INSTRUKCJA UŻYTKOWANIA APLIKACJI SERWEROWEJ

Spis treści

Rozdział 1

Użytkowanie aplikacji serwerowej

1.1 Uruchamianie aplikacji

Aplikacja serwerowa do działania wymaga zainstalowania *Java Runtime Enviroment* w wersji co najmniej 1.8. Aplikację można uruchomić wydając polecenie `java -jar chess-backend.jar` będąc w głównym katalogu aplikacji serwerowej. Dodatkowo przy uruchamianiu można wskazać port na którym będzie działała aplikacja przekazując dodatkowo argument `--port=NUMER_PORTU`. Wydając polecenie `java -jar chess-backend.jar --port=80` aplikacja zostanie uruchomiona na porcie 80. Domyślnie aplikacja uruchamiana jest na porcie `8080`.

1.2 Zarządzanie użytkownikami i silnikami szachowymi

Listy użytkowników i silników wczytywane są jednorazowo przy każdym uruchomieniu aplikacji. Aby zapisane w nich zmiany zostały wykorzystane przez aplikację, należy uruchomić ją ponownie.

1.2.1 Zarządzanie użytkownikami

W zasobach aplikacji istnieje domyślnie jeden użytkownik o loginie `test` i haśle `111111`. Aby zmodyfikować listę użytkowników można wykorzystać aplikację konfiguracyjną opisaną dalej lub ręcznie zmodyfikować plik `users.json` znajdujący się w głównym katalogu aplikacji serwerowej. Początkowa zawartość pliku przedstawiona jest na listingu C.1. Aby dodać nowego użytkownika należy dodać nowy element do listy o tej samej strukturze co przedstawiony na listingu. Hasło powinno być haszowane algorytmem *SHA256* i zapisane w postaci

szesnastkowej. Aby usunąć użytkownika wystarczy usunąć odpowiedni element z listy i zapisać plik.

Listing 1.1: Początkowa zawartość pliku config.json

```
[
  {
    "login":"test",
    "password":"
        bcb15f821479b4d5772bd0ca866c00ad5f926e3580720659cc80d39c9d09802a"
  }
]
```

1.2.2 Zarządzanie silnikami

Aby móc wykorzystywać silnik szachowy należy dodać do pliku `engines/config.json` wpis zawierający nazwę silnika, ścieżkę do silnika oraz opis. Początkowo w pliku `config.json` znajduje się wpis jednego silnika przedstawiony na listingu C.2. Do modyfikacji listy silników można też użyć aplikacji konfiguracyjnej, której użytkowanie opisano w następnej sekcji.

Listing 1.2: Początkowa zawartość pliku users.json

```
[
  {
    "name":"Stockfish 7 x64",
    "path":"stockfish_7_x64.exe",
    "description":"Stockfish Chess Engine"
  }
]
```

1.2.3 Używanie aplikacji konfiguracyjnej

W celu uproszczenia edycji list użytkowników i silników, można wykorzystać aplikację konfiguracyjną znajdującą się w głównym katalogu aplikacji serwerowej. By uruchomić aplikację konfiguracyjną należy wydać polecenie `java -jar config.jar`. Aplikacja udostępnia następujące polecenia (w kolejności alfabetycznej):

- `engine add --name=NAME --path=PATH [--description=DESCRIPTION]` umożliwia dodanie nowego silnika, lub nadpisanie istniejącego. Silniki identyfikowane są po nazwie

i ścieżce, więc jeżeli silnik będzie miał tę samą ścieżkę i inną nazwę lub na odwrót to zostanie zidentyfikowany jako nowy silnik. Parametr `NAME` oznacza nazwę silnika (może być dowolna wybrana przez użytkownika). Parametr `PATH` oznacza ścieżkę do pliku silnika, względem katalogu `engines`. Parametr `DESCRIPTION` pozwala na dodanie opisu silnika, domyślnie zostanie ustawiony na pusty. Jeżeli `NAME` lub `PATH` zawiera spacje, należy otoczyć tę wartość znakami cudzysłowu.

- `engine list` wyświetla listę silników.
- `engine rm --name=NAME --path=PATH` usuwa silnik o nazwie `NAME` i ścieżce `PATH`.
- `help` wyświetla listę dostępnych poleceń.
- `user add --username=USERNAME --password=PASSWORD` pozwala na dodanie nowego użytkownika lub nadpisanie istniejącego użytkownika o tej samej nazwie. Parametr `USERNAME` oznacza nazwę użytkownika, a `PASSWORD` hasło użytkownika.
- `user list` wyświetla listę użytkowników
- `user rm --username=USERNAME` usuwa użytkownika o nazwie `USERNAME`

1.3 Dokumentacja udostępnianego API

Poniżej opisano *REST API* oraz interfejs udostępniany przez *WebSocket* przez aplikację serwerową.

Uwaga: wszystkie ścieżki oprócz `POST: /user/login` wymagają podania tokenu autoryzacyjnego otrzymanego w trakcie logowania. Token autoryzacyjny powinien być przesyłany w nagłówku `Authorization` i być w formie `Bearer TOKEN`.

1.3.1 `POST: /user/login` - logowanie użytkownika

Parametry wejściowe

- `login` | `String` - login użytkownika
- `password` | `String` - hasło użytkownika

Listing 1.3: Przykładowe zapytanie

```
POST http://192.168.1.163:8080/user/login http/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 36

{
```

```
    "login":"test",
    "password":"111111"
}
```

Dane zwracane

- `success` | `boolean` - czy logowanie się powiodło
- `token` | `String` - token autoryzacyjny użytkownika

Listing 1.4: Przykładowa odpowiedź

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 22 Oct 2017 19:21:00 GMT

{
    "success":true,
  "token":"zdWIiOiJ0ZXN0Iiwicm9sZXMiOiJ1c2"
}
```

1.3.2 `GET: /user/logout` - wylogowanie użytkownika

Brak parametrów wejściowych i zwracanych danych. Kod odpowiedzi 200 oznacza poprawne wylogowanie.

1.3.3 `GET: /engine/available` - pobieranie listy dostępnych silników

Dane zwracane

- `success` | `boolean` - czy zapytanie zostało wykonane poprawnie
- `info` | `Array` - lista dostępnych silników
  - `name` | `String` - nazwa silnika
  - `path` | `String` - ścieżka silnika na serwerze
  - `description` | `String` - opis silnika

Listing 1.5: Przykładowa odpowiedź

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
```

```
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 22 Oct 2017 19:32:04 GMT

{
    "success":true,
    "info":[
        {
          "name":"Stockfish 7 x64",
          "path":"stockfish_7_x64.exe",
          "description":"Stockfish Chess Engine"
        }
    ]
}
```

1.3.4 `POST: /engine/start` - uruchomienie silnika

Parametry wejściowe

- `engine` | `String` - nazwa silnika do uruchomienia

Listing 1.6: Przykładowe zapytanie

```
POST http://192.168.1.163:8080/engine/start http/1.1
Content-Type: application/json; charset=UTF-8
Content-Length: 28
Authorization: Bearer zdWIiOiJ0ZXN0Iiwicm9sZXMiOiJ1c2

{
  "engine":"Stockfish 7 x64"
}
```

1.3.5 `POST: /engine/stop` - zatrzymanie aktualnie działającego silnika

Dane zwracane

- `sucess` | `boolean` - czy silnik został zatrzymany
- `info` | `String` - dodatkowe informacje

Listing 1.7: Przykładowa odpowiedź

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sun, 22 Oct 2017 19:32:04 GMT

{
    "success":true,
    "info":"Engine stopped"
}
```

1.3.6 Komunikacja przez *WebSocket*

By połączyć się z serwerem przez *WebSocket* należy użyć ścieżki `/ws_engine`. Połączenie może być nawiązane niezależnie od tego czy jakikolwiek silnik jest uruchomiony. Cała komunikacja przez *WebSocket* odbywa się zgodnie ze standardem *UCI* [Kah04]. Komunikaty powinny być przesyłane w *UTF-8*. Przy nawiązywaniu połączenia należy wysłać w nagłówku `Authorization` token autoryzacyjny użytkownika. Wszystkie wiadomości przesłane do serwera zostaną przekazane na standardowe wejście procesu silnika szachowego natomiast wszystkie dane z standardowego wyjścia procesu silnika zostaną przesłane do klienta połączonego z serwerem.

Bibliografia

[Kah04] Stefan-Meyer Kahlen. Description of the universal chess interface (uci). 2004. [online] `http://wbec-ridderkerk.nl/html/UCIProtocol.html` Dostęp 29-07-2017.