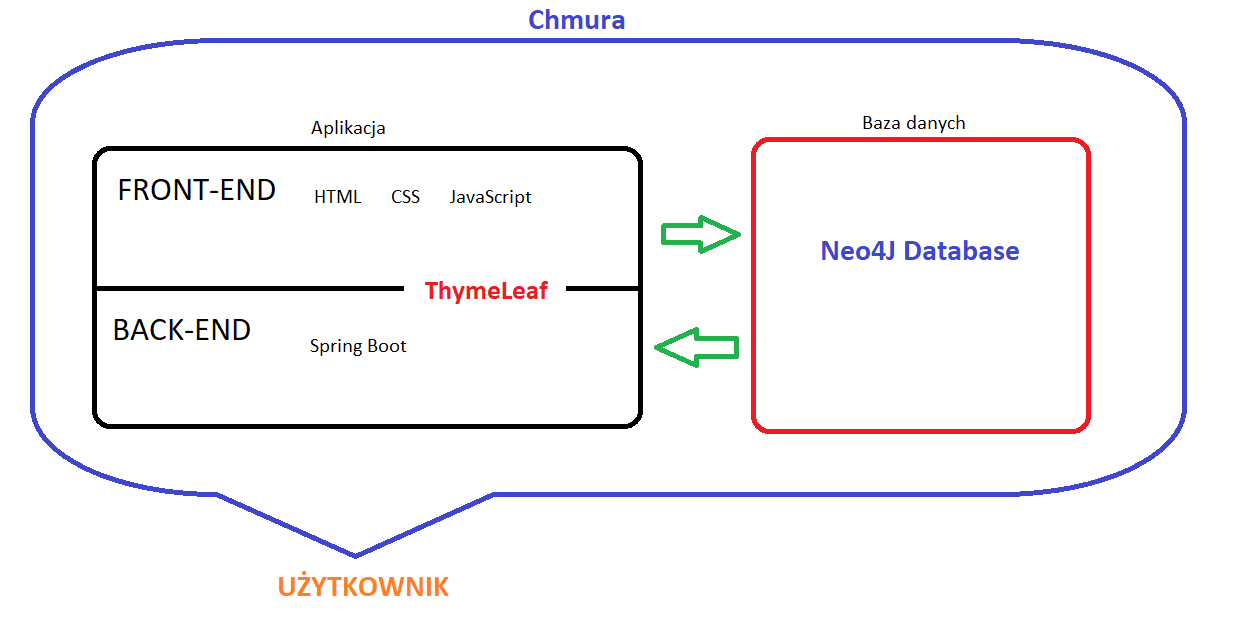
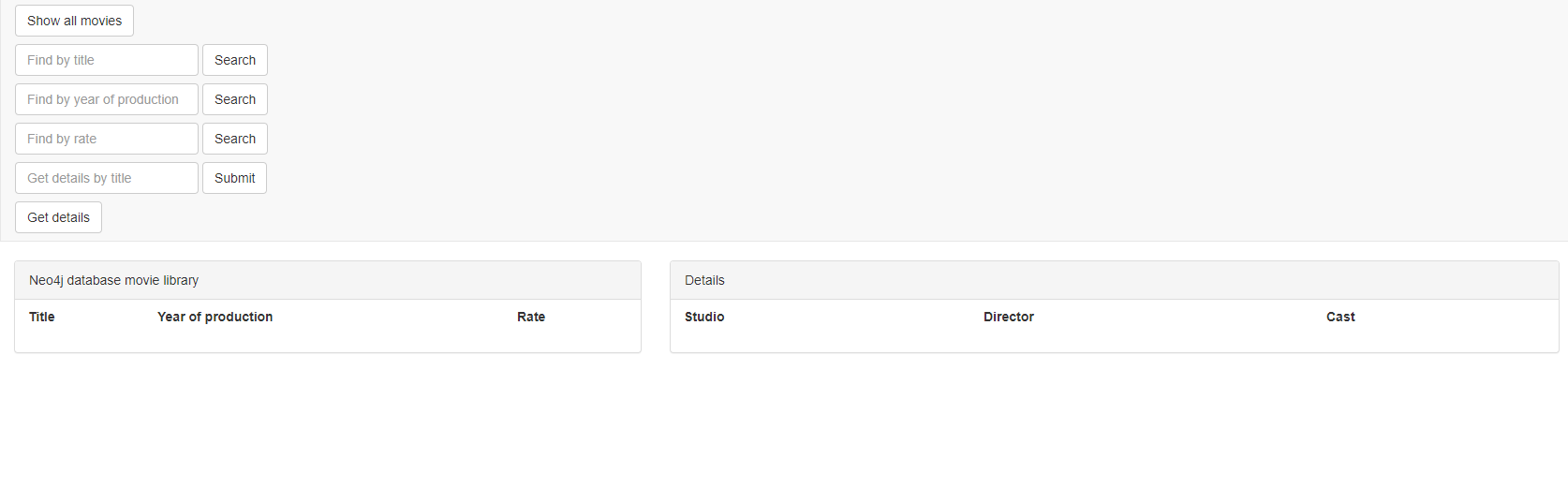
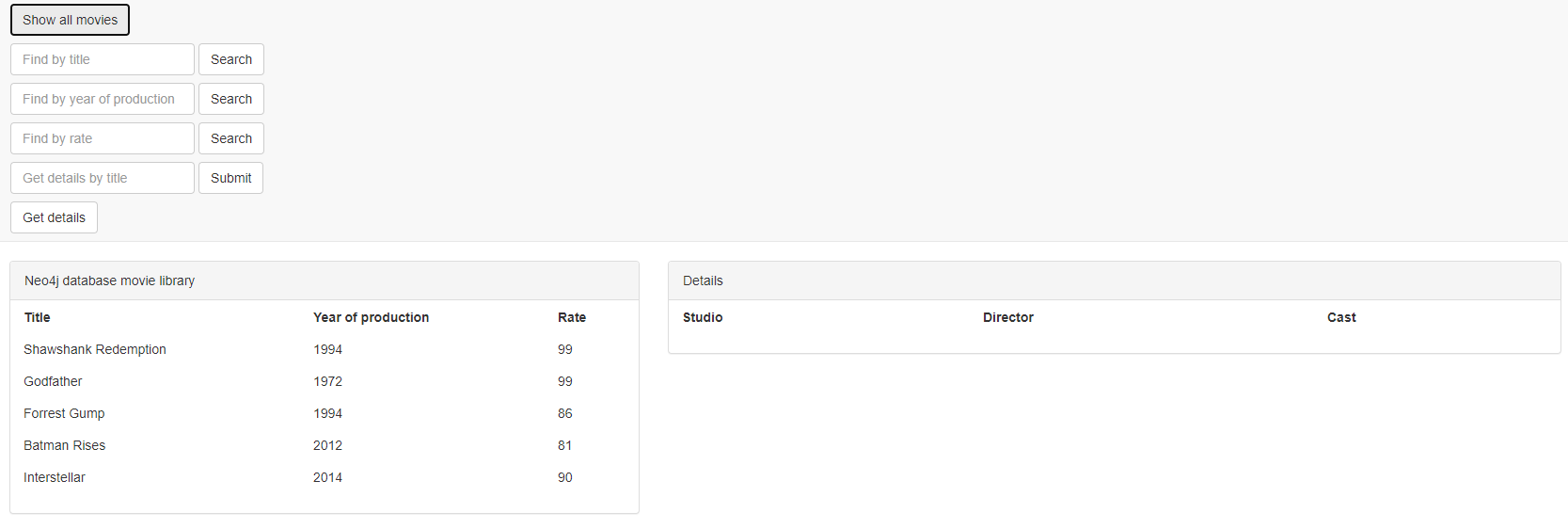
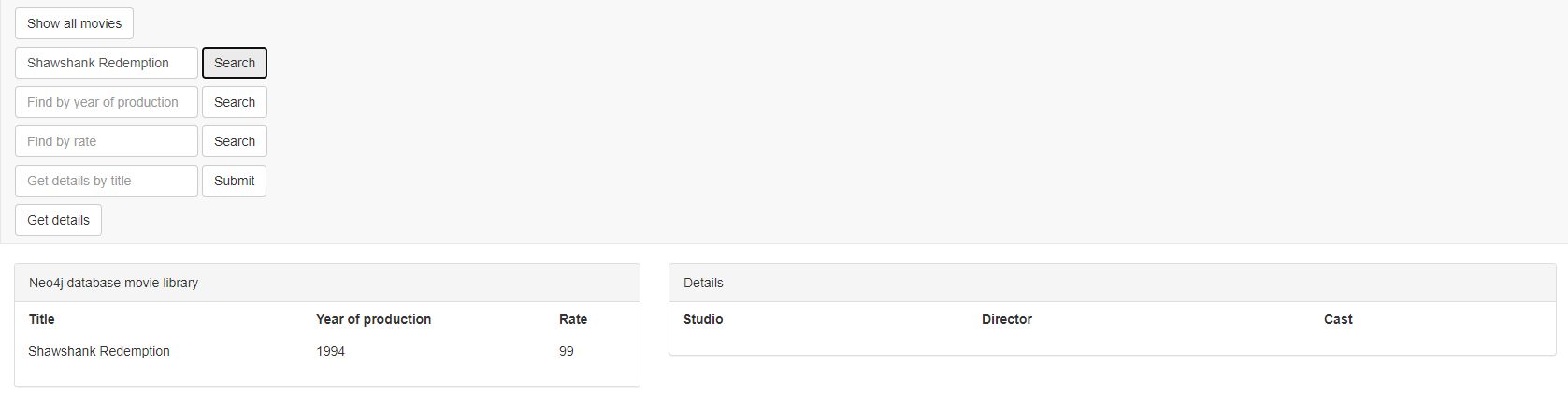
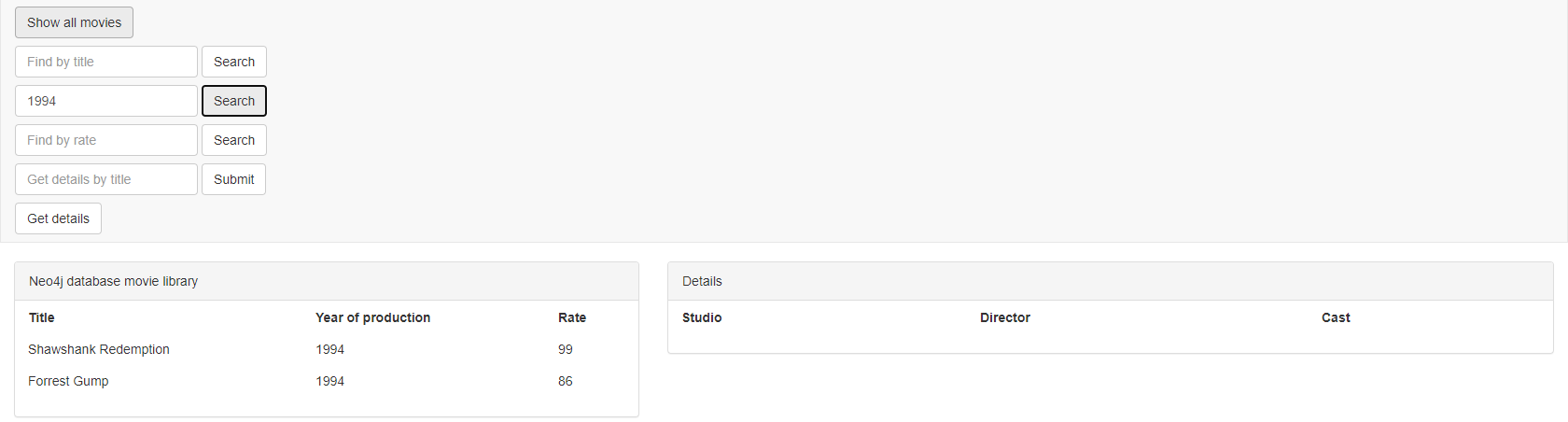
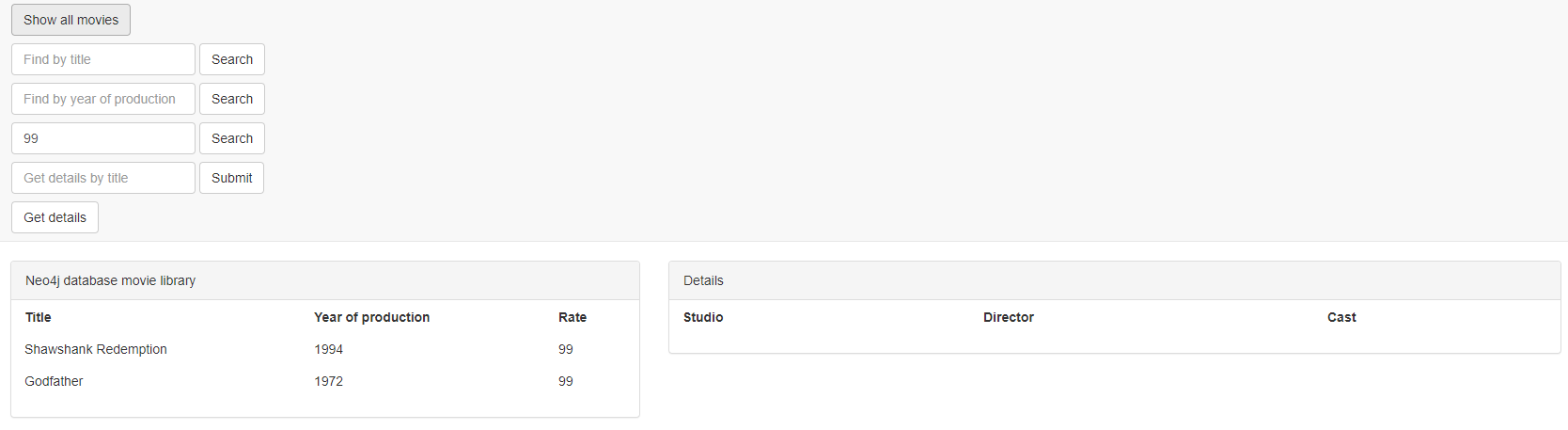
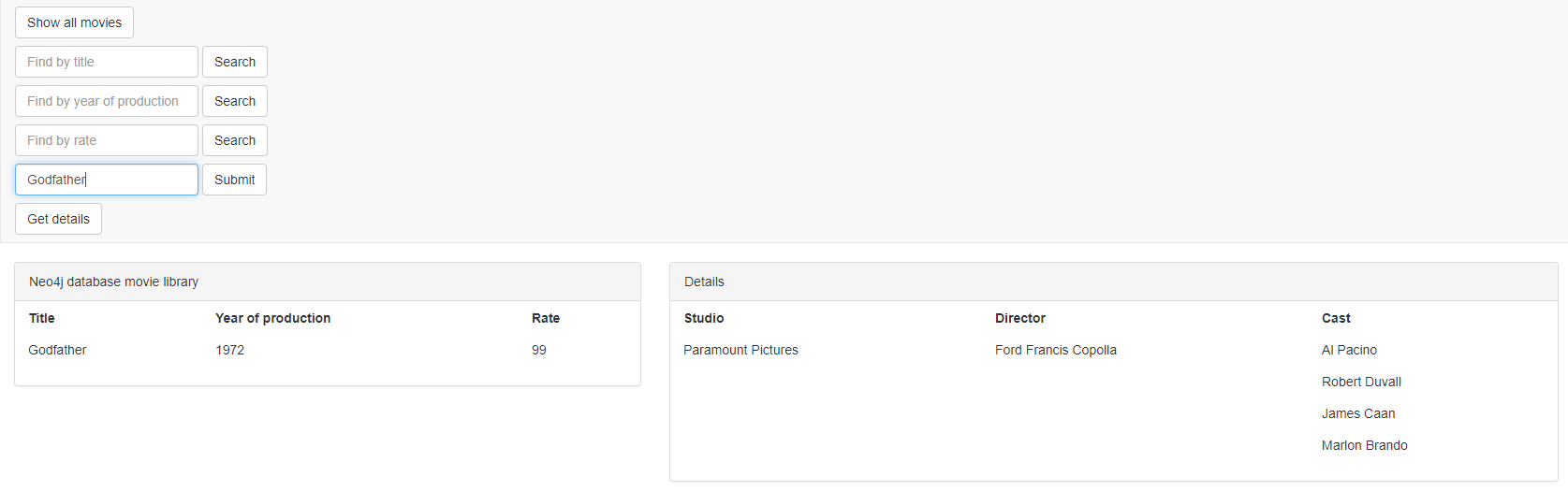
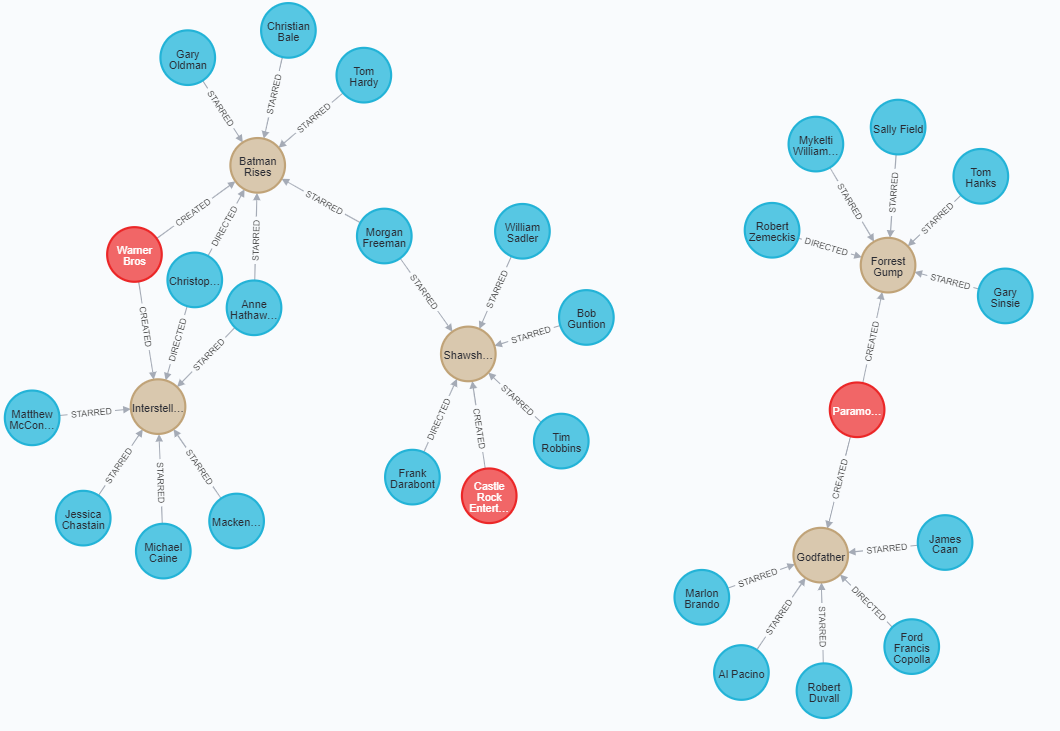
Michał Stefaniuk

Przetwarzanie w Chmurach Obliczeniowych 2020/2021  
Projekt Semestralny

1. Opis projektu  
     
   Celem projektu było stworzenie aplikacji webowej osadzonej w chmurze, która za pomocą operacji REST komunikuje się z grafową bazą danych Neo4j, również osadzoną w chmurze.  
     
   Link do aplikacji: (...)  
     
   Tematem aplikacji jest serwis odwzorowujący bibliotekę filmową, dający możliwość użytkownikowi zwracać zawartość bazy i wysyłać do niej bardziej skomplikowane zapytania.  
     
     
     
   Do stworzenia aplikacji wykorzystano technologie:  
   - Spring Boot  
   - Thymeleaf  
   - Javascript  
   - CSS  
   - HTML  
     
   Zdecydowano się na użycie Spring Boot, gdyż jest to narzędzie umożliwiające w bardzo intuicyjny i zwinny sposób implementowanie warstwy back-end serwisów webowych. Thymeleaf to plugin do Spring, za pomocą którego przesyłane są informacje między warstwą back-end oraz front-end. Pozostałe technologie (JS + HTML + CSS) są dosyć oczywistym wyborem przy reprezentowaniu interfejsu graficznego serwisu.
2. Przykładowe użycie  
     
   a) Okno aplikacji  
     
   b) Listowanie wszystkich filmów w bazie (“Show all movies”)  
     
     
   c) Szukanie filmów po tytule  
     
     
   d) Szukanie filmów po roku produkcji  
     
     
   e) Szukanie filmów po ocenie  
     
     
   f) Wyświetlanie szczegółów filmu (“Get details”)  
   
3. Implementacja (Diagram) (Kod + Database)  
     
   a) Baza danych  
     
     
     
   Zawartość bazy danych zaprezentowana jest na grafice powyżej. Komendy za pomocą, których stworzono bazę znajdują się w pliku “queries.sql”.  
     
   W bazie głównymi obiektami zainteresowania są filmy (aktualnie jest ich 5), połączone relacjami z takimi obiektami jak studio filmowe czy osoba.  
     
   Osoba może połączyć się z filmem relacją “STARRED” dla aktorów, którzy wystąpili w filmie lub “DIRECTED” jeśli osoba była reżyserem filmu. Studia filmowe łączą się z filmami relacją “CREATED”. Jak widać na diagramie istnieją relacje, gdzie studio wyprodukowało więcej niż jeden film, bądź relacje gdzie aktor wystąpił w kilku filmach.  
     
   Typy obiektów:  
   PERSON(name, born)  
   MOVIE(released, score, title)  
   STUDIO(name)  
     
   Typy relacji:  
   PERSON - **STARRED**(in) -> MOVIE  
   PERSON - **DIRECTED** -> MOVIE  
   STUDIO - **CREATED** -> MOVIE  
   Baza została osadzona w chmurze i zdeployowana w serwisie (...)  
     
   b) Implementacja od strony kodu - backend  
     
   W zasadzie cała logika biznesowa serwisu znajduje się w jednym kontrolerze w pliku **MovieLibraryController.java.**Jest to komponent Spring Boot realizujący REST’owe polecenia (anotacja @Controller). Wewnątrz znajdują się dwa endpointy. Pierwszy jest endpointem testowym, w którym można zrealizować sobie dowolne zapytanie do bazy, które zostanie wyrenderowane do **testpage.html.**Właściwa funkcjonalność serwisu znajduje się w endpoincie obsłużonym funkcją **getAllMovies().** Obsługuje on ścieżkę “/”, a więc symuluje on SAP (Single page application) i realizuje metody GET oraz POST. Wewnątrz kontrolera otwierana jest sesja i w przypadku zawołania endpointu, a więc użycia metody GET, zostaną wykonane trzy zapytania w kodzie Cypher do bazy Neo4j. Każde z nich zwraca odpowiednio listę tytułów filmów, listę z rankingiem poszczególnych filmów oraz listę z datą produkcji filmów. Po wykonaniu zapytań listy te zostają dodane do modelu za pomocą **addAttribute()** dzięki czemu stają się one dostępne z poziomu kodu HTML. Tak właśnie Thymeleaf realizuje komunikacją między Spring a warstwą front.   
     
   W przypadku, gdy endpoint ten zostanie zawołany metodą POST, a więc zostanie do niego dostarczona jakaś informacja (w tym przypadku String movieTitle przechowujący tytuł filmu) - to zostaną wykonane poszczególne zapytania do bazy, ale już w oparciu o parametr movieTitle. Konkretnie do bazy zostaną wysłane zapytania o studio które wyprodukowało film, reżysera filmu, aktorów oraz dodatkowo tytuł zostanie również zwrócony. Następnie tak jak w przypadku metody GET, pobrane atrybuty zostają dodane poprzez Thymeleaf do przestrzeni dostępnej z poziomu kodu HTML. W zapytaniach tych uwzględnione zostają także relacje między obiektami, jak na przykład:  
     
   *“MATCH (a:Movie {title: '" + movieTitle +"'}) OPTIONAL MATCH (a)<-[:CREATED]-(x) RETURN x”*Aplikacja zostaje uruchomiona w **DemoApplication.java** za pomocą wywołania metody **run()** na obiekcie *SpringApplication.*   
     
   Co jest również ważne, to że połączenie z bazą zostaje nawiązane “pod spodem”, a nie z poziomu kodu jak to się dzieje zazwyczaj. Dane bazy podajemy w komponencie Springa **application.properties** i po podaniu URL, loginu i hasła do bazy mamy połączenie z bazą.  
     
   c) Implementacja od strony kodu - frontend  
     
   Strona główna aplikacji to kod pliku **mainPage.html**. Wewnątrz znajdują się trzy komponenty. Pierwszy przechowujący przyciski do obsługi serwisu i pozostałe dwa, które są tabelami. Domyślnie są one puste i wypełniają się takimi danymi o jakie poprosi użytkownik. Co jest ciekawe to na samym początku ciała dokumentu inicjalizuje zmienne w kodzie JavaScript, które są niczym innym jak wyłuskaniem wartości, które Thymeleaf przetrzymuje dla nas w przestrzeni globalnej. Dzięki temu możemy wykorzystać w naszym dokumencie to co przygotował dla nas Spring po stronie backend.  
     
   Obsługa dynamicznej warstwy strony to plik **action.js.** Znajdujące się w nim funkcje:  
     
   > clearDetailsTable() - czyści tabele przed ponownym jej uzupełnieniem  
   > appendToTable() - dodaje do tabeli wszystkie filmy  
   > searchByTitle() - dodaje tylko wyszukany po tytule film do tabeli  
   > searchByYear() - dodaje tylko wyszukany po roku produkcji film do tabeli  
   > searchByScore() - dodaje tylko wyszukany po rankingu film do tabeli  
   > getDetails() - pobiera szczegóły dotyczące filmu i umiezcza w tabeli  
   > detailsAlert() - informuje usera w formie alertu o tytule filmu  
   > detailsByTitle() - dodaje do tabeli informacje o filme, którego dotyczą wyświetlane szczegóły  
     
   Oprócz tego znajduje się także plik **main.css** który po prostu zawiera stylowanie komponentów strony.  
     
   Aplikacja została zdeployowana w serwisie Heroku.