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.

2. Tematyka

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

- dodanie filmu
- dodanie reżysera
- dodanie aktora
- dodanie relacji aktora z filmem
- dodanie relacji reżysera z filmem
- pobranie listy filmów wraz z reżyserami i aktorami

3. Actor

I. int Id
II. string FirstName
III. string LastName
IV. 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.

4. Director

I. int Id
II. string FirstName
III. string LastName
IV. 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.

5. Movie

I. int Id
II. string Name
III. DateTime ProductionYear
IV. 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.

6. Tools

I. Windows
   a) setup.ps1 - inicjalizujący skrypt
   b) start-application.ps1 - skrypt uruchamiający
   c) cleanup.ps1 - skrypt czyszczący z artefaktów
II. Unix
   a) setup-install-docker.sh - skrypt instalujący Docker Compose
   b) setup.sh - inicjalizujący skrypt
   c) start-application.sh - skrypt uruchamiający
   d) cleanup.sh - skrypt czyszczący z artefaktów

7. Controllers

I. Actor
   a) `[HttpGet]`
      - `GetAll()`
      - `GetSingle(int Id)`
   b) `[HttpPost]`
      - `AddActor([FromBody] Actor actor)`
      - `Update(Actor actor)`
      - `Remove(Actor actor)`
II. Movies
   a) `[HttpGet]`
      - `GetAll()`
   b) `[HttpPost]`
      - `Add([FromBody] Actor actor)`

8. Database

Zostało stworzonych 5 tabel:

- Movie
- Actor
- Director

- ActorMovie
- DirectorMovie

9. Testy

I. 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

II. 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

10. Podsumowanie

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