|  |  |  |
| --- | --- | --- |
| POLITECHNIKA WARSZAWSKA  Wydział Elektryczny | **Projekt w ramach przedmiotu:**  Testowanie Oprogramowania | |
| Członkowie zespołu:  Dawid Bieniek  Sebastian Górka  Andrzej Grabowski  Miłosz Moroz  Alexey Selivonets | Dzień tygodnia: piątek  Godzina: 18:00  Data: 28.10.2022 | Ocena |

1. Wstęp

Celem projektu było stworzenie metod CRUD (create, read, update, delete) oraz ich przetestowanie. Zastosowaną w projekcie technologią jest C#, .NET SDK oraz Docker Compose.

1. Tematyka

Wybranym tematem projektu jest stworzenie strony internetowej z filmami. API umożliwia:

* dodanie filmu
* dodanie reżysera
* dodanie aktora
* pobranie listy filmów wraz z reżyserami i aktorami

1. Actor
2. int Id
3. string FirstName
4. string LastName
5. DateTime DateOfBirth

Klasa aktora zawiera 4 pola – Id typu integer informujące o numerze identyfikacyjnym aktora, FirstName typu string informujące o imieniu aktora, LastName typu string informujące o nazwisku aktora oraz DateOfBirth typu DateTime informujące o dacie urodzenia aktora.

1. Director
2. int Id
3. string FirstName
4. string LastName
5. DateTime DateOfBirth

Klasa reżysera zawiera 4 pola – Id typu integer informujące o numerze identyfikacyjnym reżysera, FirstName typu string informujące o imieniu reżysera, LastName typu string informujące o nazwisku reżysera oraz DateOfBirth typu DateTime informujące o dacie urodzenia reżysera.

1. Movie
2. int Id
3. string Name
4. DateTime ProductionYear
5. int BoxOffice

Klasa filmu zawiera 4 pola – Id typu integer informujące o numerze identyfikacyjnym filmu, Name typu string informujące o nazwie filmu, ProductionYear typu DateTime informujące o roku produkcji filmu oraz BoxOffice typu int informujące o ocenie filmu.

1. Tools
2. Windows
3. setup.ps1 - inicjalizujący skrypt
4. start-application.ps1 - skrypt uruchamiający
5. cleanup.ps1 - skrypt czyszczący z artefaktów
6. Unix
7. setup-install-docker.sh - skrypt instalujący Docker Compose
8. setup.sh - inicjalizujący skrypt
9. start-application.sh - skrypt uruchamiający
10. cleanup.sh - skrypt czyszczący z artefaktów
11. Controllers
12. Actor
13. [HttpGet]

* GetAll()
* GetSingle(int Id)

1. [HttpPost]
   * AddActor([FromBody] Actor actor)
   * Update(Actor actor)
   * Remove(Actor actor)
2. Movies
3. [HttpGet]

* GetAll()

1. [HttpPost]

* Add([FromBody] Actor actor)

1. Database

Zostało stworzonych 5 tabel:

* Movie
* Actor
* Director
* ActorMovie
* DirectorMovie

1. Testy
   * 1. Controllers

* ActorController\_GetAll\_ReturnOk()

- pobranie wszystkich aktorów

* ActorController\_GetSingle\_ReturnOk()

- pobranie jednego aktora

* ActorController\_AddActor\_ReturnOk()

- dodanie aktora

* ActorController\_Remove\_ReturnOk()

- usunięcie aktora

* ActorController\_Update\_ReturnOk()

- uaktualnienie aktora

* + 1. Repository
* GetSingle\_Returns\_1Actor()

- pobranie 1 aktora z repozytorium

* GetAllWhenContainsElements\_Returns\_AllElements()

- pobranie wszystkich aktorów z niepustego repozytorium

* GetAllWhenEmpty\_Returns\_EmptyList()

- pobranie wszystkich aktorów z pustego repozytorium

* GetQueryableMockDbSet<T>(List<T> sourceList)

- pobranie mocka dbSet

1. Podsumowanie

W repozytorium został ustawiony pipeline służący do automatycznego testowania. Wykonane testy są od siebie niezależne.