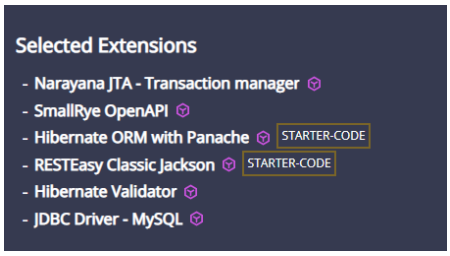
Academia Tehnică Militară "Ferdinand I“
Facultatea de Sisteme Informatice şi Securitate Cibernetică

Paradigme de programare (în Java)

Lab 13/Curs 13- Quarkus Framework

Col(r) Traian Nicula

A. Se creează proiectul Quarkus ***lab13*** folosind interfața web ***Quarkus Configure your application*** (https://code.quarkus.io/) astfel:
   - Se deschide în browser link-ul https://code.quarkus.io/
   - Se completează datele ca în figura de mai jos și se aleg extensiile *quarkus-resteasy-jackson, quarkus-smallrye-openapi, quarkus-hibernate-validator, quarkus-hibernate-orm-panache, quarkus-narayana-jta, quarkus-jdbc-mysql*



   - Se apasă butonul ***Generate your application*** și apoi ***DOWNLOAD THE ZIP***
   - Se extrage folder-ul ***lab13*** din arhivă și se copiază în folder-ul cu numele studentului
   - Se deschide proiectul ***lab13*** în VSC
   - Se adaugă în fișierul ***application.properties*** proprietățile de vai jos. Parola este cea stabilită la instalare.

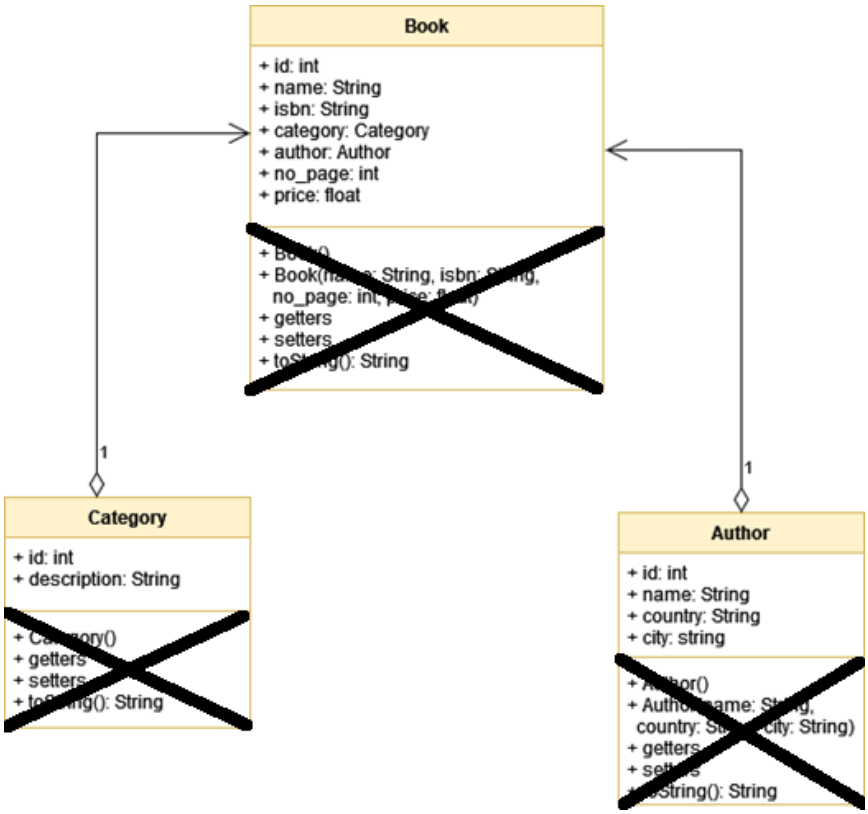
```
# datasource configuration
quarkus.datasource.db-kind = mysql
quarkus.datasource.username = user
quarkus.datasource.password = qaz123QAZ!@#
quarkus.datasource.jdbc.url = jdbc:mysql://localhost:3306/testdb

# drop and create the database at startup
quarkus.hibernate-orm.database.generation=drop-and-create
```

   - Creează fișierul resources/***import.sql***, copiază conținutul fișierului ***lab12.sql*** și se salvează.

Se rezolvă următoarele exerciții:

1. Scrieți un program Java care: (**2.5p**)
   - creează clasele conform diagramei UML de mai jos, cu modificatorul de acces ***public***, fără constructori și metode getters și setters



- convertește clasa ***Category*** în entitate ***Panache*** și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "categorySequence",`
	`sequenceName = "hibernate_sequence",`
	`allocationSize = 1, initialValue = 50)`
	`@GeneratedValue(generator = "categorySequence")`

description	Nenul cu dimensiunea maximă de 25 caractere

- convertește clasa ***Author*** în entitate ***Panache*** și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "authorSequence",`
	`sequenceName = "hibernate_sequence",`
	`allocationSize = 1, initialValue = 50)`
	`@GeneratedValue(generator = "authorSequence")`

name	Nenul
country	Nenul
city	Nenul

- convertește clasa ***Book*** în entitate ***Panache*** și adaugă constrângerile:

Câmp	Constrângere

id	Cheie primară autogenerată
	`@Id`
	`@SequenceGenerator(name = "bookSequence", sequenceName = "hibernate_sequence", allocationSize = 1, initialValue = 50)`
	`@GeneratedValue(generator = "bookSequence")`

name	Nenul
isbn	Nenul cu dimensiunea maximă de 15 caractere
category	Nenul
author	Nenul
no_page	Nenul, pozitiv
price	Partea întreagă are 4 cifre, partea fracțională are 2 cifre

2. Scrieți un program Java care: (**2.5p**)
   - creează clasa resursă ***CategoriesResource*** cu URI ***/categories*** având:
   - metoda resursă *List<Category> get()* care:
     a. poate fi accesată cu metoda **GET** la URI ***/categories***
     b. returnează reprezentarea JSON a listei categoriilor cu codul de stare 200 OK
     c. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui)
   - metoda resursă *Response create(Category cayegory)* care :
     a. poate fi accesată cu metoda **POST** la URI ***/categories***
     b. persistă obiectul ***Category*** în baza de date
     c. returnează obiectul ***Category*** în format JSON cu codul de stare 201 CREATED
     d. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui) cu JSON:
     ```
     {"description":"Literature"}
     ```
3. Scrieți un program Java care: (**2.5p**)
   - creează clasa resursă ***AuthorsResource*** cu URI ***/authors*** având:
   - metoda resursă *List<Author> get()* care:
     a. poate fi accesată cu metoda **GET** la URI ***/authors***
     b. returnează reprezentarea JSON a listei autorilor sortați după nume cu codul de stare 200 OK
     c. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui)
   - metoda resursă *Response create(Author author)* care :
     a. poate fi accesată cu metoda **POST** la URI ***/authors***
     b. persistă obiectul ***Author*** în baza de date
     c. returnează obiectul ***Author*** în format JSON cu codul de stare 201 CREATED
     d. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui) cu JSON:
     ```
     {"name":"Marin Preda","country":"Romania","city":"Bucharest"}
     ```
4. Scrieți un program Java care: (**2.5p**)
   - creează clasa resursă ***BooksResource*** cu URI ***/books*** având:
   - metoda resursă *List<Book> get()* care:
     a. poate fi accesată cu metoda **GET** la URI ***/books***

   b. returnează reprezentarea JSON a listei cărților sortate după nume cu codul de stare 200 OK
   c. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui)
- metoda resursă *Book getBookById(int id)* care:
   a. poate fi accesată cu metoda **GET** la URI ***/books/{id}***
   b. returnează reprezentarea agentului în format JSON cu codul de stare 200 OK
   c. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui)

Temă pentru acasă:

5. Scrieți un program Java care: (**1p**)
   - adaugă la clasa ***BooksResource*** metoda *Response create(Book book)* care:
      a. poate fi accesată cu metoda **POST** la URI ***/books***
      b. persistă obiectul ***Book*** în baza de date
      c. returnează obiectul ***Book*** în format JSON cu codul de stare 201 CREATED
      d. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui) cu JSON:
      ```
      {"name":"Morometzii","isbn":"9786069098103","category":{"id":50,"description":"Literature"},"author":{"id":51,"name":"Marin Preda","country":"Romania","city":"Bucharest"},"no_page":1024,"price":70.00}
      ```
6. Scrieți un program Java care: (**1p**)
   - adaugă la clasa ***BooksResource*** metoda *Response update(int id, Book book)* care:
      a. poate fi accesată cu metoda **PUT** la URI ***/books/{id}***
      b. caută obiectul ***Book*** în baza de date
      c. actualizează obiectul ***Book*** cu valorile transmise prin reprezentarea JSON ca parametru
      d. returnează obiectul ***Book*** în format JSON cu codul de stare 200 OK
      e. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui) cu JSON:
      ```
      {"name":"Morometii","isbn":"9786069098103","no_page":1000,"price":100.00}
      ```
   - adaugă la clasa ***BooksResource*** metoda *Response delete (int id)* care:
      a. poate fi accesată cu metoda **DELETE** la URI ***/books/{id}***
      b. caută obiectul ***Book*** în baza de date
      c. șterge obiectul ***Book***
      d. returnează răspuns fără corp cu codul de stare 204 NO_CONTENT
      e. folosește Swagger UI pentru verificare (http://localhost:8080/q/swagger-ui). Șterge obiectul ***Book*** creat și modificat anterior.