

Technologie wspierające wytwarzanie oprogramowania.

Laboratorium nr 2. Vagrant

Przemysław Pietrzak (238083)

Piątek, 13:15

1 Opis wdrażanej aplikacji

W ramach wykonywanego zadania wdrożono aplikację "Cinema Tickets Application" przygotowaną na zajęcia "Projektowanie i implementacja systemów webowych". Aplikacja przeznaczona jest dla kin, które chcą ułatwić swoim klientom rezerwację stanowisk na sali kinowej oraz planowanie seansów. Użytkownicy po zalogowaniu mogą wyszukać seans, którym są zainteresowani oraz zarezerwować miejsca, którymi są zainteresowani.

Aplikacja składa się z trzech komponentów. Części frontendowej odpowiadającej za interfejs użytkownika, części backendowej odpowiadającej za logikę biznesową oraz bazy danych.

2 Wykorzystywane technologie

W warstwie prezentacji wykorzystano bibliotekę React¹ do budowania komponentów składających się na interfejs użytkownika oraz bibliotekę Redux² do zarządzania stanem aplikacji. Całość uruchomiona jest w środowisku Node³, natomiast build produkcyjny pakowany jest to pliku .jar, który uruchamiany jest na serwerze Tomcat⁴.

Do implementacji warstwy logiki biznesowej został wykorzystany framework Spring Boot⁵ oraz framework Hibernate⁶ umożliwiający łatwo komunikację z bazą danych z poziomu aplikacji. Całość została zaimplementowana przy wykorzystaniu języka Java⁷ w wersji 11.

¹React. A JavaScript library for building user interfaces. <https://reactjs.org>

²Redux. A Predictable State Container for JS Apps. <https://redux.js.org>

³Node.js. <https://nodejs.org/en>

⁴Apache Tomcat. <http://tomcat.apache.org>

⁵Spring Boot. <https://spring.io/projects/spring-boot>

⁶Hibernate Everything data. <https://hibernate.org>

⁷Java. <https://java.com>

W warstwie danych wykorzystany został natomiast system zarządzania relacyjną bazą danych MariaDB⁸.

3 Zrzuty ekranu

Rysunek 1: Strona logowania

[← Login page](#)

Welcome

Username / Email

root

Password

👁

☒ Remember Me

Log in!

If you don't have account yet, please:

[Create account](#)


Rysunek 2: Główna strona serwisu

Cinema tickets sale system [Log out](#)


Title: [Search](#)

Date: → [🗲](#)


Genre:



Schindler's List
Release Date: 03 Feb 1994



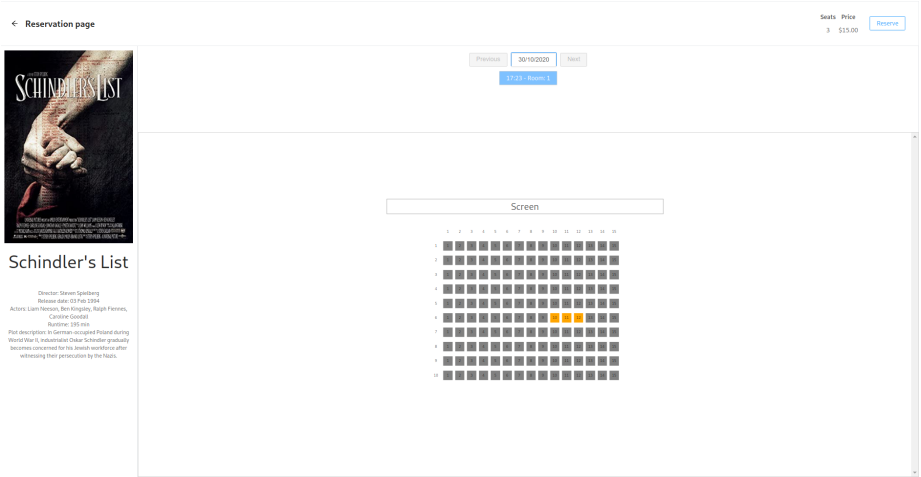
The Matrix
Release Date: 30 Mar 1999



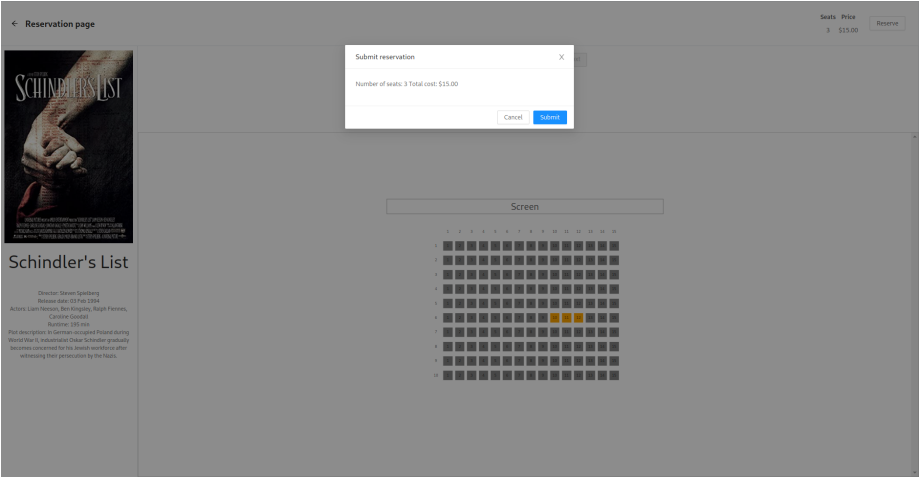
The Departed
Release Date: 05 Oct 2006

⁸MariaDB Server: The open source relational database. <https://mariadb.org>

Rysunek 3: Ekran wybierania seansów



Rysunek 4: Ekran rezerwacji miejsc



4 Wnioski

Instalacja narzędzia w środowisku Linux jest bardzo prosta, ponieważ sprowadza się do wywołania jednej komendy pobierającej program z repozytorium i wykonującej instalację. Dodatkowo składnia języka Ruby wykorzystywana w plikach konfiguracyjnych Vagranta pozwoliła na łatwe zrozumienie zachodzących procesów, a możliwość wykonywania skryptów ze zdefiniowanymi parametrami pozwoliła na dużą elastyczność w budowaniu aplikacji.

O ile samo narzędzie Vagrant nie sprawiało żadnych problemów to oryginalnie wykorzystywany obraz 'hashicorp/bionic64' okazał się problematyczny, ponieważ uniemożliwił on poprawne skonfigurowanie połączenia ssh. Zmiana obrazu na 'ubuntu/bionic64' rozwiązała ten problem i umożliwiła dalszą pracę nad problemem. Problematiczne okazało się także zużycie pamięci w trakcie przygotowywania buildu produkcyjnego aplikacji, sama konfiguracja maszyny także zajmowała dużo czasu, jednak zostało to rozwiązane poprzez ustawianie odpowiedniej ilości pamięci dla konfigurowanej maszyny.

Podsumowując, Vagrant jest świetnym narzędziem wspomagającym proces implementowania i testowania aplikacji, ze względu na prostotę nie zaleca się go jednak do używania w środowiskach produkcyjnych. Możliwość przygotowania środowiska, które jest całkowicie odtwarzalne znacznie przyspiesza proces przygotowywania i testowania aplikacji. W przypadku konfiguracji opisywanej w tym dokumencie Vagrant pozwolił na zdefiniowanie skryptów pobierających składniki potrzebne do zbudowania i uruchomienia programów napisanych w językach Java oraz TypeScript, a także skonfigurowanie bazy danych i wypełnienie ich danymi z kopii zapasowej.

5 Konfiguracje

- Plik *Vagrant*

```
1
2 Vagrant.configure("2") do |config|
3   config.vm.box = "ubuntu/bionic64"
4   config.vm.synced_folder ".", "/vagrant"
5
6   config.vm.network "forwarded_port", guest: 8080, host: 8080
7   config.vm.network "forwarded_port", guest: 8081, host: 8081
8
9   config.vm.provision "shell",
10     path: "bootstrap/bootstrap.sh",
11     env: {
12       "DB_NAME" => "cinema_tickets_app_db",
13       "DB_USERNAME" => "test",
14       "DB_PASSWORD" => "test",
15       "DB_HOST" => "localhost",
16       "DB_PORT" => 3306
17     }
```

```

18
19     config.vm.provider "virtualbox" do |vb|
20         vb.memory = "4096"
21         vb.cpus = "2"
22     end
23 end
24
25

```

- *Plik bootstrap.sh*

```

1  #!/usr/bin/env bash
2
3  echo "Start VM initialization"
4
5  # Update repository and Ubuntu
6  apt-get update -y
7
8  # Java installation script
9  chmod +x /vagrant/bootstrap/java.sh
10 /vagrant/bootstrap/java.sh
11
12 # Setup database
13 chmod +x /vagrant/bootstrap/database.sh
14 sudo /vagrant/bootstrap/database.sh -d $DB_NAME -u $DB_USERNAME
15     -p $DB_PASSWORD
16
17 # Start application
18 chmod +x /vagrant/bootstrap/cinema-tickets-app.sh
19 sudo /vagrant/bootstrap/cinema-tickets-app.sh -d $DB_NAME -U
20     $DB_USERNAME -P $DB_PASSWORD -h $DB_HOST -p $DB_PORT
21
22 # Default to main dir
23 echo "cd /vagrant" >> /home/vagrant/.bashrc
24
25

```

- *Plik java.sh*

```

1  #!/usr/bin/env bash
2
3  apt-get update
4
5  echo "Install Java..."
6  apt-get install -y openjdk-11-jdk-headless
7  echo "Java installation complete"
8
9  java --version
10

```

- *Plik database.sh*

```

1  #!/usr/bin/env bash
2  while getopts "d:u:p:" flag;
3  do
4      case $flag in
5          d) database=${OPTARG};;
6          u) username=${OPTARG};;

```

```

7     p) password=${OPTARG};;
8     esac
9 done
10
11 echo "Initializing Database..."
12
13 # Update repo, install maria-db
14 apt-get -y update
15 apt-get -y install mariadb-server
16
17 # Check Status
18 systemctl status mariadb
19
20 # Restart (ensure it's running)
21 sudo systemctl restart mariadb
22
23 echo "GRANT ALL PRIVILEGES ON *.* TO '$username'@'localhost'
    IDENTIFIED BY '$password' WITH GRANT OPTION; FLUSH
    PRIVILEGES;"
24 sudo mysql -e "GRANT ALL PRIVILEGES ON *.* TO '$username'@'%'
    IDENTIFIED BY '$password' WITH GRANT OPTION; FLUSH
    PRIVILEGES;"
25
26 # Create database;
27 echo "DROP TABLE IF EXISTS $database;"
28 mysql -u $username -p$password -e "DROP DATABASE IF EXISTS ${
    database};"
29
30 echo "CREATE DATABASE $database;"
31 mysql -u $username -p$password -e "CREATE DATABASE ${database};
    "
32
33 for sql_file in `ls /vagrant/bootstrap/database_scripts/*.sql`
34 do
35     echo "Processing file $sql_file..."
36     mysql -u $username -p$password $database < $sql_file
37 done
38
39 # Restart again
40 sudo systemctl restart mariadb
41

```

- *Plik cinema-tickets-app.sh*

```

1 #!/usr/bin/env bash
2 while getopts "d:U:P:h:p:" flag;
3 do
4     case $flag in
5         d) database=${OPTARG};;
6         U) username=${OPTARG};;
7         P) password=${OPTARG};;
8         h) host=${OPTARG};;
9         p) port=${OPTARG};;
10    esac
11 done
12
13 cd /vagrant

```

```

14
15 ls -a
16
17 echo "Initializing Cinema Tickets App..."
18 echo "Installing app..."
19
20 ./munw clean install -DskipTests
21
22 echo "Installation done..."
23 echo "Starting..."
24
25 ./munw -pl backend spring-boot:run -DskipTests -Ddb.name=
    $database -Ddb.user=$username -Ddb.password=$password -Ddb.
    host=$host -Ddb.port=$port &
26 ./munw -pl ui spring-boot:run &
27
28 echo "Finishing..."
29

```

Kod źródłowy programu, wraz z konfiguracją narzędzia Vagrant, dostępny jest na platformie GitHub: https://github.com/eipc16/PIISW_App_Vagrant