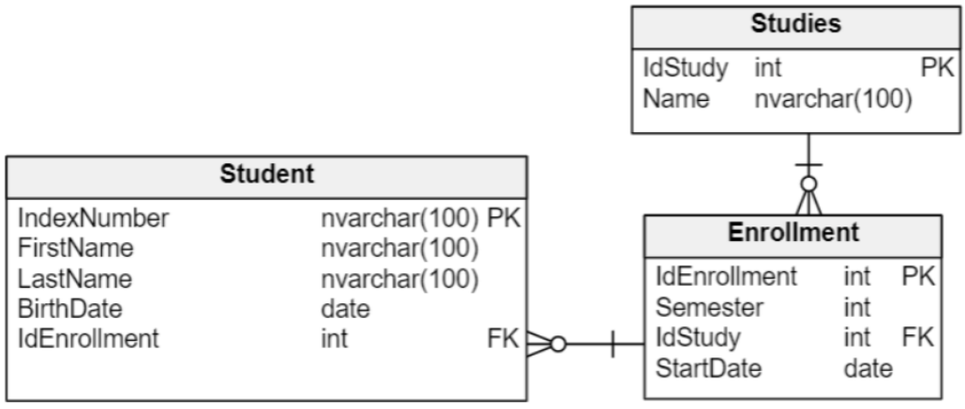
APBD - Ćwiczenie 5

pgago@pja.edu.pl

30 marca 2020

1 Wstęp

W niniejszych ćwiczeniach rozwijamy kod z poprzednich zajęć. Dla przypomnienia poniżej przedstawiony jest diagram bazy na której pracujemy.



Zadanie 1 - zapisanie studenta na studia

Ze względu na nowe wymagania musimy dodać do naszego kodu końcówkę, która pozwoli nam zarejestrować nowego studenta i zapisać go na studia.

- Należy dodać nowy kontroler o nazwie EnrollmentsController.
- Kontroler powinien odpowiadać na żądania kierowane na adres api/enrollments
- Kontroler powinien zawierać metodę, która pozwoli na przekazanie danych nowego studenta i zapisanie go na semestr.
- Poniżej przekazane jest żądanie na które powinien reagować nasz kontroler.

```
POST /api/enrollments HTTP/1.1
Host: localhost:51932
Content-Type: application/json

{
    "IndexNumber": "s1234",
"FirstName": "Andrzej",
"LastName": "Malewski",
"BirthDate": "30.03.1993"
"Studies": "IT"
}
```

- Końcówka powinna najpierw sprawdzić czy przekazane zostały wszystkie dane. W przeciwnym razie zwracamy błąd 400. Następnie sprawdzamy czy istnieją studia w tabeli Studies zgodne z wartością przesłaną przez klienta. W przeciwnym wypadku zwracamy błąd 400. Następnie odnajdujemy najnowszy wpis w tabeli Enrollments zgodny ze studiami studenta i wartością Semester=1 (student zapisuje się na pierwszy semestr). Jeśli tak wpis nie istnieje to dodajemy go do bazy danych (StartDate ustawiamy na aktualną datę). Na końcu dodajemy wpis w tabeli Students. Pamiętamy o tym, aby sprawdzić czy indeks podany przez studenta jest unikalny. W przeciwnym wypadku zgłaszamy błąd.
- Wszystkie opisane czynności powinni się odbyć w ramach pojedynczej transakcja. Jeśli zaszedł jakikolwiek błąd chcemy wycofać (rollback) wszystkie zmiany. Wykorzystaj klasę SqlConnection i metodę BeginTransaction() przedstawioną na ostatnim wykładzie.
- Jeśli student został poprawnie zapisany na semestr to zwracamy kod 201. W ciele żądania zwracamy przypisany do studenta obiekt Enrollment reprezentujący semestr na który został wpisany.

Zadanie 2 - dodajemy końcówkę do promocji studentów

W tym zadaniu dodajemy nową końcówkę, która umożliwi nam promocję studentów na nowy semestr.

- Dodajemy nową końcówkę do kontrolera Enrollments. Końcówke powinna reagować na żądanie POST wysłane na adres /api/enrollments/promotions. Poniżej mamy przykład żądania.

```
POST /api/enrollments/promotions HTTP/1.1
Host: localhost:51932
Content-Type: application/json

{
"Studies": "IT",
"Semester": 1
}
```

- Powinniśmy upewnić się, że w tabeli Enrollment istnieje wpis o podanej wartości Studies i Semester. W przeciwnym razie zwracamy błąd 404 Not Found.
- Następnie powinniśmy dodać do bazy danych procedurę składowaną. Tym razem nasza końcówka skorzysta z procedury w celu realizacji wymaganej funkcjonalności. Procedura powinna przyjmować parametry Studies i Semester. Następnie procedura aktualizuje wszystkich studentów wpisanych na dany semestr. Każdy z nich jest awansowany na nowy semestr. W tym wypadku szukamy wpisu w tabeli Enrollment, który dotyczy tych samych studiów z wartością Semester większą o 1. Jeśli takiego wpisu nie ma to go dodajemy. Na końcu aktualizuje pole IdEnrollment u danych studentów.
- Pamiętamy, że w tym wypadku uruchamiamy procedurę składowaną. Całą logika powinna być zawarta wewnątrz niej.
- Na końcu zwracamy kod 201 wraz z zawartością reprezentującą nowy obiekt Enrollment.

Zadanie 3 - dodanie osobnej warstwy komunikacji z bazą danych

Ostatnim zadaniem jest umieszczenie całej logiki bazodanowej w klasie poza naszym kontrolerem.

- Dodajemy nowy folder Services.
- Wewnątrz folderu dodajemy interfejs IStudentsDbService. Interfejs powinien definiować wszystkie niezbędne metody wykorzystywane w naszych kontrolerach.
- Następnie tworzymy konkretną implementację wspomnianego interfejsu o nazwie SqlServerDbService. Wewnątrz umieszczmy kod korzystający z SqlConnection i SqlCommand.
- Warstwę dostępu do danych wstrzykujemy poprzez konstruktor naszych kontrolerów. Pamiętamy o zarejestrowaniu zależności w klasie Startup.cs
- Na końcu zadania cała logika bazodanowa powinna być wydzielona do osobnej klasy.