Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Звіт з лабораторної роботи №2
з дисципліни «Архітектура програмного забезпечення»
Розробка серверної частини програмної системи

Виконав:
студент групи ПЗПІ-21-6
Розов Артем Олексійович

Перевірив:
ст. викл. Сокорчук І. П.

Харків 2024

Для лабораторної роботи було розроблено UML діаграму розгортання. ЇЇ наведено на рисунку 1.1

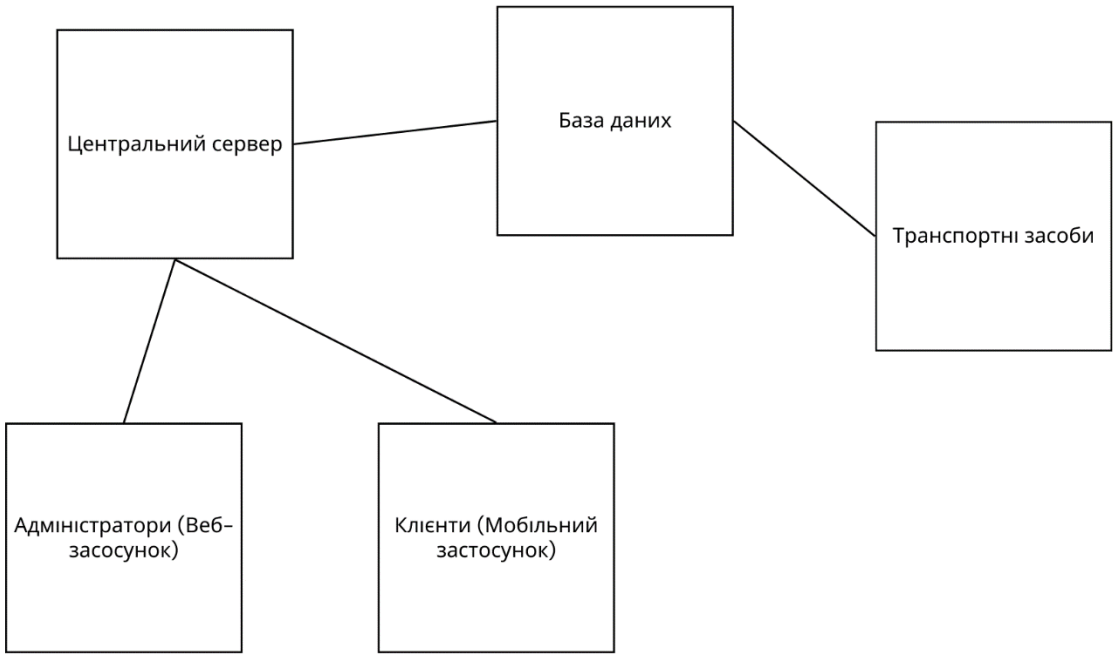

Рисунок 1.1 – UML діаграма розгортання

Наступним чином для даної лабораторної роботи було розроблено UML діаграму прецедентів. Дану діаграму відображено на рисунку 1.2.

Рисунок 1.2 – UML діаграма прецедентів

Наступний шлях – це розробка бази даних. Для створення бази даних було розроблено схему бази даних. Схему баз даних відображено на рисунку 1.3.

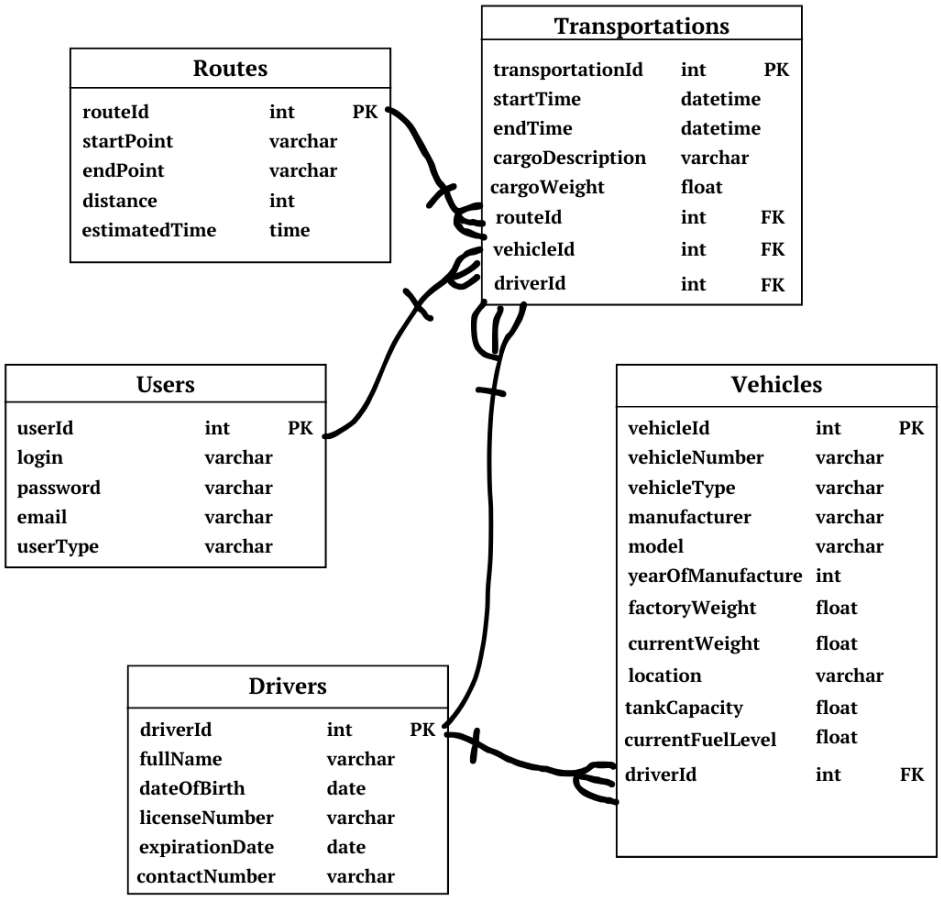

Рисунок 1.3 – Схема бази даних

Таблиця Routes має в собі інформацію про маршрути переміщення транспортних засобів. Стартову точку, кінцеву точку, дистанцію та розрахунковий час.

Таблиця Users має в собі інформацію про користувачів програмної системи. Має такі поля: логін (потрібен для входу в акаунт), пароль, пошта та тип користувача (або клієнт, або адміністратор).

Таблиця Drivers має в собі інформацію про водіїв. Має такі поля: ПІБ, дата народження, номер водійського посвідчення, термін придатності водійського посвідчення та номер телефону.

Таблиця Vehicles має в собі інформцію про автомобілі корпорацій. Має такі поля: номер авто, тип авто, марка, модель, рік випуску авто, вага авто з заводу, поточна вага авто, місцезнаходження, об'єм паливного баку, поточний процент заповнення паливного баку та водія, що користується цим авто.

Таблиця Transportations – таблиця, що має інформацію про кожне перевезення. Має такі поля: початок перевезення за часом, кінцевий час, опис вантажу, вага вантажу, певний шлях, певного водія та певне авто.

Rest специфікацію відображено в таблиці 1.1.

Таблиця 1.1 – Rest специфікація.

Метод	Посилання	Опис
GET	https://localhost:7001/api/Users/{id}	Отримання інформації про користувача по id
POST	https://localhost:7001/api/Users/register	Реєстрація нового користувача
POST	https://localhost:7001/api/Users/login	Перевірка правильності даних для входу від користувача
GET	https://localhost:7001/api/Drivers	Отримання списку всіх водіїв
GET	https://localhost:7001/api/ Drivers /{id}	Отримання інформації про водія за id
POST	https://localhost:7001/api/ Drivers	Додавання нового водія
PUT	https://localhost:7001/api/ Drivers /{id}	Оновлення інформації про водія
DELETE	https://localhost:7001/api/ Drivers /{id}	Видалення водія за його id
GET	https://localhost:7001/api/ Vehicles	Отримання інформації по всіх авто
GET	https://localhost:7001/api/ Vehicles/{id}	Отримання інформації про авто за id
POST	https://localhost:7001/api/ Vehicles	Додавання нового авто
PUT	https://localhost:7001/api/ Vehicles/{id}	Оновлення інформації про авто

Продовження таблиці 1.1 – Rest специфікація

Метод	Посилання	Опис
DELETE	https://localhost:7001/api/ Vehicles /{id}	Видалення авто за id
GET	https://localhost:7001/api/Routes	Отримання всіх маршрутів
GET	https://localhost:7001/api/ Routes /{id}	Отримання певного маршруту за його id
POST	https://localhost:7001/api/ Routes	Додавання нового маршруту
PUT	https://localhost:7001/api/ Routes/{id}	Оновлення інформації про маршрут
DELETE	https://localhost:7001/api/ Routes/{id}	Видалення маршруту за id
GET	https://localhost:7001/api/Transportations	Отримання інформації про всі перевезення
GET	https://localhost:7001/api/ Transportations /{id}	Отримання інформації про певне перевезення за його id
POST	https://localhost:7001/api/ Transportations	Додавання нового перевезення
PUT	https://localhost:7001/api/ Transportations/{id}	Оновлення інформації про перевезення
DELETE	https://localhost:7001/api/ Transportations/{id}	Видалення перевезення за id

Код реалізації наведено в додатках.

Додатки

Додаток А

Файл: MyDbContext.cs

```
1   using Microsoft.EntityFrameworkCore;
2   using VehiclesTrackingSystem.Models;
3
4   namespace VehiclesTrackingSystem
5   {
6       public class MyDbContext : DbContext
7       {
8           public MyDbContext(DbContextOptions<MyDbContext> options) :
9   base(options)
10          {
11          }
12
13          public DbSet<Vehicle> Vehicles { get; set; }
14          public DbSet<User> Users { get; set; }
15          public DbSet<Driver> Drivers { get; set; }
16          public DbSet<Models.Route> Routes { get; set; }
17          public DbSet<Transportation> Transportations { get; set; }
18      }
19  }
```

Файл: Vehicle.cs

```
1   using System.Reflection;
2
3   namespace VehiclesTrackingSystem.Models
4   {
5       public class Vehicle
6       {
7           public int VehicleId { get; set; }
8           public string VehicleNumber { get; set; }
9           public string VehicleType { get; set; }
10          public string Manufacturer { get; set; }
11          public string Model { get; set; }
12          public int YearOfManufacture { get; set; }
13          public int OwnerId { get; set; }
14          public double FactoryWeight { get; set; }
15          public double CurrentWeight { get; set; }
16          public string Location { get; set; }
17          public double TankCapacity { get; set; }
18          public double CurrentFuelLevel { get; set; }
19      }
20  }
```

Файл: User.cs

```
1   namespace VehiclesTrackingSystem.Models
2   {
3       public class User
4       {
5           public int UserId { get; set; }
6           public string Login { get; set; }
7           public string Password { get; set; }
```

```
        public string Email { get; set; }
        public string UserType { get; set; }

    }
}
```

Файл: Transportate.cs

```
namespace VehiclesTrackingSystem.Models
{
    public class Transportation
    {
        public int TransportationId { get; set; }
        public int VehicleId { get; set; }
        public int DriverId { get; set; }
        public int RouteId { get; set; }
        public DateTime StartTime { get; set; }
        public DateTime EndTime { get; set; }
        public string CargoDescription { get; set; }
        public double CargoWeight { get; set; }
    }

}
```

Файл: Route.cs

```
namespace VehiclesTrackingSystem.Models
{
    public class Route
    {
        public int RouteId {  get; set; }
        public string StartPoint {  get; set; }
        public string EndPoint {  get; set; }
        public double Distance {  get; set; }
        public TimeSpan EstimatedTime {  get; set; }

    }
}
```

Файл: RegistrationModel.cs

```
namespace VehiclesTrackingSystem.Models
{
    public class RegistrationModel
    {
        public string Login { get; set; }
        public string Password { get; set; }
        public string Email { get; set; }
        public string UserType { get; set; }
    }
}
```

Файл: LoginModel.cs

```
namespace VehiclesTrackingSystem.Models
{
    public class LoginModel
```

```
    {
        public string Login { get; set; }
        public string Password { get; set; }
    }
}
```

Файл: Driver.cs

```
namespace VehiclesTrackingSystem.Models
{
    public class Driver
    {
        public int DriverId {  get; set; }
        public string FullName {  get; set; }
        public DateTime DateOfBirth { get; set;  }
        public string LicenseNumber { get; set; }
        public DateTime ExpirationDate {  get; set; }
        public string ContactNumber {get; set; }

    }
}
```

Файл: VehiclesController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;
using VehiclesTrackingSystem;
using VehiclesTrackingSystem.Models;

[ApiController]
[Route("api/[controller]")]
public class VehiclesController : ControllerBase
{
    private readonly MyDbContext _context;

    public VehiclesController(MyDbContext context)
    {
        _context = context;
    }

    // GET: api/Vehicles
    [HttpGet]
    public async Task<ActionResult<IEnumerable<Vehicle>>>
GetVehicles()
    {
        return await _context.Vehicles.ToListAsync();
    }

    // GET: api/Vehicles/5
    [HttpGet("{id}")]
    public async Task<ActionResult<Vehicle>> GetVehicle(int id)
    {
        var vehicle = await _context.Vehicles.FindAsync(id);

        if (vehicle == null)
```

```
34          {
35              return NotFound();
36          }
37
38          return vehicle;
39      }
40
41      // POST: api/Vehicles
42      [HttpPost]
43      public async Task<ActionResult<Vehicle>> PostVehicle(Vehicle
44  vehicle)
45      {
46          _context.Vehicles.Add(vehicle);
47          await _context.SaveChangesAsync();
48
49          return CreatedAtAction(nameof(GetVehicle), new { id =
50  vehicle.VehicleId }, vehicle);
51      }
52
53      // PUT: api/Vehicles/5
54      [HttpPut("{id}")]
55      public async Task<IActionResult> PutVehicle(int id, Vehicle
56  vehicle)
57      {
58          if (id != vehicle.VehicleId)
59          {
60              return BadRequest();
61          }
62
63          _context.Entry(vehicle).State = EntityState.Modified;
64
65          try
66          {
67              await _context.SaveChangesAsync();
68          }
69          catch (DbUpdateConcurrencyException)
70          {
71              if (!VehicleExists(id))
72              {
73                  return NotFound();
74              }
75              else
76              {
77                  throw;
78              }
79          }
80
81          return NoContent();
82      }
83
84      // DELETE: api/Vehicles/5
85      [HttpDelete("{id}")]
86      public async Task<IActionResult> DeleteVehicle(int id)
87      {
88          var vehicle = await _context.Vehicles.FindAsync(id);
89          if (vehicle == null)
90          {
91              return NotFound();
```

```
        }

        _context.Vehicles.Remove(vehicle);
        await _context.SaveChangesAsync();

        return NoContent();
    }

    private bool VehicleExists(int id)
    {
        return _context.Vehicles.Any(e => e.VehicleId == id);
    }
}
```

Файл: UsersController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using VehiclesTrackingSystem.Models;

namespace VehiclesTrackingSystem.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class UsersController : ControllerBase
    {
        private readonly MyDbContext _context;

        public UsersController(MyDbContext context)
        {
            _context = context;
        }

        [HttpPost("login")]
        public async Task<IActionResult> Login(LoginModel loginModel)
        {
            var user = await _context.Users.FirstOrDefaultAsync(u =>
u.Login == loginModel.Login && u.Password == loginModel.Password);
            if (user == null)
            {
                return NotFound("Пользователь не найден или неверные
учетные данные.");
            }


            return Ok(user);
        }

        [HttpPost("register")]
        public async Task<IActionResult> Register(RegistrationModel
registerModel)
        {
            if (await _context.Users.AnyAsync(u => u.Login ==
registerModel.Login))
            {
                return Conflict("Пользователь с таким логином уже
существует.");
            }
```

```
43
44              var newUser = new User
45              {
46                  Login = registerModel.Login,
47                  Password = registerModel.Password,
48                  Email = registerModel.Email,
49                  UserType = registerModel.UserType
50              };
51
52              _context.Users.Add(newUser);
53              await _context.SaveChangesAsync();
54
55              return CreatedAtAction(nameof(GetUserById), new { id =
56  newUser.UserId }, newUser);
57          }
58
59          [HttpGet("{id}")]
60          public async Task<IActionResult> GetUserById(int id)
61          {
62              var user = await _context.Users.FindAsync(id);
63              if (user == null)
64              {
65                  return NotFound("Пользователь не найден.");
66              }
67
68              return Ok(user);
69          }
70      }
71  }
```

Файл: TransportationsController.cs

```
1   using Microsoft.AspNetCore.Mvc;
2   using Microsoft.EntityFrameworkCore;
3   using VehiclesTrackingSystem.Models;
4
5   namespace VehiclesTrackingSystem.Controllers
6   {
7       [ApiController]
8       [Route("api/[controller]")]
9       public class TransportationsController : ControllerBase
10      {
11          private readonly MyDbContext _context;
12
13          public TransportationsController(MyDbContext context)
14          {
15              _context = context;
16          }
17
18          // GET: api/Transportations
19          [HttpGet]
20          public async Task<ActionResult<IEnumerable<Transportation>>>
21  GetTransportations()
22          {
23              return await _context.Transportations.ToListAsync();
24          }
25
26          // GET: api/Transportations/5
```

```
27          [HttpGet("{id}")]
28          public async Task<ActionResult<Transportation>>
29  GetTransportation(int id)
30          {
31              var transportation = await
32  _context.Transportations.FindAsync(id);
33
34              if (transportation == null)
35              {
36                  return NotFound();
37              }
38
39              return transportation;
40          }
41
42          // POST: api/Transportations
43          [HttpPost]
44          public async Task<ActionResult<Transportation>>
45  PostTransportation(Transportation transportation)
46          {
47              _context.Transportations.Add(transportation);
48              await _context.SaveChangesAsync();
49
50              return CreatedAtAction(nameof(GetTransportation), new { id
51  = transportation.TransportationId }, transportation);
52          }
53
54          // PUT: api/Transportations/5
55          [HttpPut("{id}")]
56          public async Task<IActionResult> PutTransportation(int id,
57  Transportation transportation)
58          {
59              if (id != transportation.TransportationId)
60              {
61                  return BadRequest();
62              }
63
64              _context.Entry(transportation).State =
65  EntityState.Modified;
66
67              try
68              {
69                  await _context.SaveChangesAsync();
70              }
71              catch (DbUpdateConcurrencyException)
72              {
73                  if (!TransportationExists(id))
74                  {
75                      return NotFound();
76                  }
77                  else
78                  {
79                      throw;
80                  }
81              }
82
83              return NoContent();
84          }
```

```
85
86          // DELETE: api/Transportations/5
87          [HttpDelete("{id}")]
88          public async Task<IActionResult> DeleteTransportation(int id)
89          {
90              var transportation = await
91  _context.Transportations.FindAsync(id);
92              if (transportation == null)
93              {
94                  return NotFound();
95              }
96
97              _context.Transportations.Remove(transportation);
98              await _context.SaveChangesAsync();
99
100             return NoContent();
101         }
102
103         private bool TransportationExists(int id)
104         {
105             return _context.Transportations.Any(e =>
106 e.TransportationId == id);
107         }
108     }
109 }
```

Файл: RoutesController.cs

```
1   using Microsoft.AspNetCore.Mvc;
2   using Microsoft.EntityFrameworkCore;
3
4   namespace VehiclesTrackingSystem.Controllers
5   {
6       [ApiController]
7       [Route("api/[controller]")]
8       public class RoutesController : ControllerBase
9       {
10          private readonly MyDbContext _context;
11
12          public RoutesController(MyDbContext context)
13          {
14              _context = context;
15          }
16
17          // GET: api/Routes
18          [HttpGet]
19          public async Task<ActionResult<IEnumerable<Models.Route>>>
20  GetRoutes()
21          {
22              return await _context.Routes.ToListAsync();
23          }
24
25          // GET: api/Routes/5
26          [HttpGet("{id}")]
27          public async Task<ActionResult<Models.Route>> GetRoute(int id)
28          {
29              var route = await _context.Routes.FindAsync(id);
30
```

```
            if (route == null)
            {
                return NotFound();
            }

            return route;
        }

        // POST: api/Routes
        [HttpPost]
        public async Task<ActionResult<Models.Route>>
PostRoute(Models.Route route)
        {
            _context.Routes.Add(route);
            await _context.SaveChangesAsync();

            return CreatedAtAction(nameof(GetRoute), new { id =
route.RouteId }, route);
        }

        // PUT: api/Routes/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutRoute(int id, Models.Route
route)
        {
            if (id != route.RouteId)
            {
                return BadRequest();
            }

            _context.Entry(route).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!RouteExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // DELETE: api/Routes/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteRoute(int id)
        {
            var route = await _context.Routes.FindAsync(id);
            if (route == null)
            {
```

```
                return NotFound();
            }

            _context.Routes.Remove(route);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool RouteExists(int id)
        {
            return _context.Routes.Any(e => e.RouteId == id);
        }
    }
}
```

Файл: DriversController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace VehiclesTrackingSystem.Controllers
{
    using Microsoft.AspNetCore.Mvc;
    using Microsoft.EntityFrameworkCore;
    using System.Collections.Generic;
    using System.Linq;
    using System.Threading.Tasks;
    using VehiclesTrackingSystem.Models;

    [ApiController]
    [Route("api/[controller]")]
    public class DriversController : ControllerBase
    {
        private readonly MyDbContext _context;

        public DriversController(MyDbContext context)
        {
            _context = context;
        }

        // GET: api/Drivers
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Driver>>>
GetDrivers()
        {
            return await _context.Drivers.ToListAsync();
        }

        // GET: api/Drivers/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Driver>> GetDriver(int id)
        {
            var driver = await _context.Drivers.FindAsync(id);

            if (driver == null)
            {
                return NotFound();
            }
```

```
41
42              return driver;
43          }
44
45          // POST: api/Drivers
46          [HttpPost]
47          public async Task<ActionResult<Driver>> PostDriver(Driver
48  driver)
49          {
50              _context.Drivers.Add(driver);
51              await _context.SaveChangesAsync();
52
53              return CreatedAtAction(nameof(GetDriver), new { id =
54  driver.DriverId }, driver);
55          }
56
57          // PUT: api/Drivers/5
58          [HttpPut("{id}")]
59          public async Task<IActionResult> PutDriver(int id, Driver
60  driver)
61          {
62              if (id != driver.DriverId)
63              {
64                  return BadRequest();
65              }
66
67              _context.Entry(driver).State = EntityState.Modified;
68
69              try
70              {
71                  await _context.SaveChangesAsync();
72              }
73              catch (DbUpdateConcurrencyException)
74              {
75                  if (!DriverExists(id))
76                  {
77                      return NotFound();
78                  }
79                  else
80                  {
81                      throw;
82                  }
83              }
84
85              return NoContent();
86          }
87
88          // DELETE: api/Drivers/5
89          [HttpDelete("{id}")]
90          public async Task<IActionResult> DeleteDriver(int id)
91          {
92              var driver = await _context.Drivers.FindAsync(id);
93              if (driver == null)
94              {
95                  return NotFound();
96              }
97
98              _context.Drivers.Remove(driver);
```

```
99                 await _context.SaveChangesAsync();
100
101                 return NoContent();
102             }
103
104             private bool DriverExists(int id)
105             {
106                 return _context.Drivers.Any(e => e.DriverId == id);
107             }
108         }
109
110     }
```