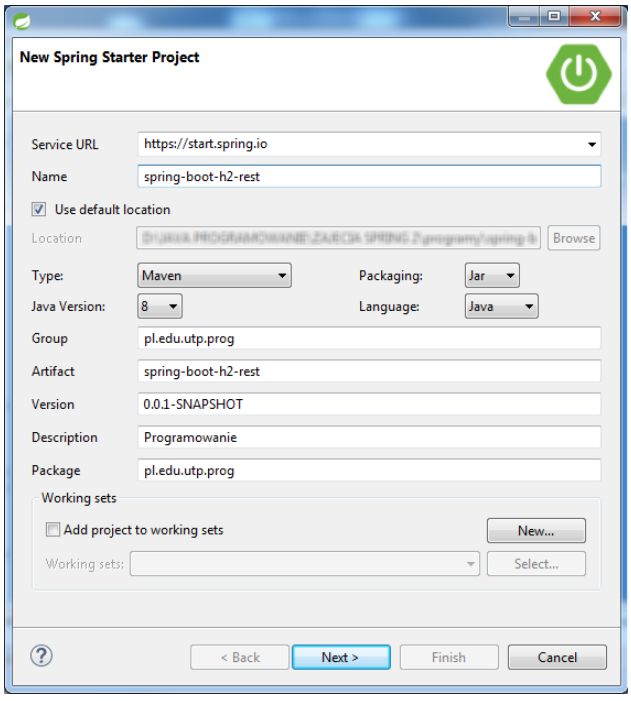
ĆWICZENIE: Aplikacja RESTful (Serwer)
(4 godziny zajęciowe)

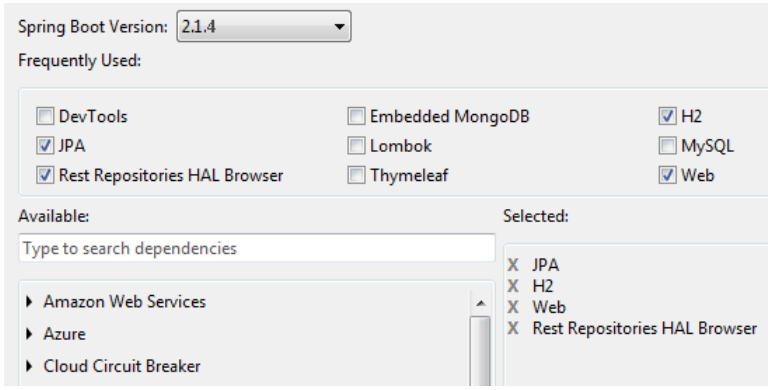
Zaimplementować aplikację RESTful, która będzie umożliwiać pobieranie informacji o studentach za pomocą interfejsu REST API. Kontroler powinien zawierać w tym przypadku tylko dwie metody, `getStudent(Long id)` oraz `getStudents()`, które pozwolą odpowiednio na pobieranie informacji o studencie z okreslonym identyfikatorem id oraz listę wszystkich studentów. W przypadku, gdy klient będzie żądał informacji o studencie, który nie istnieje w bazie danych, powinien dostać informację zwrotną z kodem błędu 404 i komunikatem ":(*Client with id=… not found* ".

REST (**RE**presentational **S**tate **T**ransfer) jest swego rodzaju architektonicznym stylem do projektowania rozproszonych aplikacji internetowych. W przypadku **Web API opiera się on na protokole HTTP**. Każdy zasób można pobrać, zmodyfikować, utworzyć i usunąć standardowymi metodami HTTP. Dla przykładu, popularne akcje takie jak Create, Read, Update, Delete (CRUD) odpowiadają metodom POST, GET, PUT i DELETE. Na potrzeby niniejszego ćwiczenia stworzymy uproszczone API zgodnie z tabelą poniżej, które daje nam wyłącznie możliwość pobierania informacji o studentach, bez sposobności ich tworzenia, usuwania i modyfikowania.

Metoda HTTP	URI zasobu	Opis
GET	`/students`	Pobiera listę studentów
GET	`/students/id`	Pobiera dane studenta o identyfikatorze id, np. `/students/2`

- ✓ Stwórz nowy projekt Spring Boot (File/New/Spring Starter Project), podaj wymagane dane oraz wybierz potrzebne zależności.





Zawartość wygenerowanego pliku *pom.xml*

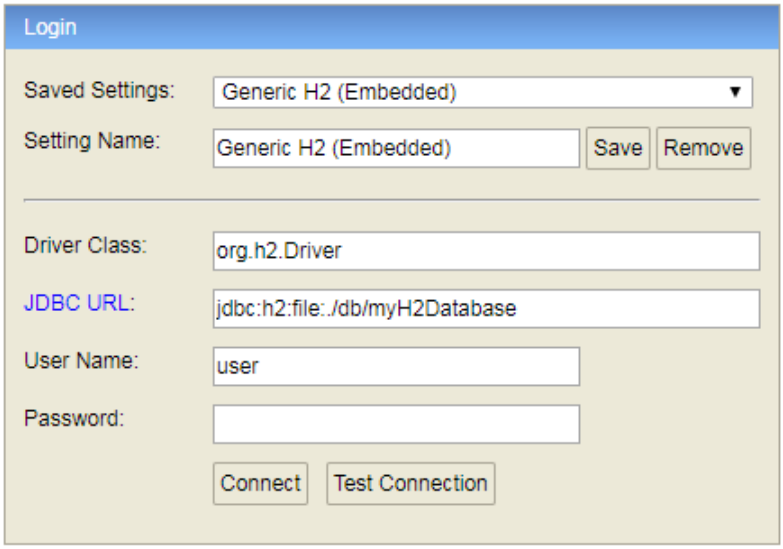
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.1.4.RELEASE</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>pl.edu.utp.prog</groupId>
        <artifactId>spring-boot-h2-rest</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>spring-boot-h2-rest</name>
        <description>Programowanie</description>
        <properties>
                <java.version>1.8</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.data</groupId>
                        <artifactId>spring-data-rest-hal-browser</artifactId>
                </dependency>

                <dependency>
                        <groupId>com.h2database</groupId>
                        <artifactId>h2</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>
</project>
```

- ✓ Aby skonfigurować i uzyskać dostęp do bazy danych H2 za pomocą konsoli H2, wpisz w pliku **`application.properties`** następującą konfigurację.

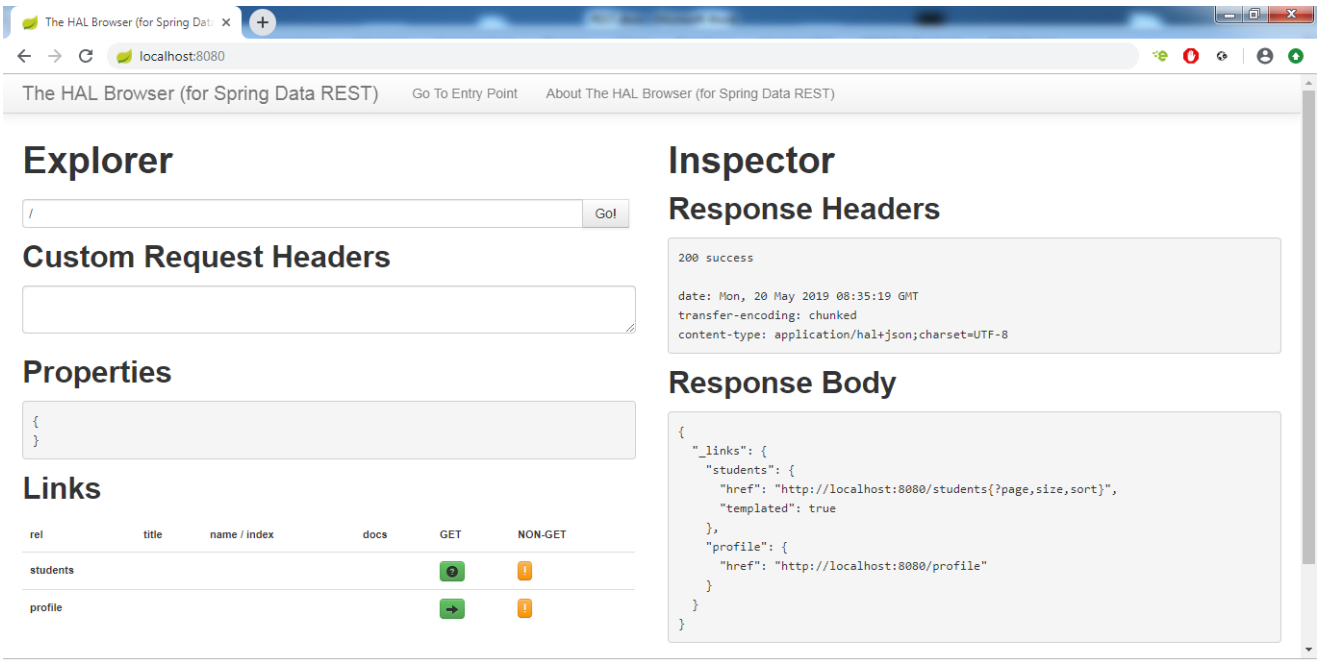
```
spring.datasource.url=jdbc:h2:file:./db/myH2Database
spring.datasource.username=user
spring.datasource.password= Wpisz swoje hasło
spring.datasource.driverClassName=org.h2.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true

spring.h2.console.enabled=true
spring.h2.console.path=/h2
```

W sprawozdaniu należy zawrzeć wyjaśnienie i opis powyższych linii, które konfigurują naszą aplikację.

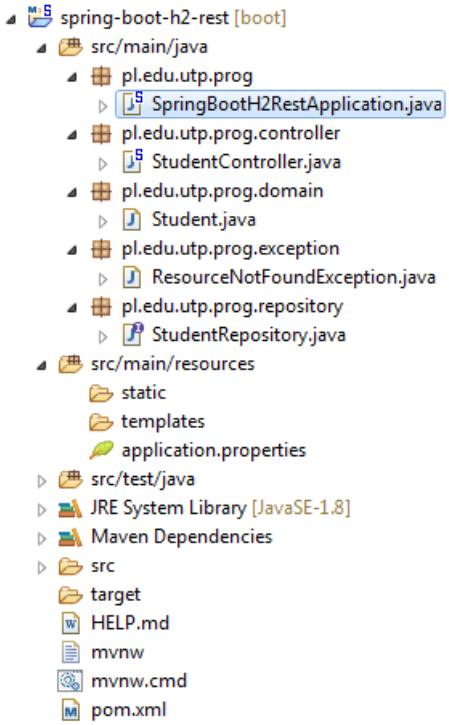
- ✓ Dostęp do konsoli osadzonej bazy danych H2 możemy uzyskać wpisując po uruchomieniu aplikacji następujący adres w przeglądarce internetowej: http://localhost:8080/h2. Z poziomu konsoli możemy przeglądać zawartość naszej bazy danych i wykonywać zapytania SQL.



- ✓ Przeglądarka internetowa, z której zwykle korzystamy do przeglądania stron WWW, jest niewystarczająca do testowania REST API. Wynika to z faktu, że interesują nas nie tylko metody HTTP takie jak GET czy POST, ale również PUT, DELETE, PATCH etc. Moglibyśmy pobrać i zainstalować osobną aplikację do tego celu (np. popularną aplikację Postman), ale skorzystamy z gotowego, dostępnego rozwiązania dla aplikacji Spring Boot. Możemy dołączyć do naszego projektu przeglądarkę **HAL Browser**, co wymaga tylko podania odpowiedniej zależności w pliku `pom.xml`. Dostęp do tej przeglądarki po uruchomieniu aplikacji otrzymujemy poprzez adres http://localhost:8080.



✓ Przykładowa struktura naszego projektu



Zadanie dodatkowe (dla chętnych). Zaimplementować pozostałe metody zgodnie z tabelą poniżej. Wzbogacić aplikację RESTful o funkcję walidacji tworzonych i aktualizowanych studentów.

Metoda HTTP	URI zasobu	Opis
GET	`/students`	Pobiera listę studentów
GET	`/students/id`	Pobiera dane studenta o identyfikatorze id, np. `/students/2`
POST	`/students`	Tworzy nowego studenta
PUT	`/students/id`	Aktualizuje dane studenta o identyfikatorze id
DELETE	`/students/id`	Usuwa studenta o identyfikatorze id