

Sprawozdanie Kostyra Mateusz

Utworzenie projektu

The screenshot shows the Spring Initializr web application interface. The top navigation bar includes the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Includes radio buttons for "Maven Project" (selected) and "Gradle Project".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for various versions: "3.0.0 (SNAPSHOT)", "3.0.0 (M2)", "2.7.0 (SNAPSHOT)", "2.7.0 (RC1)", "2.6.8 (SNAPSHOT)", "2.6.7" (selected), "2.5.14 (SNAPSHOT)", and "2.5.13".
- Project Metadata:** Includes input fields for "Group" (pl.zajecia), "Artifact" (Laboratorium3), "Name" (Laboratorium3), "Description" (Demo project for Spring Boot), and "Package name" (pl.zajecia.Laboratorium3).
- Packaging:** Includes radio buttons for "Jar" (selected) and "War".
- Java:** Includes radio buttons for "18", "17", "11", and "8" (selected).
- Dependencies:** Includes a section for "Spring Web" with a "Web" dependency, "H2 Database" with an "SQL" dependency, and "Spring Data JPA" with an "SQL" dependency.

At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".

Dodanie klasy VideoCassete oraz jej uzupełnienie

```
VideoCassete.java
1 package pl.zajecia.Laboratorium3;
2 import java.time.LocalDate;
3
4 public class VideoCassete {
5     2 usages
6     private Long id;
7     2 usages
8     private String title;
9     2 usages
10    private LocalDate productionYear;
11
12    public Long getId(){
13        return id;
14    }
15
16    public void setId(Long id){
17        this.id=id;
18    }
19
20    public String getTitle(){
21        return title;
22    }
23
24    public void setTitle(String title){
25        this.title=title;
26    }
27
28    public LocalDate getProductionYear(){
29        return productionYear;
30    }
31
32    public void setProductionYear(LocalDate productionYear){
33        this.productionYear=productionYear;
34    }
35 }
```

Dodanie klasy VideoCasseteApi oraz jej uzupełnienie

```
VideoCasseteApi.java x
1 package pl.zajecia.Laboratorium3;
2
3 import org.springframework.web.bind.annotation.RestController;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 @RestController
9 public class VideoCasseteApi {
10     2 usages
11     private List<VideoCassete> videoCassetes;
12
13     public VideoCasseteApi(){
14         videoCassetes=new ArrayList<>();
15         videoCassetes.add(new VideoCassete());
16     }
17 }
```

Dodanie konstruktora parametrycznego do klasy VideoCassete

```
public VideoCassete(Long id, String title, LocalDate productionYear) {
    this.id = id;
    this.title = title;
    this.productionYear = productionYear;
}

1 usage
public VideoCassete() {
}
```

Dodaj kilka filmów

```
public VideoCasseteApi(){
    videoCassetes=new ArrayList<>();
    videoCassetes.add(new VideoCassete( id: 1L, title: "Tytuł1", LocalDate.of( year: 1999, month: 1, dayOfMonth: 1)));
    videoCassetes.add(new VideoCassete( id: 1L, title: "Tytuł2", LocalDate.of( year: 1998, month: 2, dayOfMonth: 3)));
}
```

Utwórz metody, która będzie pobierała wszystkie elementy z listy

```
@GetMapping
public List<VideoCassete> getAll(){
    return VideoCassetes;
}
```

Nad klasą dodaj adnotację requestMapping z parametrem, którym będzie adres za pomocą którego będzie można odpytać tworzone Api

```
@RestController
@RequestMapping("/api/cassetts")
public class VideoCasseteApi {
```

Do adnotacji metody getAll - GetMapping dopisz ścieżkę /all

```
@GetMapping("/all")
public List<VideoCassete> getAll(){
    return VideoCassetes;
}
```

Zmodyfikuj wyrażenie aby użyć typu Optional, który zabezpiecza przed wyjątkiem związanym z wystąpieniem wartości typu null.

```
@GetMapping
public VideoCassete getById(@RequestParam int index){
    Optional<VideoCassete> first = videoCassetes
        .stream()
        .filter(element -> element.getId()==index)
        .findFirst();
    return first.get();
}
```

Uruchom aplikację Postmana i przetestuj jej działanie. Do pobierania elementów służy metoda GET

GET http://localhost:8080/api/cassetts/all

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 314 ms 279 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "title": "Tytuł1",
5     "productionYear": "1999-01-01"
6   },
7   {
8     "id": 2,
9     "title": "Tytuł2",
10    "productionYear": "1998-02-03"
11  }
12 }
```

GET http://localhost:8080/api/cassetts?index=1

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> index	1			
Key	Value	Description		

Body Cookies Headers (8) Test Results 404 Not Found 310 ms 361 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-04-29T07:37:02.392+00:00",
3   "status": 404,
4   "error": "Not Found",
5   "path": "/api/cassetts"
6 }
```

Uruchom jeszcze raz

GET http://localhost:8080/api/cassetts?index=1

Save Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> index	1			
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 35 ms 220 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "Tytuł1",
4   "productionYear": "1999-01-01"
5 }
```

GET http://localhost:8080/api/cassetts?index=3

Save Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> index	3			
Key	Value	Description		

Body Cookies Headers (4) Test Results 500 Internal Server Error 300 ms 267 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-04-29T07:39:10.631+00:00",
3   "status": 500,
4   "error": "Internal Server Error",
5   "path": "/api/cassetts"
6 }
```

Przejdź do implementacji metody addVideo realizującej dodawanie elementów, przy użyciu metody http POST - użyj adnotacji PostMapping. Elementem dodawanym będzie obiekt klasy VideoCassettes. Będzie to metoda webowa więc do parametru przekazanego metodzie dodaj adnotację RequestBody, co oznacza, że ten obiekt zostanie przesłany do aplikacji w postaci serializowanej.

```
@PostMapping
public boolean addVideo(@RequestBody VideoCassete videoCassete){
    return videoCassetes.add(videoCassete);
}
```

Następnie zaimplementuj metodę do modyfikacji obiektu - użyj adnotacji PutMapping. Metodzie podajemy id i na podstawie podanych wartości będzie ona nadpisywać dany element.

```
@PutMapping
public boolean updateVideo(@RequestBody VideoCassete videoCassete){
    return videoCassetes.add(videoCassete);
}
```

Przy implementacji ostatniej metody, która będzie służyła do usuwania obiektu z listy – użyj adnotacji DeleteMapping.

```
@DeleteMapping
public boolean deleteVideo(@RequestParam int index){
    return videoCassetes.removeIf(element->element.getId()==index);
}
```

Użyj POST do dodania elementu

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/api/cassetts
- Body (JSON):**

```
{  "id": 3,  "title": "Tytuł3",  "productionYear": "1980-12-30"}
```
- Response:** 200 OK, 20 ms, 168 B. The response body is `true`.

Pobierz wszystkie elementy z listy

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/api/cassetts/all
- Body (JSON):**

```
{  "id": 1,  "title": "Tytuł1",  "productionYear": "1999-01-01"},  {  "id": 2,  "title": "Tytuł2",  "productionYear": "1998-02-03"},  {  "id": 3,  "title": "Tytuł3",  "productionYear": "1980-12-30"}
```
- Response:** 200 OK, 34 ms, 336 B.

Przetestuj metodę, która służy do usuwania elementów

The screenshot shows a Postman interface with a DELETE request to `http://localhost:8080/api/cassetts?index=1`. The request is saved and ready to be sent. The response is a 200 OK status with a response time of 12 ms and a body of `true`.

KEY	VALUE	DESCRIPTION
index	1	
Key	Value	Description

```
1 true
```

Pobierz wszystkie elementy z listy aby sprawdzić czy faktycznie element o indeksie 1 został usunięty

The screenshot shows a Postman interface with a GET request to `http://localhost:8080/api/cassetts/all`. The request is saved and ready to be sent. The response is a 200 OK status with a response time of 8 ms and a JSON body containing two cassette objects.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "id": 2,
3   "title": "Tytuł2",
4   "productionYear": "1998-02-03"
5 }
6 ,
7 {
8   "id": 3,
9   "title": "Tytuł3",
10  "productionYear": "1980-12-30"
11 }
12 }
```


Po przetestowaniu API aplikacji przystąpimy do realizacji aplikacji zapisującej dane w bazie danych. W pliku `application.properties` ustaw właściwości dotyczące nawiązywanego połączenia z bazą danych.

```
application.properties x
1  spring.jpa.properties.hibernate.hbm2ddl.auto=create
2  spring.datasource.url=jdbc:h2:file:./bazaDanych1
3  spring.h2.console.enabled=true
4  spring.h2.console.path=/console
```

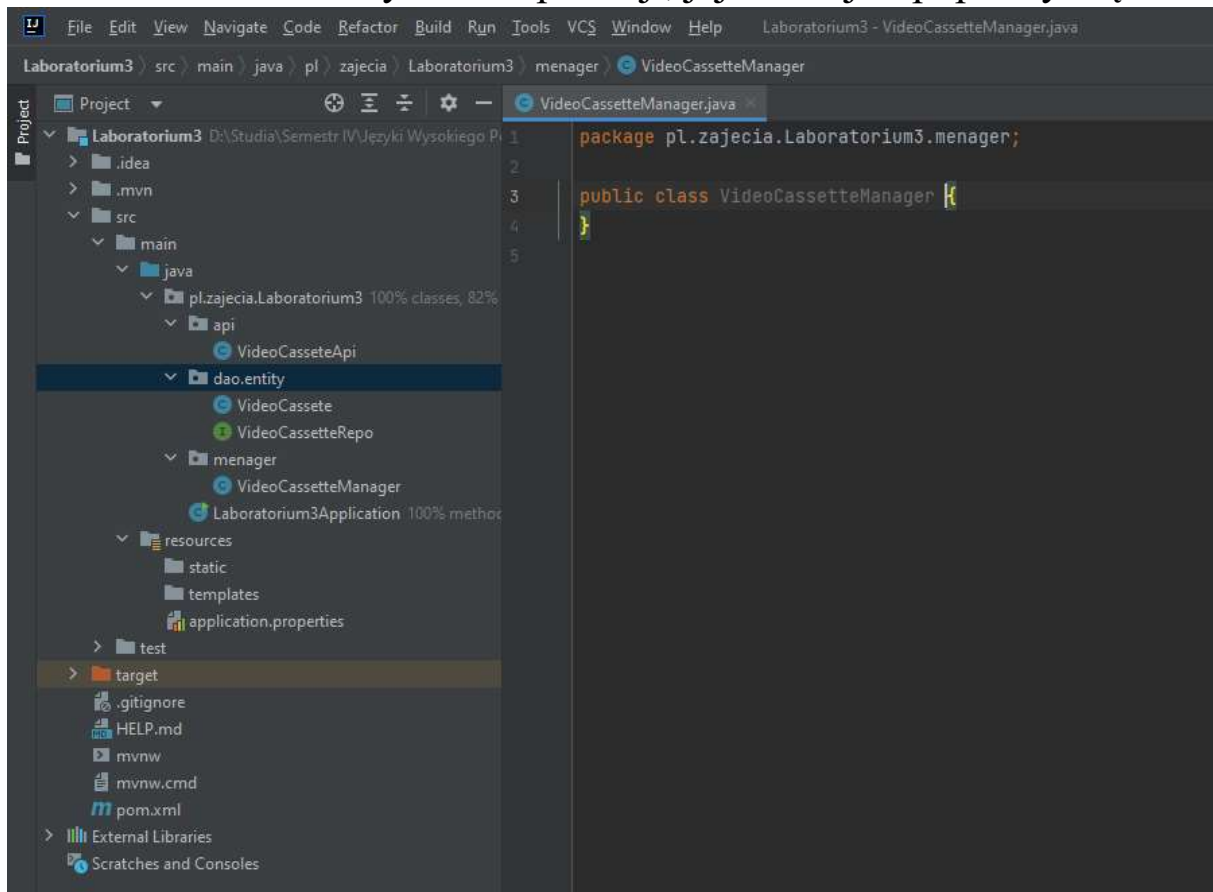
W momencie, gdy chcemy obiekty klasy przechowywać w bazie danych należy klasę zamienić na encję.

```
8 usages
@Entity
public class VideoCassete {
    3 usages
    @GeneratedValue (strategy = GenerationType.IDENTITY)
    @Id
    private Long id;|
```

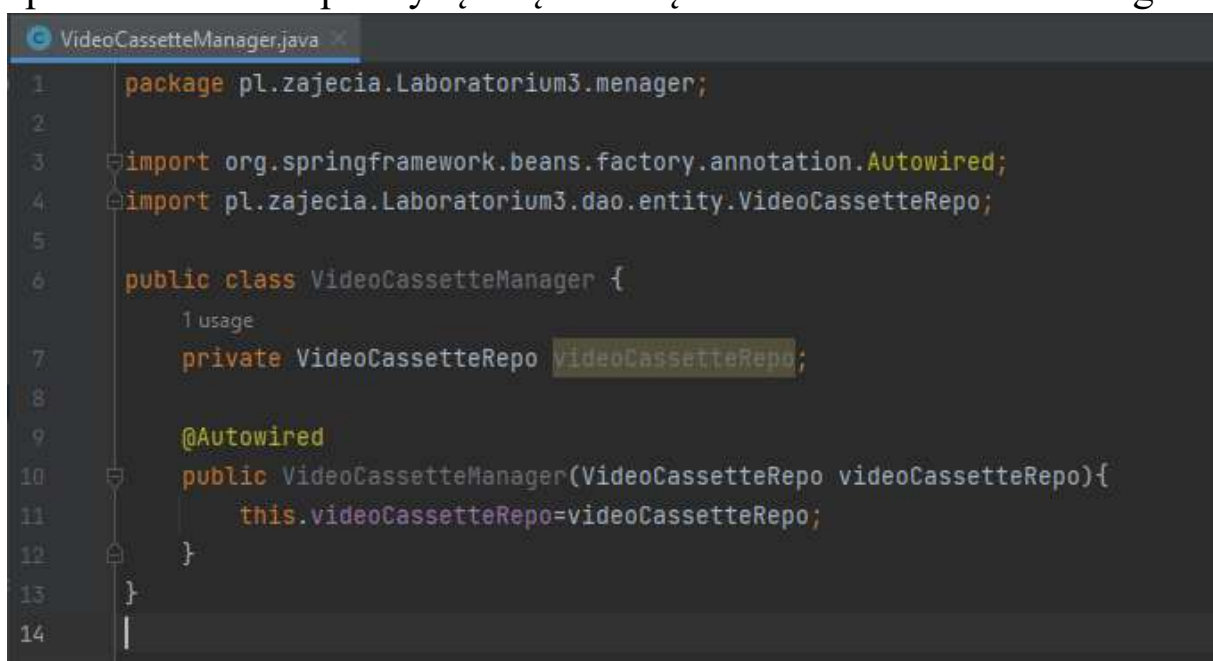
Utwórz repozytorium - interfejs `VideoCasseteRepo`

```
VideoCasseteRepo.java x
1  package pl.zajecia.Laboratorium3;
2
3  import org.springframework.data.repository.CrudRepository;
4  import org.springframework.stereotype.Repository;
5
6  @Repository
7  public interface VideoCasseteRepo extends CrudRepository<VideoCassete, Long> {
8
9  }
10 |
```

Uporządkuj kod tworzonego projektu, aby miał strukturę warstwową, co ułatwi zadanie utrzymania aplikacji, jej rozwoju i poprawy błędów.



Tworzenie warstwy pośredniej między warstwą dostępu do danych a api stanowi dobrą praktykę - będzie nią klasa VideoCassetteManager.



Utwórz metody, które będą odpowiadały metodom znajdującym się w repozytorium i będą wywoływane w ramach implementacji metod klasy api aplikacji.

```
public Optional<VideoCassete> findById(Long id){  
    return videoCassetteRepo.findById(id);  
}
```

Druga metoda o nazwie findAll ma pobierać wszystkie elementy z bazy danych. Typ Iterable oznacza elementy, które da się iterować czyli jakiegoś typu kolekcje.

```
public Iterable<VideoCassete> findAll(){  
    return videoCassetteRepo.findAll();  
}
```

Zaimplementuj metodę realizującą dodawanie elementu do bazy – repozytorium udostępnia metodę save. Metoda zwraca element, który został zapisany w bazie.

```
public VideoCassete save(VideoCassete videoCassete){  
    return videoCassetteRepo.save(videoCassete);  
}
```

Zaimplementuj w serwisie metodę do usuwania elementów, która na podstawie dostarczonego id usunie dany element z bazy danych.

```
public void delete (Long id){  
    videoCassetteRepo.deleteById(id);  
}
```

Skopiuj z konstruktora klasy api tworzenie dwóch obiektów i wklej w klasie menagera w ramach metody fillDB. Użyj metody save z klasy serwisu, aby dodać te dwa obiekty - filmy do bazy danych.

```
public void filldb(){
    save(new VideoCassete( id: 1L, title: "Tytuł1", LocalDate.of( year: 1999, month: 1, dayOfMonth: 1)));
    save(new VideoCassete( id: 2L, title: "Tytuł2", LocalDate.of( year: 1998, month: 2, dayOfMonth: 3)));
}
```

Do metody dodaj specjalną adnotację eventListener z atrybutem ApplicationReadyEvent, która oznacza nasłuchiwanie zdarzenia uruchomienia aplikacji. Ciało metody fillDB oznaczone tą adnotacją, zostanie wykonane po uruchomieniu aplikacji, co spowoduje, że na jej starcie w bazie danych zostaną zapisane dwa filmy.

```
@EventListener(ApplicationReadyEvent.class)
public void filldb(){
    save(new VideoCassete( id: 1L, title: "Tytuł1", LocalDate.of( year: 1999, month: 1, dayOfMonth: 1)));
    save(new VideoCassete( id: 2L, title: "Tytuł2", LocalDate.of( year: 1998, month: 2, dayOfMonth: 3)));
}
```

W klasie api „zastąp” listę - bazą danych. Nie będziemy już potrzebowali listy, bo informacje o kasetach będą przechowywane w bazie danych - usuń listę i dotychczasowy konstruktor w którym dodajemy kasety do listy. W to miejsce utwórz deklarację obiektu klasy menagera, dodaj import, i w konstruktorze wstrzyknij instancję klasy menagera.

```
public class VideoCasseteApi {
    6 usages
    private VideoCassetteManager videoCassetes;

    @Autowired
    public VideoCasseteApi(VideoCassetteManager videoCassetes){
        this.videoCassetes=videoCassetes;
    }
}
```

Metoda getAll powinna teraz korzystać z metody serwisu findAll i zwracać typ Iterable a nie listę.

```
@GetMapping("/all")
public Iterable<VideoCassete> getAll(){
    return videoCassetes.findAll();
}
```

Kolejna metoda getById nie będzie już potrzebowała korzystać z wyrażenia lambda – zostanie zastąpiona prostszym zapisem, w ramach którego wywołujemy metodę serwisu findById.

```
@GetMapping
public Optional<VideoCassete> getById(@RequestParam Long index){
    return videoCassetes.findById(index);
}
```

Aby zmodyfikować metodę dodającą obiekt/kasetę użyjemy metody z serwisu save, która zwraca dodany obiekt kasety wideo, więc typ zwracany przez metodę add zamieniamy z boolean na VideoCassete.

```
@PostMapping
public VideoCassete addVideo(@RequestBody VideoCassete videoCassete){
    return videoCassetes.save(videoCassete);
}
```

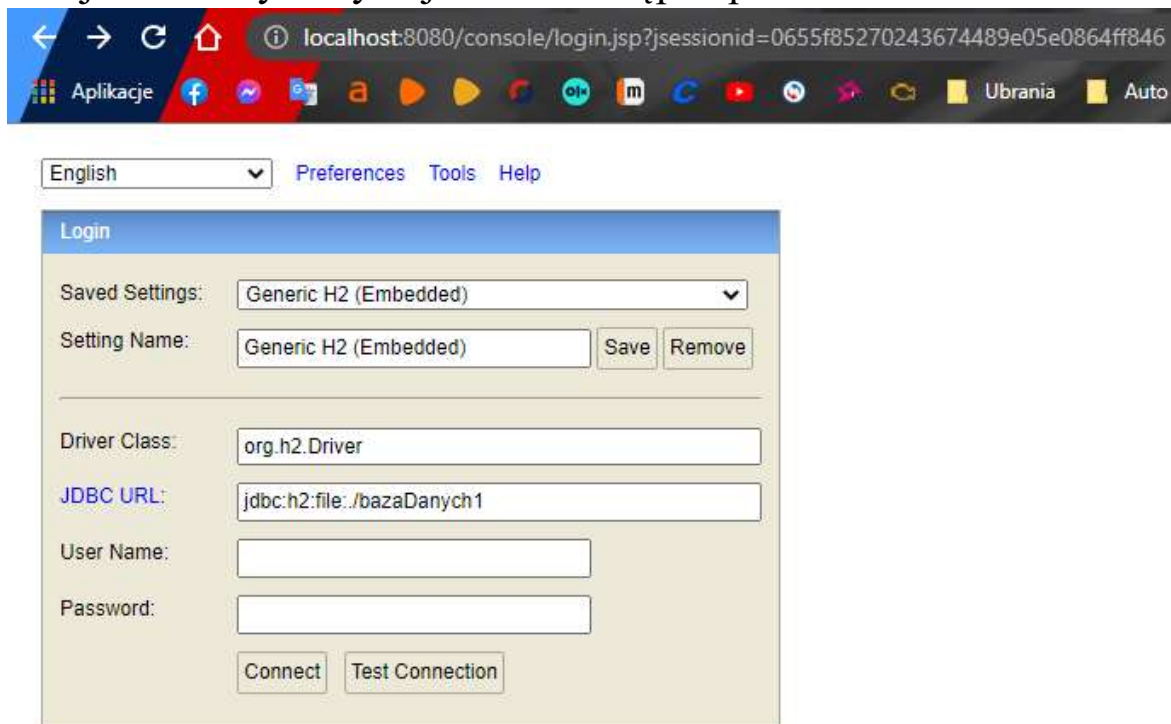
Analogicznie modyfikujemy metodę update

```
@PutMapping
public VideoCassete updateVideo(@RequestBody VideoCassete videoCassete){
    return videoCassetes.save(videoCassete);
}
```

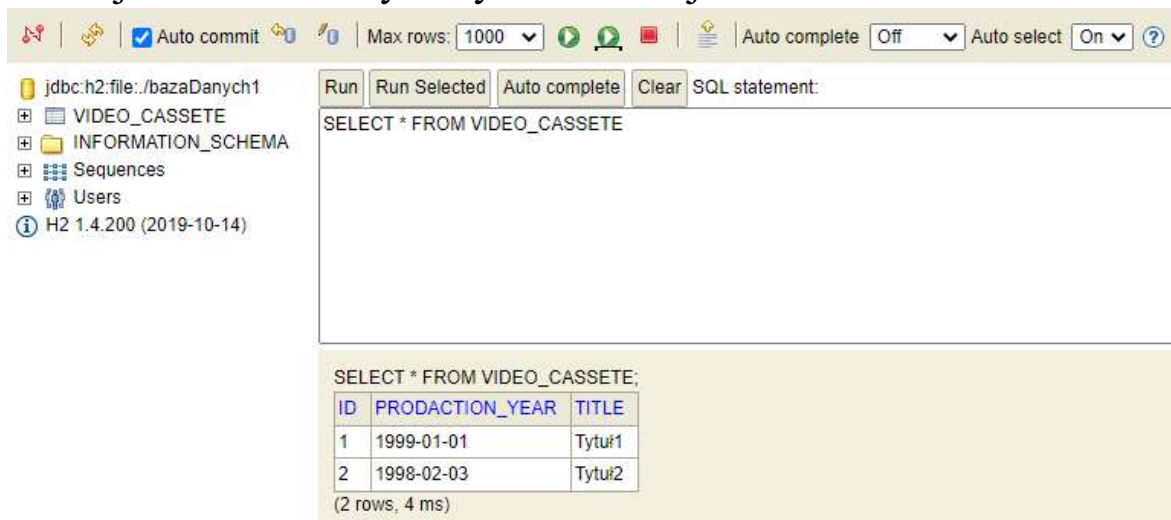
Przy modyfikacji ostatniej metody delete, korzystamy z metody z serwisu delete, która zwraca typ void.

```
@DeleteMapping
public void deleteVideo(@RequestParam Long index){
    videoCassetes.delete(index);
}
```

Przejdź do bazy danych jest ona dostępna przez: /console.



Kliknij na nazwie bazy danych i naciśnij Run



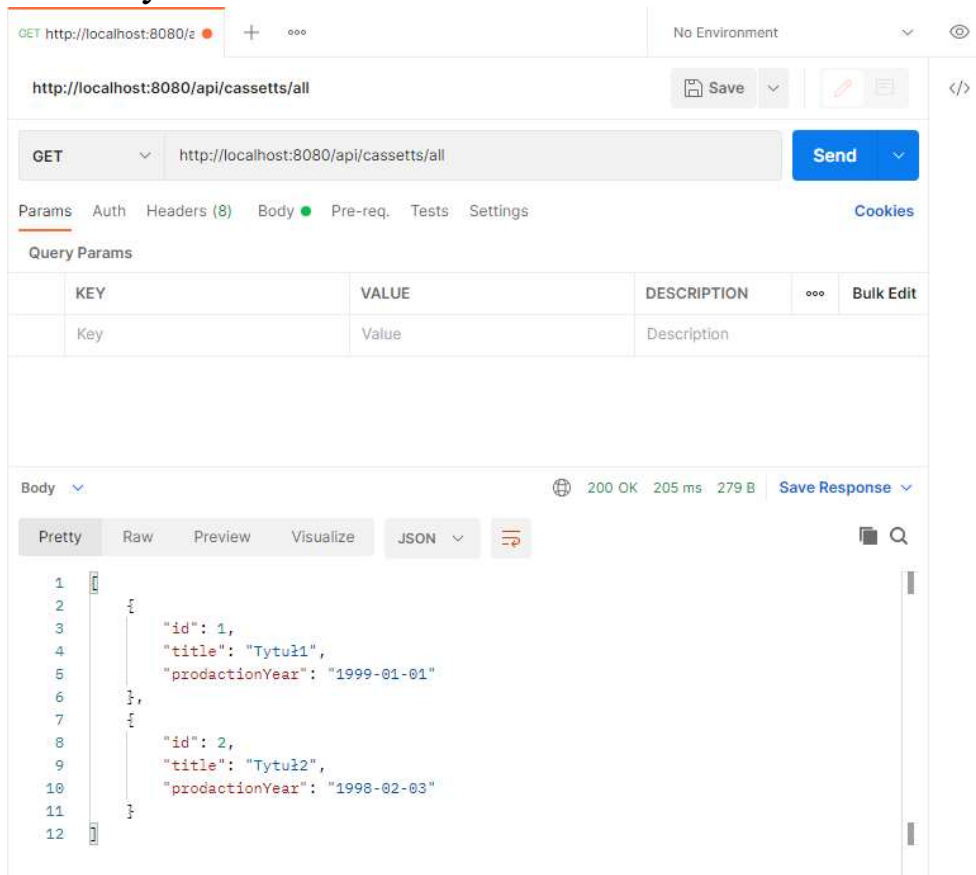
Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM VIDEO_CASSETE

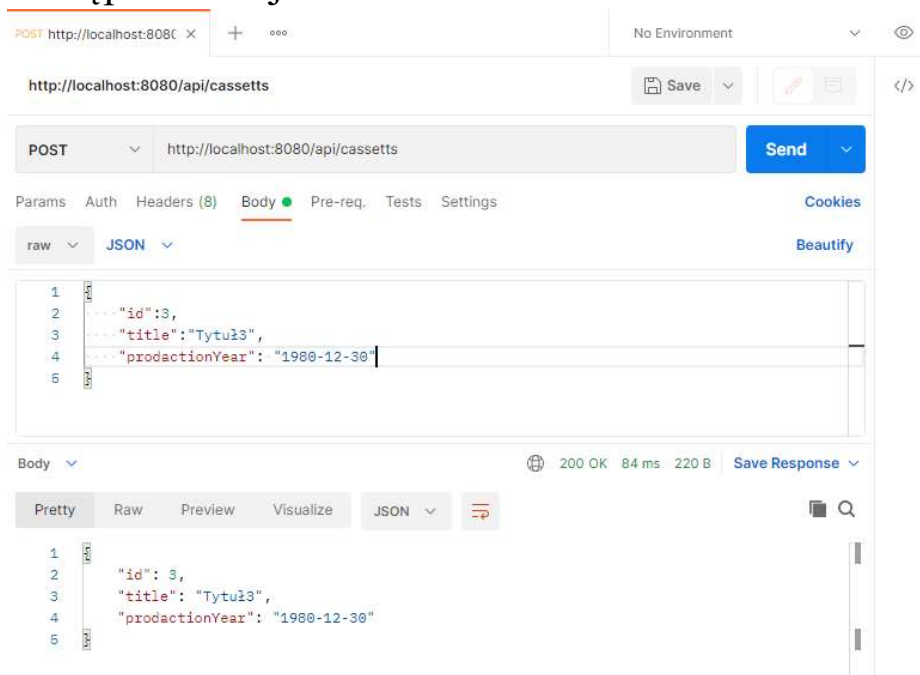
ID	PRODACTION_YEAR	Tytuł
1	1999-01-01	Tytuł1
2	1998-02-03	Tytuł2

(2 rows, 4 ms)

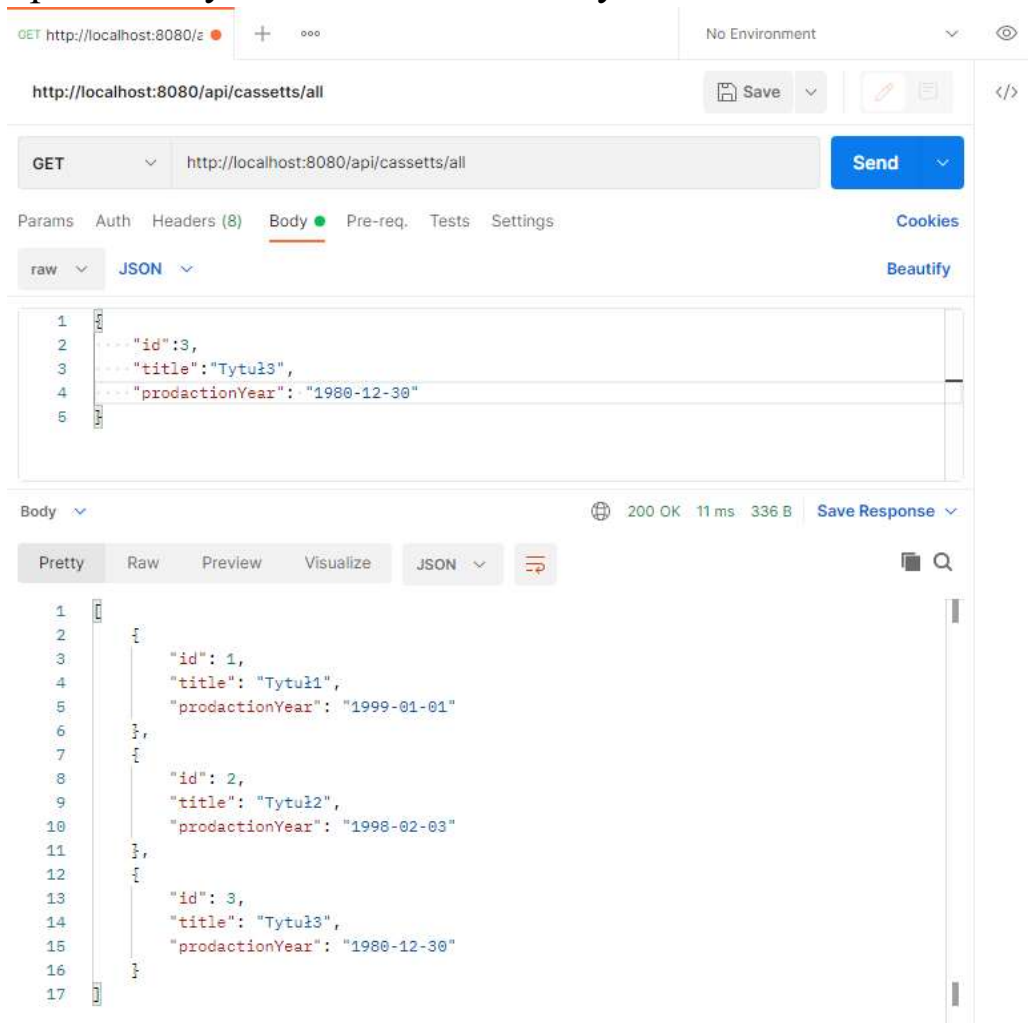
Po upewnieniu się, że w bazie danych są dwa elementy, wykonaj test metod w kliencie http – Postmanie. Najpierw pobierz wszystkie elementy.



Następnie dodaj element



Sprawdź czy element został dodany



GET http://localhost:8080/api/cassetts/all

Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies Beautify

raw JSON

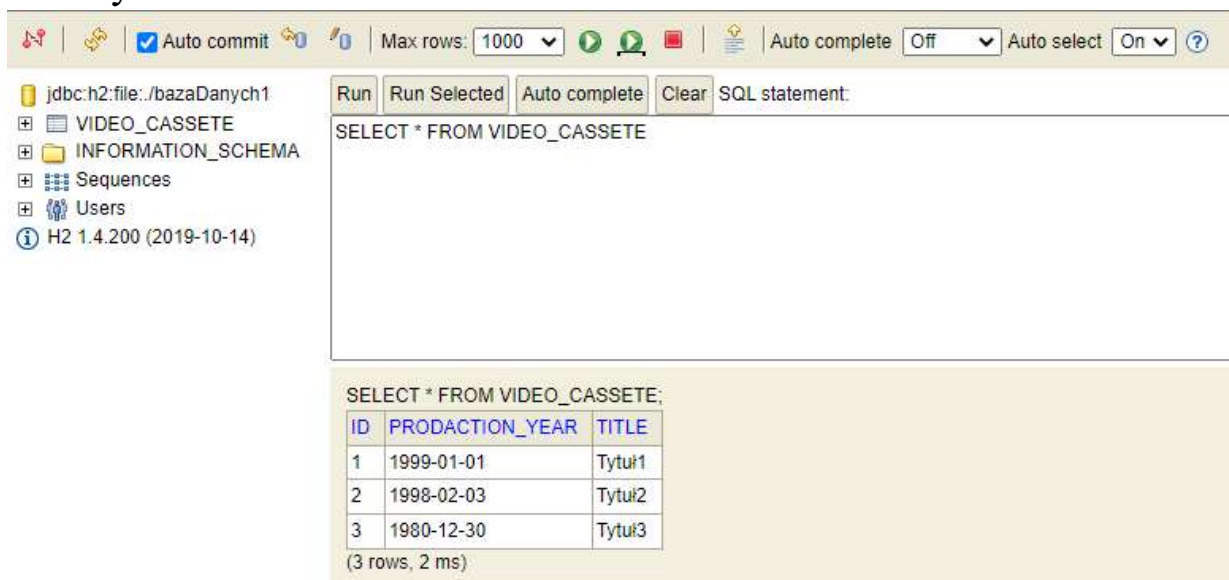
```
1 {
2   "id": 3,
3   "title": "Tytuł3",
4   "productionYear": "1980-12-30"
5 }
```

Body 200 OK 11 ms 336 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "Tytuł1",
4   "productionYear": "1999-01-01"
5 },
6 {
7   "id": 2,
8   "title": "Tytuł2",
9   "productionYear": "1998-02-03"
10 },
11 {
12   "id": 3,
13   "title": "Tytuł3",
14   "productionYear": "1980-12-30"
15 }
16 }
```

Sprawdź również z poziomu konsoli bazy danych czy element został dodany



Auto commit Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:file:./bazaDanych1

VIDEO_CASSETTE INFORMATION_SCHEMA Sequences Users H2 1.4.200 (2019-10-14)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM VIDEO_CASSETTE

SELECT * FROM VIDEO_CASSETTE:

ID	PRODACTION_YEAR	TITLE
1	1999-01-01	Tytuł1
2	1998-02-03	Tytuł2
3	1980-12-30	Tytuł3

(3 rows, 2 ms)

Przetestuj działanie metody PUT, która służy do modyfikacji pól danego elementu np. zmień tytuł którejś pozycji i datę

The screenshot shows a Postman PUT request to the URL `http://localhost:8080/api/cassetts`. The request body is a JSON object: `{ "id": 3, "title": "Tytuł33", "productionYear": "2000-12-30" }`. The response is a 200 OK status with a response time of 18 ms and a body size of 221 B. The response body is displayed in a pretty-printed JSON format: `{ "id": 3, "title": "Tytuł33", "productionYear": "2000-12-30" }`.

Sprawdź wynik w Postmanie

The screenshot shows a Postman GET request to the URL `http://localhost:8080/api/cassetts?index=3`. The response is a 200 OK status with a response time of 17 ms and a body size of 221 B. The response body is displayed in a pretty-printed JSON format: `{ "id": 3, "title": "Tytuł33", "productionYear": "2000-12-30" }`.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> index	3	
Key	Value	Description

W metodzie POST nie powinno się dodawać id bo on jest automatycznie generowany. Wykonaj zapytania

The screenshot shows a REST client interface with a POST request to `http://localhost:8080/api/cassetts`. The request body is a JSON object:

```
1 {
2   "title": "Tytuł4",
3   "productionYear": "1980-12-30"
4 }
```

The response status is 200 OK, with a response time of 14 ms and a body size of 220 B. The response body is shown in JSON format:

```
1 {
2   "id": 4,
3   "title": "Tytuł4",
4   "productionYear": "1980-12-30"
5 }
```

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/cassetts/all`. The response status is 200 OK, with a response time of 12 ms and a body size of 394 B. The response body is shown in JSON format:

```
5   "productionYear": "1999-01-01"
6 },
7 {
8   "id": 2,
9   "title": "Tytuł2",
10  "productionYear": "1998-02-03"
11 },
12 {
13   "id": 3,
14   "title": "Tytuł33",
15   "productionYear": "2000-12-30"
16 },
17 {
18   "id": 4,
19   "title": "Tytuł4",
20   "productionYear": "1980-12-30"
21 }
22 }
```

Przetestuj działanie ostatniej metody delete – usuń np. 2 element z bazy

DEL http://localhost:8080/a + ... No Environment

http://localhost:8080/api/cassetts?index=2 Save

DELETE http://localhost:8080/api/cassetts?index=2 Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	index	2			
	Key	Value	Description		

Body 200 OK 16 ms 123 B Save Response

Pretty Raw Preview Visualize Text

1

GET http://localhost:8080/ + ... No Environment ▼ 👁

http://localhost:8080/api/cassetts/all Save ✎ 🗨 </>

GET ▼ http://localhost:8080/api/cassetts/all Send ▼

Params Auth Headers (8) Body ● Pre-req. Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body ▼ 🌐 200 OK 14 ms 337 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ☰ 🔍

```

1  [
2    {
3      "id": 1,
4      "title": "Tytuł1",
5      "productionYear": "1999-01-01"
6    },
7    {
8      "id": 3,
9      "title": "Tytuł33",
10     "productionYear": "2000-12-30"
11   },
12   {
13     "id": 4,
14     "title": "Tytuł4",
15     "productionYear": "1980-12-30"
16   }
17 ]

```

🔊 🛠️ ☒ Auto commit 🔗 🔗 Max rows: 1000 🔄 🛑 📄 Auto complete Off ▼ Auto select On ▼ ?

📁 jdbc:h2:file:./bazaDanych1

- 📁 VIDEO_CASSETTE
- 📁 INFORMATION_SCHEMA
- 📁 Sequences
- 👤 Users
- 📄 H2 1.4.200 (2019-10-14)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM VIDEO_CASSETTE

SELECT * FROM VIDEO_CASSETTE;

ID	PRODACTION_YEAR	TITLE
1	1999-01-01	Tytuł1
3	2000-12-30	Tytuł33
4	1980-12-30	Tytuł4

(3 rows, 2 ms)