Persado Project Deliverable Documentation

Delivered by: FOTIS FLOROS

1 General Information

The below technologies are used for the implementation of the project.

Technology Information	
Language	Java 1.8
Framework	Spring Boot
Database	H2 which is an embedded, open-source, and in-memory database. It is a relational database management system written in Java.
Repository	Github has been used for version control. Link: https://github.com/fotf91/persadodeliverable.git

1.1 Get Started

Clone the project by using the following command:

- clone https://github.com/fotf91/persadodeliverable.git

To run the project, navigate inside the project folder in order to run the application locally, then execute the following commands:

- mvn clean install
- mvn spring-boot:run

2 Database

Below are the database tables of the application.

Moreover every time the Application starts, data are automatically inserted in the below tables.

Database can be accessed from the browser using:

- Endpoint: http://localhost:8080/h2
- Username: user
- Password: pass

2.1 Table AUTHOR

Name	Type	Comment

AUTHOR_ID	Int	Primary Key. Automatically generated.
FIRST_NAME	Varchar(50)	Not Null
LAST_NAME	Varchar(50)	Not Null
EMAIL_ADDRESS	Varchar(50)	Not Null, Unique
DOB	Date	Not Null

2.2 Table PUBLISHER

Name	Type	Comment
PUBLISHER_ID	Int	Primary Key. Automatically generated.
NAME	Varchar(50)	Not Null
TELEPHONE	Varchar(10)	Not Null
ADDRESS	Varchar(100)	Not Null

2.3 Table BOOK

Name	Type	Comment
ISBN	Varcar(17)	Primary Key.
TITLE	Varchar(50)	Not Null
DESCRIPTION	Varchar(250)	Not Null
VISIBLE	Boolean	Not Null
CREATION_DATE	Date	Not Null
AUTHOR_ID	Int	Not Null. Foreign key to column AUTHOR_ID of table AUTHOR.
PUBLISHER_ID	Int	Foreign key to column PUBLISHER_ID of table PUBLISHER.

3 Business Objects

This chapter describes the business objects along with their validations. They are used in order to communicate the information between the API consumer and the database.

3.1 Author Business Object

This object contains the same fields also used in the AUTHOR table.
Describes the JSON used as payload in operation:

- Create Author Service.

Name	Type	Comment
authorId	Long	Automatically Generated.
firstName	String	Restrictions • Cannot be null • Cannot be empty • Size must be between 3 and 50
lastName	String	Restrictions • Cannot be null • Cannot be empty • Size must be between 3 and 50
email	String	Restrictions • Cannot be null • Cannot be empty • Must be unique • A valid email must contain at least one character, an @ symbol, then a non whitespace character.
DOB	LocalDate	Restrictions • Cannot be null • Valid Date format is Year-Month-Day. Example: 1970-12-30.

3.2 Publisher Business Object

This object contains the same fields also used in the PUBLISHER table.
Describes the JSON used as payload in operation:

- Create Publisher Service.

Name	Type	Comment
publiserId	Long	Automatically Generated.
name	String	Restrictions • Cannot be null • Cannot be empty • Size must be between 3 and 50
telephone	String	Restrictions • Cannot be null • Cannot be empty • Size must be exactly 10 characters. It must contain only digits.
address	String	Restrictions • Cannot be null • Cannot be empty • Size must be between 3 and 100.

3.3 Book Business Object

This object contains the same fields also used in the BOOK table.
Describes the JSON used as payload in operation:

- Create Book Service.
- Update an Existing Book Service.

Name	Type	Comment
isbn	String	Restrictions • Must be unique. • Must have the format XXX-X-XX-XXXXXX-X where X is a digit.
title	String	Restrictions • Cannot be null • Cannot be empty • Size must be between 3 and 50
description	String	Restrictions • Cannot be null • Cannot be empty • Size must be

		between 3 and 250.
visible	Boolean	Default value is false.
creationDate	LocalDate	Restrictions • Cannot be null • Valid Date format is Year-Month-Day. Example: 1970-12-30.
authorId	Long	Restrictions • Cannot be null. • It must be a valid author Id according to the data of AUTHOR table.
publisherId	Long	Restrictions • It must be a valid publisher Id according to the data of PUBLISHER table.

3.4 Book With Publisher Business Object

This object describes the JSON response of the operations:

- Retrieve Visible Books by Publisher Id Service
- Retrieve Visible Books With Publisher Service

No restrictions are described since it is used for read purposes.

Name	Type	Comments
title	String	The title of a Book.
description	String	The description of a Book.
isbn	String	The isbn of a Book.
authorsFullName	String	The First Name and the Last Name of an Author.

3.5 Book With Details Business Object

This object describes the JSON response of the operation:

- Retrieve Books by Isbn Service.

No restrictions are described since it is used for read purposes.

Name	Type	Comments
title	String	The title of a Book.
description	String	The description of a Book.
isbn	String	The isbn of a Book.
creationDate	String	The book creation date. (Format example: 21/09/1937)
authorsFullName	String	The First Name and the Last Name of an Author.
authorsEmail	String	The Email of the Author.
authorsDateOfBirth	String	The author's date of birth. (Format example: 21st of November 1924).
publisherName	String	The name of the publisher.
publisherAddress	String	The address of the publisher.

4 Services

This chapter describes the services that are implemented.

By default the project runs on localhost:8080, thus all the endpoints of this document are described with that endpoint.

4.1 Create Author Service

This service can be used for the creation of an Author Entity.

Resource Verb	POST
Endpoint	http://localhost:8080/api/author
Expected Input Format	JSON
Expected Output Format	JSON
Output HTTP Status	200 OK

Input	A JSON of the format of Author Business Object. • If authorId is provided in the input it will be ignored, since it is automatically generated.
Output	A JSON of the format of Author Business Object.

Sample Input	{
	"firstName": "Fotis",
	"lastName": "Floros",
	"email": "fotf91@gmail.com",
	"dateOfBirth": "1990-07-19"
	}

Sample Output	{
	"authorId": 7,
	"firstName": "Fotis",
	"lastName": "Floros",
	"email": "fotf91@gmail.com",
	"dateOfBirth": "1990-07-19"
	}

Validations	The restrictions of the Author Business Object will apply. Moreover, if the Input Payload does not meet the restrictions of the Business Object, then the response and HTTP Code of chapter "Input Payload Validation Failure" will be returned.

4.2 Create Publisher Service

This service can be used for the creation of a Publisher Entity.

Resource Verb	POST
Endpoint	http://localhost:8080/api/publisher
Expected Input Format	JSON
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	A JSON of the format of Publisher Business Object. • If publisherId is provided in the input it will be ignored, since it is automatically generated.

Output	A JSON of the format of Publisher Business Object.

Sample Input	{
	"name": "Patakis",
	"telephone": "2104315881",
	"address": "Ermou 10"
	}

Sample Output	{
	"publisherId": 7,
	"name": "Patakis",
	"telephone": "2104315881",
	"address": "Ermou 10"
	}

Validations	The restrictions of the Publisher Business Object will apply.

4.3 Create Book Service

This service can be used for the creation of a Book Entity.

Resource Verb	POST
Endpoint	http://localhost:8080/api/book
Expected Input Format	JSON
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	A JSON of the format of Book Business Object. • In order to reference the Author or Publisher the authorId and publisherId elements must be used. • If the field visible is not provided, then false will be the default value.
Output	A JSON of the format of Book Business Object.
Sample Input	{ "title": "Book Title", "description": "Book description.", "isbn": "000-0-00-000000-1", "visible": **true**, "creationDate": "1990-05-10",

	<pre>"authorId": "1",
	"publisherId": null
	}</pre>

Sample Output	<pre>{
	"isbn": "000-0-00-000000-1",
	"title": "Book Title",
	"description": "Book description.",
	"visible": true,
	"creationDate": "1990-05-10",
	"authorId": 1,
	"publisherId": null
	}</pre>

Validations	The restrictions of the Book Business Object will apply. Moreover, if the Input Payload Contains reference to non existing resources, then the response and HTTP Code of chapter "Creation of a new resource with reference to non existing entity" will be returned.

4.4 Update an Existing Book Service

This service can be used in order to update a Book Entity.

Resource Verb	PUT
Endpoint	http://localhost:8080/api/book/<isbn>
Expected Input Format	JSON
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	The parameter isbn of the endpoint must refer to an existing book isbn. A JSON of the format of Book Business Object. • In order to reference the Author or Publisher the athorId and publisherId elements must be used. • If the field visible is not provided, then false will be the default value.
Output	A JSON of the format of Book Business Object.

<table>
<tr><td>Sample Input</td><td>URI: http://localhost:8080/api/book/978-0-26-110334-0
{
"title": "My Book",
"description": "My description.",
"isbn": "978-0-26-110334-0",
"visible": true,
"creationDate": "1990-05-10",
"authorId": "1",
"publisherId": null
}</td></tr>
<tr><td>Sample Output</td><td>{
"isbn": "978-0-26-110334-0",
"title": "My Book",
"description": "My description.",
"visible": true,
"creationDate": "1990-05-10",
"authorId": 1,
"publisherId": null
}</td></tr>
<tr><td>Validations</td><td>The restrictions of the Book Business Object will apply.
Moreover the following errors may occur:
● If the endpoint parameter isbn, refers to a non existing book isbn a 400 Bad Request response along with the respective description will be returned.
○ Response example
{
"statusCode": "BAD_REQUEST",
"timestamp": "2021-03-16T22:01:31.656+00:00",
"description": "uri=/api/book/978-0-26-110111-0",
"errors": [
"Could not find resource with search term: 978-0-26-110111-0"
]
}
● If the JSON input isbn, refers to an existing book isbn(which means that the isbn will be updated to that one), a 400 Bad Request response along with the respective error messages will be returned.</td></tr>
</table>

	○ Response example
	<pre>{
	  "statusCode": "BAD_REQUEST",
	  "timestamp": "2021-03-16T22:02:24.922+00:00",
	  "description": "uri=/api/book/978-3-16-148410-0",
	  "errors": [
	  "A book with the new ISBN 090-8-60-666487-1 already exists."
	  ]
	}</pre>

4.5 Delete an Existing Book Service

This service can be used in order to delete a Book Entity.

Resource Verb	DELETE
Endpoint	http://localhost:8080/api/book/<isbn>
Expected Input Format	N/A
Expected Output Format	N/A
Output HTTP Status	200 OK
Input	The parameter isbn of the endpoint must refer to an existing book isbn.
Output	HTTP Status 200 OK will be returned.
Sample Input	URI: http://localhost:8080/api/book/978-0-26-110334-0
Sample Output	HTTP Status 200 OK

Validations	In case the isbn defined on the endpoint parameter does not refer to an existing book, then a 400 Bad Request response along with the respective error messages will be returned.
	● Response Example:
	<pre>{
	  "statusCode": "BAD_REQUEST",
	  "timestamp": "2021-03-16T22:03:20.680+00:00",</pre>

	"description": "uri=/api/book/978-0-26-1111-0", "errors": ["Could not find resource with search term: 978-0-26-1111-0"] }

4.6 Retrieve Visible Books by Publisher Id Service

This service can be used in order to retrieve visible books with the respective author information (as Book With Publisher Business Object). The retrieval takes place based on the publisher Id.
The books will be ordered as follows:

- Alphabetically by author's last name
- If books of the same author exist, they will be ordered by isbn.

Resource Verb	GET
Endpoint	http://localhost:8080/api/book?publisherId=<publisherId>
Expected Input Format	N/A
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	The request endpoint will contain a publisher Id.
Output	A list of Book With Publisher Business Object as JSON.
Sample Input	http://localhost:8080/api/book?publisherId=1
Sample Output	[{ "title": "Dune", "description": "Dune is a 1965 science-fiction novel by American author Frank Herbert, originally published as two ...", "isbn": "478-0-60-005171-0", "authorsFullName": "Frank Herbert" }, { "title": "The Lost Road and Other Writings", "description": "The Lost Road and Other Writings description.", "isbn": "039-5-71-041838-0",

	"authorsFullName": "Christopher Tolkien"
	},
	{
	"title": "The War of the Jewels",
	"description": "The War of the Jewels description.",
	"isbn": "978-0-26-110334-0",
	"authorsFullName": "Christopher Tolkien"
	},
	{
	"title": "The Hobbit",
	"description": "The Hobbit description.",
	"isbn": "978-3-16-148410-0",
	"authorsFullName": "Christopher Tolkien"
	}
	]

Validations	In case no books are found, then the response and HTTP Code of chapter "A GET operation cannot find any entity" will be returned.

4.7 Retrieve Visible Books With Publisher Service

This service can be used in order to retrieve visible books that have a publisher, with the respective publisher information (as Book With Publisher Business Object).
The books will be ordered as follows:

- Alphabetically by author's last name
- If books of the same author exist, they will be ordered by isbn.

Resource Verb	GET
Endpoint	http://localhost:8080/api/book/published
Expected Input Format	N/A
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	N/A
Output	A list of Book With Publisher Business Object as JSON.
Sample Input	http://localhost:8080/api/book?publisherId=1

Sample Output


```
[
  {
    "title": "Slinky Malinki",
    "description": "Slinky Malinki description.",
    "isbn": "090-8-60-666487-1",
    "authorsFullName": "Lynley Dodd"
  },
  {
    "title": "Hairy Maclary from Donaldsons Dairy",
    "description": "Hairy Maclary from Donaldsons Dairy description.",
    "isbn": "190-8-60-620615-7",
    "authorsFullName": "Lynley Dodd"
  },
  {
    "title": "Story of my life",
    "description": "This is the story of my life.",
    "isbn": "000-0-00-000511-0",
    "authorsFullName": "Fotis Floros"
  },
  {
    "title": "Dune",
    "description": "Dune is a 1965 science-fiction novel by American author Frank Herbert, originally published as two ...",
    "isbn": "478-0-60-005171-0",
    "authorsFullName": "Frank Herbert"
  },
  {
    "title": "Dune",
    "description": "Dune is a 1965 science-fiction novel by American author Frank Herbert, originally published as two ...",
    "isbn": "978-9-60-306372-0",
    "authorsFullName": "Frank Herbert"
  },
  {
    "title": "The Lost Road and Other Writings",
    "description": "The Lost Road and Other Writings description.",
    "isbn": "039-5-71-041838-0",
    "authorsFullName": "Christopher Tolkien"
  },
  {
    "title": "The War of the Jewels",
    "description": "The War of the Jewels description.",
    "isbn": "978-0-26-110334-0",
    "authorsFullName": "Christopher Tolkien"
```

	},
	{
	"title": "The Hobbit",
	"description": "The Hobbit description.",
	"isbn": "978-3-16-148410-0",
	"authorsFullName": "Christopher Tolkien"
	}
	]

Validations	In case no books are found, then the response and HTTP Code of chapter "A GET operation cannot find any entity" will be returned.

4.8 Retrieve Books by Isbn Service

This service can be used in order to retrieve Books according to the Isbn value.

Resource Verb	GET
Endpoint	http://localhost:8080/api/book/<isbn>
Expected Input Format	N/A
Expected Output Format	JSON
Output HTTP Status	200 OK
Input	N/A
Output	A list of Book With Details Business Object as JSON.
Sample Input	http://localhost:8080/api/book/978-3-16-148410-0

Sample Output	{
	"title": "The Hobbit",
	"description": "The Hobbit description.",
	"isbn": "978-3-16-148410-0",
	"creationDate": "21/09/1937",
	"authorsFullName": "Christopher Tolkien",
	"authorsEmail": "christophertolkien@gmail.com",
	"authorsDateOfBirth": "21st of November 1924",
	"publisherName": "Houghton Mifflin",
	"publisherAddress": "Antistratou 15"
	}

Validations	In case no books are found, then the response and HTTP Code of chapter "A GET operation cannot find any entity" will be returned.

5 General Services Errors

This chapter gives a general description of services errors that may occur in multiple cases. Those errors have not been described explicitly in the Validations part of the Services Chapter.

5.1 Non Existing Endpoint

Case	A non existing endpoint is used (example: http://localhost:8080/api/authoraaa).
Output HTTP Status	404 Not Found

Example of Expected Output	{
	"timestamp": "2021-03-16T20:04:43.654+00:00",
	"status": 404,
	"error": "Not Found",
	"message": "No message available",
	"path": "/api/authoraaa"
	}

5.2 Input Payload Validation Failure

Case	The input payload of a service that persists (saves/updates) entities, contains data that may result to validation failure.
Output HTTP Status	400 Bad Request
Example Scenario Description	The API consumer wants to create a publisher by invoking the endpoint http://localhost:8080/api/publisher with Verb POST. In this example the following errors validation errors for name and telephone are expected.
Example Scenario Input Payload	{ "name": "A",

	<pre>"telephone": "12345678901",
	"address": "Ermou 10"
	}</pre>

Expected Output	<pre>{
	"statusCode": "BAD_REQUEST",
	"timestamp": "2021-03-16T20:22:41.394+00:00",
	"description": "uri=/api/publisher",
	"errors": [
	"name: size must be between 3 and 50",
	"telephone: Telephone can contain only digits and it must have length of 10."
	]
	}</pre>

5.3 Creation of a new resource with reference to non existing entity

Case	The input payload contains data which describe a reference to a non existing resource.
Output HTTP Status	400 Bad Request
Example Scenario Description	The API consumer wants to create a book by invoking the endpoint http://localhost:8080/api/book with Verb POST. In this example the author Id does not exist in the database.

Example Scenario Input Payload	<pre>{
	"title": "Book Title",
	"description": "Book description.",
	"isbn": "000-0-00-000000-1",
	"visible": **true**,
	"creationDate": "1990-05-10",
	"authorId": "100",
	"publisherId": **null**
	}</pre>

Expected Output	<pre>{
	"statusCode": "BAD_REQUEST",
	"timestamp": "2021-03-16T21:19:30.579+00:00",
	"description": "uri=/api/book",</pre>

	<pre>"errors": [
	  "There is a reference to a non existing entity."
	]
	}</pre>

5.4 A GET operation cannot find any entity

Case	A GET operation cannot retrieve any resources.
Output HTTP Status	404 Not Found
Example Scenario Description	The API consumer wants to retrieve visible books according to the publisherId, by invoking the endpoint http://localhost:8080/api/book?publisherId=10 where 10 is a non existing publisherId.

Expected Output	<pre>{
	"statusCode": "NOT_FOUND",
	"timestamp": "2021-03-16T22:14:16.246+00:00",
	"description": "uri=/api/book",
	"errors": [
	  "Could not find resource with search term: 10"
	]
	}</pre>