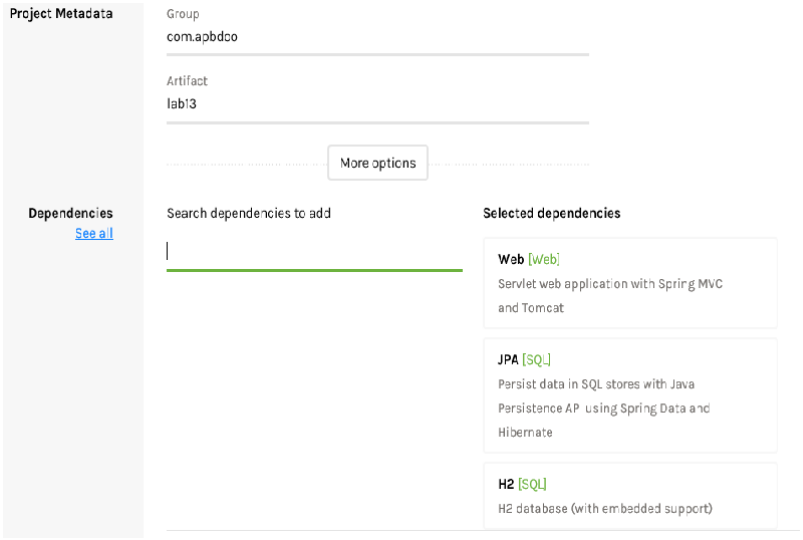
Laborator 12: RESTful Webservices

1. Pentru crearea unui nou proiect spring boot se poate utilizas https://start.spring.io/:



2. Rulati aplicatia 13 in mod debug. In application.properties setati:

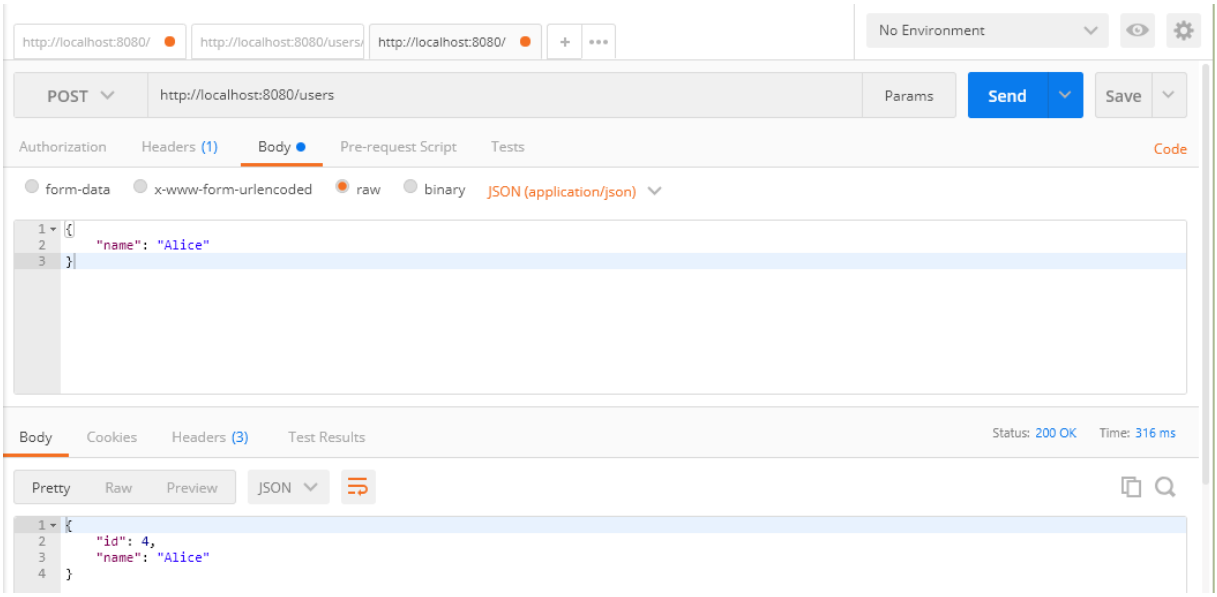
```
logging.level.org.springframework =debug
```

3. Testați în Postman.

```
http://localhost:8080/users/
http://localhost:8080/users/1
```

4. Adăugați un request de tip POST. Testați în POSTMAN.

```
@PostMapping(path = "/users")
    public User save(@RequestBody User user){
        return userService.save(user);
}
```



ResponseEntity
Adaugă headere și status-codes obiectului Response. Spre exemplu, dacă vom utiliza metoda *created* care primește ca argument o locație (URI) a unui nou obiect creat, se va adăuga URI-ul ca header și status code returnat de request va fi 201.

ServletUriComponentsBuilder
Poate fi utilizat în metodele din Controller pentru a genera dinamic URI-uri.

5. Utilizați ResponseEntity și ServletUriComponentsBuilder pentru a returna adresa URI a resursei create și status code 201 *CREATED*.

```
@PostMapping(path = "/users")
public ResponseEntity<User> save(@RequestBody User user){
   User savedUser = userService.save(user);
        ….
   return ResponseEntity.created(locationUri).build();
}
```

6. Testați în POSTMAN un request pentru a obține un user care nu va fi găsit. Este status codul returnat corect :500 Internal Server Error?

```
http://localhost:8080/users/1000
```

7. Creați o clasă care va extinde RuntimeException și care va fi adnotată ResponseStatus(HttpStatus.NOT_FOUND).

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFound extends RuntimeException {
    public UserNotFound(String message) {
        super(message);
    }
}
```

8. Modificați clasa UserServiceImpl astfel încât să utilizeze clasa de eroare UserNotFound.

ResponseEntityExceptionHandler
este clasa utilizată pentru tratarea centralizată a erorilor. Metodele @ExceptionHandler returnează un ResponseEntity care prin intermediul unui converter include un mesaj de eroare custom.

https://www.baeldung.com/exception-handling-for-rest-with-spring

9. Creați un pattern pentru mesajele de eroare. Creați clasa:

```
@Data
@AllArgsConstructor
public class ExceptionPattern {
    Date date;
    String message;
    String description;
}
```

10. Extindeți ResponseEntityExceptionHandler:

```
@ControllerAdvice
@RestController
public class CustomizedExceptionHandler
    extends ResponseEntityExceptionHandler {

    @ExceptionHandler(UserNotFound.class)
    protected ResponseEntity<Object>
handleException(RuntimeException ex, WebRequest request) {
      ExceptionPattern exception = new ExceptionPattern(new
Date(), ex.getMessage(), request.getDescription(true));
      return new ResponseEntity(exception,HttpStatus. .NOT_FOUND);
    }
}
```

protected void configure(AuthenticationManagerBuilder auth) throws Exception
este metoda utilizată pentru configurarea utilizatorilor (in memory, database, ldap etc.)

11. Completați metoda:

```
@DeleteMapping("/users/{userId}")
    public void deleteUser(@PathVariable Long userId){

}
```

12. Adăugați constrângerea ca numele să aibă cel mult 20 de caractare și testați pentru a obține codul 400 BAD REQUEST.

13. Suprascrieți metoda handleMethodArgumentNotValid a clasei ResponseEntityExceptionHandler.

HATEOS Hypermedia as the engine of application state
Componentă a aplicațiilor REST prin care împreună cu o resursă, utilizatorul primește informații suplimentare cu privire la acțiunile pe care le poate face (linkuri).

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```