VANIER COLLEGE
TEACHER: SLEIMAN RABAH

420-511-VA WEB SERVICES – WINTER 2025

COMPUTER SCIENCE DEPARTMENT

ASSIGNMENT #1

IMPLEMENTING A DATA-EXCHANGE REST-BASED WEB SERVICE

Due date: scheduled on LÉA

CONSTRAINTS

- This is an **individual** assignment,
- You are not permitted to collaborate nor share code with any of your classmates,
- Using ChatGPT, GitHub Copilot, or any similar tools is not permitted,
- It is your responsibility to be familiar with the Cheating and Plagiarism Policy (Section 11.4)
- **Read the above points five more times.**

OBJECTIVES

- To implement a REST data exchange Web service.
- To consume and test a REST Web service using a REST client.
- To understand the fundamentals of HTTP message passing mechanism within the context of Web services.

REQUIRED TOOLS

- ***XAMPP*** and VS Code
- REST client VS Code Thunder Client extension
- Slim framework documentation
- World Cup database
- Frostybee's Slim app starter template

OVERVIEW

In this assignment, we will implement a REST-based web service for data exchange, capable of managing and sharing information about FIFA World Cup tournaments. A database containing information about all tournaments that have taken place to date is provided, along with a Slim-based application skeleton.

The web service to be implemented will expose a set of resources, which are detailed in the Implementation section below.

Please note that you are not required to implement a client application for this assignment. However, you are required to use Thunder Client to interact with and test the implementation of your web service.

PART I: SETTING UP THE DEVELOPMENT ENVIRONMENT

You are provided with a working Slim app template that must be used for this assignment. This Slim app needs to be deployed on an Apache web server and is already configured with the required dependencies.

WHAT NEEDS TO BE DONE:

1) Open *XAMPP* console and run both *Apache* and *MySQL*.
2) Import the provided **`worldcup`** database into the database using *phpMyAdmin*.

3) Deploy the provided *Slim* app on Apache Web server. Refer to the `README.md` file for the instructions and more details about the required path configurations that you need to adjust.
   - The subdirectory containing the code of your Web service must be named **`worldcup-api`**
4) Using Thunder Client, test your app deployment by initiating an HTTP request to the `/ping` resource.

PART II: IMPLEMENTATION

The REST web service you are implementing must expose some of the data stored in the provided World Cup database as web resources. Some of these resources must be exposed as collection resources, and therefore, the name of a collection resource must be plural, as instructed.

Note: You **must explore** the relationships between the database tables before you begin working on the implementation. These relationships can be easily visualized using using *phpMyAdmin*'s *Designer* feature.

GENERAL REQUIREMENTS

Your Web service must:

1) Handle errors gracefully. Errors returned to the client application must be encoded in JSON.
2) Validate data provided in query string parameters (GET requests) and URI template.
3) Support **pagination** and **filtering operations** on the exposed collection resources specified below.
4) Produce an appropriate response status code based on the type of request and any issues encountered while processing the request.
5) Provide a root resource (that is, /) whose content includes all the resources the WS exposes along with the full URI for each resource and their respective description all encoded in JSON.
6) Produce a valid HTTP response message structure. For each response, there must be an appropriate media type associated with the representation produced, along with a status code included in the response.
7) Encode resource representations exchanged between an HTTP client and your web service in JSON (including errors).
8) Implement a set of custom HTTP exception classes that extend the *`HttpSpecializedException`* class. Such classes must be used to handle server-side, request-handling-related errors and runtime exceptions.

REQUIREMENTS FOR DOCUMENTING AND ORGANIZING YOUR CODE

- Your code must be documented using PHPDoc.
- Your code must be organized within the pre-created folder (Controllers, Models, Exceptions, etc.)
- Create controller classes in which you will implement callback functions responsible for handling client requests. A controller class must be associated with exactly one main resource. However, a controller class may, and in some cases should, contain the definition of one or more callback functions.
- For each exposed ***collection*** resource:
  - Add a route to the list of routes managed by your Slim app
  - Create a model class for interacting with the database.

REQUIREMENTS FOR DATA RETRIEVAL OPERATIONS

The resources documented below must be exposed by your service. However, keep in mind that:

1) **Data must be pulled from the corresponding child tables.**
2) Sorting must be supported on at least two collection resources of your choice. A sorting parameter of your choice should be included in the request URI query string. By default, result sets should be sorted in ascending order.

#	Resource URI & Description	Filters to be supported
1)	`/players` Gets a list of zero or more players resources that match the request's filtering criteria	- First name - Last name - Date of birth, condition: greater than the specified date - Position: goal keeper, forward, defender, OR midfielder - Gender
2)	`/players/{player_id}` Gets the details of the specified player	N/A
3)	`/players/{player_id}/goals` Gets a list of zero or more goal resources scored by the specified player that match the request's filtering criteria.	- Tournament - Match
4)	`/players/{player_id}/appearances` Gets a list of the specified player's appearances.	N/A
5)	`/teams` Gets a list of zero or more teams matching the specified filter.	- Region
6)	`/teams/{team_id}/appearances` Gets a list of zero or more appearances of the specified team.	- Match result
7)	`/tournaments` Gets a list of zero or more World Cup tournaments resources.	- Start date, condition: between two dates - Winner - Host country - Tournament type: women/men
8)	`/tournaments/{tournament_id}/matches` Gets the list of matches that took place in the specified tournament and match the request filter.	- Stage
9)	`/matches/{match_id}/players` Gets the list of players who played in the specified match.	- Position: goal keeper, forward, defender, OR midfielder
10)	`/stadiums` Gets the list of stadiums where World Cup matches took place.	- Country - City - Capacity: greater than the specified number
11)	`/stadiums/{stadium_id}/matches` Gets List of matches that took place the specified stadium.	- Tournament - Stage name

PART III: TESTING YOUR WEB SERVICE

You should thoroughly test your implementation using a REST client. Your test cases must cover operations on all exposed resources, including filtering, paging, and sorting.

EVALUATION CRITERIA

		POINTS
Functionality	Compliance of the implementation with the stated requirements.	60
Correctness	No errors, the program works well and meets all the specifications. Ability to perform as required and produce correct output.	20
Code readability and good programming practices	Programming style, good naming convention, and clarity. The use of meaningful self-explanatory names for identifiers (classes, data members, variables, constants, methods, etc.).	10
Documentation	Description of purpose of program. Use of PHPDoc comment style for all methods and fields, comments on non-trivial steps in all methods.	10
	Total	**100**

WHAT TO SUBMIT

- Compress your `worldcup-api` folder and upload it to LÉA before the due date.