LEANTECH

Backend Test

SQuaRE

Test Overview

This test is designed to evaluate and determine the level of understanding and management of the different tools and skills of the developer. To achieve this, the test is divided into two phases, a theoretical/practical and a practical phase.

Phase one

In this phase, all software skills are tested, focusing on putting this knowledge into practice in the proposed challenges. The main topics to evaluate would be the ability to generate precise documentation, the implementation of a project scalable structure and good practices in language coding.

Please create a github account with your Lean Tech email, create a new **PRIVATE** repository named "lean_tech_backend_test" and upload your project here.

1 Title: Hello world
Tags: Version Control, Command Language, Programming language
Level: Basic
Description: 1. Create a project base in any of the following languages (Python, NodeJS, C#). 2. Add the resource http://localhost/holamundo/txt to the project, this url must return the following: "**Hello world**" I must indicate in the header that the content type is "**text/plain**". 3. Add the resource http://localhost/helloworld/json to the project, this url must return the following **{"message": "Hello world"}** I have to indicate in the header that the content type is "**application/json".** 4. In the **readme** file add the installation and execution guide of the project in markdown format. 5. Commit with the following message "**Hello world from backend**".
Specific Criteria to Evaluate: Programming language
Basic Time: 1h **Mid Time: 30m** **Senior Time: 15m**

2 Title: Data Structure NoSQL,
Tags: NoSQL
Level: Basic
Description: 1. From the data given in the onboarding board about transport logics, define a general document-based NoSQL schema to manage carrier, order, moves, shipments, BOLs, accessorials, etc. 2. Made a document called **nosqlstructure** with the schema**.** 3. Commit with the following message “**no sql structure**”
Link:
Specific Criteria to Evaluate: Data Structure
Basic Time: 4h **Mid Time: 2h** **Senior Time: 1h**

3 Title: Data Structure SQL,
Tags: SQL
Level: Basic
Description: 1. From the data given in the onboarding board about transport logics, define an general ERD scheme to manage carrier, order, moves, shipments, BOLs, accessorials, etc. 2. Based on the ERD schema, create the sql commands to create the database and its tables in a file called **sqlstructure.** 3. Commit with the following message “**sql structure**”
Link:
Specific Criteria to Evaluate: Data Structure, Normalización de Datos
Basic Time: 4h **Mid Time: 2h** **Senior Time: 1h**

4 Title: Create CRUD SQL

Tags:

Level: Basic

Description:

1. From the ERD structure defined above, create the CRUDs with an API interface of each of the entities.
2. The defined resources must be under the **/sql** link, example **/sql/entityname**, for each entity create the CRUD using the following methods: GET, POST, DELETE, PUT; I have to indicate in the header that the content type is **“application/ json”.**
3. The **/sql/index.html** resource should only allow the GET method and it should return an html guide with the list of available resources, methods, defined structures and usage guide for each of the entities. I have indicated in the header that the content type is **"text/html".**
4. For operations with your database manager of your preference (mysql, postgresql) do not use **ORM**, use the direct connection driver to the database.
5. Commit with the following message **“sql CRUD”.**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation

Basic Time: 2h
Mid Time: 1h
Senior Time: 30m

5 Title: Create CRUD NoSQL

Tags:

Level: Basic

Description:

1. From the NoSQL data structure defined above, create the CRUDs with an API interface of each of the entities.
2. The defined resources must be under the **/nosql** link, example **/nosql/entityname**, for each entity create the CRUD using the following methods: GET, POST, DELETE, PUT; I have to indicate in the header that the content type is **“application/json”**
3. The **/sql/index.html** resource should only allow the GET method and it should return an html guide with the list of available resources, methods, defined structures and usage guide for each of the entities. I have indicated in the header that the content type is **"text/html"**.
4. For operations with your database manager of your preference, do not use orm, use the direct connection driver to the database.
5. Commit with the following message **“sql CRUD”.ç**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation

Basic Time: 2h
Mid Time: 1h
Senior Time: 30m

6 Title: US Energy Information Administration

Tags:

Level: Basic

Description:

1. Generate a resource that connects through the API to the US Energy Information Administration (https://www.eia.gov/) and returns the current weekly price of Regular Conventional Retail gasoline.
2. In the **readme** file add the guide section to use of these resources.
3. Make a commit with the following message **“api eia”.**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance

Basic Time: 1h
Mid Time: 30m
Senior Time: 10m

7 Title: Demo

Tags:

Level: Mid

Description:

1. Based on the following data structure (Data Schema), define and implement the different resources for each of its entities with an API interface with the following GET, POST, PUT, DELETE methods, these resources must be defined under the following link **/demo**, I have to indicate in the header that the content type is **“application/json”**
2. In the shipment entity implement the following search system: the user in the parameter **q** can write several search criteria (example: fedex ltl broker), the system must remove the criteria that are common contextual connectors. These criteria must be searched in all fields and must return any type of match by criteria. Also, if the user prefers it, they can include the **date** parameter to indicate the date they want the records. Keep in mind that this filter can return hundreds of records.
3. In the **readme** file add the guide section use of these resources.
4. Make a commit with the following message **“Initial structure section demo”.**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance

Basic Time: 2h
Mid Time: 1h
Senior Time: 30m

8 Title: Authentication
Tags:
Level: Mid
Description: 1. Implement an authentication system with the following roles: **admin** will have access to all resources and **readonly** will only have access to resources with GET methods 2. Implement basic authentication and token or jwt to all resources using the **/demo** path. 3. In the **readme** file describe and argue for your authentication system as well as define the usage guide. 4. Make a commit with the following message **“Authentication”**
Link:
Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance
Basic Time: 2h **Mid Time: 1h** **Senior Time: 30m**

9 Title: Import Data
Tags:
Level: Senior
Description: 1. With the resources defined in the previous section (CRUD Demo) for the shipments entity, it is necessary to define and implement a data import system through an API interface, the system must support files in **csv** format separated by commas and **xls**; the system must support files of more than **20mb** and the user must know the status of the final import, indicating the imported registration number and non-imported registration number and why.

2. In the **readme** file describe and argue your import system as well as define the usage guide.
3. Commit with the following message “Import Data”.

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance

Basic Time: 4h
Mid Time: 2.5h
Senior Time: 1.5m

10 Title: Export to Google Drive

Tags:

Level: Senior

Description:

1. With the resources defined in the previous section (CRUD Demo) define and implement a user data export system through an API interface, the system must generate files (See File) in csv format separated by commas or xls the user must indicate what format they want, by default it is xls format; the files must be uploaded to google drive in a shared folder; all reports must be separated or classified by user box and the format of the names must be YYYYMMDDmm. [cvs, xls]. Once the report is generated, the user must be notified via email of the completion of the report with the link to view or download.
2. In the **readme** file describe and argue your export system as well as define the usage guide.
3. Commit with the following message **“Export Data”**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance

Basic Time: 4h
Mid Time: 2.5h
Senior Time: 1.5m

11 Title: Report

Tags:

Level: Senior

Description:

1. Define and implement the resource /demo/status through an API interface with the GET method to be able to return the status of the files generated in Google Drive, the report must be per logged in user, if the user has the administrator role and he decides to see all users, he must indicate it. The report must be able to be delivered in format PDF (Ver Archivo) or json at the request of the user by default must be in PDF format. The report consists of two sections: a statistics section that should show me the number of files generated per month, and week; and the other with the list of all the files generated with the generation date, completion date, generation status and link to view and download the file.
2. In the **readme** file describe and argue your reporting system as well as define the usage guide.
3. Commit with the following message **"Report"**

Link:

Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance

Basic Time: 16h
Mid Time: 8h
Senior Time: 5m

12 Title: Deployment

Tags:
Level: Senior
Description: 1. On a SAAS platform with a free layer. 2. In the **readme** file describe and argue your reporting system as well as define the usage guide. 3. Commit with the following message **“Deployment”.**
Link:
Specific Criteria to Evaluate: Programming language, Architecture, implementation, performance
Basic Time: 16h **Mid Time: 8h** **Senior Time: 5h**