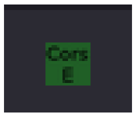
SpringBoot API Documentation

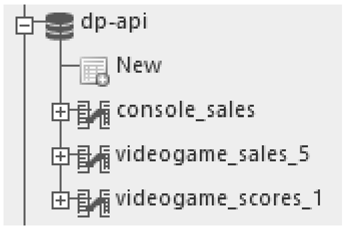
Konstantin Boguev
4669886
Data processing resit

Set-up

1. Download the repository from GitHub
2. Download Xampp (https://www.apachefriends.org/index.html)
3. Download a CORS bypasser such as CORSE (https://addons.mozilla.org/en-US/firefox/addon/cors-everywhere/)
4. Open the website.html page found in the “src/main/website” folder in a local server, the developer used the “Live Server” extension which runs inside Visual Studio Code (https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer)
5. Activate the CORS bypass extension by simply clicking the extension logo in the top right of your browser

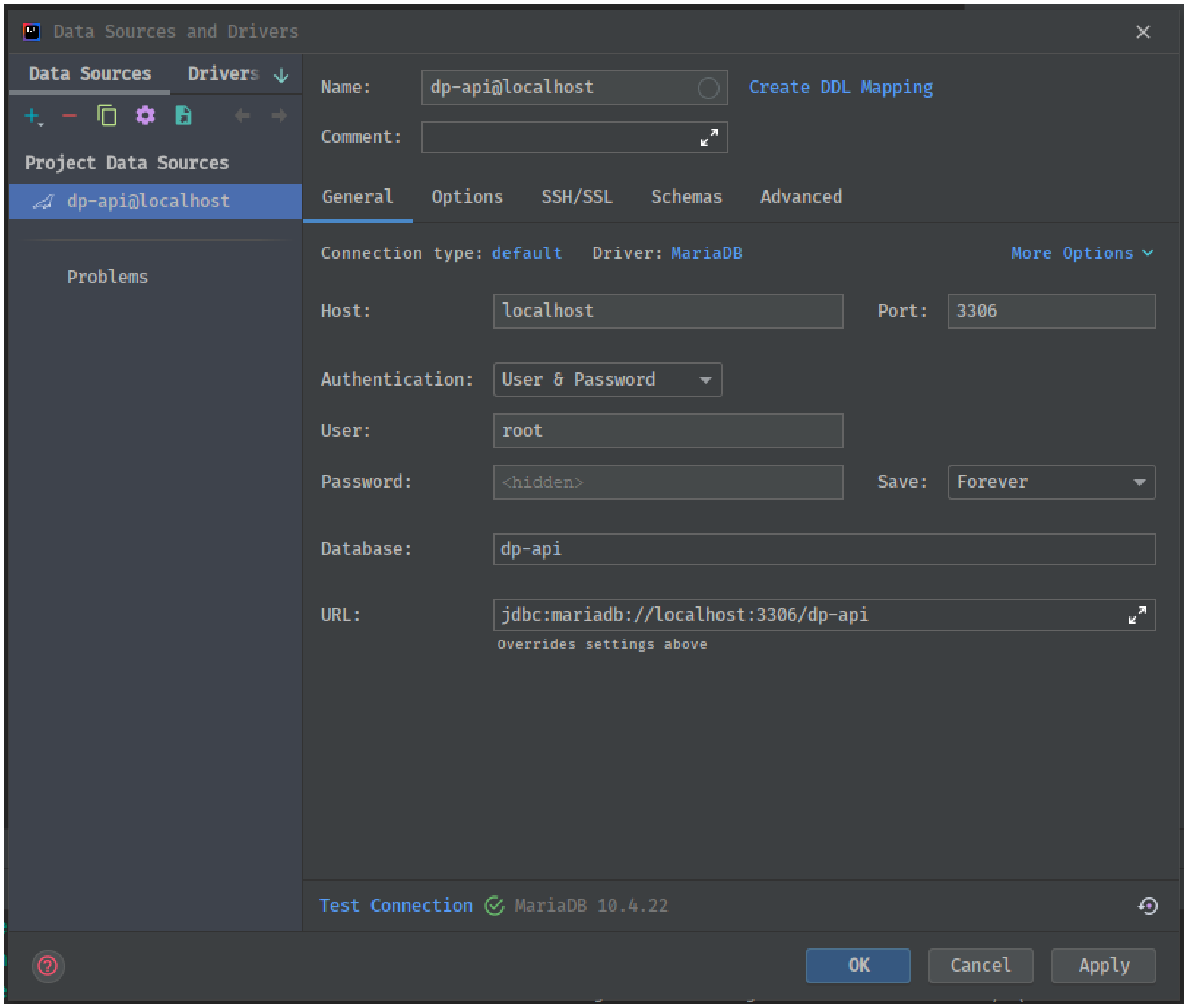


6. Set up the xampp database by creating a new database called “dp-api” and importing the “dp-api.sql” file found inside the “src/main/DB-DATA” folder in the repository. This should result in the following:



```
dp-api
  New
  console_sales
  videogame_sales_5
  videogame_scores_1
```

7. Open the application inside an IDE such as “IntelliJ”
8. Set up the data sources in IntelliJ by clicking the databases icon on the right and inserting a new “MariaDB” source with the following configuration (password is “toor”):



Data Sources and Drivers

Name: dp-api@localhost

Comment:

Connection type: default Driver: MariaDB

Host: localhost Port: 3306

Authentication: User & Password

User: root

Password: <hidden> Save: Forever

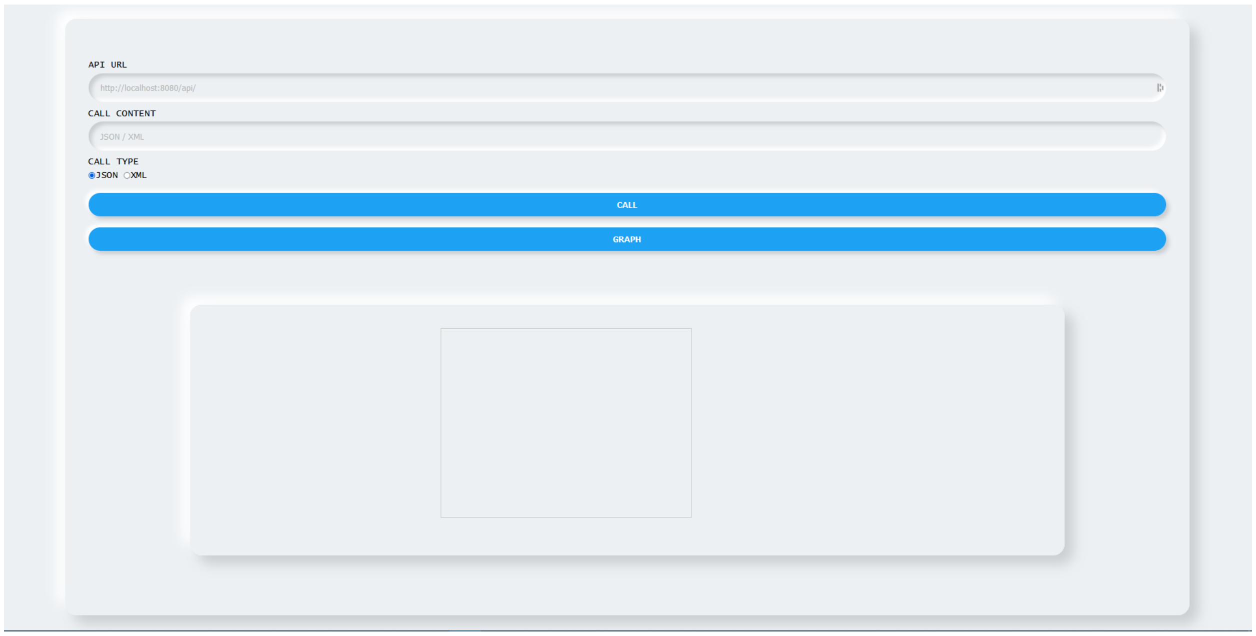
Database: dp-api

URL: jdbc:mariadb://localhost:3306/dp-api

Overrides settings above

Test Connection ✓ MariaDB 10.4.22

9. Run “DataProcessingApplication”.
10. On the website you should see the following:



- API url is the path of the API which you want to call, refer to the table below to know all paths the API offers
- Call content is the JSON / XML body of the request
- Call type determines which type of content is being sent and received. To send a JSON body and receive a JSON response use the JSON button, for XML use the XML button.
- The CALL button executes the call with the provided path and body
- The GRAPH button generates a graph comparing the sales and ratings of the Wii and the XBOX360
- The call results will be displayed inside the small box in the interface where you can scroll. Unfortunately, due to how HTML processes XML it is not possible to show symbols such as “<” and “>” as they get interpreted as HTML content. I could not find a workaround for this.

Design choices

SpringBoot was utilized as the developer already had some previous experience with the framework. SpringBoot also provides fast speeds and ample module compatibility.

PUT was chosen instead of PATCH as the developer wanted the user of the API to provide the API with a call body which contains all attributes instead of only the ones which he wishes to modify. This way all methods can be validated using a single JSON / XML schema instead of having to implement multiple ones.

The database host chosen was XAMPP as the developers already had experience with it and it is easy to set up as well as import data.

Graph.js was used as it offers good speed while maintaining good looks. The developer was also experienced with this library.

Endpoints (http://localhost:8080/api/...)

DataSet	Path	Method	Accepts	Returns
ConsoleSales	consoleSales/post	POST	JSON/XML	Success or not
	consoleSales/getall	GET	///	JSON / XML
	consoleSales/put/{id}	PUT	JSON / XML & REQUIRES ID	Success or not
	consoleSales/delete/{id}	DELETE	ID	Success or not
GameRatings	gameRatings/post	POST	JSON/XML	Success or not
	gameRatings/getall	GET	///	JSON / XML
	gameRatings/put/{id}	PUT	JSON / XML & REQUIRES ID	Success or not
	gameRatings/delete/{id}	DELETE	ID	Success or not
VideogameSales	videogameSales/post	POST	JSON/XML	Success or not
	videogameSales/getall	GET	///	JSON / XML
	videogameSales/put/{id}	PUT	JSON / XML & ID	Success or not
	videogameSales/delete/{id}	DELETE	ID	Success or not

Body format for calls (all required)

ConsoleSales		GameRatings		VideogameSales	
Property	**Type**	**Property**	**Type**	**Property**	**Type**
consoleId	string	name	string	name	string
consoleName	string	platform	string	platform	string
manufacturer	string	releaseYear	string	releaseYear	number
releaseYear	integer (min 1900)	score	number (min 0 max 100)	genre	string
sales	number (min 0)	userScore	number (min0 max 10)	publisher	string
		developer	string	globalSales	number
		genre	string	critScore	number

All JSON body inputs must be enclosed by “{}”, XML calls must begin by declaring the XML version and encoding with the following format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
// Body content
</root>
```