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
- 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. Director
   a) `[HttpGet]`
      - `GetAll()`
      - `GetSingle(int Id)`
   b) `[HttpPost]`
      - `AddDirector([FromBody] Director director)`
      - `Update(Director direactor)`
      - `Remove(Director director)`

III. Movies
   a) `[HttpGet]`
      - `GetAll()`
      - `GetSingle(int Id)`
   b) `[HttpPost]`
      - `AddMovie([FromBody] Movie Movie)`

- `Update(Movie Movie)`
- `Remove(Movie Movie)`

8. Database

Zostało stworzonych 5 tabel:

- Movie
- Actor
- Director
- ActorMovie
- DirectorMovie

9. Testy

I. Controllers

a) actor

- 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

b) director

- DirctorController_GetAll_ReturnOk()
  - pobranie wszystkich reżyserów
- DirctorController_GetSingle_ReturnOk()
  - pobranie jednego reżysera
- DirctorController_AddActor_ReturnOk()
  - dodanie reżysera
- DirctorController_Remove_ReturnOk()
  - usunięcie reżysera
- DirctorController_Update_ReturnOk()
  - uaktualnienie reżysera

c) movies

- MoviesController_GetAll_ReturnOk()
  - pobranie wszystkich filmów
- MoviesController_GetSingle_ReturnOk()
  - pobranie jednego filmu
- MoviesController_AddActor_ReturnOk()
  - dodanie filmu
- MoviesController_Remove_ReturnOk()
  - usunięcie filmu
- MoviesController_Update_ReturnOk()
  - uaktualnienie filmu

II. Repository

a) actor

- GetSingle_ValidId_RecordWithGivenIdReturned()
- GetSingle_InvalidId_NullReturned(int checkedId)
- GetAllAsync_PopulatedTable_PopulatedListReturned()
- GetAllAsync_EmptyTable_EmptyListReturned()
- Add_NValidRecords_DbSetAddCalledNTimes(int recordsToAdd)
- Remove_NValidRecords_DbSetRemoveCalledNTimes(int recordsToRemove)
- Remove_InvalidRecord_DbSetRemoveCalledOnce()
- Remove_FromEmptyTable_DbSetRemoveCalledOnce()
- Update_NValidRecords_DbSetUpdateCalledNTimes(int recordsToRemove)
- Update_InvalidRecord_DbSetUpdateCalledOnce()
- Update_FromEmptyTable_DbSetUpdateCalledOnce()

b) director

- GetSingle_ValidId_RecordWithGivenIdReturned()
- GetSingle_InvalidId_NullReturned(int checkedId)
- GetAllAsync_PopulatedTable_PopulatedListReturned()
- GetAllAsync_EmptyTable_EmptyListReturned()
- Add_NValidRecords_DbSetAddCalledNTimes(int recordsToAdd)
- Remove_NValidRecords_DbSetRemoveCalledNTimes(int recordsToRemove)
- Remove_InvalidRecord_DbSetRemoveCalledOnce()
- Remove_FromEmptyTable_DbSetRemoveCalledOnce()
- Update_NValidRecords_DbSetUpdateCalledNTimes(int recordsToRemove)
- Update_InvalidRecord_DbSetUpdateCalledOnce()
- Update_FromEmptyTable_DbSetUpdateCalledOnce()

c) movie

- GetSingle_ValidId_RecordWithGivenIdReturned()
- GetSingle_InvalidId_NullReturned(int checkedId)
- GetAllAsync_PopulatedTable_PopulatedListReturned()
- GetAllAsync_EmptyTable_EmptyListReturned()
- Add_NValidRecords_DbSetAddCalledNTimes(int recordsToAdd)
- Remove_NValidRecords_DbSetRemoveCalledNTimes(int recordsToRemove)
- Remove_InvalidRecord_DbSetRemoveCalledOnce()
- Remove_FromEmptyTable_DbSetRemoveCalledOnce()
- Update_NValidRecords_DbSetUpdateCalledNTimes(int recordsToRemove)
- Update_InvalidRecord_DbSetUpdateCalledOnce()
- Update_FromEmptyTable_DbSetUpdateCalledOnce()

10. Podsumowanie

W repozytorium został ustawiony pipeline służący do automatycznego testowania. Wykonane testy są od siebie niezależne. W sumie zostało wykonanych 15 testów do kontrolerów oraz 33 do repozytorium - razem 48 testów. Kod został napisany zgodnie z systemem notacji ciągów tekstowych camelCase.