**Ref.: JO 135892, Associate Information Systems Officer, P2 7 October 2020**

# Written Assessment

You have 2h00 to complete the written assessment.

**DO NOT WRITE YOUR NAME ON YOUR TEST ANSWERS.**

Please return your test answers to:

Gloria Alexandre: e-mail : [gloria.alexandre@cbd.int](mailto:gloria.alexandre@cbd.int) and copy Gianina Del Carpio: e-mail: [gianina.delcarpio@cbd.int](mailto:gianina.delcarpio@cbd.int)

It is recommended to read the entire assessment entirely before starting to answer.

The assessment is divided into two (2) sections:

1. Technical questions
2. A mandate

IMPORTANT NOTE: You are free to use any available *non-human* resources such as books; internet, etc.; interactive assistance, such as assistance through discussion forums (new postings). Direct messaging is of course forbidden for the purpose of this exercise.

# 1. Technical questions

*Please respond to the questions below to demonstrate your understanding. Use your own words. Do not copy-paste answer(s) from the Internet. Keep your answers short and brief*.

[PLEASE ANSWER DIRECTLY INTO THE DESIGNED SPACES OF THIS WORD DOCUMENT]

|  |
| --- |
| Q.1: Can you describe what is *encodeURIComponent* and when to use it? |
| encodeURLComponent allow us to encodes a URI by replacing each characters by one, two or three escape sequence to represent UTF8 encoding the characters. |

|  |
| --- |
| Q.2: Can you describe what is *sanitization*, why it is important and when to use it? |
| Sanitization is a process of examining HTML inputs. Usually we sanitize the user input to protect against attack (cross-site scripting). We shouldn’t allow user to use input elements to add code (example: javascript codes) |

|  |
| --- |
| Q.3: Can you describe what is a *closure function* and when it is commonly used? |
| Closure is created when a child function to keep the value of the parent scope function. Even the parent function already executed. It is one of the interesting feature of JavaScript.  For example:  const continent = ()=>{  const cont = ‘Africa’  return function(){  const country=’Egypt’;  console.log(`Country ${country} is found in ${cont}`);  }  }  On the above example: the variable ‘cont’ accessible inside the returned anonymous function by closure. |

|  |
| --- |
| Q.4: Can you describe what is *asynchronous programming*? |
| Asynchronous programming allows us to not to block execution in single treaded language. For example, JavaScript is a single treaded language, but it is non-blocking. This non-blocking nature came from the asynchronous feature of the language.  Asynchronous allow multiple things to happen at the same time. For example in JavaScript when we execute an expression it goes to ‘call stack’, if the expression takes time to execute, it doesn’t block the other code, the JavaScript engine moves the expression to ‘call back queue’ and move to the next expression |

|  |
| --- |
| Q.5: Can you describe one or two *asynchronous programming patterns* commonly used in *JavaScript* language and what problem it solves? |
| Asynchrounous javascript pattern:   1. Callback: in JavaScript we can pass function as a parameter (high order function). Call back is a function that is passed as an argument to another function. The callback should be invoked whenever the asynchronous work is finished. Disadvantage: Call back hell. 2. Promise: returns a resolve or a reject results. It has at most, one resolution value. Advantage: somehow resolve call back hell, also has a better error handling. 3. Async.. wait: is the elegant way of handling async JavaScript. We create a function with async keyword, then we add await for the request. This let us to run async code as synchronous code |

|  |
| --- |
| Q.6: Can you describe what is a *relational database* and a *document database* (known as NoSQL database) and give an example / a description of when to use or not-use each of them? |
| Relational database used to represent the data in a table and the data stored in a table. Each table has it’s own columns( attributes) and rows (records). In order to pull data from RDBMS we use SQL language. Examples: MS-SQL, MySQL, Postgres, Oracle, SQL-lite.  Document based Database: stores the data in a collection (similar to table). Each record represented as documents.  We usually use RDMS in business applications, especially application developed in ASP.net, PHP, Java. Document based databased is used in modern web and mobile applications. Data represented in document based databases usually in REST API. |

|  |
| --- |
| Q.7: Can you describe what is an *Application Container*, what are the advantages and when to use them? |
| Application containers helps us to virtualize our application in a very small container with it’s own dependencies that helps us to use system resource efficiently. It is easy to deploy our application, OS independent. A code works on local machine can be shipped and deployed in any machine. We should use application container (like Docker) for application development. |

|  |
| --- |
| Q.8: Can you describe what is a *REST API* and how it serves web applications? |
| REST API helps us to communicate application with other applications or back end services over HTTP. In REST api we usually do HTTP request using GET, POST, PUT, PATCH, AND DELETE operations.  REST API serves usually in JSON or XML format. For example when we develop an application we have a frontend and a backend application. We develop a backend application to expose/consume data using REST api. The front end application uses the REST api generated by the backend applicaton |

|  |
| --- |
| Q.9: Can you describe what are *Continuous Integration* and *Continuous Deployment* (CI/CD) processes and what are the advantages of using them? |
| CI helps us to integrate merge changes to main branch as often as possible. The code change are validated by creating a build and running automated test against the build.  For instance: when you make a change a code   * Create a new branch in git * Git add, commit and push to the new branch * Then we do a ‘Pull Request’ and at that stage we can implement automated Integration using (example: TravisCI). * If the TravisCI test passes, merge the new branch * Then for the deployment * - we can do a pull request * - create a new release by labelling (tagging) the release * - publish release * - run Jenkins deploy job and build with parameter (with the release tag/label name) |

|  |
| --- |
| Q.10: Imagine that you have web applications used by clients all around the world. Can you suggest solution(s) to improve your application performance for the end user and ensure its high availability? |
| To increase the performance of the application we can use different technologies to enhance the user experience:   * Caching: we can do a client side or server side caching. * Use CDN: for media contents. For example if we store our media contents in Amazon S3 bucket, we can use Cloud Front Service. * Optimize the application. * Using Progressive Web App: By implementing Service worker in our web/mobile app. |

# 2. Mandate

Your mandate consists of the following:

1. Implement a basic Model-View client-side application (a web page / app);
2. Develop the technical specifications of a basic REST API;

[PLEASE ATTACH ONLY ONE ZIP FILE FOR THE MANDATE: INDIVUDUAL EXERCICES (A & B) MUST BE IN SEPRATE FOLDERS]

## A. Client-Side

Implement a very basic client-side application (a web page) listing the Parties and the Non-Parties to the Convention on Biological Diversity (CBD). The page should consist of a top section displaying the “Total Number of Parties” and the “Name of the Party” which last accessed the Convention, followed by a second section containing a table consisting of three columns, namely '#', 'Country' and 'Date'.

Also list of Non-Parties should consist of a top section displaying the 'Total Number of Non-Parties', followed by a second section containing a table consisting of two columns, namely '#', 'Country'.

All lists by be order by Country name.

*Example of the expected web page:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Parties to the Convention on Biological Diversity  **Number of Parties**: 196  **Last accession**: Andorra on 2015-05-05   |  |  |  | | --- | --- | --- | | **#** | **Country** | **Date** | | 1 | Afghanistan | 2002-12-18 | | 2 | Albania | 1994-04-05 | | 3 | Algeria | 1995-11-12 | | 4 | Andorra | 2015-05-05 | | … | …. | … | | 195 | Zambia | 1993-12-29 | | 196 | Zimbabwe | 1995-02-09 |   **Number of Non-Parties**: 2   |  |  | | --- | --- | | **#** | **Country** | | 1 | Holy See | | 2 | United States of America | |

**IMPORTANT**: For the purpose of the exercise, although very basic, the webpage must be developed as Model-View driven client-side application. That is, apart from static files, all your client application needs from the server is the data (from REST API specification you (will) develop in the section *B. Server-Side* below).

Your adherence to SOA design principles, Model-View design pattern and asynchronous programming patterns will be tested more than actual code. You will be judged on coding simplicity, readability and maintainability. You will not be judged on coding style (e.g. curly braces being on the same line, etc.) or aesthetics (i.e. no style, CSS class necessary).

The expected deliverables are:

* Markup files (HTML or equivalent)
* Script files (JavaScript or equivalent)
* Any other relevant information, quick drawings/sketches, etc.
* Do not include any auto-generated or network packages files into your deliverables

Your solution should be written in a web programming language (e.g.: JavaScript) and may depend on an existing web client framework. Your solution will be manually reviewed by developers. That is, the solution does not need to be syntactically correct (compile or run) and may even contain some pseudo code (if needed).

## B. Server-Side

*Based on requirement of section A above (Client-Side):*

Develop the technical specifications of a basic REST API that returns the list of all UN member States (countries) and their date of accession, when applicable, to the Convention on Biological Diversity (CBD).

**IMPORTANT**: For the purpose of the exercise, the REST API must return all UN member States including those that haven't accessed the Convention yet.

*This part of your mandate does not involve any coding. That is, it only involves writing the technical specifications and a sample data output in JSON format.*

The expected deliverable is a document describing:

* Technical specifications (should be very short as the REST API is very basic)
* Sample HTTP request
* Sample HTTP response

**Solution**:

To design the backend application

Technology: Node, Express, Mongoose and MongoDB database

If the application is hosted on AWS cloud infrastructure.

* I will create a Virtual Private Cloud
* 2 EC2 instances (for the frontend and backend app) – also I enable ‘Auto Scaling’ horizontally
* Database: Document based database – Dyanmo DB
* Since I don’t have static contents, I don’t use S3 service.
* All request should come from DNS (Route 53)
* Load banker: ELB(Elastic Load Balancer)

Application code:

* MVC style
  + Controller: fetch member states from the model and response a json
  + Model: A mongoose schema
  + A Db connection separate file holds connection string to mongodb
  + Routers
    - A general routing file which redirects to specific router
    - Member state router accepts a get request and calls a controller

* + Error handling middle ware for ‘Not Found’ and ‘other errors’
  + Features Not added:
    - We can add ‘caching’ to the get requests
    - We can also implement our own CDN for each API endpoints.

Note: I also added both the frontend and backend source codes with the email. You can also found it here <https://github.com/sanoylab/exam>