Deksop delete és updata

Bevezető

A desktop programunkban már működik a diák adatának módosítása backend oldali adatkezeléssel. Most a delete és update funkciót valósítjuk meg.

A backend kódok

Mivel a kiinduló kódunk nem tartalmazza a insert és update backend tananyag kódjait és a refaktorációkat, pótoljuk ezeket a backend oldalon (ha az ön kódja tartalmazza ezeket, akkor ezt a részt átlépheti):

- Repository réteg
- HasId a Student osztályba
- Controller réteg

Service réteg

A service réteget kiszerveztük egy új, Kreta.HttpService projektben. Itt található az IStudentService interface és a StudentService osztály.
Kialakítunk egy egységes service réteget, felhasználva az alábbi tudnivalókat:
https://stackoverflow.com/questions/45739753/correct-use-of-ensuresuccessstatuscode-and-issuccessstatuscode

- Interface a szükséges metódusokkal

```
public interface IStudentService
 {
     public Task<List<Student>> SelectAllStudent();
     public Task<ControllerResponse> Update(StudentDto studentDto);
     public Task<ControllerResponse> DeleteAsync(Guid id);
     public Task<ControllerResponse> InsertAsync(Student student);
 }
```

Először az update

- Először a backend oldali BadReqest választ dolgozzuk fel

```
if (httpResponse.StatusCode == HttpStatusCode.BadRequest)
{
    string content = await httpResponse.Content.ReadAsStringAsync();
    ControllerResponse? response = JsonConvert.DeserializeObject<ControllerResponse>(content);
    if (response is null)
    {
        defaultResponse.ClearAndAddError("A törlés http kérés hibát okozott!");
    }
    else return response;
}
```

- Minden más hiba esetbe úgy teszünk, mintha jól lefutott volna a http kérés (biztosítjuk a prototípusunk működését): `EnsureSuccessStatusCode`. Ilyen esetbe a defaultResponse változó a return előtt egy felhasználóbarát hibaüzenetet kap.

```
else if (!httpResponse.IsSuccessStatusCode)
{
        httpResponse.EnsureSuccessStatusCode();
}
```

- Ha nincs hiba, nem kerül semmi a defaultResponseba (akkor nincs hiba) és evvel az üres defaultResponsal térünk vissza

```
else
{
        return defaultResponse;
}
```

Az update teljes kódja:

```
public async Task<ControllerResponse> Update(StudentDto studentDto)
 {
     ControllerResponse defaultResponse = new();
     if (_httpClient is not null)
     {
         try
         {
             HttpResponseMessage httpResponse = await _httpClient.PutAsJsonAsync("api/Student", studentDto);
             if (httpResponse.StatusCode == HttpStatusCode.BadRequest)
             {
                 string content = await httpResponse.Content.ReadAsStringAsync();
                 ControllerResponse? response = JsonConvert.DeserializeObject<ControllerResponse>(content);
                 if (response is null)
                 {
                     defaultResponse.ClearAndAddError("A törlés http kérés hibát okozott!");
                 }
                 else return response;
             }
             else if (!httpResponse.IsSuccessStatusCode)
             {
                 httpResponse.EnsureSuccessStatusCode();
             }
             else
             {
                 return defaultResponse;
             }
         }
         catch (Exception ex)
         {
             Console.WriteLine($"{ex.Message}");
         }
     }
     defaultResponse.ClearAndAddError("Az adatok frissítés nem lehetséges!");
     return defaultResponse;
 }
```

Ez alapján írjuk meg a delete és insert kódokat:

- A törlés http kérés és kezelése
- Új adat mentése http kérés és kezelése

Frontend oldalon a StudentViewModel dolgozunk

Az update metódushoz hasonlóan átírjuk a törlés kódját. Refaktorálás során észrevesszük, hogy az OnPropertyChanged metódus hívásra nincs szükség.

A már megírt DoSave metódus OnPropertyChanged nélkül:

```
[RelayCommand]
public async Task DoSave(Student newStudent)
{
    if (_studentService is not null)
    {
        ControllerResponse result = await _studentService.Update(newStudent.ToStudentDto());
        if (!result.HasError)
        {
            await UpdateView();
        }
    }
}
```

Az új DoRemove metódus:

```
        [RelayCommand]
        public async Task DoRemove(Student studentToDelete)
        {
            if (_studentService is not null)
            {
                ControllerResponse result = await
_studentService.DeleteAsync(studentToDelete.Id);
```

```
            if (!result.HasError)
            {
                await UpdateView();
            }
        }
    }
```

- Diák adatának törlése backendről működik
- A mentés gombon az új diák adatának mentése és id-vel rendelkező diák adatának frissítése
- A Student ID alapértelmezett konstruktorban legyen Guid.Empy

Az új diák felvitele és diák törlése is működik:

- Stréber Szonja törölve
- Tornász Tomi felvéve