# Anexo EndPoints:

|  |  |
| --- | --- |
| **POST (login)**  **User authentication** | |
| Request | Response |
| Content-type: application/json  Body:  {  username: string <the name of the user to be authenticated>  password: string <the password of the user to be authenticated>  } | 1. If everything is 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: Bearer string <user identification will be given by the Authorization header (as well as other endpoints)>  Body: empty | 1. Logout confirmed:  Status: 200/OK  Content-type: application/json  Body:  {  message: "Logout successful"  }  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)**  **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 header Authorization (as well as other endpoints)>  Parameters:  Query string parameters, both optional:  ● pageSize: int <determines the size of each search page> (default is 10)  ● page: int <determines 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 <user identification will be given by the Authorization header (as well as other endpoints)>  Parameters:  required:  ● q: string <corresponds to the search term that should be applied to the name of the searched items>  ● category: string <type of part to be fetched (must match the name of the part on the endpoint, for example motherboards or processors)>  optional:  ● pageSize: int <determines the size of each search page> (default is 10)  ● page: int <determines current search page> (default is 1) | 1. Normal 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 (as well as 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  ● memoryId: RAM memory ID  ● memoryAmount: amount of RAM memories  ● storageDevices: array of JSON objects that contains:  ○ 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”: <ID of the created machine>,  …  }  ]  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 (as well as other endpoints)>  Body:  {  <same as 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 successfully updated:  Status: 200/OK  Content-Type: application/json  Body:  {  <machine properties described in the subsection “Entity Details”>  }  2. The 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: Bearer string <user identification will be given by the Authorization header (as well as 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 found"  }  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: Bearer string <user identification will be given by the Authorization header (as well as other endpoints)>  Body: a JSON object with some mandatory (marked \*) and optional properties:  JSON Properties:  ● motherboardId\*: Motherboard ID  ● powerSupplyId\*: Power supply ID  ● processorId: processor ID  ● memoryId: RAM memory ID  ● memoryAmount: amount of RAM memories  ● storageDevices: array of JSON objects that contains:  ○ 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 given “id” (which is the value returned in “imageUrl” in the API listings** | |
| Request | Response |
| Authorization: Bearer string <user identification will be given by the Authorization header (as well as other endpoints)>  Parameters:  Required:  ● id: int <ID of the searched image> | 1. Normal flow:  Status: 200/OK  Content-type: image/string <value corresponding to 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"  } |