Projeto Teste

PHP  
Tecnologias WEB

**Submetido por:**

Glauco de Sousa Rocha, DF

Gustavo Henrique da Silva, SP

Leonardo Victor Fernandes Ferreira, AL

**Conteúdo**

1. [**Introdução 3**](#_heading=h.3znysh7)
2. [**Glossário 3**](#_heading=h.2et92p0)
3. [**PHP – 3 horas 3**](#_heading=h.tyjcwt)
4. [**Descrição do projeto e tarefas 3**](#_heading=h.3dy6vkm)
   1. [Descrição do formato da entrega 4](#_heading=h.1t3h5sf)
   2. [Causas de incompatibilidade 4](#_heading=h.4d34og8)
5. [**Detalhes da API 5**](#_heading=h.2s8eyo1)
   1. [Autenticação e casos gerais 5](#_heading=h.17dp8vu)
6. [**Detalhes das Entidades 5**](#_heading=h.3rdcrjn)
7. [**API 6**](#_heading=h.26in1rg)
8. [**Ficha de Avaliação Simplificada 15**](#_heading=h.lnxbz9)

# Introduction

A computer sales and assembly company in the city of Maceió/AL is expanding its business to the rest of Brazil. To ensure that the machines sold to their customers have compatible parts, they need a Web Service that tests the parts for compatibility.

The service must be able to:

• register spare parts;

• assembly of entire machines with parts that are compatible with each other;

• if there is any incompatibility, it must be reported to the user.

The registration of templates of entire machines must be dynamic for the end user.

# Glossary

* Socket: is the connector, located on the motherboard, that defines the electrical interface for data transmission to and from the processor.
* TDP (Thermal Design Power): is the maximum heat generated by a computer chip or component (which can be a processor, video card, etc.) that the computer's cooling system is designed to dissipate without any overload.
* RAM (Random Access Memory): type of memory present in computers for fast and random processing. Typically used for active data during user work.
* SLI/Crossfire: Technologies used by graphics cards from Nvidia and AMD, respectively, to work together and increase the total graphics processing power of the computer.
* PCI Express (Peripheral Component Interconnect Express): is a high-speed serial connection standard present on motherboards.

# PHP – 3 hours

# Project description and tasks

Project execution takes 3 hours. At this time you should create an API using PHP (optionally with a framework) and a relational database.

You will receive .json files, with initial data, the database creation file and images of each one of the different types of entities that must be registered in your system to allow the execution of tests.

## Delivery format description

The following requirements must be met upon delivery of the test project:

• The database must be created with the name AlatechMachines and must use the utf8mb4 collation;

• Your API must be accessible from /XX/alatech/api (it is your responsibility to provide the host address and port. XX corresponds to the abbreviation of your delegation).

## Incompatibility Causes

Your API must report incompatibility when machines are created or edited in the following cases:

• Motherboard socket type is different from the processor socket type;

• Processor TDP is greater than the maximum TDP supported by the motherboard;

• Motherboard RAM memory type is different from RAM memory type;

• Amount of RAM memories is greater than the amount of slots present on the motherboard (there must always be at least 1 RAM memory on the machine);

• Number of video cards is greater than the number of PCI Express slots on the motherboard (there must always be at least 1 video card in the machine);

• Number of SATA storage devices is greater than the number of SATA slots on the motherboard;

• Number of M2 type storage devices is greater than the number of M2 slots on the motherboard;

• Total sum of storage devices equals zero (ie there must be at least 1 SATA device or 1 M2 device);

• Number of video cards is greater than 1 and the video card model does not support SLI/Crossfire;

• Power supply power is less than the minimum video card power times (multiplied) by the number of video cards.

# API Details

## Authentication and general cases

For security reasons, only the login endpoint and images should be public. When accessing any other address that does not match those mentioned above, the API must return the status code HTTP 401 Unauthorized if the Authorization header is not present in the request. If there is a header, but it is invalid, the HTTP 403 Forbidden status code must be returned.

User authentication in the API must be done through a token that will be sent with each request in the Authorization header. This token must be:

• encrypted;

• in the format: Bearer {token}; and

• secure to prevent user password extraction.

# Entity Details

After user authentication, he must have access to endpoints that will allow him to manipulate the following entities (each entity must have its endpoint) in order to mount machines (which will be a separate endpoint):

• Motherboards List

○ Must have name, image URL, brand, socket type, RAM memory type, number of RAM slots, maximum TDP, number of SATA slots, number of M2 slots, number of PCI Express slots.

• Processor Listing

○ Must have name, image URL, brand, socket type, number of physical cores, base frequency, maximum frequency, cache amount, TDP.

• RAM memory listing

○ Must have name, image URL, brand, amount of memory, type of RAM memory, frequency.

• Storage Device Listing

○ Must have name, image URL, brand, device type (HDD or SSD), amount of memory, input type (SATA or M2).

• Video Card Listing

○ Must have name, image URL, brand, amount of memory, memory type (GDDR5 or GDDR6), minimum power supply, support SLI/Crossfire.

• Listing of Power Supplies

○ Must have Name, Image URL, Brand, Power (in Watts), 80 Plus rating (White, Bronze, Silver, Gold, Platinum, Titanium or not applicable).

• Brand Listing

○ Must have name

• Management of Entire Machines

○ Must have name, image URL, motherboard, processor, memories (minimum 1 unit), storage devices (minimum 1 unit), video cards (minimum 1 unit), power supply.

All products must have a photo (the photos are provided on the media) and they must be accessible through the endpoint specified in the API section below.

# API

The API must always receive and return data in JSON format. It should have the following endpoints, which are relative to /alatech/api):

|  |  |
| --- | --- |
| **POST (login)**  **User authentication** | |
| Request | Response |
| **Content-type:** application/json **Body:**  {  username: string < the name of the user you want to authenticate >  *password*: string < the password of the user to be authenticated > } | **1. Se tudo OK: Status:** 200/OK  **Content-type**: application/json  **Body:**  {  token: string <*token* that authorizes the user to access system endpoints.> }  **2. If any data is incorrect:**  **Status:** 400/BAD REQUEST  **Content-type**: application/json  **Body:**  {  message: “Invalid credentials” }  **3. If there is already active authentication for the user**:  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “User already authenticated” } |

|  |  |
| --- | --- |
| **DELETE (*logout*)**  **Removing user authentication** | |
| Request | Response |
| **Authorization:** Bearerstring < User identification will be given by the Authorization header (just like the other endpoints)**>**  **Body: empty** | **1. Logout confirmed: Status:** 200/OK **Content-type**: application/json  **Body:**  {  message: “logout successfully” }  **2. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **3. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **GET (*motherboards, processors, ram-memories, storage-devices, graphic-cards, power-supplies, machines, brands*)**  **Listings of motherboards, processors, RAM memories, storage devices, video cards, power supplies and entire machines, respectively.** | |
| Request | Response |
| **header: Authorization** string < User identification will be given by the Authorization header (just like the other endpoints)**>**  **Parameters:**  **Query string parameters, both optional:**   * pageSize: int < determines the size of each search page > (default is 10) * page: int < determines the current search page > (default is 1) | **1. normal flow: Status:** 200/OK **Content-type**: application/json  **Body:**  [  {  < properties described in the subsection "Entity Details" for the referenced entity >  },  {  < properties described in the subsection "Entity Details" for the referenced entity >  },  ...  ]  **2. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **3. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **GET (search/{category}?q={q})**  **Product search** | |
| Request | Response |
| **Authorization:** Bearer string < Product search will be done by the Authorization header (just like the other endpoints)**>**  **Parameters:**  **required:**   * q: string < matches the search term that should be applied to the name of the searched items > * category: string < type of part to look for (must match the name of the part on the endpoint, eg **motherboards** or **processors**)>   **opcionais:**   * pageSize: int < determines the size of each search page> (default is 10) * page: int < determines the current search page> (default is 1) | **1Normal flow:**  **-** listed products  - The total of items from all groups must not exceed the pageSize value  **Status:** 200/OK **Content-type**: application/json  **Body:**  [  {  < properties described in the subsection "Entity Details" for the referenced entity >  },  {  < properties described in the subsection "Entity Details" for the referenced entity >  },  ...  ]  **2. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **3. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **POST (machines)**  **Create a new machine model.** | |
| Request | Response |
| **Authorization:** Bearer string < User identification will be given by the Authorization header (just like the other endpoints**>**  **Body:** a JSON object that must contain all of the following properties:  **JSON Properties:**  ● motherboardId: Motherboard ID  ● powerSupplyId: Power supply ID  ● processorId: processor ID  ● ramMemoryId: RAM memory ID  ● ramMemoryAmount: amount of RAM memories  ● storageDevices: array of JSON objects containing:  ○ storageDeviceId: storage device IDs  ○ amount: number of devices in this model  ● graphicCardId: video card ID  ● graphicCardAmount: number of video cards | **1. Successfully created machine:**  **Status:** 201/CREATED  **Content-type**: application/json  **Body:**  [  {  “id”: < created machine id >,  …  }  ]  **2. Image not provided, parts quantities were invalid or incompatibilities between them were found:**  **Status:** 400/BAD REQUEST  **Content-type**: application/json  **Body:**  {  “<property with error>”: “<description of error or incompatibility>”,  “<property with error>”: “<description of error or incompatibility>”,  …  }  **3. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **4. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **PUT (machines/{id})**  **Change an existing machine model.** | |
| Request | Response |
| **Authorization:** Bearer string < User identification will be given by the Authorization header (just like the other endpoints)**>**  **Body:**  {  <mesmo que POST (machine)>  }  If imageBase64 is not provided, the machine image (“imageUrl” property) must remain unchanged.  **Mandatory parameters:**   * id: Machine ID to be changed. | **1. Machine model updated successfully:**  **Status:** 200/OK  **Content-Type:** application/json  **Body:**  {  < machine properties described in the subsection "Entity Details">  }  **2. Quantities of parts were invalid or incompatibilities between them were found:**  **Status:** 400/BAD REQUEST  **Content-Type:** application/json  **Body:**  {  “<property with error>”: “<description of error or incompatibility>”,  “<property with error>”: “<description of error or incompatibility>”,  …  }  **3. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **4. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **DELETE (machines/{id})**  **Remove an existing machine model.** | |
| Request | Response |
| **Authorization:** Bearerstring < User identification will be given by the Authorization header (just like the other endpoints)**>**  **Mandatory parameters:**   * id: int <ID of machine to be deleted> | **1. Machine model successfully removed:**  **Status:** 204/NO CONTENT  **2. If the machine model does not exist:**  **Status:** 404/NOT FOUND  **Content-type**: application/json  **Body:**  {  message: “machine model not foundo” }  **3. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **4. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **POST (verify-compatibility)**  **Check the compatibility between two or more parts, not necessarily a machine, however there should always be a motherboard and a power supply.** | |
| Request | Response |
| **Authorization:** Bearerstring < User identification will be given by the Authorization header (just like the other endpoints)**>**  **Body:** a JSON object with some mandatory (marked \*) and some optional properties:  **JSON Properties:**  ● motherboardId\*: Motherboard ID  ● powerSupplyId\*: Power supply ID  ● processorId: processor ID  ● ramMemoryId: RAM memory ID  ● ramMemoryAmount: amount of RAM memories  ● storageDevices: array of JSON objects containing:  ○ storageDeviceId: storage device IDs  ○ amount: number of devices in this model  ● graphicCardId: video card ID  ● graphicCardAmount: number of video cards | **1. successful verification:**  **Status:** 200/NO CONTENT  **Body:**  {  message: “Valid machine”  }  **2. Incompatibilities found:**  **Status:** 400/BAD REQUEST  **Content-Type:** application/json  **Body:**  {  “<property with error>”: “<description of error or incompatibility>”,  “<property with error>”: “<description of error or incompatibility>”,  …  }  **3. If the token is not present in the request:**  **Status:** 401/UNAUTHORIZED  **Content-type**: application/json  **Body:**  {  message: “Authentication required” }  **4. If the token does not match any valid authentication:**  **Status:** 403/FORBIDDEN  **Content-type**: application/json  **Body:**  {  message: “Invalid Token” } |

|  |  |
| --- | --- |
| **GET (images/{id})**  **Returns the image with the data “id” (which is the value returned in “imageUrl” in the API listings)** | |
| Request | Response |
| **Parameters:**  **Required:**   * id: int < searched image id > | **1. normal flow:**  **Status:** 200/OK **Content-type**: image/string < string being the value corresponding to the image type >  **Body:** the image itself  **2. If the image does not exist:**  **Status:** 404/NOT FOUND  **Content-type**: application/json  **Body:**  {  message: “image not found” } |

**Simplified Evaluation Sheet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CRITERION | SECTION | JUDGE POINTS | MEASUREMENT POINTS | TOTAL |
| C1 | server side development | 0,00 | 10,00 | 10,00 |
| Total | | 0,00 | 10,00 | **10,00** |