

SPRAWOZDANIE

Imię Nazwisko: Klaudia Kromołowska

Temat zajęć laboratoryjnych: Implementacja API w oparciu o wzorzec architektoniczny REST w języku JAVA

1. Zwięzły opis aplikacji realizowanej podczas zajęć laboratoryjnych.

W ramach zajęć przygotowano aplikację udostępniającą prosty interfejs REST w oparciu o Spring Boot i język JAVA

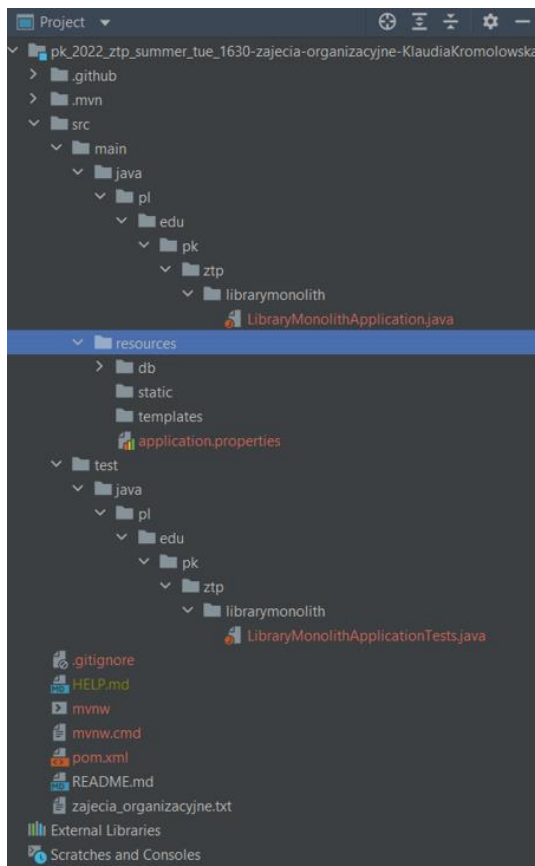
2. Użyte narzędzia.

(język programowania, najistotniejsze biblioteki, moduły baza danych itp. Nie opisujemy, tylko wypisujemy w postaci listy)

Spring, Java, Swagger, Yaml, Postman, baza danych H2, moduł Flyway

3. Sprawozdanie zgodne z plikiem README

1. Zrzut ekranu struktury projektu po zaimportowaniu do IDE



2. Wynik uruchomienia z komentarzem wyjaśniającym poszczególne etapy inicjalizacji aplikacji oraz źródło błędu.

```

      ____          _            __ _ _
     /  _ \        | |          // (_) |
    /  ___ \       | |__   _// /_>| |
   /  ___  \       | '_ \| // | |_|_|
  /  ___  \       | |_) || | | |_) |
 /  ___  \       | |__|/ |_| |__| |
/_  ___  \       |_____/____/_____|_|

:: Spring Boot ::                (v2.6.5)


2022-03-29 17:24:08.115 INFO 4452 -- [main] p.e.p.z.l.LibraryMonoLithApplication : Starting LibraryMonoLithApplication using Java 17.0.1 on DESKTOP-F5TIVUR wit
2022-03-29 17:24:08.126 INFO 4452 -- [main] p.e.p.z.l.LibraryMonoLithApplication : No active profile set, falling back to 1 default profile: "default"
2022-03-29 17:24:15.912 INFO 4452 -- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-03-29 17:24:15.961 INFO 4452 -- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-03-29 17:24:15.962 INFO 4452 -- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.60]
2022-03-29 17:24:16.483 INFO 4452 -- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-03-29 17:24:16.483 INFO 4452 -- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 8016 ms
2022-03-29 17:24:16.758 INFO 4452 -- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-03-29 17:24:17.520 INFO 4452 -- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-03-29 17:24:17.710 INFO 4452 -- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:Ll
2022-03-29 17:24:20.846 INFO 4452 -- [main] o.f.c.internal.license.VersionPrinter : Flyway Community Edition 8.0.5 by Redgate
2022-03-29 17:24:20.848 INFO 4452 -- [main] o.f.c.i.database.base.BaseDatabaseType : Database: jdbc:h2:mem:librarydb (H2 1.4)
2022-03-29 17:24:21.351 INFO 4452 -- [main] o.f.core.internal.command.DbValidate : Successfully validated 1 migration (execution time 00:00.200s)
2022-03-29 17:24:21.402 INFO 4452 -- [main] o.f.c.i.s.JdbcTableSchemaHistory : Creating Schema History table "PUBLIC"."flyway_schema_history" ...
2022-03-29 17:24:21.659 INFO 4452 -- [main] o.f.core.internal.command.DbMigrate : Current version of schema "PUBLIC": << Empty Schema >>
2022-03-29 17:24:21.722 INFO 4452 -- [main] o.f.core.internal.command.DbMigrate : Migrating schema "PUBLIC" to version "1.0 - create librarydb schema"
2022-03-29 17:24:22.238 INFO 4452 -- [main] o.f.core.internal.command.DbMigrate : Successfully applied 1 migration to schema "PUBLIC", now at version v1.0 (ex
2022-03-29 17:24:23.114 INFO 4452 -- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'
2022-03-29 17:24:23.159 INFO 4452 -- [main] p.e.p.z.l.LibraryMonoLithApplication : Started LibraryMonoLithApplication in 17.368 seconds (JVM running for 21.1)
```

Powyższy zrzut ekranu nie zawiera już błędu. Był on spowodowany brakiem konfiguracji połączenia do bazy danych.

Kolejne logi: start, informacja o braku czytania profili (używamy np. gdy środowisko testowe i to na którym pracujemy wymagają różnych baz danych) – w projekcie jest on domyślny. Następnie uruchamiany jest serwer Tomcat, który obsługuje zapytania restowe. Później inicjalizowana jest konfiguracja aplikacji webowej. Kolejno występuje połączenie do bazy danych (właśnie tu pojawiał się błąd). Za pomocą Flyway tworzony jest schemat bazy danych, po czym aplikacja zostaje uruchomiona na porcie 8080.

3. Wyjaśnienie znaczenia poszczególnych opcji konfiguracyjnych

```
spring.datasource.url=jdbc:h2:mem:librarydb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=admin
spring.datasource.password=password
spring.h2.console.enabled=true
```

Są to opcje konfiguracji bazy danych: url zawiera nazwę, driver odpowiedni sterownik (umożliwia tłumaczenie SQL na komunikaty zrozumiałe przez bazę danych), kolejne wiersze to nazwa użytkownika i hasło. Ostatni wiersz pozwala na dostęp do konsoli przeglądarkowej H2.

4. Wyjaśnienie celu i działania adnotacji @Repository oraz @Autowired

@Repository (z frameworku Spring) pozwala oznaczyć klasę aplikacji jako element umożliwiający dostęp do bazy danych

@Autowired - wstrzykiwanie zależności – zamiast ręcznie tworzyć obiekty i przekazywać konfigurację, ta adnotacja pozwala na automatyczną konfigurację w oparciu o application.properties.

5. UserController i BookController – elementy konfiguracji zgodnie z dokumentacją oraz realizacja obsługi błędów

W obu wspomnianych klasach zdefiniowane są metody, które będą pozwalały na wykonywanie różnych akcji na aplikacji.

@RestController – informacja o reprezentacji kontrolera, umożliwiającego wykonanie akcji.

USER CONTROLLER:

```
@GetMapping
```

```
List<UserDTO> getAllUsers()
```

Zwraca wszystkich użytkowników z bazy danych

```
@GetMapping("/{id}")
```

```
UserDTO getUserById(final Integer userID)
```

Zwraca informacje o użytkowniku znajdujące się w rekordzie o podanym ID. W przypadku braku wskazanego ID w bazie danych zwraca błąd 404

```
@DeleteMapping("/{id}")
```

```
void deleteUser(final Integer userID)
```

Usuwa użytkownika o podanym numerze ID. Jeżeli nie ma takiego użytkownika zwraca błąd 404, jeżeli z jakiegoś powodu nie da się go usunąć, błąd 403

```
@PostMapping
```

```
UserDTO postUser(final UserDTO user)
```

Tworzy nowego użytkownika w bazie danych.

BOOKS CONTROLLER:

```
@GetMapping
```

```
List<BookDTO> getAllBooks(final boolean showOnlyAvailable)
```

Zwraca informacje o wszystkich książkach znajdujących się w bazie danych

```
@GetMapping("/{id}")
```

```
BookDTO getBookRentals(final Integer bookID)
```

Zwraca informację o książce o podanym numerze ID uwzględniając listę jej wypożyczeń. W przypadku, kiedy nie ma w bazie danych książki o podanym ID zwraca błąd 404.

```
@PatchMapping("/return/{id}")
```

```
BookDTO patchRentBook(final Integer bookID, final Integer userID)
```

Usuwa informację o wypożyczeniu książki o podanym ID (następuje „zwrot”). Jeżeli nie podano numeru ID użytkownika lub nie ma go w bazie – błąd 401. Jeżeli nie można zwrócić książki (np. nie było wypożyczonej o tym numerze ID) – 409.

```
@PatchMapping("/rent/{id}")
```

```
BookDTO patchReturnBook(final Integer bookID, final Integer userID)
```

Dodaje informację o wypożyczeniu książki o podanym ID. Jeżeli nie podano numeru ID użytkownika lub nie ma go w bazie – błąd 401. Jeżeli wskazana książka nie jest dostępna lub nie można jej wypożyczyć – błąd 409.

6. BookRepository – opis publicznych metod

```
List<BookDTO> findAll
```

W zależności od podanego parametru funkcja zwróci wszystkie książki lub jedynie te które są dostępne do wypożyczenia.

```
boolean isBookAvailable
```

Funkcja sprawdza czy jest możliwe wypożyczenie książki – czy ilość książek wypożyczonych o podanym numerze ID jest mniejsza od ilości książek o wskazanym ID w całej bazie.

```
BookDTO findBookById
```

Zwraca informacje o książce o podanym numerze ID z tabeli tbl_books. W przypadku, kiedy nie występuje książka o takim ID, zwracany jest błąd 400.

```
List<BookRentalDTO> findRentalByBookId
```

Zwraca informacje o wypożyczeniach książki o podanym numerze ID

```
public boolean returnBook
```

Sprawdza czy możliwe jest wykonanie operacji zwracania książki

```
public boolean rentBook
```

Sprawdza czy możliwe jest wypożyczenie książki

7. Wyniki testów:

All Tests

Passed (9)

Failed (0)

Iteration 1

GET

GET all books

http://localhost:8080/books

/ GET all books

200 OK

792 ms

664 B

Pass

Status code is 200

PATCH

PATCH return book

http://localhost:8080/books/return/3

/ PATCH return book

200 OK

134 ms

354 B

Pass

Status code is 200

PATCH

PATCH return book - missing book

http://localhost:8080/books/return/111

/ PATCH return book - missing book

400 Bad Request

60 ms

251 B

Pass

Status code is 400

PATCH

PATCH return book - missing user

http://localhost:8080/books/return/1

/ PATCH return book - missing user

401 Unauthorized

22 ms

280 B

Pass

Status code is 401

PATCH

PATCH rent book

http://localhost:8080/books/rent/4

/ PATCH rent book

200 OK

21 ms

335 B

Pass

Status code is 200

PATCH

PATCH rent unavailable book

http://localhost:8080/books/rent/1

/ PATCH rent unavailable book

409 Conflict

19 ms

270 B

Pass

Status code is 409

PATCH

PATCH rent book - no user

http://localhost:8080/books/rent/1

/ PATCH rent book - no user

401 Unauthorized

18 ms

278 B

Pass

Status code is 401

GET

GET book rentals

http://localhost:8080/books/3

/ GET book rentals

200 OK

20 ms

354 B

Pass

Status code is 200

GET

GET available books

http://localhost:8080/books?available=true

/ GET available books

200 OK

18 ms

541 B

Pass

Status code is 200