGameImportData();
44     foreach (var attribute in attributes)
45     {
46         record.SetAttribute(attribute.Name, attribute);
47     }
48     var record = new BoardGameImportData();
49     foreach (var attribute in attributes)
50     {
51         record.SetAttribute(attribute.Name, attribute);
52     }
53     return records;
54 }
55
56 private void AddRecord(BoardGameImportData record)
57 {
58     if (record.BoardGameId == null)
59         throw new ArgumentException("BoardGameId is null.", nameof(record));
60     var category = GetCategory(record.CategoryHeaderColumnName, record.BoardGameId);
61     foreach (var record in csv.GetRecords(typeof(BoardGameImportData)))
62     {
63         records.Add(record);
64     }
65     var families = records
66         .GroupBy(r => r.Family)
67         .Select(g => new Family(g.Key, g.ToList()))
68         .ToList();
69     foreach (var family in families)
70     {
71         records.AddRange(family.Records);
72     }
73 }

```



Рисунок Г.19 – Слайд №19 (рисунок виконано самостійно)

# Тестування: мануальне

Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	У базі даних є створена подія, а користувач, який її створив, авторизований	Подію можна скасувати через API	Пройдено
Скасування події			
№	Опис випадку	Очікуваний результат	Висновок
1	Надіслати POST-запит до /api/Events/{id}/cancel автором події	Сервер повертає HTTP-статус 200 (OK), подія позначається як скасована	Пройдено
2	Надіслати POST-запит до /api/Events/{id}/cancel, якщо подія вже скасована	Сервер повертає HTTP-статус 400 (Bad Request) з валідаційною помилкою	Пройдено
3	Надіслати POST-запит до /api/Events/{id}/cancel, де id — неіснуючої події	Сервер повертає HTTP-статус 404 (Not Found) з повідомленням "Подію не знайдено"	Пройдено
4	Надіслати POST-запит до /api/Events/{id}/cancel користувачем, який не є автором	Сервер повертає HTTP-статус 400 (Bad Request) з повідомленням "Немає прав на скасування цієї події"	Пройдено



Рисунок Г.20 – Слайд №20 (рисунок виконано самостійно)

# Тестування: автоматизоване

## Unit-тестування

```

120 [Theory]
121 [InlineData(false, "2023-05-20", "2023-05-20", "2023-05-20", EventStatus.Planned)]
122 [InlineData(false, "2023-05-20", "2023-05-20", "2023-05-20", EventStatus.Ongoing)]
123 [InlineData(false, "2023-05-20", "2023-05-20", "2023-05-21", EventStatus.Finished)]
124 [InlineData(true, "2023-05-20", "2023-05-20", "2023-05-20", EventStatus.Finished)]
125 [Fact]
126 public void GetEventStatus_Should_ReturnCorrectStatus()
127 {
128     // Arrange
129     var evt = Event.Create("Test Event", null, null, null,
130         DateTime.UtcNow, DateTime.UtcNow.AddDays(1),
131         null, EventType.Meeting, null, null, 1, null);
132     if (IsCancelled)
133     {
134         evt.Cancel();
135     }
136     DateTimeOffset currentTime = DateTimeOffset.UtcNow;
137     // Act
138     var status = evt.GetEventStatus(currentTime);
139     // Assert
140     Assert.Equal(EventStatus.Cancelled, status);
141 }
142 [Fact]
143 public void Cancel_Should_SetIsCancelled_IfNotCancelled()
144 {
145     // Arrange
146     var evt = Event.Create("Cancelable Event", null, null, null,
147         DateTime.UtcNow.AddDays(1), DateTime.UtcNow.AddDays(2), false,
148         null, EventType.Meeting, null, null, 1, null);
149     // Act
150     evt.Cancel();
151     // Assert
152     Assert.True(evt.IsCancelled);
153 }

```

## Інтеграційне тестування (Сервіс + Репозиторії)

```

1 // Інтеграційні тести
2 public class EventsServiceTests
3 {
4     [Fact]
5     public async Task GetEventParticipantsDetails_ReturnsExpectedResult()
6     {
7         // Arrange
8         var fixture = new EventsServiceFixture();
9         var user = User.RegisterRegular("email@gmail.com", "pass");
10        fixture.DbContext.SetUsers(user);
11        fixture.DbContext.SaveChanges();
12        var player = PlayerProfile.CreateRegular(playerId, "John");
13        fixture.DbContext.SetPlayerProfiles(player);
14        fixture.DbContext.SaveChanges();
15        var eventParticipation = new EventParticipation(eventId, playerId, new DateTimeOffset(2023, 1, 1));
16        fixture.DbContext.SetEventParticipations(eventParticipation);
17        fixture.DbContext.SaveChanges();
18        // Act
19        var result = await fixture.EventsService.GetEventParticipantsDetailsAsync(eventParticipationId, cancellationToken);
20        // Assert
21        Assert.Single(result);
22        Assert.Equal("John", result[0].Nickname);
23        Assert.Equal(1, result[0].Id);
24    }
25 }

```



Рисунок Г.21 – Слайд №21 (рисунок виконано самостійно)

# Публікація результатів

УДК 004.89  
**ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ АЛГОРИТМІВ ДЛЯ ПЕРСОНАЛІЗАЦІЙНИХ РЕКОМЕНДАЦІЙ НАСТІЛЬНИХ ІГОР**  
 Цивик В.І., Іосукан І.І., Іванова В.М.  
 email: vbyduval@ua.fm@yandex.ua, iova.rosalana@yandex.ua

Університет Харківського національного університету радіоелектроніки, каф. ПІ  
 м. Харків, Україна

This work is devoted to the analysis of recommendation algorithms for board games based on the content-based approach. The study explores various recommendation methods, including vector-based similarity calculations and hybrid techniques to improve accuracy. The problem of cold start and data normalization is addressed, ensuring better personalization for users. Additionally, the effectiveness of the proposed system is assessed using precision, recall, and F1-score metrics. The research highlights the importance of combining multiple filtering techniques for enhancing recommendation relevance.

На сьогоднішній день сучасні програмні системи все частіше включають в себе рекомендаційні підсистеми для персоналізації контенту та підвищення залученості користувачів. Вони використовуються в різних сферах, зокрема в електронній комерції, соціальних мережах та освітніх платформах. В останні роки зростає інтерес до впровадження рекомендаційних алгоритмів у сфері настільних ігор, що обумовлено зростаючим різноманітністю ігрового контенту та необхідністю допомогти користувачам у виборі відповідних ігор. Традиційні методи пошуку нових настільних ігор, такі як рейтинги або відгуки, мають певні обмеження, оскільки вони не враховують індивідуальні вподобання гравців та їхній попередній досвід.

Основною проблемою у створенні такої системи є велика кількість параметрів, що характеризують ігри, а також необхідність ефективного аналізу вподобань користувачів. У цьому дослідженні розглядаються методи побудови рекомендаційних систем для настільних ігор, зокрема підхід, заснований на контентній фільтрації (Content-based Filtering) [1], а також можливість поєднання різних алгоритмів для підвищення точності рекомендацій.

Серед основних методів для побудови рекомендаційних систем можна виділити наступні:  
 – контентна фільтрація (Content-Based Filtering) – рекомендація об'єктів (настільних ігор) на основі даних характеристик та схожості з тим, що вже сподобалося користувачу;  
 – колаборативна фільтрація (Collaborative Filtering) – рекомендації, що базуються на даних про взаємодію користувачів з об'єктами, не враховуючи їх характеристики;

– гібридні методи (Hybrid) – поєднання контентної та колаборативної фільтрації для забезпечення більшої релевантності рекомендацій.

При відсутності великої кількості даних про взаємодію користувачів варто використовувати Content-Based Filtering [2], наприклад, у випадку пошуку системи, де поточному користувачу є лише дані про настільні ігри та їх характеристики. У такій програмній системі варто передбачити можливість оцінювати об'єкти взаємодію користувачами. Отримавши такі дані можна застосувати методи Collaborative Filtering і об'єднати їх з попередніми алгоритмами сформувати гібридну рекомендаційну систему, яка б поєднувала обидва підходи.

При застосуванні Content-Based Filtering потрібно враховувати такі властивості настільних ігор, як популярність, складність, ігрові механіки, тематика, категорії, новині обмеження, тривалість партії, кількість гравців, залученість від зміни мови (кількість мовної/текстової інформації). Також необхідно врахувати попередній досвід гравця, а саме: зі скількох ігор, які сподобалися користувачу. Одним з джерел даних, яке б містило може стати сервіс BoardGameGeek, де зібрано дано більше ніж про 20000 настільних ігор [3].

Кожна настільна гра може бути представлена у вигляді векторного профілю, що містить її характеристики. Основне завдання алгоритму – порівняти ці профілі з іграми, які вже сподобалися користувачу, та вибрати найбільш схожі варіанти. Для цього спочатку будеться матриця ознак, де кожен рядок відображає певну гру, а стовпці містять значення таких параметрів, як складність, тематика, механіки, категорії, тривалість партії, популярність тощо.

Одним із найпоширеніших способів обчислення схожості між іграми є використання косинусової подібності (cosine similarity) [4] між певним вектором представленим Косинусом подібності визначає ступінь схожості між двома векторами за допомогою косинуса кута між ними. Чим більше значення до 1, тим більше схожі ігри. Інший підхід – евклідова відстань, яка визначає відстань між двома різними даними ряди у багатовимірному просторі ознак. Для категоризаційних ознак (наприклад, механіки, тематика, жанри) можна використовувати методи One-Hot Encoding або TF-IDF для текстової інформації, що дозволяє будувати багатовимірне представлення гри та порівнювати їх між собою.

Важливим аспектом у впровадженні чисельних параметрів, таких як рейтинг, складність чи тривалість партії, оскільки вони мають різні масштаби. Наприклад, рейтинг може мати значення від 1 до 10, тоді як тривалість партії – від 5 до 300 хвилин. Щоб уникнути домінування окремих параметрів, застосовується міні-макс нормалізація або z-score стандартизація.

Одним із викликів цього підходу є проблема холодного старту, коли у нового користувача немає історії вподобань, і система не має достатньо

інформації для формування рекомендацій. У таких випадках можна використовувати Content-Based Filtering з іншими методами, наприклад, використовувати популярні ігри з високою релевантністю або враховувати схожість з профілями інших користувачів (елементи колаборативної фільтрації).

Після впровадження рекомендаційної системи варто вивчити, наскільки добре розроблений алгоритм прогнозу користувачів рекомендацій ігор. Для цього можна використовувати такі метрики, як Precision, Recall та F1-score, які аналізують, яка частка рекомендованих ігор дійсно цікава користувачу (Precision) і наскільки повною мірою система покриває всі можливі рекомендації (Recall). F1-score комбінує ці два показники, дозволяючи більш повно оцінити модель.

Для реалізації описаних вище алгоритмів Content-Based Filtering існує декілька підходів і технологій, які можуть бути використані.  
 По-перше, можна використовувати мову Python, бібліотеки для машинного навчання та роботи з аналізом даних, такі як Scikit-learn, Pandas та NumPy.  
 По-друге, для платформі NET існують бібліотеки ML.NET та Mahout для роботи з машинним навчанням та великими чисельними даними.  
 По-третє, існують хмарні (cloud) рішення, такі як Google Cloud AI чи Azure AI для роботи та навчання моделей машинного навчання [5].

Отже, при проектуванні програмної системи рекомендацій настільних ігор варто ретельно підійти до проектування та вибору алгоритму. У випадку відсутності даних про взаємодію користувачів з об'єктами варто використовувати Content-Based Filtering, який у процесі розвитку системи можна поєднувати з Collaborative Filtering.

- Список використаних джерел:
1. Ma, Tejashri Sharda Phalle, Content Based Filtering And Collaborative Filtering: A Comparative Study, Journal of Advanced Zoology, 2024, Vol. 45, P. 96-100.
  2. Marwa Hussein Mohamed, Mohamed Khalifa, Mohamed Haan Ibrahim, Recommender Systems Challenges and Solutions Survey, International Conference on Innovative Trends in Computer Engineering, URL: <https://doi.org/10.1109/ITCE.2019.8646651> (date of access: 04.03.2025).
  3. Phil Woodward, Sam Woodward, Mining the Boardgamegeek, Significance, 2019, Vol. 16, Issue 2, P. 24-29.
  4. Muhammad Fahd, Dewi Handayani, Content-based filtering using cosine similarity algorithm for alternative selection on training programs, Journal of Soft Computing Exploration, 2023, Vol. 4, P. 204 - 212.
  5. Манук О., Кравець Н. Дослідження методів створення персоналізованих програмних систем у Azure, Computer Systems and Information Technologies, 2022, №2(1), С. 38-47.



Рисунок Г.22 – Слайд №22 (рисунок виконано самостійно)

## Підсумки

- ❖ Розроблена система цілком відповідає початковому баченню та поставленим вимогам
- ❖ Можливість подальшої монетизації та популяризації
- ❖ Створена система може бути повноцінно використана після налаштування інфраструктури
- ❖ Подальший розвиток: Mobile (PWA), більше аналітики, просунуті рекомендації



Рисунок Г.23 – Слайд №23 (рисунок виконано самостійно)

Дякуємо за увагу!



Рисунок Г.24 – Слайд №24 (рисунок виконано самостійно)

## ДОДАТОК Д

Копії тез доповіді на 29-му міжнародному молодіжному форумі

«Радіоелектроніка та молодь У ХХІ столітті»

УДК 004.89

### ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ АЛГОРИТМІВ ДЛЯ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ НАСТІЛЬНИХ ІГОР

Цвик В.І., Посукан І. І., Ляпота В.М.

email: [vladyslav.tsvyk@nure.ua](mailto:vladyslav.tsvyk@nure.ua), [inna.posukan@nure.ua](mailto:inna.posukan@nure.ua), [vitaliy.lyapota@nure.ua](mailto:vitaliy.lyapota@nure.ua)

Харківський національний університет радіоелектроніки, каф. ПІ

м. Харків, Україна

This work is devoted to the analysis of recommendation algorithms for board games based on the content-based approach. The study explores various recommendation methods, including vector-based similarity calculations and hybrid techniques to improve accuracy. The problem of cold start and data normalization is addressed, ensuring better personalization for users. Additionally, the effectiveness of the proposed system is assessed using precision, recall, and F1-score metrics. The research highlights the importance of combining multiple filtering techniques for enhancing recommendation relevance.

На сьогоднішній день сучасні програмні системи все частіше включають в себе рекомендаційні підсистеми для персоналізації контенту та підвищення залученості користувачів. Вони використовуються в різних сферах, зокрема в електронній комерції, пошукових сервісах, соціальних мережах та освітніх платформах. В останні роки зростає інтерес до впровадження рекомендаційних алгоритмів у сфері настільних ігор, що обумовлено значною різноманітністю ігрового контенту та необхідністю допомагати користувачам у виборі відповідних ігор. Традиційні методи пошуку нових настільних ігор, такі як рейтинги або відгуки, мають певні обмеження, оскільки вони не враховують індивідуальні вподобання гравців та їхній попередній досвід.

Основною проблемою у створенні такої системи є велика кількість параметрів, що характеризують ігри, а також необхідність ефективного аналізу вподобань користувачів. У цьому дослідженні розглядаються методи побудови рекомендаційних систем для настільних ігор, зокрема підходи, засновані на контентній фільтрації (Content-based Filtering) [1], а також можливості поєднання різних алгоритмів для підвищення точності рекомендацій.

Серед основних методів для побудови рекомендаційних систем можна виділити наступні:

– контентна фільтрація (Content-Based Filtering) – рекомендація об'єктів (настільних ігор) на основі їхніх характеристик та схожості з тим, що вже сподобалося користувачу;

– колаборативна фільтрація (Collaborative Filtering) – рекомендації, що базується на даних про взаємодію користувачів з об'єктами, не враховуючи їх характеристик;

– гібридні методи (Hybrid) – поєднання контентної та колаборативної фільтрації для забезпечення більшої релевантності рекомендації.

При відсутності великої кількості даних про взаємодію користувачів варто використовувати Content-Based Filtering [2], наприклад, у випадку запуску системи, де початковими даними є лише дані про настільні ігри та їх характеристики. У такій програмній системі варто передбачити можливість оцінювати об'єкти взаємодії користувачами. Отримавши такі дані можна застосувати методи Collaborative Filtering і об'єднавши їх з попередніми алгоритмами сформувати гібридну рекомендаційну систему, яка б поєднувала обидва підходи.

При застосуванні Content-Based Filtering потрібно враховувати такі властивості настільних ігор, як популярність, складність, ігрові механіки, тематика, категорії, вікові обмеження, тривалість партії, кількість гравців, залежність від знання мови (кількість важливої текстової інформації). Також необхідно врахувати попередній досвід гравця, а саме зі список ігор, які сподобалися користувачу. Одним з джерел даних, яке б містило може стати сервіс BoardGameGeek, де зібрано дані більше ніж про 20000 настільних ігор [3].

Кожна настільна гра може бути представлена у вигляді векторного профілю, що містить її характеристики. Основне завдання алгоритму – порівняти ці профілі з іграми, які вже сподобалися користувачу, та знайти найбільш схожі варіанти. Для цього спочатку будується матриця ознак, де кожен рядок відповідає певній грі, а стовпці містять значення таких параметрів, як складність, тематика, механіки, категорії, тривалість партії, популярність тощо.

Одним із найпоширеніших способів обчислення схожості між іграми є використання косинусної подібності (cosine similarity) [4] між їхніми векторними представленнями. Косинусна подібність визначає ступінь схожості між двома векторами за допомогою косинуса кута між ними. Чим ближче значення до 1, тим більше схожі ігри. Інший підхід – евклідова відстань, яка визначає наскільки близько розташовані дві гри у багатовимірному просторі ознак. Для категоріальних ознак (наприклад, механіки, тематика, жанри) можна використовувати методи One-Hot Encoding або TF-IDF для текстових описів, що дозволяє будувати багатовимірне представлення гри та порівнювати їх між собою.

Важливим аспектом є нормалізація числових параметрів, таких як рейтинг, складність чи тривалість партії, оскільки вони мають різні масштаби. Наприклад, рейтинг може мати значення від 1 до 10, тоді як тривалість партії – від 5 до 300 хвилин. Щоб уникнути домінування окремих параметрів, застосовується min-max нормалізація або z-score стандартизація.

Одним із викликів цього підходу є проблема холодного старту, коли у нового користувача немає історії вподобань, і система не має достатньо інформації для формування рекомендацій. У таких випадках можна комбінувати Content-Based Filtering з іншими методами, наприклад, використовувати популярні ігри з високими рейтингами або враховувати схожість з профілями інших користувачів (елементи колаборативної фільтрації).

Після впровадження рекомендаційної системи варто визначити, наскільки добре розроблений алгоритм пропонує користувачам релевантні ігри. Для цього можна використовувати такі метрики, як Precision, Recall та F1-score, які

аналізують, яка частка рекомендованих ігор дійсно цікава користувачу (Precision) і наскільки повно система покриває всі можливі релевантні варіанти (Recall). F1-score комбінує ці два показники, дозволяючи збалансовано оцінити модель.

Для реалізації описаних вище алгоритмів Content-Based Filtering існує декілька підходів і технологій які можуть бути використані.

По-перше, можна використовувати мову Python, бібліотеки для машинного навчання та роботи з аналізом даних, такі як Scikit-learn, Pandas та NumPy.

По-друге, для платформи .NET існують бібліотеки ML.NET та MathNet для роботи з машинним навчанням та великими числовими даними.

По-третє, існують хмарні (cloud) рішення, такі як Google Cloud AI чи Azure AI для розробки та навчання моделей машинного навчання [5].

Отже, при проектуванні програмної системи рекомендації настільних ігор варто ретельно підійти до проектування та вибору алгоритмів. У випадку відсутності даних про взаємодію користувача з об'єктами варто використовувати Content-Based Filtering, який у процесі розвитку системи можна скомбінувати з Collaborative Filtering.

Список використаних джерел:

1. Ms. Tejashri Sharad Phalle. Content Based Filtering And Collaborative Filtering: A Comparative Study. Journal of Advanced Zoology. 2024. Vol. 45. P. 96–100.
2. Marwa Hussien Mohamed, Mohamed Khafagy, Mohamed Hasan Ibrahim. Recommender Systems Challenges and Solutions Survey. International Conference on Innovative Trends in Computer Engineering. URL: <http://dx.doi.org/10.1109/ITCE.2019.8646645> (date of access: 04.03.2025).
3. Phil Woodward, Sam Woodward. Mining the Boardgamegeek. Significance. 2019. Vol. 16, Issue 5. P. 24–29.
4. Muhammad Falah, Dewi Handayani. Content-based filtering using cosine similarity algorithm for alternative selection on training programs. Journal of Soft Computing Exploration. 2023. Vol. 4. P. 204 – 212.
5. Макеєв О., Кравець Н. Дослідження методів створення сервісо-орієнтованих програмних систем у Azure. Computer Systems and Information Technologies. 2023. №2(11). С. 38–47.