WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

Projekt

*Przedmiot**: Zarządzanie Danymi***

Prowadzący:

inż. Eryk Chrustek

Wykonawcy:

Wojciech Czapiga, w65469
Piotr Burakowski, w66745

Semestr, symbol kierunku i grupa:

6IIZ/2024-SP1

Wstęp

Prezentowany projekt jest stroną internetową przykładowej biblioteki. Jako miejsce fizyczne wymagające kontaktu interpersonalnego do spełnienia usług, aplikacja może co najwyżej pozwolić na udostępnienie danych wewnętrznych o stanie książek poprzez media informatyczne.

Specyfikacja Wymagań

- Dostęp do listy książek na stronie głównej
- Dostęp do informacji o API (Swagger)
- Dostęp do informacji o stanie i dostępności książek poprzez API
- Dostęp do informacji o stanie wynajmów poprzez API

Użyte technologie

- ASP.NET MVC C#
- Swagger
- Entity Framework 6.0
- .Net Core 6.0

Wizualna prezentacja strony

Powyżej widać stronę główną projektu. Nie zawiera ona na ten moment nic więcej niż odnośniki do podstron.

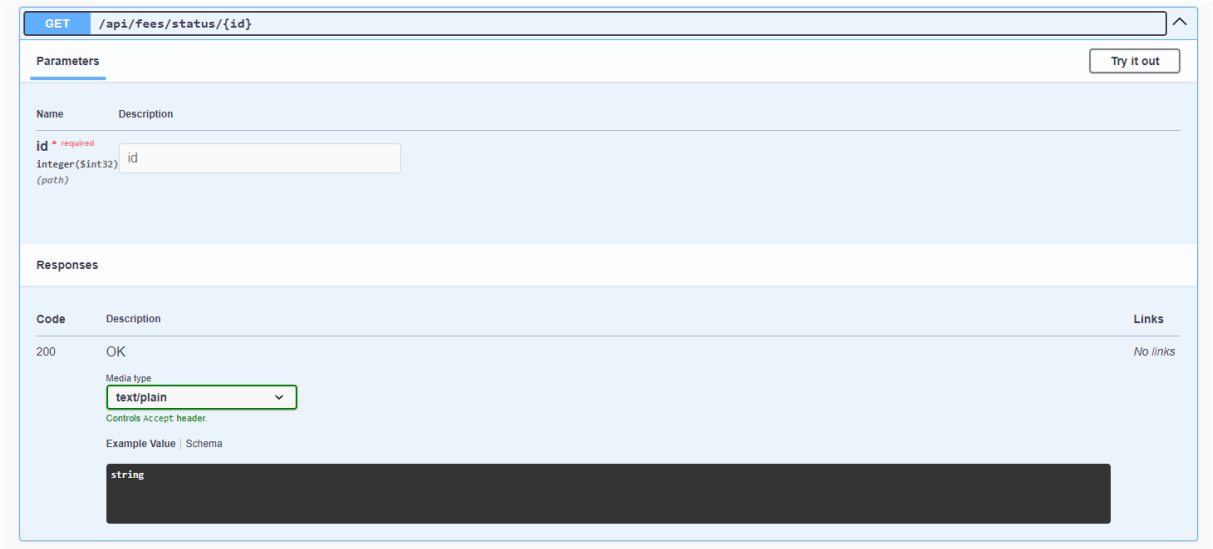
Następnie dostępny jest spis książek z przykładowymi danymi.

Ostatnią ważną częścią jest oczywiście Swagger.

API

Dostępne jest wiele funkcji pozwalających użytkownikowi na uzyskanie dokładnych informacji o stanie książek, opłat jak i funkcja pozwalająca na dogenerowanie brakujących opłat. Taka funkcja może na przykład zostać zastosowana przez jakąś aplikacje zewnętrzną organizującą dane.

Dostępny jest także spis schematów pozwalający na podgląd struktury każdego obiektu jak ich możliwe wartości.



Powyżej znajduje się przykładowa funkcja API pozwalająca na translację statusu opłaty z numerycznej na tekstową.

Baza danych

Za pomocą Entity Framework została zbudowana lokalna baza w SQLite na wzór powyższej struktury.

Kod

Strona wizualna aplikacji używa CSHTML łączący fragmenty HTML oraz C#

```
<div class="text-center">
    <h1>Nasze książki</h1>
    <table class="border-1 border-dark w-75" style="border: 1px solid black">
        <tr>
            <th>Tytuł</th>
            <th>Opis</th>
            <th>Autor</th>
            <th>Rok Pub.</th>
            <th>Kategoria</th>
            <th>Dostępna?</th>
        </tr>
    @foreach(var b in c.Books){
            var ava = b.CurrentlyAvailable();
            <tr style="background: @(!ava?"lightcoral":"transparent"); border-top: 0.5px solid black">
                <td>@b.Title</td>
                <td>@b.Description</td>
                <td>@b.Author</td>
                <td>@b.PublicationYear</td>
                <td>@b.Genre</td>
                <td>@(ava?"Dostępna":"Niedostępna")</td>
            </tr>
    }
    </table>
</div>
```

Taki kod kompilowany jest na HTML i wysyłany do użytkownika w momencie pobrania strony.

Backend, zbudowany za pomocą kontrolerów używa zaś C# połączonego ze stroną znacznikami

```
[HttpGet("overdue/fees/generate")]
Odwołania: 0
public ActionResult<List<Fee>> GenerateFees()
{
    List<Fee> fees = new();
    foreach(Rental r in PrimaryContext.Rentals.Where(f => f.DueDate <= DateTime.Today && !f.Returned).ToList())
    {
        var fee = new Fee() { Rental = r};
        fees.Add(fee);
        PrimaryContext.Fees.Add(fee);
    }
    PrimaryContext.SaveChanges();
    return fees;
}
[HttpGet("overdue/fees/{id}/due")]
Odwołania: 0
public ActionResult<double> GetFeeAmount(int id)
{
    if (PrimaryContext.Fees.FirstOrDefault(f => f.Id == id) is Fee fee)
        return fee.GetDueAmount();
    return NotFound();
}
```

Na powyższym obrazie widać dwie metody. Pierwsza, wywołana poprzez ”./api/overdue/fees/generate” pozwala na wygenerowanie nowych opłat i przy okazji zwrócenie owych nowo powstałych odsetek. Druga metoda wyszukuje opłatę po jej ID a następnie kalkuluje ilość pieniędzy jaką wypożyczający jest winny bibliotece.

Pełny kod dostępny jest na [Github.]