csarp-back-01-01-01-student-update

# Diák adatok frissítése a desktop alkalmazásban

(az előző, 07-00-01-wpf-mvvm-1-refactoring-backend folytatása)

## Diák adatok frissítése

### Service réteg

Először megírjuk a service réteget, amely a backenddel kommunikálva elvégzi a diák adatának frissítését.

Az adatok fogadása miatt szükség lesz a Newtonsoft.Json csomagra.

1. Interface bővítése

public interface IStudentService

{

public Task<List<Student>> SelectAllStudent();

**public Task<ControllerResponse> Update(StudentDto studentDto);**

}

2.

Implementáljuk az interface-t a StudentService osztályban majd elkészítjük a metódus visszatérési értékét:

public Task<ControllerResponse> Update(StudentDto studentDto)

{

**ControllerResponse defaultResponse = new();**

**return defaultResponse;**

}

Ha van http kliensünk, akkor az api hívást egy try-catch szerkezetbe helyezzük el (ez máshol is alkalmazható api híváskor), hogyha valami hiba keletkezik megjelenítjük azt a consolon:

ControllerResponse defaultResponse = new();

if (\_httpClient is not null)

{

try

{

}

catch (Exception ex)

{

Console.WriteLine($"{ex.Message}");

}

}

A httpClient segítségével elküldjük a post kérést a megadott url-re és becsomagolva a diák adatokat a kérelembe. Ezután kiolvassuk a választ, és a választ a Newston.Json csomag segítségével ConrollerResponse típusba konvertáljuk.

(a pirossal kiemelt rész nem jó a commitban, a végén lett korigálva)

try

{

**HttpResponseMessage httpResponse = await \_httpClient.PutAsJsonAsync("api/Student", studentDto);**

**string content = await httpResponse.Content.ReadAsStringAsync();**

**ControllerResponse? response = JsonConvert.DeserializeObject<ControllerResponse>(content);**

}

Ha van válasz, és az api kérés sikeressen lefutott, akkor visszatérünk az üres válasszal amiből kiolvasható lesz, hogy az api kérés sikeresen megtörténet.

Ha hiba történt kiírjuk a hibaüzenetet a consolre (később ezekből a kiírásokbol logok lehetnek).

ControllerResponse? response = JsonConvert.DeserializeObject<ControllerResponse>(content);

**if (response is not null)**

**{**

**if (**response.IsSuccess**)**

**{**

**return defaultResponse;**

**}**

**else**

**{**

**Console.WriteLine($"{response.Error}");**

**}**

**}**

A kód csak akkor ér a try-catch ágon túl, ha hiba történt. Egy elfogadható és a UI-on megjeleníthető hibaüzenetet rakunk a válaszba mielőtt visszatérünk a metódusból.

### ViewModel réteg

* DoSave és új Initialize async

[RelayCommand]

public async Task DoSave(Student newStudent)

{

if (\_studentService is not null)

{

ControllerResponse result = await \_studentService.Update(newStudent.ToStudentDto());

await UpdateView();

}

OnPropertyChanged(nameof(Students));

}

…

public override async Task InitializeAsync()

{

await UpdateView();

}

private async Task UpdateView()

{

if (\_studentService is not null)

{

List<Student> students = await \_studentService.SelectAllStudent();

Students = new ObservableCollection<Student>(students);

}

}

* Post helyett Put a StudentService rétegben

## Teszt

Stréber Szonja neve sikeresen módosítva Tanuló Szonjára!

