ДОДАТОК А

Специфікація ПЗ

А.1 Вступ

А.1.1 Мета документа

Даний документ є формалізованим описом вимог до програмної системи для планування мультитранспортних поїздок із пересадками та продажу квитків. Він визначає функціональність, обмеження, інтерфейси та інші ключові аспекти системи.

Документ призначений для:

- розробників – як основа для проектування та реалізації;
- тестувальників – для створення тестових сценаріїв;
- замовників – для перевірки відповідності очікуваним вимогам;
- менеджерів проекту – для планування робіт.

А.1.2 Область застосування

Для пасажирів:

- пошук оптимальних маршрутів із врахуванням часу, вартості та кількості пересадок;
- купівля квитків через інтегровані платіжні системи;
- отримання інформації про зміни в реальному часі (затримки, скасування рейсів).

Для перевізників (транспортних компаній):

- управління розкладами, транспортом та персоналом;
- моніторинг технічного стану транспортних засобів;
- формування звітів з аналітикою завантаженості маршрутів.

Для туристичних агенцій та агрегаторів:

- інтеграція через API для продажу комбінованих квитків.

А.1.3 Визначення, акроніми та скорочення

API (Application Programming Interface) – інтерфейс для взаємодії між компонентами системи.

СУБД (Система управління базами даних) – програмне забезпечення для роботи з даними (PostgreSQL).

GDS (Global Distribution System) – система бронювання авіаквитків (наприклад, Amadeus, Sabre).

TMS (Transportation Management System) – система управління транспортними операціями.

HTTP/REST – протокол передачі даних та архітектурний стиль API.

JWT (JSON Web Token) – метод автентифікації користувачів.

PCI DSS – стандарт безпеки для платіжних систем.

А.2 Загальний опис

А.2.1 Перспектива продукту

Система є централізованою платформою, яка об’єднує дані різних транспортних компаній (автобуси, поїзди, літаки) та надає єдиний інтерфейс для:

- планування маршрутів із пересадками;
- продажу квитків;
- управління ресурсами перевізників.

А.2.2 Функції системи

А.2.2.1 Модуль планування маршрутів

Пошук оптимальних маршрутів за часом поїздки, вартістю вартістю, кількістю пересадок, врахування динамічних змін (затримки, скасування). Візуалізація маршрутів на мапі (інтеграція з Google Maps API).

А.2.2.2 Платіжний модуль

- підтримка різних методів оплати: кредитні картки (Stripe), електронні гаманці (Privat24, PayPal), криптовалюти (BTCPay);
- генерація цифрових квитків (PDF, QR-код);
- механізм повернення коштів.

А.2.2.3 Модуль управління ресурсами

- облік транспортних засобів (наявність, технічний стан);
- управління персоналом (водії, диспетчери);
- формування розкладів на основі доступності транспорту.

А.2.2.4 Модуль аналітики

- звіти про завантаженість маршрутів;
- фінансова статистика (продажі квитків, доходи).

А.2.3 Класи користувачів

- пасажир: пошук маршрутів, купівля квитків, перегляд історії поїздок;
- адміністратор компанії-перевізника: створення маршрутів та поїздок, перегляд статистики поїздок, управління автопарком.
- робітник компанії-перевізника: перевірка квитків пасажирів, перегляд маршруту поїздки, повідомлення про зміни;
- адміністратор: управління адміністраторами компаній, доступ до загальної аналітики, налаштування системи.

А.2.4 Обмеження

Технологічні:

- використання HTTP API (без графічного інтерфейсу);
- підтримка тільки реляційної СУБД PostgreSQL на першому етапі.

Безпека:

- відповідність PCI DSS для платіжних операцій;
- обов’язкова автентифікація через JWT.

Продуктивність:

- час відгуку API не повинен перевищувати 2 секунди;
- система повинна витримувати до 10 000 одночасних користувачів.

А.3 Детальні вимоги

А.3.1 Функціональні вимоги

А.3.1.1 Модуль планування маршрутів

Вхідні дані:

- пункт відправлення та призначення;
- бажана дата та час;
- критерії оптимізації (час/вартість/пересадки).

Обробка:

- перетворення даних у графову модель (станції – вершини, маршрути – ребра);
- застосування алгоритму Дейкстри та пошуку в ширину для пошуку найкоротшого та всіх шляхів відповідно;
- фільтрація результатів (наявність вільних місць, допустимі дати).

Вихідні дані:

- список маршрутів із деталями;
- загальний час у дорозі;
- вартість;
- кількість пересадок;
- деталі кожного сегмента (вид транспорту, час відправлення/прибуття);

А.3.1.2 Платіжний модуль

Вимоги до оплати:

- підтримка мультивалютності (USD, EUR, UAH);
- автоматична конвертація валют за поточним курсом.

Вимоги до квитків:

- генерація у форматах PDF та QR-код;

- можливість скасування з автоматичним поверненням коштів.

А.3.1.3 Модуль управління ресурсами

Управління транспортом:

- додавання/редагування транспортних засобів (автобуси, поїзди, літаки);
- відстеження технічного стану.

Управління персоналом:

- розподіл працівників за графіками;
- розмежування прав доступу (ролі: адміністратор компанії-перевізника, працівник компанії-перевізника, адміністратор).

А.3.2 Нефункціональні вимоги

А.3.2.1 Продуктивність

- час відгуку API: ≤ 2 секунди для 95% запитів;
- пропускна здатність: до 10 000 одночасних користувачів;
- масштабування: можливість додавання серверів під навантаженням.

А.3.2.2 Безпека

Шифрування даних:

- AES-256 для зберігання;
- TLS 1.3 для передачі.

Автентифікація та авторизація:

- JWT-токени з терміном дії 1 година.

Платежі та обробка даних:

- PCI DSS для платіжних операцій;
- GDPR для обробки персональних даних.

А.3.2.3 Юзабіліті

- підтримка української та англійської мов;
- локалізація дат, часу, валют;

- документація API у форматі OpenAPI (Swagger).

А.3.3 Інтерфейси

А.3.3.1 Зовнішні інтерфейси

API для клієнтів:

- RESTful HTTP з JSON;
- Документація через Swagger UI.

Інтеграція з платіжними системами:

- Stripe, Privat24, BTCPay.

Інтеграція з транспортними платформами:

- GDS (для авіакомпаній);
- TMS (для залізниць).

А.3.3.2 Внутрішні інтерфейси

- СУБД: PostgreSQL з ORM Entity Framework.

А.3.4 Вимоги до даних

А.3.4.1 Структура бази даних

Основні сутності:

- countries, regions, cities, addresses – географічні дані;
- routes – маршрути;
- vehicles – транспортні засоби;
- vehicle_enrollments – поїздки;
- tickets – квитки пасажирів.

Денормалізація:

- таблиця ticket_groups для швидкого пошуку агрегованих даних.

А.3.4.2 Цілісність даних

- обов’язковість полів (наприклад, часу відправлення у поїздці);
- унікальність записів (наприклад, електронної пошти користувача).

ДОДАТОК Б

Приклади коду створеного ПЗ

Б.1 Використання шаблону "Посередник"

```
using MediatR;

namespace cuqmbr.TravelGuide.Application
    .Addresses.Commands.AddAddress;

public record AddAddressCommand : IRequest<AddressDto>
{
    public string Name { get; set; }

    public double Longitude { get; set; }

    public double Latitude { get; set; }

    public VehicleType VehicleType { get; set; }

    public Guid CityGuid { get; set; }
}

public class AddAddressCommandHandler :
    IRequestHandler<AddAddressCommand, AddressDto>
{
    // Об'являємо поля класів для взаємодії с БД та ін.

    public AddAddressCommandHandler(// Залежності)
    {
        // Задаємо значення полям
    }

    public async Task<AddressDto> Handle(
        AddAddressCommand request,
        CancellationToken cancellationToken)
    {
        // Перевіряємо на наявність дублікату
        // Перевіряємо чи існує батьківський об'єкт (City)
        // Додаємо інформацію про адресу в сховище даних
        // Повертаємо модель даних, що відображає додану адресу
    }
}
```

Б.2 Конвеєр журналювання

```
using System.Diagnostics;
using MediatR;
using Microsoft.Extensions.Logging;

```

```
 5 namespace cuqmbr.TravelGuide.Application.Common.Behaviours;
 6
 7 public class LoggingBehaviour<TRequest, TResponse> :
 8     IPipelineBehavior<TRequest, TResponse>
 9     where TRequest : notnull
10 {
11     private readonly ILogger _logger;
12     private readonly Stopwatch _stopWatch;
13
14     public LoggingBehaviour(ILogger<TRequest> logger)
15     {
16         _logger = logger;
17         _stopWatch = new Stopwatch();
18     }
19
20     public async Task<TResponse> Handle(
21         TRequest request,
22         RequestHandlerDelegate<TResponse> next,
23         CancellationToken cancellationToken)
24     {
25         _logger.LogDebug("Started handling MediatR request.");
26
27         _stopWatch.Start();
28
29         var response = await next();
30
31         _stopWatch.Stop();
32
33         _logger.LogInformation(
34             "MediatR request handled in {Duration}ms.",
35             _stopWatch.ElapsedMilliseconds);
36
37         if (_stopWatch.ElapsedMilliseconds > 500)
38         {
39             _logger.LogWarning(
40                 "MediatR request handled slowly ({Duration}ms).",
41                 _stopWatch.ElapsedMilliseconds);
42         }
43
44         return response;
45     }
46 }
```

Б.3 Узагальнений інтерфейс репозиторію

```
1 using System.Linq.Expressions;
2 using cuqmbr.TravelGuide.Application.Common.Models;
3 using cuqmbr.TravelGuide.Domain.Entities;
4
5 namespace cuqmbr.TravelGuide.Application
6     .Common.Persistence.Repositories;
```

```
 7
 8 public interface BaseRepository<TEntity>
 9     where TEntity : EntityBase
10 {
11     Task<TEntity> AddOneAsync(
12         TEntity entity,
13         CancellationToken cancellationToken);
14
15     Task<TEntity?> GetOneAsync(
16         Expression<Func<TEntity, bool>> predicate,
17         CancellationToken cancellationToken);
18
19     Task<PaginatedList<TEntity>> GetPageAsync(
20         Expression<Func<TEntity, bool>> predicate,
21         int pageNumber, int pageSize,
22         CancellationToken cancellationToken);
23
24     Task<TEntity> UpdateOneAsync(
25         TEntity entity,
26         CancellationToken cancellationToken);
27
28     Task DeleteOneAsync(
29         TEntity entity,
30         CancellationToken cancellationToken);
31 }
```

Б.4 Абстрактна реалізація репозиторію для взаємодії з СУБД PostgreSQL

```
 1 using System.Linq.Expressions;
 2 using cuqmbr.TravelGuide.Application
 3     .Common.Persistence.Repositories;
 4 using cuqmbr.TravelGuide.Application.Common.Models;
 5 using cuqmbr.TravelGuide.Domain.Entities;
 6 using Microsoft.EntityFrameworkCore;
 7
 8 namespace cuqmbr.TravelGuide.Persistence.PostgreSql.Repositories;
 9
10 public abstract class PostgreSqlBaseRepository<Tentity>
11     : BaseRepository<TEntity>
12     where TEntity : EntityBase
13 {
14     protected readonly DbSet<TEntity> _dbSet;
15
16     public PostgreSqlBaseRepository(PostgreSqlDbContext dbContext)
17     {
18         _dbSet = dbContext.Set<TEntity>();
19     }
20
21     public async Task<TEntity> AddOneAsync(
22         TEntity entity,
23         CancellationToken cancellationToken)
```

```
24      {
25          await _dbSet.AddAsync(entity, cancellationToken);
26          return entity;
27      }
28
29      public async Task<TEntity?> GetOneAsync(
30          Expression<Func<TEntity, bool>> predicate,
31          CancellationToken cancellationToken)
32      {
33          return await _dbSet
34              .SingleOrDefaultAsync(predicate, cancellationToken);
35      }
36
37      public async Task<PaginatedList<TEntity>> GetPageAsync(
38          Expression<Func<TEntity, bool>> predicate,
39          int pageNumber, int pageSize,
40          CancellationToken cancellationToken)
41      {
42          var count = await _dbSet
43              .Where(predicate)
44              .CountAsync(cancellationToken);
45
46          var entities =
47              await _dbSet
48                  .Where(predicate)
49                  .Skip((pageNumber - 1) * pageSize).Take(pageSize)
50                  .ToListAsync(cancellationToken);
51
52          return new PaginatedList<TEntity>(
53              entities, count,
54              pageNumber, pageSize);
55      }
56
57      public Task<TEntity> UpdateOneAsync(
58          TEntity entity,
59          CancellationToken cancellationToken)
60      {
61          _dbSet.Update(entity);
62          return Task.FromResult(entity);
63      }
64
65      public Task DeleteOneAsync(
66          TEntity entity,
67          CancellationToken cancellationToken)
68      {
69          _dbSet.Remove(entity);
70          return Task.CompletedTask;
71      }
72  }
```

Б.5 Конкретний інтерфейсу репозиторію

```
 1 using cuqmbr.TravelGuide.Domain.Entities;
 2
 3 namespace cuqmbr.TravelGuide.Application
       .Common.Persistence.Repositories;
 4
 5 public interface AddressRepository : BaseRepository<Address> { }
```

Б.6 Конкретний клас репозиторію

```
 1 using
cuqmbr.TravelGuide.Application.Common.Persistence.Repositories;
 2 using cuqmbr.TravelGuide.Domain.Entities;
 3
 4 namespace cuqmbr.TravelGuide.Persistence.PostgreSql.Repositories;
 5
 6 public sealed class PostgreSqlAddressRepository :
 7     PostgreSqlBaseRepository<Address>, AddressRepository
 8 {
 9     public PostgreSqlAddressRepository(PostgreSqlDbContext
dbContext)
10         : base(dbContext) { }
11 }
```

Б.7 Інтерфейс класу "Одиниця роботи"

```
 1 using
cuqmbr.TravelGuide.Application.Common.Persistence.Repositories;
 2
 3 namespace cuqmbr.TravelGuide.Application.Common.Persistence;
 4
 5 public interface UnitOfWork : IDisposable
 6 {
 7     AddressRepository AddressRepository { get; }
 8
 9     Task<int> SaveAsync(CancellationToken cancellationToken);
10 }
```

Б.8 Реалізація класу "Одиниця роботи"

```
 1 using cuqmbr.TravelGuide.Application.Common.Persistence;
 2 using
cuqmbr.TravelGuide.Application.Common.Persistence.Repositories;
 3 using cuqmbr.TravelGuide.Persistence.PostgreSql.Repositories;
 4
 5 namespace cuqmbr.TravelGuide.Persistence.PostgreSql;
 6
 7 public sealed class PostgreSqlUnitOfWork : UnitOfWork
```

```
 8 {
 9     private readonly PostgreSqlDbContext _dbContext;
10
11     public PostgreSqlUnitOfWork(
12         PostgreSqlDbContext dbContext)
13     {
14         _dbContext = dbContext;
15
16         AddressRepository =
               new PostgreSqlAddressRepository(_dbContext);
17     }
18
19     public AddressRepository AddressRepository { get; init; }
20
21     public async Task<int> SaveAsync(
           CancellationToken cancellationToken)
22     {
23         return await
_dbContext.SaveChangesAsync(cancellationToken);
24     }
25
26     public void Dispose()
27     {
28         Dispose(disposing: true);
29         GC.SuppressFinalize(this);
30     }
31
32     public void Dispose(bool disposing)
33     {
34         if (disposing)
35         {
36             _dbContext.Dispose();
37         }
38     }
39 }
```

Б.9 Створення та асинхронне виконання задач конвертації цін квитків

```
1 var convertTasks = new List<Task>();
2
3 foreach (var t in ticketGroup.Tickets)
4 {
5     convertTasks.Add(Task.Factory.StartNew(() =>
6     {
7         lock (_lock)
8         {
9             var convertedCost =
_currencyConverter.ConvertAsync(t.Cost,
10                t.Currency, _sessionCurrencyService.Currency,
11                cancellationToken)
12            .Result;
```

```
13
14              t.Cost = _sessionCurrencyService
15                  .Currency.Round(convertedCost);
16          }
17      }));
18
19      foreach (var rad in t.VehicleEnrollment.RouteAddressDetails)
20      {
21          convertTasks.Add(Task.Factory.StartNew(() =>
22          {
23              lock (_lock)
24              {
25                  var convertedCost =
_currencyConverter.ConvertAsync(
26                     rad.CostToNextAddress, t.VehicleEnrollment.Currency,
27                     _sessionCurrencyService.Currency, cancellationToken)
28                  .Result;
29
30                  rad.CostToNextAddress = _sessionCurrencyService
31                      .Currency.Round(convertedCost);
32              }
33          }));
34      }
35  }
36
37  Task.WaitAll(convertTasks);
```

Б.10 Валідатор даних команд, що використовує методи для локалізації

```
 1  using cuqmbr.TravelGuide.Application.Common.Services;
 2  using FluentValidation;
 3  using Microsoft.Extensions.Localization;
 4
 5  namespace cuqmbr.TravelGuide.Application
        .Addresses.Commands.AddAddress;
 6
 7  public class AddAddressCommandValidator :
        AbstractValidator<AddAddressCommand>
 8  {
 9      public AddAddressCommandValidator(
10          IStringLocalizer localizer,
11          SessionCultureService cultureService)
12      {
13          RuleFor(v => v.Name)
14              .NotEmpty()
15              .WithMessage(localizer["FluentValidation.NotEmpty"])
16              .MaximumLength(64)
17              .WithMessage(
18                  String.Format(
19                      cultureService.Culture,
20                      localizer["FluentValidation.MaximumLength"],
```

```
21                      64));
22      }
23 }
```

Б.11 Приклад файлів локалізації в форматі json

```
1 {
2   "FluentValidation": {
3     "NotEmpty": "Must not be empty.",
4     "MaximumLength": "Length must less than or equal to {0}
characters.",
5   }
6 }
```

```
1 {
2   "FluentValidation": {
3     "NotEmpty": "Не повинно бути порожнім.",
4     "MaximumLength": "Довжина повинна бути меншою або дорівнювати
{0} символам.",
5   }
6 }
```

Б.12 Клас-перелічення для валют

```
1 namespace cuqmbr.TravelGuide.Domain.Enums;
2
3 // ISO-4217 Currency Codes dated 2025-03-31
4
5 public abstract class Currency : Enumeration<Currency>
6 {
7     public static readonly Currency Default = new
DefaultCurrency();
8     public static readonly Currency USD = new USDCurrency();
9     public static readonly Currency EUR = new EURCurrency();
10    public static readonly Currency UAH = new UAHCurrency();
11
12    protected Currency(int value, string name) : base(value, name) {
}
13
14    protected virtual byte DecimalDigits { get; } = byte.MaxValue;
15
16    public decimal Round(decimal amount)
17    {
18        return Math.Round(amount, DecimalDigits);
19    }
20
21    // No currency is specified
22    private sealed class DefaultCurrency : Currency
23    {
24        public DefaultCurrency() : base(Int32.MaxValue, "DEFAULT")
{ }
```

```
25
26          protected override byte DecimalDigits => 2;
27      }
28
29      private sealed class USDCurrency : Currency
30      {
31          public USDCurrency() : base(840, "USD") { }
32
33          protected override byte DecimalDigits => 2;
34      }
35
36      private sealed class EURCurrency : Currency
37      {
38          public EURCurrency() : base(978, "EUR") { }
39
40          protected override byte DecimalDigits => 2;
41      }
42
43      private sealed class UAHCurrency : Currency
44      {
45          public UAHCurrency() : base(980, "UAH") { }
46
47          protected override byte DecimalDigits => 2;
48      }
49  }
```

Б.13 HTTP контролер фреймворку ASP.NET

```
 1  using Microsoft.AspNetCore.Mvc;
 2  using Swashbuckle.AspNetCore.Annotations;
 3  using cuqmbr.TravelGuide.Domain.Enums;
 4  using cuqmbr.TravelGuide.Application.Addresses;
 5  using
cuqmbr.TravelGuide.Application.Addresses.Commands.AddAddress;
 6  using cuqmbr.TravelGuide.Application.Addresses.ViewModels;
 7
 8  namespace cuqmbr.TravelGuide.HttpApi.Controllers;
 9
10  [Route("addresses")]
11  public class AddressesController : ControllerBase
12  {
13      [HttpPost]
14      public async Task<ActionResult<AddressDto>> Add(
15          [FromBody] AddAddressViewModel viewModel,
16          CancellationToken cancellationToken)
17      {
18          return StatusCode(
19              StatusCodes.Status201Created,
20              await Mediator.Send(
21                  new AddAddressCommand()
22                  {
```

```
                    Name = viewModel.Name,
                    Longitude = viewModel.Longitude,
                    Latitude = viewModel.Latitude,
                    VehicleType =
VehicleType.FromName(viewModel.VehicleType),
                    CityGuid = viewModel.CityUuid
                },
                cancellationToken));
    }
}
```

Б.14 Конвеєр обробки виключень ASP.NET

```
using cuqmbr.TravelGuide.Application.Common.Exceptions;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Localization;
using System.Diagnostics;

namespace cuqmbr.TravelGuide.HttpApi.Middlewares;

public class GlobalExceptionHandlerMiddleware : IMiddleware
{
    private readonly Dictionary<
        Type, Func<HttpContext, Exception, Task>>
_exceptionHandlers;
    private readonly ILogger _logger;
    private readonly IStringLocalizer _localizer;

    public GlobalExceptionHandlerMiddleware(
        ILogger<GlobalExceptionHandlerMiddleware> logger,
        IStringLocalizer localizer)
    {
        _exceptionHandlers = new()
        {
            { typeof(ValidationException),
HandleValidationException },
        };

        _logger = logger;
        _localizer = localizer;
    }

    public async Task InvokeAsync(
        HttpContext context, RequestDelegate next)
    {
        try
        {
            await next(context);
        }
        catch (Exception exception)
```

```
36          {
37              var exceptionType = exception.GetType();
38
39              if (_exceptionHandlers.ContainsKey(exceptionType))
40              {
41                  _logger.LogInformation(
42                      "Interrupted with planned exception
{ExceptionType}.",
43                      exception.GetType());
44
45                  context.Response.ContentType =
"application/problem+json";
46
47                  await _exceptionHandlers[exceptionType]
48                      .Invoke(context, exception);
49                  return;
50              }
51
52              _logger.LogError(
53                  exception,
54                  "Interrupted with unhandled exception
{ExceptionType}.",
55                  exception.GetType());
56
57              await HandleUnhandledException(context, exception);
58          }
59      }
60
61      private async Task HandleValidationException(
62          HttpContext context,
63          Exception exception)
64      {
65          var ex = (ValidationException)exception;
66
67          context.Response.StatusCode =
StatusCodes.Status400BadRequest;
68
69          await context.Response.WriteAsJsonAsync(
70              new HttpValidationProblemDetailsWithTraceId(ex.Errors)
71          {
72              Status = StatusCodes.Status400BadRequest,
73              Type =
"https://datatracker.ietf.org/doc/html/rfc7231#section-6.5.1",
74              Title =
_localizer["ExceptionHandling.ValidationException.Title"],
75              Detail =
_localizer["ExceptionHandling.ValidationException.Detail"]
76          });
77      }
78
79      private async Task HandleUnhandledException(
80          HttpContext context, Exception exception)
```

```
81     {
82         context.Response.StatusCode =
StatusCodes.Status500InternalServerError;
83
84         await context.Response.WriteAsJsonAsync(new
ProblemDetailsWithTraceId()
85         {
86             Status = StatusCodes.Status500InternalServerError,
87             Type =
"https://datatracker.ietf.org/doc/html/rfc7231#section-6.6.1",
88             Title =
_localizer["ExceptionHandling.UnhandledException.Title"],
89             Detail =
_localizer["ExceptionHandling.UnhandledException.Detail"]
90         });
91     }
92 }
```

Б.15 Періодичний сервіс для видалення застарілих зарезервованих квитків

```
 1 using MediatR;
 2 using cuqmbr.TravelGuide.Application
 3     .TicketGroups.Commands.RemoveOldReservedTicketGroups;
 4
 5 namespace cuqmbr.TravelGuide.HttpApi.HostedServices;
 6
 7 public class ReservedTicketRemoverHostedService :
BackgroundService
 8 {
 9     private Timer _timer = null;
10
11     public ReservedTicketRemoverHostedService(
12         IServiceProvider serviceProvider)
13     {
14         ServiceProvider = serviceProvider;
15     }
16
17     public IServiceProvider ServiceProvider;
18
19     protected override async Task ExecuteAsync(
20         CancellationToken cancellationToken)
21     {
22         _timer = new Timer(async (state) =>
23             {
24                 using var scope = ServiceProvider.CreateScope();
25                 var _mediator = scope.ServiceProvider
26                     .GetRequiredService<IMediator>();
27
28                 await _mediator.Send(
29                     new RemoveOldReservedTicketGroupsCommand()
30                     {
```

```
                        ReservedFor = TimeSpan.FromMinutes(10)
                    }, cancellationToken);
            },
            null, TimeSpan.Zero, TimeSpan.FromMinutes(1));
    }

    public override async Task StopAsync(
        CancellationToken cancellationToken)
    {
        _timer?.Dispose();
        await base.StopAsync(cancellationToken);
    }
}
```

Б.16 Код алгоритму пошуку усіх маршрутів із пересадками

```
var paths =
    new List<List<TaggedEdge<Address, RouteAddressDetail>>>();
var queue =
    new Queue<(
        TaggedEdge<Address, RouteAddressDetail> edge,
        List<TaggedEdge<Address, RouteAddressDetail>> path)>();

foreach (var edge in graph.OutEdges(departureAddress))
{
    queue.Enqueue(
        (edge,
        new List<TaggedEdge<Address, RouteAddressDetail>>()
            { edge }));
}

while (queue.Count > 0)
{
    var current = queue.Dequeue();

    if (current.edge.Target.Equals(arrivalAddress))
    {
        paths.Add(current.path);
        continue;
    }

    foreach (var edge in graph.OutEdges(current.edge.Target))
    {
        var neighbor = edge;
        if (!current.path.Contains(neighbor))
        {
            var newPath =
                new List<TaggedEdge<Address, RouteAddressDetail>>
                    (current.path) { neighbor };
            queue.Enqueue((neighbor, newPath));
        }
```

```
    }
}
```

Б.17 Метод підміни сервісу автентифікованої сесії

```
protected void SetAuthenticatedUserRoles(
    IdentityRole[] roles = default!)
{
    _serviceCollection
        .AddScoped<SessionUserService>(_ =>
        {
            var mock = new Mock<SessionUserService>();

            var guid = Guid.NewGuid();

            mock.Setup(s => s.Email).Returns(guid.ToString());
            mock.Setup(s => s.Guid).Returns(guid);
            mock.Setup(s => s.Guid).Returns(Guid.NewGuid());
            mock.Setup(s => s.IsAuthenticated).Returns(true);
            mock.Setup(s => s.Roles).Returns(roles);

            return mock.Object;
        });
}
```

Б.18 Метод тестування додавання країни

```
[Fact]
public async Task AddCountry_WithAdminRole_CountryCreated()
{
    SetAuthenticatedUserRoles(new[]
{ IdentityRole.Administrator });

    var mediator = GetService<IMediator>();

    string name = "Name";

    var createCountryResult = await mediator.Send(
        new AddCountryCommand()
        {
            Name = name
        }, TestContext.Current.CancellationToken);

    var getCountryResult = await mediator.Send(
        new GetCountryQuery()
        {
            Guid = createCountryResult.Uuid,
        }, TestContext.Current.CancellationToken);

    Assert.NotNull(getCountryResult);
```

```
    Assert.NotNull(getCountryResult.Name);
    Assert.Equal(name, getCountryResult.Name);
}
```

Б.19 Метод тестування неможливості додання дублікату країни

```
[Fact]
public async Task
    AddDuplicateCountry_WithAdminRole_ThrowsDuplicateEntityException()
{
    SetAuthenticatedUserRoles(new[]
{ IdentityRole.Administrator });

    var mediator = GetService<IMediator>();

    string name = "Name";

    await mediator.Send(
        new AddCountryCommand()
        {
            Name = name
        }, TestContext.Current.CancellationToken);

    await Assert.ThrowsAsync<DuplicateEntityException>(() =>
        mediator.Send(new AddCountryCommand()
        {
            Name = name
        }, TestContext.Current.CancellationToken));
}
```

Б.20 Метод тестування граничних значень довжини назви країни

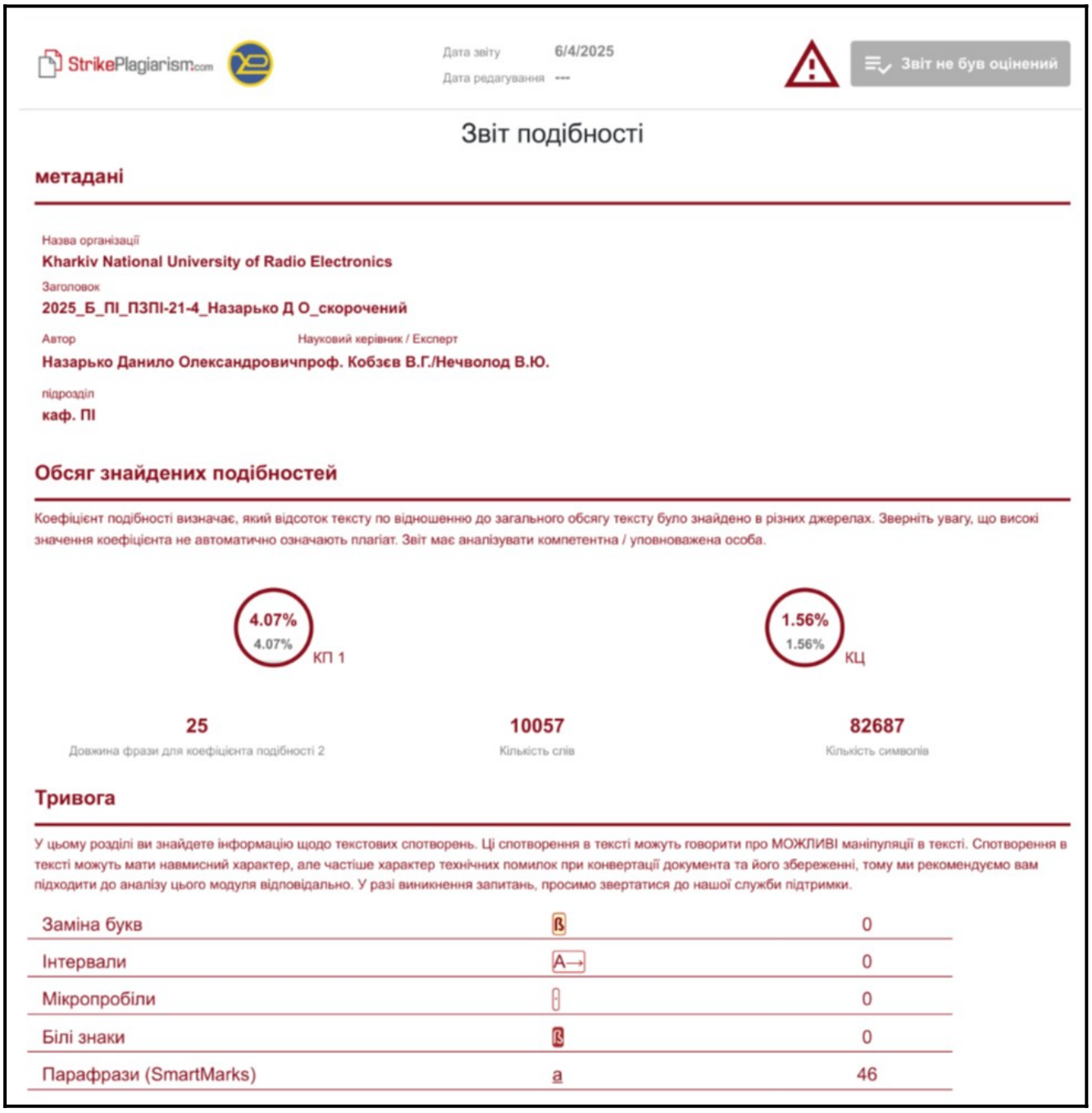
```
[Theory]
// Empty
[InlineData("")]
// Length > 64 (65)
[InlineData("012345678901234567890123456789012345678901234567890123456789012345678901234")]
public async Task
    AddCountry_WithInvalidName_WithAdminRole_ThrowsValidationException
        (string name)
{
    SetAuthenticatedUserRoles(new[]
{ IdentityRole.Administrator });

    var mediator = GetService<IMediator>();

    await Assert.ThrowsAsync<ValidationException>(() =>
        mediator.Send(new AddCountryCommand()
        { Name = name }, TestContext.Current.CancellationToken));
}
```

ДОДАТОК В

Результати перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 6/4/2025
Дата редагування ---

Звіт не був оцінений

Звіт подібності

метадані

Назва організації
Kharkiv National University of Radio Electronics
Заголовок
2025_Б_ПІ_ПЗПІ-21-4_Назарько Д О_скорочений
Автор
Назарько Данило Олександровичпроф. Кобзєв В.Г./Нечволод В.Ю.
Науковий керівник / Експерт
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.

4.07% 4.07% КП 1

1.56% 1.56% КЦ

25	10057	82687
Довжина фрази для коефіцієнта подібності 2	Кількість слів	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	ß	0
Інтервали	A→	0
Мікропробіли	¦	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	46

ДОДАТОК Г

Слайди презентації

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кваліфікаційна робота бакалавра

Програмна система планування маршрутів поїздок та продажу квитків з урахуванням пересадок на різні види транспорту. API на основі протоколу HTTP.

Здобувач:
Назарько Д.О.
ПЗПІ-21-4

Керівник:
проф. кафедри ПІ
Кобзєв В.Г.

2025

Мета роботи

Розробити HTTP API уніфікованої платформи для:

- планування оптимальних маршрутів з пересадками;
- автоматизації продажу квитків;
- інтеграції даних різних транспортних компаній;
- підвищення ефективності управління транспортом.

2

Аналіз проблеми

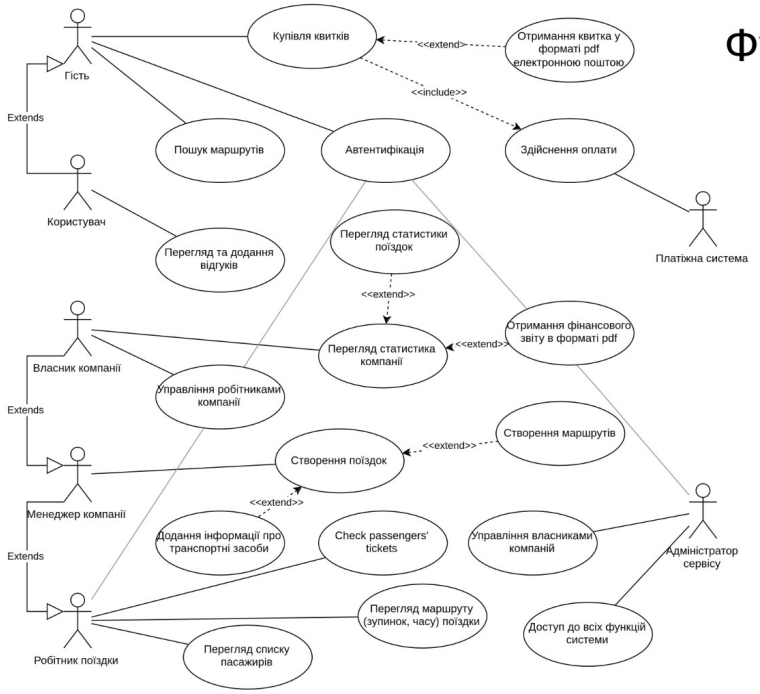
Назва сервісу	Переваги	Недоліки
Google Maps	Інтеграція з різними видами транспорту (автобуси, поїзди, таксі)	Відсутність продажу квитків (лише посилання на сторонні сервіси)
	Реалізація алгоритмів пошуку маршрутів (на основі даних у реальному часі)	Обмежена підтримка мультитранспортних поїздок у деяких країнах
Rome2Rio	Пошук комбінованих маршрутів (літаки + поїзди + автобуси)	Відсутність єдиного кошика для мультитранспортних квитків
	Підтримка бронювання квитків через партнерські сервіси	Обмежена інтеграція з локальними перевізниками
Укрзалізниця	Продаж квитків на поїзди з підтримкою електронного квитка	Обмеження лише залізничним транспортом
	Інтеграція з системою бронювання місць	Відсутність пошуку маршрутів з пересадками на інші види транспорту

3

Постановка задачі

- побудова централізованої БД із підтримкою цілісності та швидкості обробки даних;
- розробка функціоналу: пошук маршрутів та прибдання квитків для пасажирів, управління транспортом та робітниками для перевізників, аналітика для адміністраторів та перевізників;
- розробка алгоритмів пошуку маршрутів: використання теорії графів, алгоритм Дейкстри та пошуку в ширину;
- інтеграція з платіжними системами: підтримка LiqPay, Stripe, BTCPay, єдиний кошик для мультитранспортних квитків;
- захист даних: при передачі (TLS 1.3), зберіганні (Data at Rest Encryption, хешування паролів), JWT-токени для авторизації;
- підтримка надійності за допомогою модульних та інтеграційні тестів.

4



Функціональні можливості

- пошук маршрутів за часом, вартістю та кількістю пересадок;
- миттєве оновлення даних про затримки та скасування;
- інтеграція з платіжними системами для придбання квитків;
- управління транспортом і персоналом для перевізників.

5

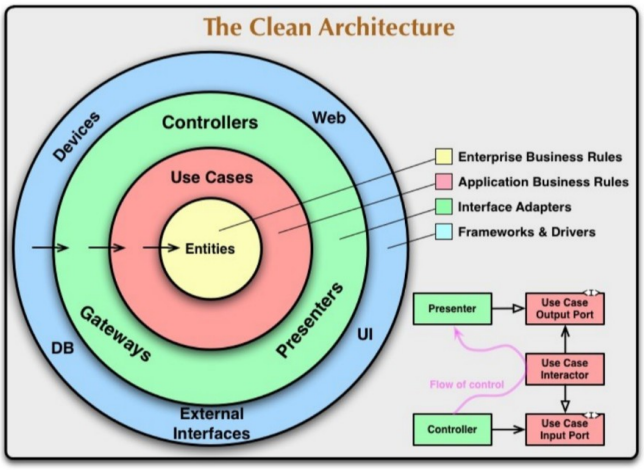
Використані технології

- застосунок: C#, ASP.NET, PostgreSQL;
- алгоритми: Дейкстри та пошуку в ширину для знаходження маршрутів;
- інфраструктура: Proxmox, Terraform, Ansible, Docker, GitHub Actions.

6

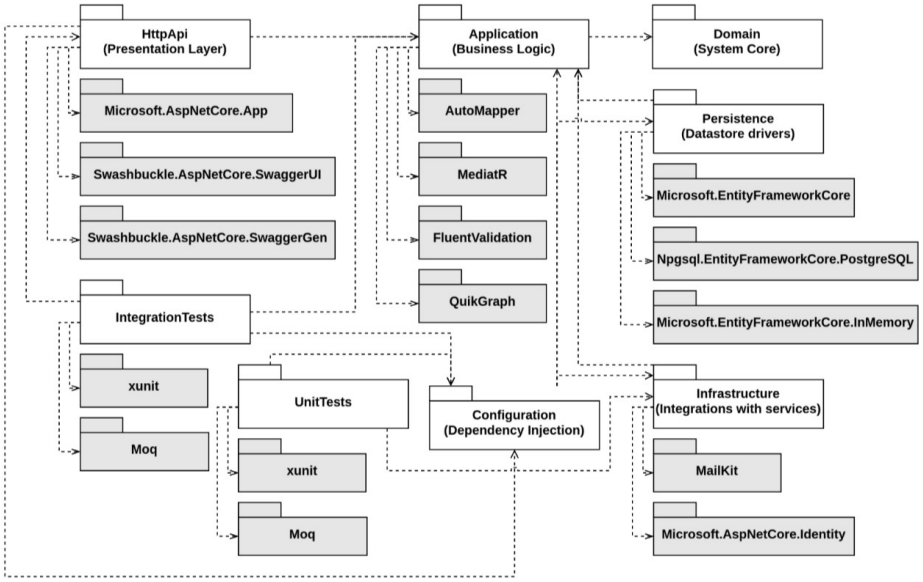
Архітектура системи

- багатошарова “Чиста” архітектура;
- незалежність від фреймворків;
- легкість тестування;
- гнучкість у виборі СУБД.



7

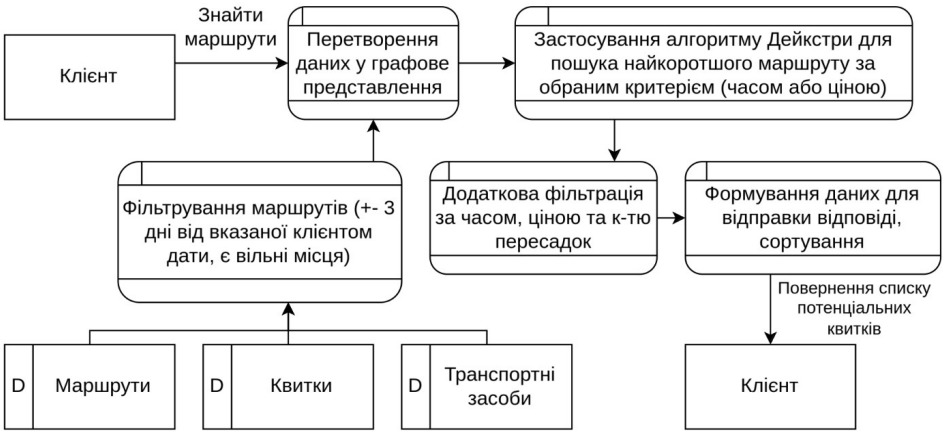
Архітектура системи



8

Алгоритми пошуку маршрутів

- графова модель, вершини – станції, ребра – транспортні зв'язки;
- оптимізація: час, вартість;
- фільтрація: вільні місця, дати, к-ть пересадок.

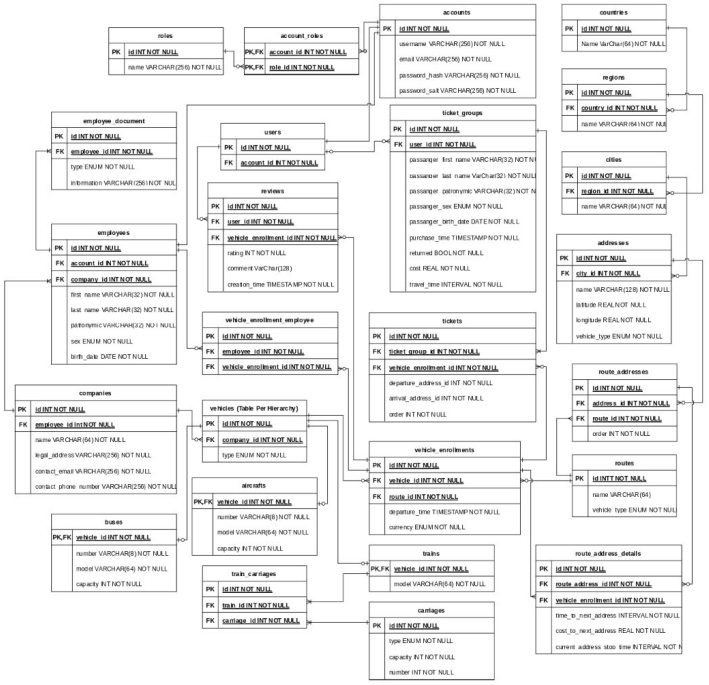


9

Підтримка різних мов і валют

- локалізація: українська та англійська мови (rfc5646);
- часові пояси: автоматичне перетворення (IANA Timezone Database);
- валюти: конвертація за курсом між USD, EUR, UAH (ISO-4217).

10



Інтеграція з СУБД

- ORM Entity Framework, Code First Approach, Fluent Configuration;
- Шаблони Repository та Unit of Work.

11

Фреймворк ASP.NET

- HTTP контролери, Mediator pattern;
- глобальний конвеєр обробки помилок;
- періодичні задачі;
- документування за допомогою OpenAPI Swagger.

12

Тестування

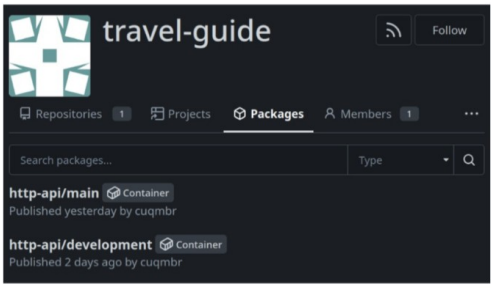
- юніт-тести: перевірка модулів (xUnit, Moq);
- інтеграційні тести: взаємодія компонентів;
- ручне тестування.

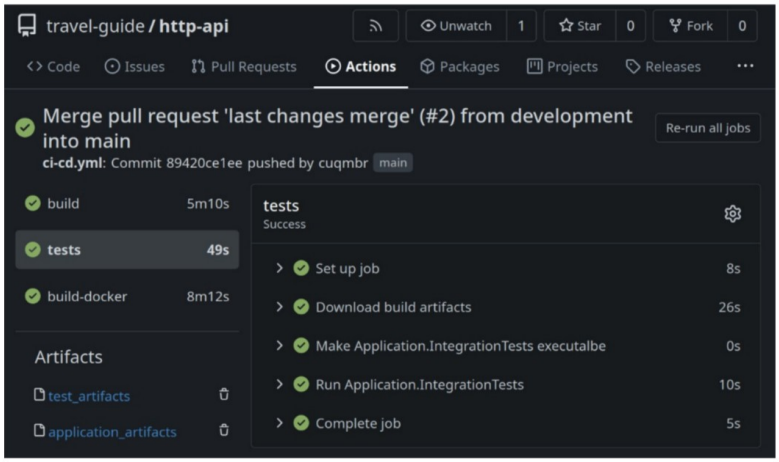
```
Test run summary: Passed! - tst/Application.IntegrationTests
  total: 133
  failed: 0
  succeeded: 133
  skipped: 0
  duration: 3s 639ms
```

13

Неперервна інтеграція

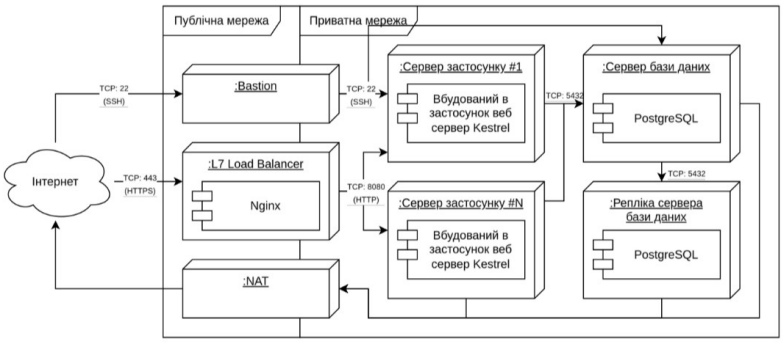
- Git, Gitea;
- дві гілки;
- GitHub Actions.

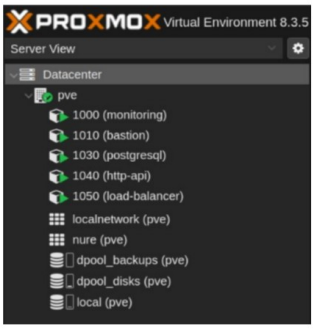


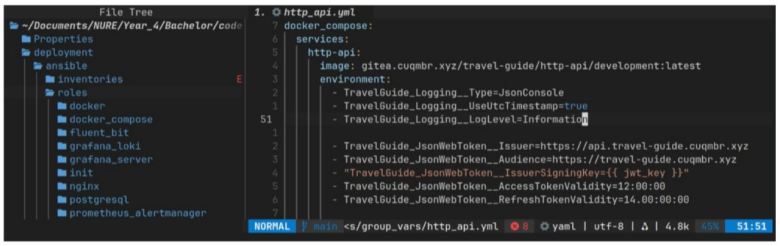


14

Розгортання







15

Підсумки

Протягом виконання роботи було ралізовано:

- централізовану БД із підтримкою цілісності та швидкості обробки даних;
- функціонал пошук маршрутів та прибдання квитків для пасажирів, управління транспортом та робітниками для перевізників;
- модифіковано алгоритм пошуку в ширину в графі для пошуку маршрутів з пересадками;
- інтеграцію з платіжною системою LiqPay;
- захист даних при передачі (TLS 1.2, 1.3), зберіганні (Data at Rest Encryption, хешування паролів), JWT-токени для авторизації;
- забезпечення надійності за допомогою проведення тестування.

16