**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 5

**З дисципліни:** *“Програмування в Інтернет”*

**На тему:** *“* *PHP як засіб написання сценаріїв ”*

**Лектор:**

ст. викладач

Купльовський Б. Є.

**Виконав:**

ст. групи ПЗ-25

Бурлаков О.О.

**Прийняв:**

доц. каф. ПЗ

Фоменко А. В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема:** PHP як засіб написання сценаріїв

**Мета:** засвоїти елементи програмування серверної частини застосувань мовою PHP.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

PHP (Препроцесор Гіпертексту) є мовою сценаріїв загального призначення з відкритим вихідним кодом. PHP використовується для ведення web-розробок і може записуватись безпосередньо в HTML-коді.

Синтаксис мови бере початок з C, Java, Perl і є легким для вивчення. Переважним призначенням PHP є надання web-розробникам можливості швидкого створення динамічно генерованих web-сторінок, однак сфера застосування PHP лише цим не обмежується.

В офіційній документації мова РНР подається як HTML – вбудована мова програмування (скриптів) на стороні сервера. Це означає, що:

* + оброблення РНР-коду відбувається на сервері ще до того, як web-сторінка буде передана браузеру;
  + РНР-код може бути вбудований безпосередньо в HTML-код сторінки. Цим він відрізняється від Perl;
  + РНР не є ні компілятором, ні інтерпретатором, а чимось середнім між ними. РНР обробляє сценарії, які подаються на вхід. Він транслює його в спеціальний байт-код (внутрішнє представлення). Після цього РНР виконує байт-код. У такому разі виконавчий файл не створюється. Байт-код значно компактніший, ніж звичайний код програми. Тому він швидше інтерпретується та виконується. Тому РНР більше інтерпретатор, ніж компілятор.

Використання інтерпретатора (а, значить, і РНР) має переваги:

* + не потрібно дбати про звільнення виділеної пам’яті, закривати файл після закінчення роботи з ним – це здійснює браузер, оскільки програма виконується під його контролем;
  + не потрібно визначати типи змінних та оголошувати змінну до її першого використання;
  + відлагоджування програми та виявлення помилок істотно полегшуються – інтерпретатор повідомить про це;
  + існує можливість створення програми, яка напише іншу програму, а потім введе в себе код щойно написаної програми і виконає його.

Виконання РНР-програм відбувається за схемою (рис. 4.1).

**браузер**

**php.exe**

**.phtml**

**.php**

*Рис. 4.1. Виконання програм РНР*

Перед надсиланням сервером браузеру файла його переглядає препроцес- інтерпретатор. Файли, що піддаються обробленню препроцесором, повинні мати розширення

.phtml або .php і містити код для препроцесора. Перед надсиланням сторінки PHP-код виконується сервером Apache і браузеру видається результат у вигляді HTML-сторінки

(вона відрізняється від тієї, що зберігається на сервері). Сторінки з розширенням

.html/.htm web-сервер відправляє браузеру без будь-якого опрацювання.

Основна відмінність від CGI-скриптів, написаних мовами Perl або C, полягає в тому, що в CGІ-програмах треба писати виведений HTML-код, а PHP – дає змогу вносити програму – скрипт у код HTML-сторінки (обмежену тегами <? і ?>).

Переваги РНР такі:

*продуктивність –* сервер може обслуговувати мільйони звернень на день; *інтеграція з базами даних –* РНР володіє вбудованим зв’язком з багатьма системами баз даних (MySQL, PostgreSQL, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, i Sybase та інших на основі ODBC-драйверів);

*простота для вивчення –* синтаксис РНР оснований на мовах програмування С та

Perl;

*можливість перенесення –* пакет РНР можна використовувати під керуванням різних операційних систем (Linux, FreeBSD, Unix, Windows);

*наявність вбудованих бібліотек –* можна генерувати

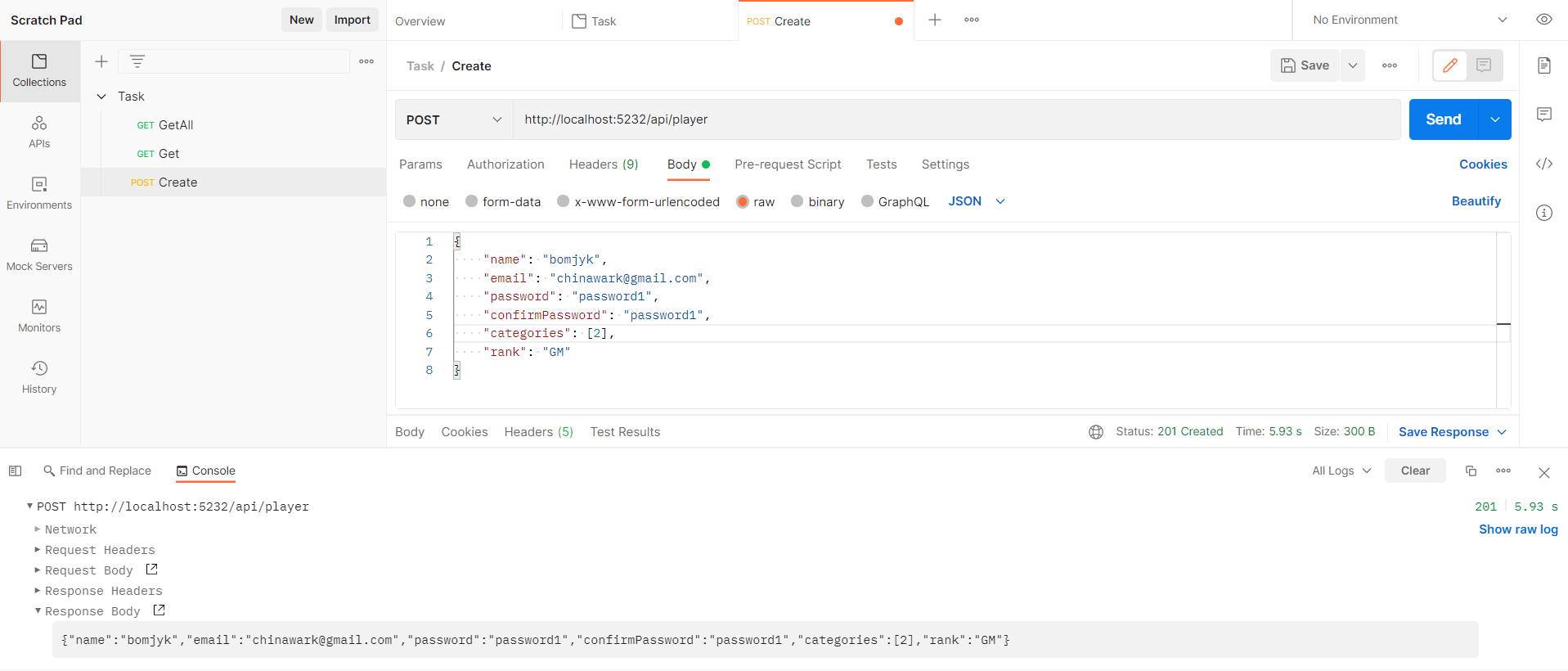
Gif-зображення, під’єднатись до багатьох мережевих служб, відправляти повідомлення електронною поштою, працювати з cookie-наборами і генерувати PDF-документи;

*вартість –* пакет РНР є безкоштовний.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

1. [Встановити](https://laravel.com/docs/8.x/installation#your-first-laravel-project) та [налаштувати](https://laravel.com/docs/8.x/configuration) Laravel.
2. Написати [api контроллер](https://laravel.com/docs/8.x/controllers#restful-partial-resource-routes) який буде приймати дані гравця в шахи для збереження (фактичне зберігання даних поки не потрібно) валідувати їх та відповідати “Дані успішно збережено” або повертати інформацію про невалідні дані.

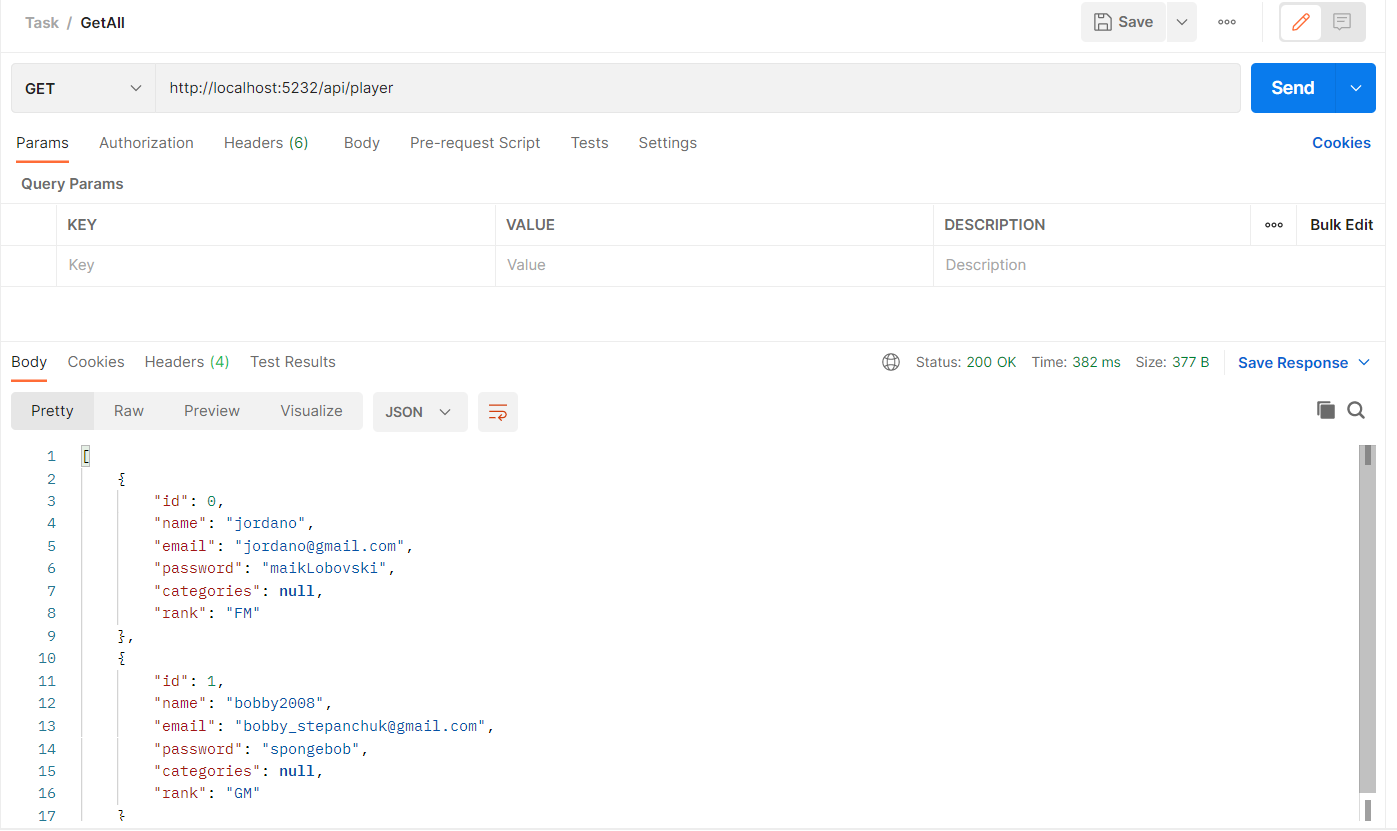
**РЕЗУЛЬТАТ ВИКОНАННЯ**



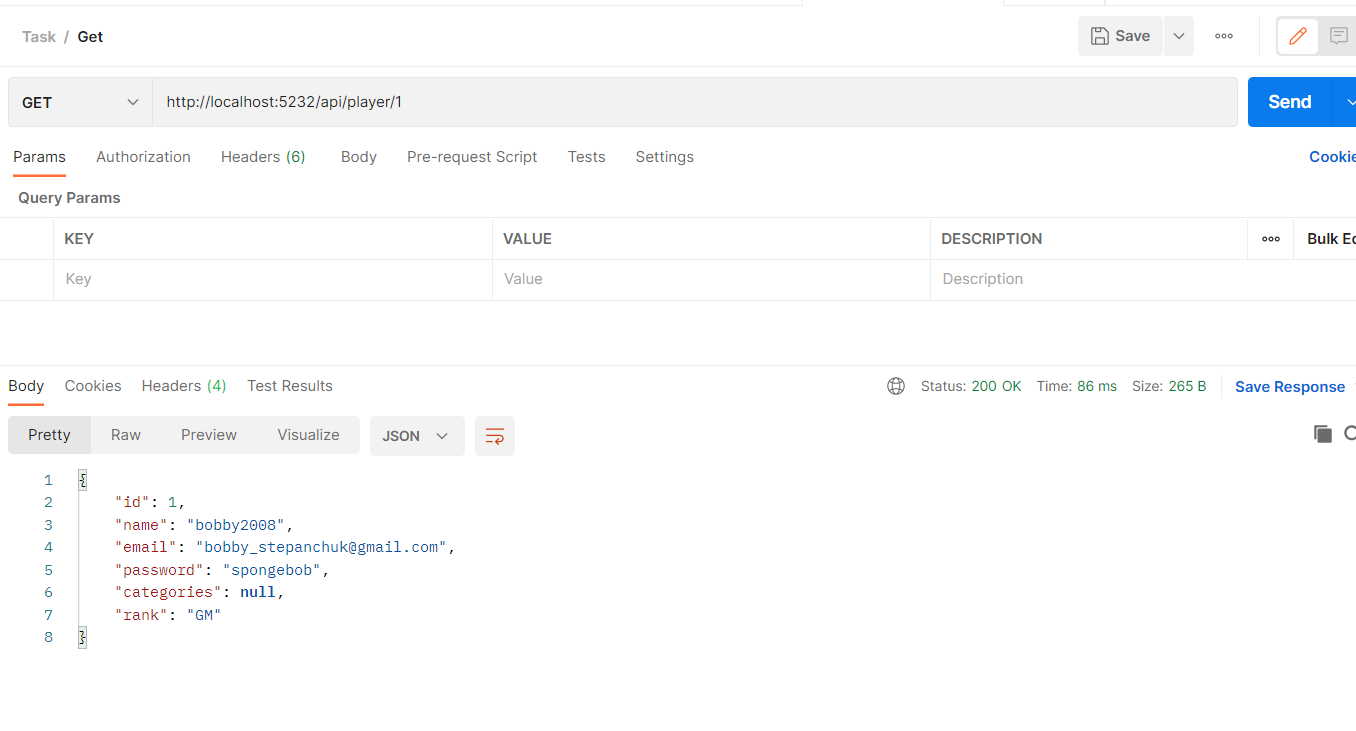
*Рис. 1. Відправлення POST запиту на контролер, та отримання позитивної відповіді.*



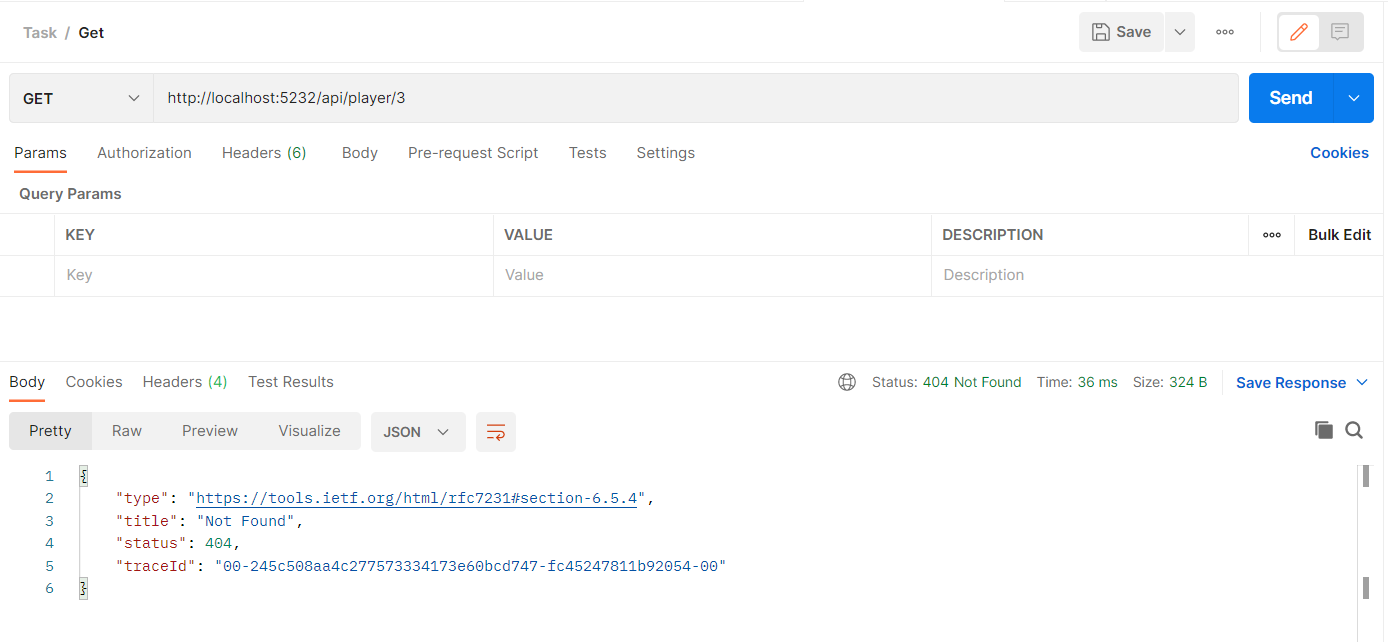
*Рис. 2. Відправлення POST запиту на контроллер та отримання помилки валідації*



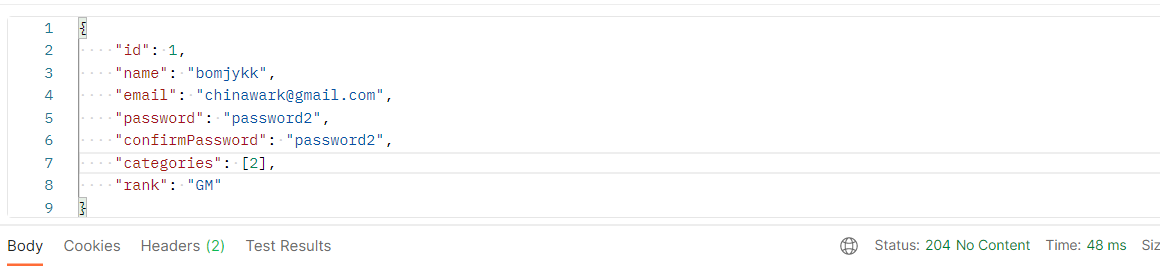
*Рис.3. Відправлення GET запиту для отримання даних усіх гравців.*



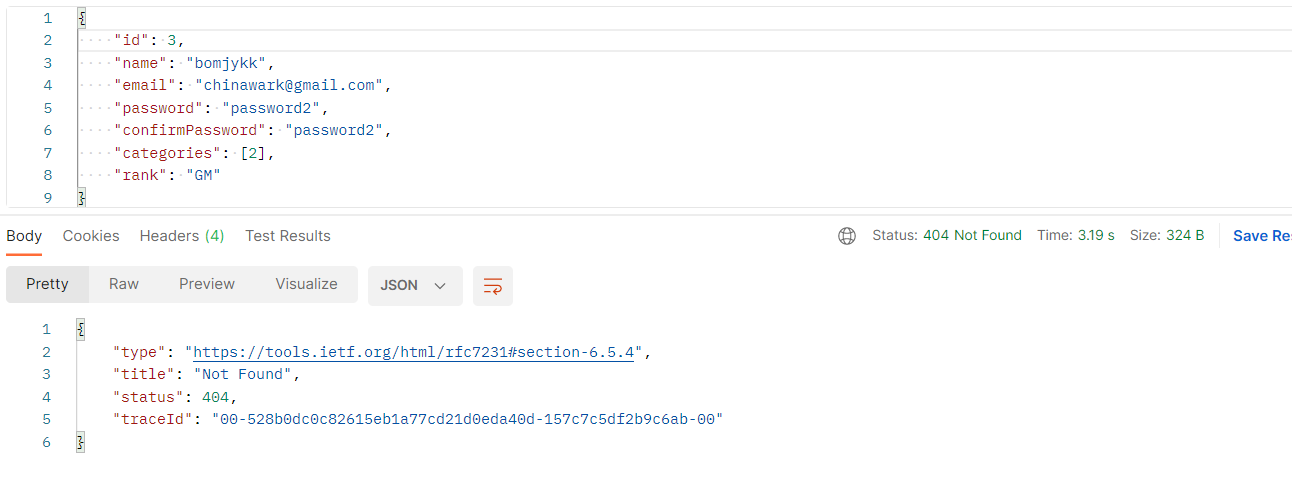
*Рис. 4. Відправлення GET запиту для отримання даних одного гравця.*



*Рис.5. Відправлення GET запиту для отримання даних одного гравця. «Гравця не знайдено»*



*Рис.6. Успішний результат виконання PUT запиту*



*Рис. 7. Результат PUT запиту на неіснуючого гравця.*

**Код програми**

**PlayerController.cs**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using WebApplication1.Models.DTO;

using WebApplication1.Models.Entities;

using WebApplication1.Models.Requests;

using WebApplication1.Services.Interfaces;

namespace WebApplication1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class PlayerController : ControllerBase

{

private readonly IPlayerService \_playerService;

private readonly ICategoryService \_categoryService;

private readonly List<Category> \_categories = new List<Category>()

{

new Category() { Id = 0, Name = "UA", },

new Category() { Id = 1, Name = "IT", },

};

private readonly List<Player> \_players = new List<Player>()

{

new Player() { Id = 0, Email = "jordano@gmail.com", Name = "jordano", Password = "maikLobovski", Rank = "FM"},

new Player() { Id = 1, Email = "bobby\_stepanchuk@gmail.com", Name = "bobby2008", Password = "spongebob", Rank = "GM"},

};

public PlayerController(IPlayerService playerService, ICategoryService categoryService)

{

\_playerService = playerService;

\_categoryService = categoryService;

}

[HttpGet]

public async Task<IActionResult> GetAll()

{

var players = \_players.ToList();

return Ok(players);

}

[HttpGet("{id}")]

public async Task<IActionResult> Get(int id)

{

var player = \_players.FirstOrDefault(p => p.Id == id);

if (player is null)

{

return NotFound();

}

return Ok(player);

}

[HttpPost]

public async Task<IActionResult> Insert([FromBody]InsertRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

\_players.Add(new Player()

{

Id = \_players.Count,

Email = request.Email,

Name = request.Name,

Password = request.Password,

Rank = request.Rank,

});

var player = \_players.Last();

return Created(nameof(Insert), player);

}

[HttpPut]

public async Task<IActionResult> Update([FromBody]UpdateRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

var player = \_players.FirstOrDefault(p => p.Id == request.Id);

if (player is null)

{

return NotFound();

}

player.Email = request.Email;

player.Id = request.Id;

player.Name = request.Name;

player.Password = request.Password;

player.Rank = request.Rank;

\_players.Remove(\_players.FirstOrDefault(p => p.Id == request.Id));

\_players.Add(player);

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> Delete(int id)

{

var player = \_players.FirstOrDefault(p => p.Id == id);

if (player is null)

{

return NotFound();

}

\_players.Remove(player);

return Ok();

}

}

}

**ВИСНОВКИ**

Під час виконання цієї лабораторної роботи я переніс раніше розроблену сторінку на React JS. Для стилізації було використано React Bootstrap з мінімальним набором елементів. З допомогою JavaScript було надіслано запит на api, та отримано і відображену картинку.

Посилання на репозиторій: https://github.com/AlexBurlak/PVI4