Tests unitaires réalisés avec Postman

[USE-CO1] Create one user without email

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without email", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-CO2] Create one user without firstname

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without firstname", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-CO3] Create one user without lastname

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without lastname", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-CO4] Create one user without phonenumber

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without phone number", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-CO5] Create one user without address

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without address", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-CO6] Create one user with invalid email format

```
pm.test("Right code for invalid email format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for for invalid email format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format d'adresse email invalide.");
});
```

[USE-CO7] Create one user with email address already exists

```
pm.test("Right code for email address already exists", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for for email address already exists", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Un compte avec cette adresse e-mail
existe déjà.");
});
```

[USE-CO8] Create one user without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "api_token" : {"type" : "string"},
              "code_role" : {"type" : "integer"}
          },
          "required": ["api_token", "code_role"]
  })
});
```

[USE-GA1] Get all users with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[USE-GA2] Get right users with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});

pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "email" : {"type" : "string"},
              "firstname" : {"type" : "string"},
              "lastname" : {"type" : "string"},
              "phonenumber" : {"type" : "string"},
              "address" : {"type" : "string"},
              "api_token" : {"type" : "null"},
              "code_role" : {"type" : "null"},
              "password_hash" : {"type" : "null"},
             "dogs" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "name" : {"type" : "string"},
                      "breed" : {"type" : "string"},
                      "sex" : {"type" : "string"},
                      "picture_serial_id" : {"type" : ["string","null"]},
                      "chip_id" : {"type" : ["string","null"]},
                      "user_id" : {"type" : "integer"},
                  },
                  "required":
["id","name","breed","sex","picture_serial_id","chip_id","user_id"]
              }
```

```
            },
            "required":
["id","email","firstname","lastname","phonenumber","address","api_token","code_rol
e","password_hash","dogs"]
        }]
    })
});
```

[USEE-GA1] Get right educator users

```
pm.test("The data structure of the response is correct", () => {
    pm.response.to.have.jsonSchema({
        "type": "array",
        "items": [{
            "type": "object",
            "properties": {
                "id" : {"type" : "integer"},
                "email" : {"type" : "null"},
                "firstname" : {"type" : "string"},
                "lastname" : {"type" : "string"},
                "phonenumber" : {"type" : "null"},
                "address" : {"type" : "null"},
                "api_token" : {"type" : "null"},
                "code_role" : {"type" : "null"},
                "password_hash" : {"type" : "null"}
            },
            "required":
["id","email","firstname","lastname","phonenumber","address","api_token","code_rol
e","password_hash"]
        }]
    })
});
```

[USE-GO1] Get one user with a user api token

```
pm.test("Authorization header is present", () => {
    pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[USE-GO2] Get one non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[USE-GO3] Get right user wtih admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "email" : {"type" : "string"},
              "firstname" : {"type" : "string"},
              "lastname" : {"type" : "string"},
              "phonenumber" : {"type" : "string"},
              "address" : {"type" : "string"},
              "api_token" : {"type" : "string"},
              "code_role" : {"type" : "integer"},
              "password_hash" : {"type" : ["string","null"]},
             "dogs" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "name" : {"type" : "string"},
                      "breed" : {"type" : "string"},
                      "sex" : {"type" : "string"},
                      "picture_serial_id" : {"type" : ["string","null"]},
                      "chip_id" : {"type" : ["string","null"]},
                      "user_id" : {"type" : "integer"},
                  },
                  "required":
["id","name","breed","sex","picture_serial_id","chip_id","user_id"]
              },
              "documents" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "document_serial_id" : {"type" : "string"},
                      "type" : {"type" : "string"},
                      "user_id" : {"type" : "string"},
```

```
                    "package_number" : {"type" : "null"},
                    "signature_base64" : {"type" : "null"},
                },
                "required": ["id","document_serial_id","type","user_id"]
            }
        },
        "required":
["id","email","firstname","lastname","phonenumber","address","api_token","code_rol
e","password_hash","dogs","documents"]
  })
});
```

[USE-UO1] Update one user with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[USE-UO2] Update one non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[USE-UO3] Update one user with invalid email format

```
pm.test("Right code for invalid email format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for for invalid email format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format d'adresse email invalide.");
});
```

[USE-UO4] Update one user without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[USE-DO1] Delete one user with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[USE-DO2] Delete one non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[USE-DO3] Delete one user without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[USE-GUA1] Get all user information of the current logged in non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[USE-GUA2] Get all user information of the current logged in user

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});

pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "email" : {"type" : "string"},
              "firstname" : {"type" : "string"},
              "lastname" : {"type" : "string"},
              "phonenumber" : {"type" : "string"},
              "address" : {"type" : "string"},
              "api_token" : {"type" : "string"},
              "code_role" : {"type" : "integer"},
              "password_hash" : {"type" : ["string","null"]},
              "dogs" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "name" : {"type" : "string"},
                      "breed" : {"type" : "string"},
                      "sex" : {"type" : "string"},
                      "picture_serial_id" : {"type" : ["string","null"]},
                      "chip_id" : {"type" : ["string","null"]},
                      "user_id" : {"type" : "integer"},
                  },
                  "required":
["id","name","breed","sex","picture_serial_id","chip_id","user_id"]
              },
              "documents" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "document_serial_id" : {"type" : "string"},
                      "type" : {"type" : "string"},
                      "user_id" : {"type" : "string"},
                      "package_number" : {"type" : "null"},
                      "signature_base64" : {"type" : "null"},
                  },
                  "required": ["id","document_serial_id","type","user_id"]
              },
              "appointments" : {
                  "type" : "array",
                  "properties" : {
                      "id" : {"type" : "integer"},
                      "datetime_appointment" : {"type" : "string"},
                      "duration_in_hour" : {"type" : "string"},
                      "note_text" : {"type" : "null"},
```

```
                        "note_graphical_serial_id" : {"type" : ["null","string"]},
                        "summary" : {"type" : ["string","null"]},
                        "user_id_customer" : {"type" : "integer"},
                        "user_id_educator" : {"type" : "integer"}
                    },
                    "required":
["id","datetime_appointment","duration_in_hour","note_text","note_graphical_serial
_id","summary","user_id_customer","user_id_educator"]
                }
            },
            "required":
["id","email","firstname","lastname","phonenumber","address","api_token","code_rol
e","password_hash","dogs","documents","appointments"]
  })
});
```

[USE-UPAU1] Update password of the current logged in non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[USE-UPAU2] Update password of the current logged in user without problems

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[USE-C1] Connect user without email

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without email", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[USE-C2] Connect user with an unrecognized email

```
pm.test("Right code for invalid login", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid login", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Identifiants de connexion invalides.");
});
```

[USE-C3] Connect user with wrong password

```
pm.test("Right code for invalid login", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid login", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Identifiants de connexion invalides.");
});
```

[USE-C4] Successful connection

```
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
            "api_token" : {"type" : "string"},
            "user_id" : {"type" : "integer"},
            "code_role" : {"type" : "string"}
          },
          "required": ["api_token","user_id","code_role"]
  })
});
```

[DOG-CO1] Create one dog with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-CO2] Create one dog without name

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without name", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-CO3] Create one dog without breed

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without breed", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-CO4] Create one dog without sex

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without sex", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-CO5] Create one dog without user_id

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without user_id", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-CO6] Create one dog for non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
```

```
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-CO7] Create one dog without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[DOG-GA1] Get all dogs with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-GA2] Get right dogs with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "name" : {"type" : "string"},
              "breed" : {"type" : "string"},
              "sex" : {"type" : "string"},
              "picture_serial_id" : {"type" : ["string","null"]},
              "chip_id" : {"type" : "string"},
              "user_id" : {"type" : "integer"}
          },
          "required":
```

```
["id","name","breed","sex","picture_serial_id","chip_id","user_id"]
      }]
  })
});
```

[DOG-GO1] Get one dog with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-GO2] Get one non-existent dog

```
pm.test("Dog not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-GO3] Get right dog with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "name" : {"type" : "string"},
              "breed" : {"type" : "string"},
              "sex" : {"type" : "string"},
              "picture_serial_id" : {"type" : ["string","null"]},
              "chip_id" : {"type" : ["string","null"]},
              "user_id" : {"type" : "integer"}
```

```
            },
            "required":
["id","name","breed","sex","picture_serial_id","chip_id","user_id"]
  })
});
```

[DOG-UO1] Update one dog with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-UO2] Update one non-existent dog

```
pm.test("Dog not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-UO3] Update one dog without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[DOG-DO1] Delete one dog with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
```

```
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-DO2] Delete one non-existent dog

```
pm.test("Dog not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-DO3] Delete one dog without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[DOG-UDP1] Upload dog picture with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOG-UDP2] Upload dog picture without dog_picture

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without dog_picture", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-UDP3] Upload dog picture without dog_id

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without dog_id", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOG-UDP4] Upload dog picture for non-existant dog

```
pm.test("Dog not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-UDP5] Upload dog picture with invalid image type format

```
pm.test("Right code for invalid image file format", function () {
    pm.response.to.have.status(415);
});
pm.test("Right message for invalid image file format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format d'image par pris en charge.");
});
```

[DOG-UDP6] Upload dog picture without problems

```
pm.test("Right code for successful uploaded ressource", function () {
    pm.response.to.have.status(200);
});
```

[DOG-DDP1] Download non-existant dog picture

```
pm.test("Dog not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOG-DDP2] Download dog picture without problems

```
pm.test("The server returns some information", () => {
    pm.expect(pm.response.body);
});
```

[DOC-CO1] Create one document with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOC-CO2] Create one document without type

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without type", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOC-CO3] Create one document without user_id

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without user_id", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOC-CO4] Create one document with invalid document type format

```
pm.test("Right code for invalid document type format", function () {
    pm.response.to.have.status(415);
```

```
});
pm.test("Right message for invalid document type format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Type de document invalide =>
(conditions_inscription,poster).");
});
```

[DOC-CO5] Create one document for non-existent user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOC-CO6] Create one conditions of registration with invalid paramater

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without type", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[DOC-CO7] Create one conditions of registration with invalid package number

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Numéro de Forfait invalide => (1 jusqu'à
5).");
});
```

[DOC-CO8] Create one conditions of registration without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[DOC-CO9] Create one PDF with invalid document format

```
pm.test("Right code for invalid document type format", function () {
    pm.response.to.have.status(415);
});
pm.test("Right message for invalid document type format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de document pas pris en charge =>
Format disponible PDF.");
});
```

[DOC-CO10] Create one poster without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[DOC-GA1] Get all documents with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOC-GA2] Get right documents with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "document_serial_id" : {"type" : "string"},
              "type" : {"type" : "string"},
```

```
                "user_id" : {"type" : "integer"}
            },
            "required": ["id","document_serial_id","type","user_id"]
        }]
    })
});
```

[DOC-GO1] Get one document with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOC-GO2] Get one non-existent document

```
pm.test("Document not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOC-GO3] Get right document with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
            "type": "object",
            "properties": {
                "id" : {"type" : "integer"},
                "document_serial_id" : {"type" : "string"},
                "type" : {"type" : "string"},
                "user_id" : {"type" : "integer"}
            },
```

```
            "required": ["id","document_serial_id","type","user_id"]
    })
});
```

[DOC-UO1] Update one document with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOC-UO2] Update one non-existent document

```
pm.test("Document not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOC-CO3] Update one document with invalid document type format

```
pm.test("Right code for invalid document type format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid document type format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Type de document invalide =>
(conditions_inscription,poster).");
});
```

[DOC-UO4] Update one document without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[DOC-DO1] Delete one document with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[DOC-DO2] Delete one non-existent document

```
pm.test("Document not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[DOC-DO3] Delete one document without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[ABS-CO1] Create one absence with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[ABS-CO2] Create one absence without date_from

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without date_from", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[ABS-CO3] Create one absence without date_to

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without date_to", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[ABS-CO4] Create one absence with invalid date_from format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_from format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_from format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[ABS-CO5] Create one absence with invalid date_to format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_to format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_to format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[ABS-CO6] Create one absence with chronological date problem

```
pm.test("Right code for chronological problem", function () {
    pm.response.to.have.status(400);
});
```

```
pm.test("Right message for chronological problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("La date ou l'heure de début est plus
récente que la date ou l'heure de fin.");
});
```

[ABS-CO7] Create one absence without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[ABS-GA1] Get all absences with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});

pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[ABS-GA2] Get right absences with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_absence_from" : {"type" : "string"},
              "date_absence_to" : {"type" : "string"},
              "description" : {"type" : ["string","null"]},
              "id_educator" : {"type" : "integer"}
          },
          "required":
["id","date_absence_from","date_absence_to","description","id_educator"]
```

```
        }]
    })
});
```

[ABS-GO1] Get one absence with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[ABS-GO2] Get one non-existent absence

```
pm.test("Absence not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[ABS-GO3] Get right absence with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_absence_from" : {"type" : "string"},
              "date_absence_to" : {"type" : "string"},
              "description" : {"type" : ["string","null"]},
              "id_educator" : {"type" : "integer"}
          },
          "required":
["id","date_absence_from","date_absence_to","description","id_educator"]
```

```
  })
});
```

[ABS-UO1] Update one absence with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[ABS-UO2] Update one non-existent absence

```
pm.test("Absence not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[ABS-UO3] Update one absence with invalid date_from format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_from format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_from format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[ABS-UO4] Update one absence with invalid date_to format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_to format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_to format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
```

```
DD).");
});
```

[ABS-UO5] Update one absence with chronological date problem

```
pm.test("Right code for chronological problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for chronological problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("La date ou l'heure de début est plus
récente que la date ou l'heure de fin.");
});
```

[ABS-UO6] Update one absence without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[ABS-DO1] Delete one absence with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[ABS-DO2] Delete one non-existent absence

```
pm.test("Absence not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[ABS-DO3] Delete one absence without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[WEE-CO1] Create one weekly schedule with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[WEE-CO2] Create one weekly schedule without date_from

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without date_from", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[WEE-CO3] Create one weekly schedule with invalid date_from format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_from format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_from format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[WEE-CO4] Create one weekly schedule with invalid date_to format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date_to format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date_to format", function () {
```

```
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[WEE-CO5] Create one weekly schedule with chronological date problem

```
pm.test("Right code for chronological problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for chronological problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("La date ou l'heure de début est plus
récente que la date ou l'heure de fin.");
});
```

[WEE-CO6] Create one weekly schedule permanant when one already exists

```
pm.test("Right code for weekly schedule already exists problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for weekly schedule already exists problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Un calendrier permanent a déjà été
créé.");
});
```

[WEE-CO7] Create one weekly schedule with overlap problem

```
pm.test("Right code for overlap problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for overlap problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Les dates chevauchent d'autres dates
déjà existantes.");
});
```

[WEE-CO8] Create one weekly schedule without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[WEE-GA1] Get all weekly schedules with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
})
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[WEE-GA2] Get right weekly schedules with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_valid_from" : {"type" : "string"},
              "date_valid_to" : {"type" : ["string","null"]},
              "id_educator" : {"type" : "integer"},
              "timeSlots": {
                  "type": "array",
                  "properties": {
                  "id" : {"type" : "integer"},
                  "code_day" : {"type" : "integer"},
                  "time_start" : {"type" : "string"},
                  "time_end" : {"type" : "string"},
                  "id_weekly_schedule" : {"type" : ["integer","null"]},
                  "id_schedule_override" : {"type" : ["integer","null"]},
                  "id_educator" : {"type" : "integer"},
                },
                "required":
["id","code_day","time_start","time_end","id_weekly_schedule","id_schedule_overrid
e","id_educator"]
              }
          },
          "required":
["id","date_valid_from","date_valid_to","id_educator","timeSlots"]
      }]
  })
});
```

[WEE-GO1] Get one weekly schedule with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[WEE-GO2] Get one non-existent weekly schedule

```
pm.test("Weekly schedule not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[WEE-GO3] Get right weekly schedule with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_valid_from" : {"type" : "string"},
              "date_valid_to" : {"type" : ["string","null"]},
              "id_educator" : {"type" : "integer"}
          },
          "required": ["id","date_valid_from","date_valid_to","id_educator"]
  })
});
```

[WEE-DO1] Delete one weekly schedule with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[WEE-DO2] Delete one non-existent weekly schedule

```
pm.test("Weekly schedule not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[WEE-DO3] Delete one weekly schedule without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[SCH-CO1] Create one schedule override with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});

pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[SCH-CO2] Create one schedule override without date

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without date", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[SCH-CO3] Create one schedule override with invalid date format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid date format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid date format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-
DD).");
});
```

[SCH-CO4] Create one schedule override with overlap problem

```
pm.test("Right code for overlap problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for overlap problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Les dates chevauchent d'autres dates
déjà existantes.");
});
```

[SCH-CO5] Create one schedule override without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[SCH-GA1] Get all schedule overrides with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
```

```
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[SCH-GA2] Get right schedule overrides with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_schedule_override" : {"type" : "string"},
              "id_educator" : {"type" : "integer"},
              "timeSlots": {
                  "type": "array",
                  "properties": {
                  "id" : {"type" : "integer"},
                  "code_day" : {"type" : "integer"},
                  "time_start" : {"type" : "string"},
                  "time_end" : {"type" : "string"},
                  "id_weekly_schedule" : {"type" : ["integer","null"]},
                  "id_schedule_override" : {"type" : ["integer","null"]},
                  "id_educator" : {"type" : "integer"},
                },
                "required":
["id","code_day","time_start","time_end","id_weekly_schedule","id_schedule_overrid
e","id_educator"]
              }
          },
          "required": ["id","date_schedule_override","id_educator","timeSlots"]
      }]
  })
});
```

[SCH-GO1] Get one schedule override with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
```

```
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[SCH-GO2] Get one non-existent schedule override

```
pm.test("Weekly schedule not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[SCH-GO3] Get right schedule override with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "date_schedule_override" : {"type" : "string"},
              "id_educator" : {"type" : "integer"}
          },
          "required": ["id","date_schedule_override","id_educator"]
  })
});
```

[SCH-DO1] Delete one schedule override with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[SCH-DO2] Delete one non-existent schedule override

```
pm.test("Weekly schedule not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[SCH-DO3] Delete one schedule override without problems

```
pm.test("Right code for successful deleted ressource", function () {
    pm.response.to.have.status(200);
});
```

[TIM-CO1] Create one time slot with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[TIM-CO2] Create one time slot without code day

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without code day", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO3] Create one time slot without time start

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without time start", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO4] Create one time slot without time end

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without time end", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO5] Create one time slot without the id weekly schedule and the id schedule override

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without the id weekly schedule and the id
schedule override", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO6] Create one time slot with the id weekly schedule and the id schedule override

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request with the id weekly schedule and the id schedule
override", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO7] Create one time slot for non-existent weekly schedule

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
```

```
pm.test("Right message for non-existent weekly schedule", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO8] Create one time slot for non-existent schedule override

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for non-existent schedule override", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[TIM-CO9] Create one time slot with invalid code_day format

```
pm.test("Right code for invalid code day format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid code day format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de jour invalide => (1 jusqu'à 7,
dimanche = 1).");
});
```

[TIM-CO10] Create one time slot with invalid time_start format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid time_start format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid time_start format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de temps invalide =>
(HH:MM:SS).");
});
```

[TIM-CO11] Create one time slot with invalid time_end format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid time_end format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid time_end format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de temps invalide =>
```

```
(HH:MM:SS).");
});
```

[TIM-CO12] Create one time slot with chronological time problem

```
pm.test("Right code for chronological problem", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for chronological problem", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("La date ou l'heure de début est plus
récente que la date ou l'heure de fin.");
});
```

[TIM-CO13] Create one time slot with time slot overlap in the same weekly schedule problem

```
pm.test("Right code for overlap in the same weekly schedule problem", function ()
{
    pm.response.to.have.status(400);
});
pm.test("Right message for overlap in the same weekly schedule problem", function
() {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Les horaires chevauchent d'autres
horaires déjà existants.");
});
```

[TIM-CO14] Create one time slot with time slot overlap in the same schedule override problem

```
pm.test("Right code for overlap in the same schedule override problem", function
() {
    pm.response.to.have.status(400);
});
pm.test("Right message for overlap in the same schedule override problem",
function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Les horaires chevauchent d'autres
horaires déjà existants.");
});
```

[TIM-CO15] Create one time slot without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[TIM-GA1] Get all time slots with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[TIM-GA2] Get right time slots with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "code_day" : {"type" : "integer"},
              "time_start" : {"type" : "string"},
              "time_end" : {"type" : "string"},
              "id_weekly_schedule" : {"type" : ["integer","null"]},
              "id_schedule_override" : {"type" : ["integer","null"]},
              "id_educator" : {"type" : "integer"}
          },
          "required":
["id","code_day","time_start","time_end","id_weekly_schedule","id_schedule_overrid
e","id_educator"]
      }]
  })
});
```

[TIM-GO1] Get one time slot with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
```

```
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[TIM-GO2] Get one non-existent time slot

```
pm.test("Time slot not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[TIM-GO3] Get right time slot with admin api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "code_day" : {"type" : "integer"},
              "time_start" : {"type" : "string"},
              "time_end" : {"type" : "string"},
              "id_weekly_schedule" : {"type" : ["integer","null"]},
              "id_schedule_override" : {"type" : ["integer","null"]},
              "id_educator" : {"type" : "integer"}
          },
          "required":
["id","code_day","time_start","time_end","id_weekly_schedule","id_schedule_overrid
e","id_educator"]
  })
});
```

[TIM-DO1] Delete one time slot with a user api token

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
```

```
});
```

[TIM-DO2] Delete one non-existent time slot

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[TIM-DO3] Delete one time slot without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[APP-CO1] Create one appointment with unauthorized user

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[APP-CO2] Create one appointment without datetime_appointment

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without datetime_appointment", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-CO3] Create one appointment without duration_in_hour

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without duration_in_hour", function () {
    const responseJson = pm.response.json();
```

```
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-C04] Create one appointment without user_id_customer

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without user_id_customer", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-C05] Create one appointment without user_id_educator

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without user_id_educator", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-C06] Create one appointment for non-existent customer user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[APP-C07] Create one appointment for non-existent educator user

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[APP-CO8] Create one appointment with invalid datetime_appointment format (dateAndTimeTestData.csv)

```
pm.test("Right code for invalid datetime_appointment format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid datetime_appointment format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format de date invalide => (YYYY-MM-DD
HH:MM:SS).");
});
```

[APP-CO9] Create one appointment for invalid appointment slot

```
pm.test("Right code for invalid datetime_appointment format", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for invalid datetime_appointment format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Impossible de prendre ce rendez-vous.");
});
```

[APP-CO10] Create one appointment without problems

```
pm.test("Right code for successful created ressource", function () {
    pm.response.to.have.status(201);
});
```

[APP-GA1] Get all appointments with unauthorized user

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[APP-GA2] Get right appointments for customer or educator

```

```

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "datetime_appointment" : {"type" : "string"},
              "duration_in_hour" : {"type" : "integer"},
              "note_text" : {"type" : "null"},
              "note_graphical_serial_id" : {"type" : "null"},
              "summary" : {"type" : ["string","null"]},
              "user_id_customer" : {"type" : "integer"},
              "user_id_educator" : {"type" : "integer"}
          },
          "required":
["id","datetime_appointment","duration_in_hour","note_text","note_graphical_serial
_id","summary","user_id_customer","user_id_educator"]
      }]
  })
});
```

[APP-GA3] Get right appointments for educator

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is right", function () {
    pm.response.to.have.status(200);
});
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "id" : {"type" : "integer"},
              "datetime_appointment" : {"type" : "string"},
              "duration_in_hour" : {"type" : "integer"},
              "note_text" : {"type" : ["string","null"]},
              "note_graphical_serial_id" : {"type" : ["string","null"]},
              "summary" : {"type" : ["string","null"]},
              "user_id_customer" : {"type" : "integer"},
              "user_id_educator" : {"type" : "integer"}
          },
```

```
          "required":
["id","datetime_appointment","duration_in_hour","note_text","note_graphical_serial
_id","summary","user_id_customer","user_id_educator"]
      }]
  })
});
```

[APP-UO1] Update one appointment with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[APP-UO2] Update one non-existent appointment

```
pm.test("Appointment not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[APP-UO3] Update one appointment without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[APP-DO1] Delete one appointment with unauthorized user

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
```

```
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[APP-DO2] Delete one non-existent appointment

```
pm.test("Appointment not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[APP-DO3] Delete one appointment with unauthorized customer or educator

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
});
```

[APP-DO4] Delete one appointment without problems

```
pm.test("Right code for successful updated ressource", function () {
    pm.response.to.have.status(200);
});
```

[APP-UNG1] Upload note graphical with a user api token

```
pm.test("Authorization header is present", () => {
  pm.request.to.have.header("Authorization");
});
pm.test("Authorization header is false", function () {
    pm.response.to.have.status(403);
});
pm.test("Right message for access without permission", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Vous n'avez pas les permissions.");
```

```
});
```

[APP-UNG2] Upload note graphical without note_graphical

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without dog_picture", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-UNG3] Upload note graphical without appointment_id

```
pm.test("Right code for invalid attributes", function () {
    pm.response.to.have.status(400);
});
pm.test("Right message for request without dog_id", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Attributs invalides.");
});
```

[APP-UNG4] Upload note graphical for non-existant appointment

```
pm.test("Appointment not found", function () {
    pm.response.to.have.status(404);
});

pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[APP-UNG5] Upload note graphical with invalid image format

```
pm.test("Right code for invalid image file format", function () {
    pm.response.to.have.status(415);
});
pm.test("Right message for invalid image file format", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Format d'image par pris en charge.");
});
```

[APP-DNG1] Download note graphical with a user api token

```
pm.test("Right code for successful uploaded ressource", function () {
    pm.response.to.have.status(200);
});
```

[APP-DNG2] Download non-existant note graphical

```
pm.test("Right code for successful uploaded ressource", function () {
    pm.response.to.have.status(200);
});
```

[APP-DNG3] Download note graphical without problems

```
pm.test("Right code for successful uploaded ressource", function () {
    pm.response.to.have.status(200);
});
```

[PLA-GO1] Get one planning for non-existent educator

```
pm.test("User not found", function () {
    pm.response.to.have.status(404);
});
pm.test("Right message for not found response", function () {
    const responseJson = pm.response.json();
    pm.expect(responseJson.error).to.eql("Le serveur n'a pas trouvé la ressource
demandée.");
});
```

[PLA-GO2] Get right planning

```
pm.test("The data structure of the response is correct", () => {
  pm.response.to.have.jsonSchema({
      "type": "array",
      "items": [{
          "type": "object",
          "properties": {
              "date" : {"type" : "string"},
              "time_start" : {"type" : "string"},
              "time_end" : {"type" : "string"}
          },
          "required": ["time_start","time_end","date"]
      }]
  })
});
```

Contenu du fichier dateAndTimeTestData.csv

date	time
12345	12345
texte	texte
122-02-2021	011:30:50
12-024-2021	11:302:50
12-02-20212	11:30:506
33-02-2021	25:02:23
12-13-2021	11:61:23
12.02.2021	11:45:68
12.02.2021	11-30-50
N12.02.2021	11.30.50
12.02.2021N	N11:61:23
12/07/2000	11:61:23N