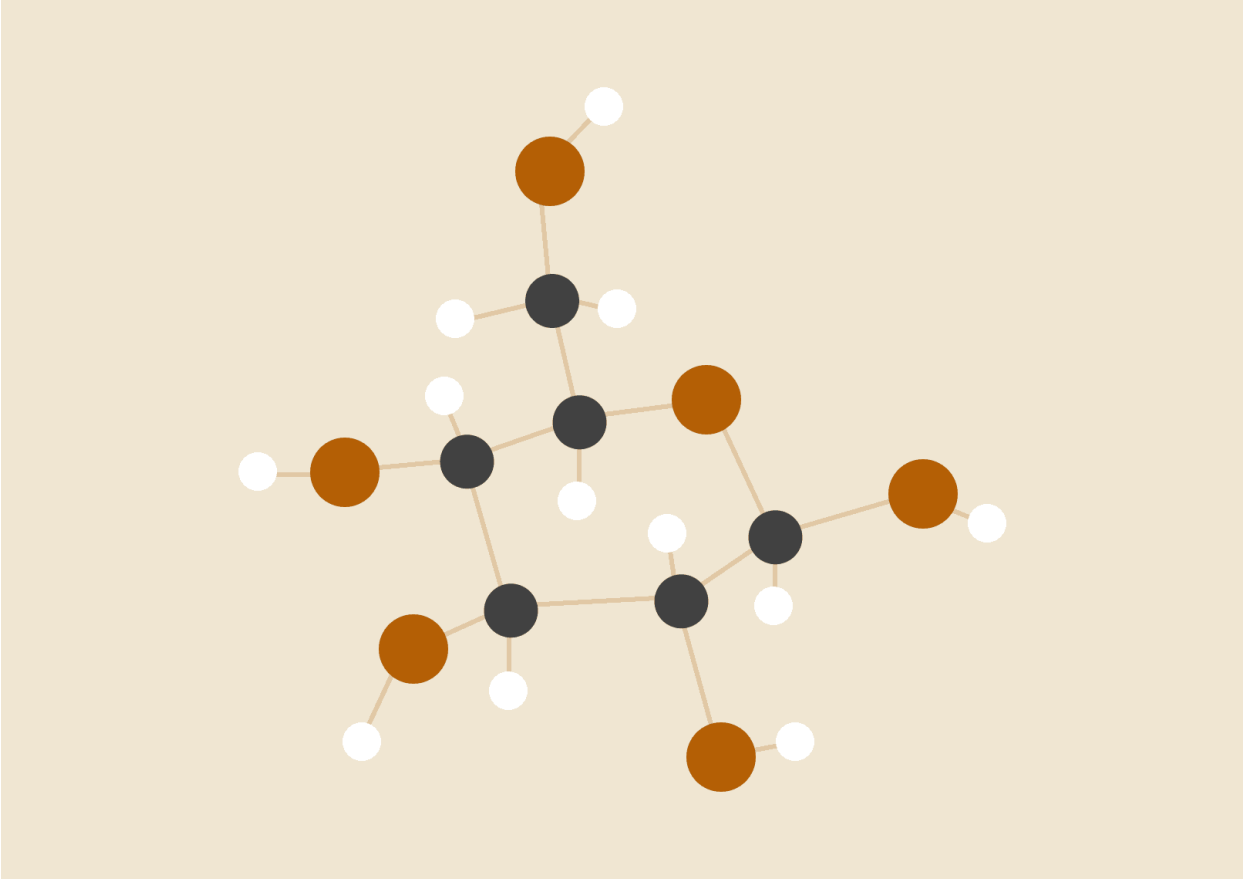
Software Testing and Quality Assurance

Prof. Mohamad Kassab



Leonardo Moreira de Araújo - 202201700

João Gabriel Cavalcante França - 202201695

July, 2024

Software Engineering

INTRODUCTION

For our Final Testing Project in Professor Mohamad Kassab's course, we decided to test the REST API created as the final project of our Software Construction course, part of our Bachelor's degree in Software Engineering. It's important to note that in this program, we have the Software Testing course in the eighth semester, but since we are only in the fifth semester, we haven't yet covered this subject, except for the Advanced Course on Software Testing and Quality Assurance by Professor Mohamad Kassab, which taught us a lot in depth with excellent didactics. Once again, we would like to thank him, along with Professor Valdemar, for organizing the course. This API was written in Java, using Spring, and does not have a front-end, so our requests were made using Postman.

API Description: The project consists of an order management system for a restaurant, facilitating day-to-day operations within this environment. Various actions are covered, such as taking orders digitally, changing the status of orders, printing the order digitally, closing table accounts, calculating profits at set intervals, keeping track of orders assigned to waiters and monitoring the establishment's stock.

Within the system, there is a distinction between two types of employee responsible for the main operations: the waiter and the manager. While the waiter concentrates on direct activities related to orders, such as writing down and controlling orders, closing accounts, among others, the manager performs the part related to stock control, finance and the restaurant's statistics. It is therefore up to the manager, when a product is found to be out of stock, to arrange for the item to be purchased and replaced.

TESTING TECHNIQUES

1. Black Box
    a. Boundary Testing
    b. Equivalence Class Testing
    c. Decision Table – Based Testing
    d. All-Pairs Testing
2. White Box
    a. Control Flow Testing
    b. Data Flow Testing (NOT APPLICABLE)
3. Exploratory Testing

REFERENCES/ TOOLS

1. Code Repository link: https://github.com/lucamartins/ufg-cs-sgr-core-api
2. Documentation Repository link: https://github.com/amadeulee/construcao-software-sgr
3. Postman
4. IntelliJ IDEA
5. Pairwise Online Tool (teremokgames.com)
6. https://www.planttext.com/
7. ~~*Testrail (we tested, but for practical usage we preferred write directly on this doc)*~~

BLACK BOX TESTING

BOUNDARY TESTING

We utilized the Boundary Testing technique with the Robustness Testing variation to test possible ages for registration in the restaurant's system. This way, we assessed whether our system could register very young or older individuals around our threshold value of 18 years. Additionally, we created two extra test cases for extreme scenarios: whether we could register people over 120 years old and those who have not yet been born.

As the class User doesn't depend on any other, we consider this a Unit testing.

TC-BB-BOUNDARY-01: AGE OF EMPLOYEES MIN-

VERSION 1.0

OBJECTIVE:

- To check whether it is possible to register an employee with an age far below 18 in the system.

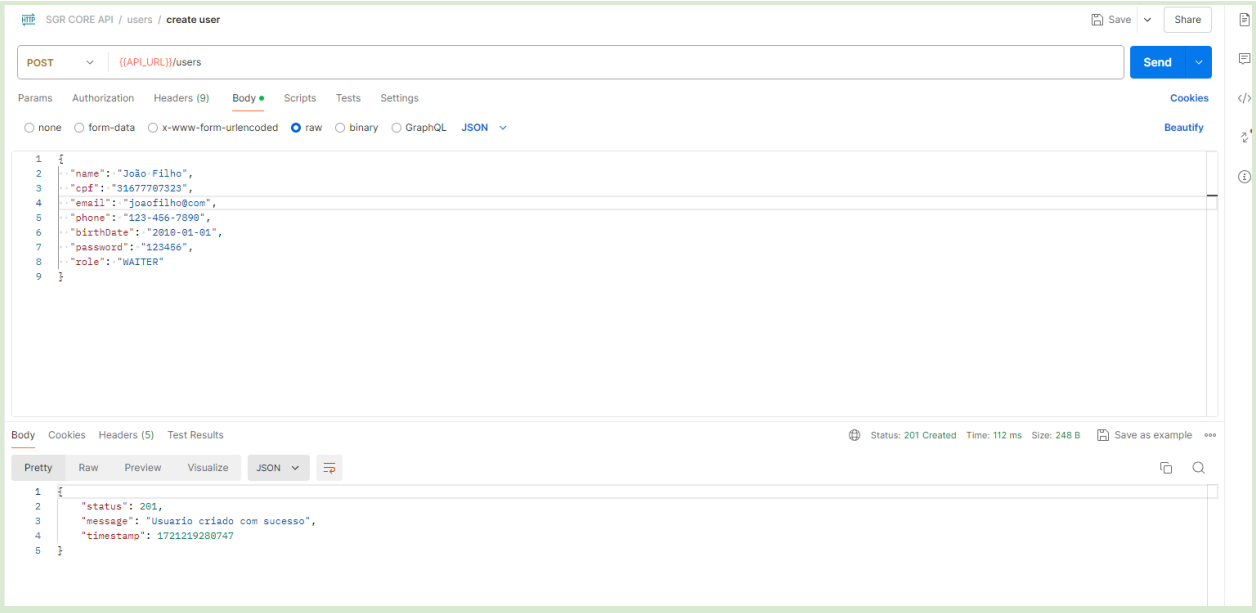
PRECONDITIONS:

- The user must be much younger than 18 on the day of registration.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "João Filho", "cpf": "31677707323", "email": joaofilho@com", "phone": "123-456-7890", "birthDate":"2010-01-01", "password": "123456", "role": "WAITER"}	{ "status": 400, "message": "Idade inválida: o funcionário deve ter pelo menos 18 anos.", "timestamp": 1721219280747}	{ "status": 201, "message": "Usuario criado com sucesso", "timestamp": 1721219280747}



TC-BB-BOUNDARY-02: AGE OF EMPLOYEES MIN+

VERSION 1.0

OBJECTIVE:

- To check whether it is possible to register, in the system, an employee with an age close to below 18.

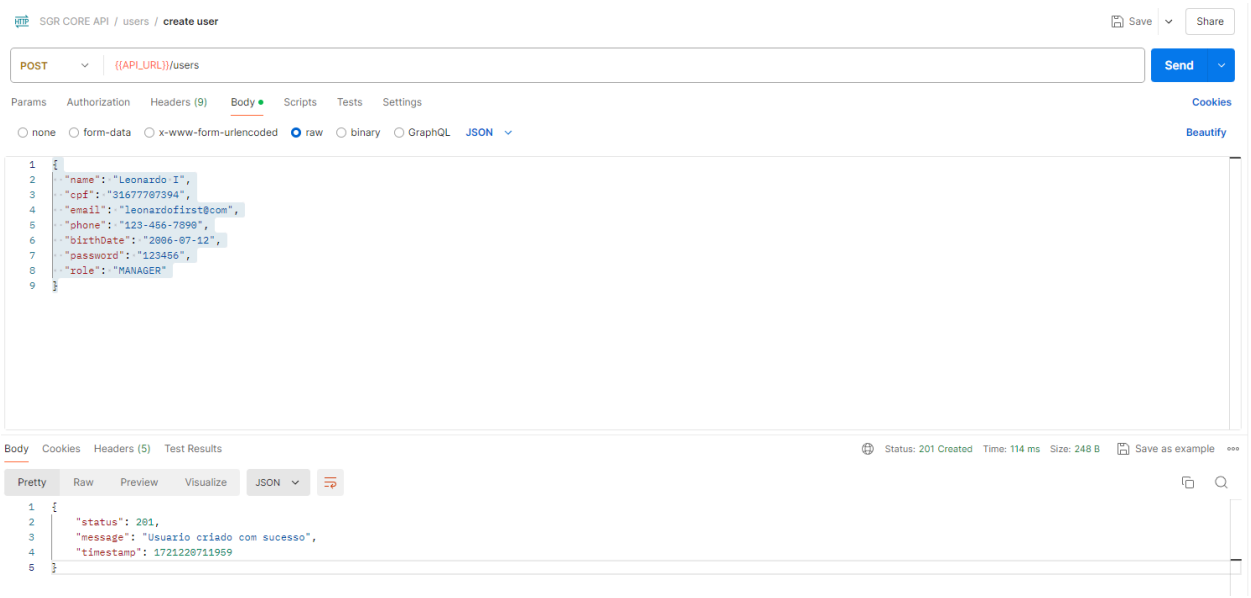
PRECONDITIONS:

- The registered user must reach the age of majority the day after registration, in other words, they are registered when they are 17, but will turn 18 the day after.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "Leonardo I", "cpf": "31677707394", "email":"leonardofirst@com", "phone": "123-456-7890", "birthDate":"2006-07-12", "password": "123456", "role": "MANAGER" }	{ "status": 400, "message": "Idade inválida: o funcionário deve ter pelo menos 18 anos.", "timestamp": 1721220711959}	{ "status": 201, "message": "Usuario criado com sucesso", "timestamp": 1721220711959}



TC-BB-BOUNDARY-03: AGE OF EMPLOYEES NORM

VERSION 1.0

OBJECTIVE:

- To check whether it is possible to register an employee aged 18 in the system.

PRECONDITIONS:

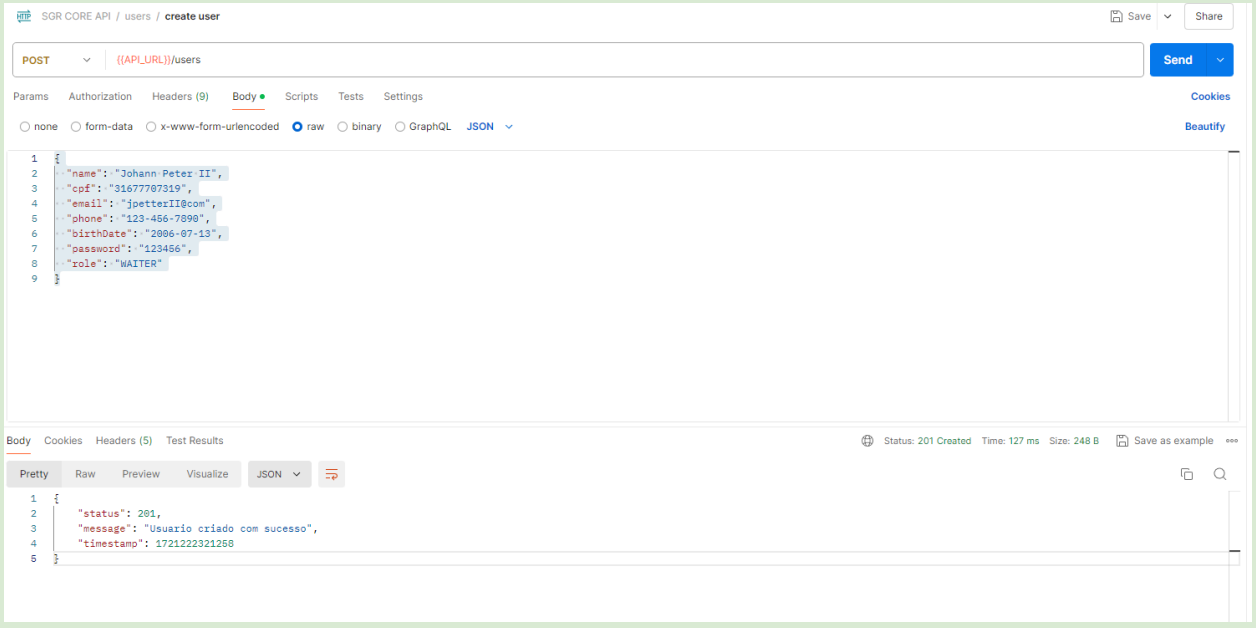
- The registered user must be 18 on the day of registration.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "Johann Peter II",	{ "status": 201,	{ "status": 201,

"cpf": "31677707319", "email":"jpetterII@com", "phone": "123-456-7890", "birthDate":"2006-07-13", "password": "123456", "role": "WAITER"}	"message": "Usuario criado com sucesso", "timestamp": 1721222321258 }	"message": "Usuario criado com sucesso", "timestamp": 1721222321258 }



TC-BB-BOUNDARY-04: AGE OF EMPLOYEES MAX-

VERSION 1.0

OBJECTIVE:

- Check whether it is possible to register an employee aged 18 or over in the system.

PRECONDITIONS:

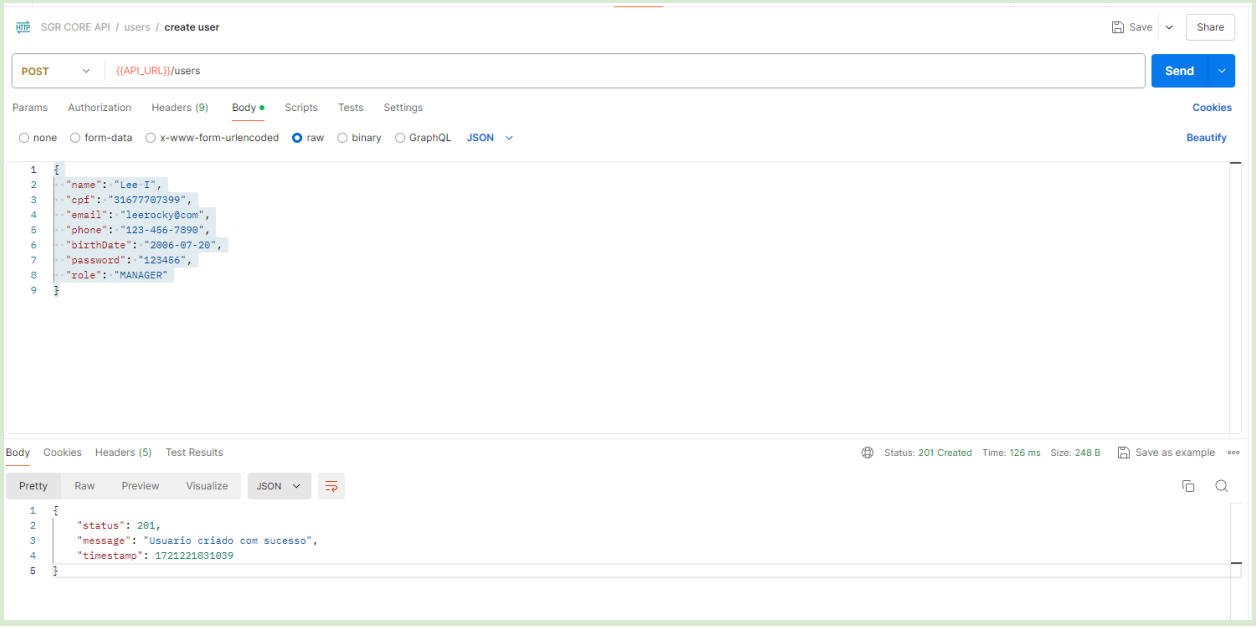
- The registered user must be just over 18 on the day of registration.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

<pre>{ "name": "Lee I",
"cpf": "31677707399",
"email": "leerocky@com",
"phone": "123-456-7890",
"birthDate": "2006-07-20",
"password": "123456",
"role": "MANAGER"}</pre>
	<pre>{ "status": 201,
	"message": "Usuario criado com sucesso"
	"timestamp": any }</pre>
		<pre>{ "status": 201,
		"message": "Usuario criado com sucesso",
		"timestamp":1721221831039}</pre>



TC-BB-BOUNDARY-05: AGE OF EMPLOYEES MAX+

VERSION 1.0

OBJECTIVE:

- To check whether it is possible to register, in the system, an employee with an age far above 18.

PRECONDITIONS:

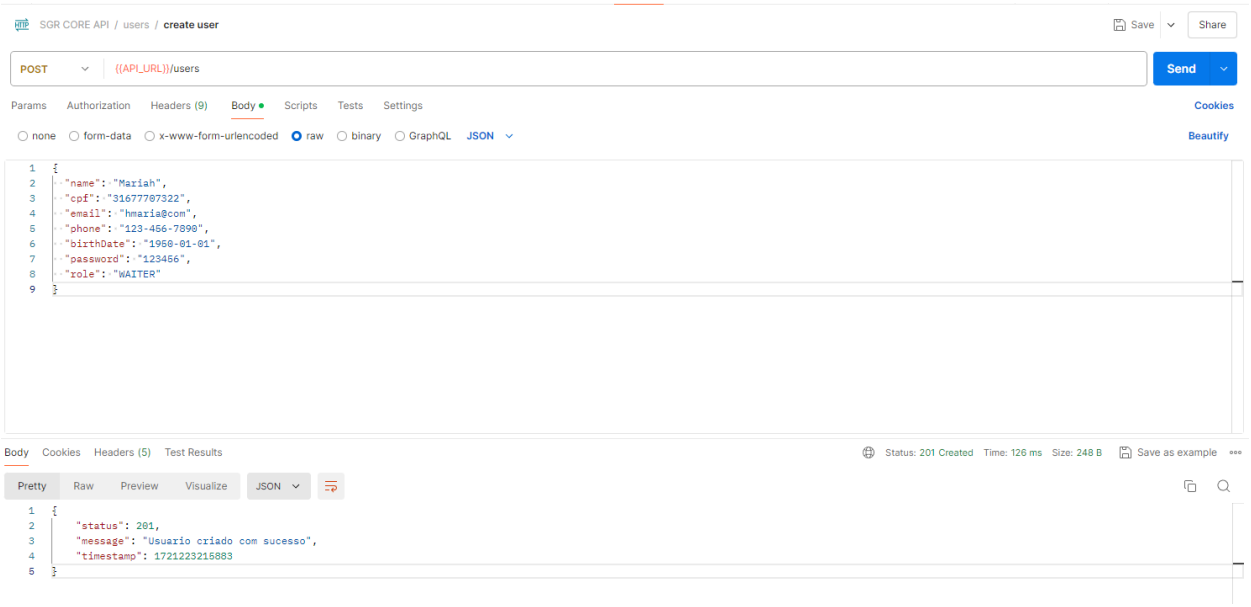
- The registered user must be well over 18 on the day of registration.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

<pre>{
"name": "Mariah",
"cpf": "31677707322",
"email": "hmaria@com",
"phone": "123-456-7890",
"birthDate": "1950-01-01",
"password": "123456",
"role": "WAITER"
}</pre>
	<pre>{
	"status": 201,
	"message": "Usuario criado com sucesso",
	"timestamp": 1721223215883
	}</pre>
		<pre>{
		"status": 201,
		"message": "Usuario criado com sucesso",
		"timestamp": 1721223215883
		}</pre>



TC-BB-BOUNDARY-06: AGE OF EMPLOYEES OUTLIER MIN

VERSION 1.0

OBJECTIVE:

- Check whether it is possible to register an employee over the age of 120 in the system.

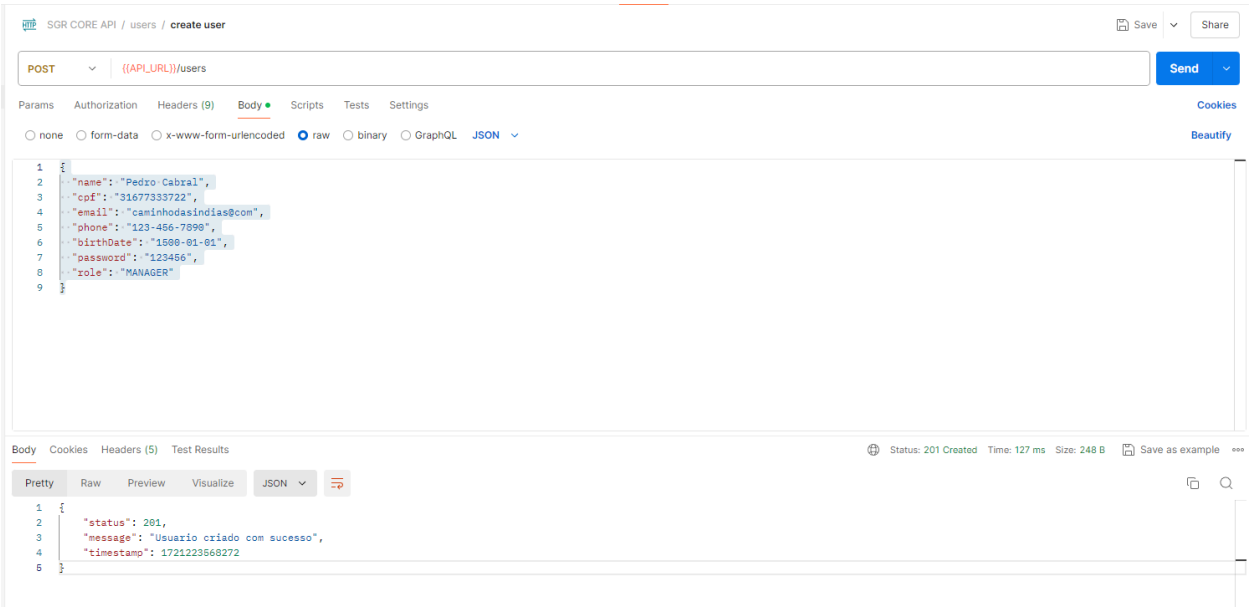
PRECONDITIONS:

- The registered user must be well over 18 on the day of registration.

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "Pedro Cabral", "cpf": "31677333722", "email": "caminhodasindias@com", "phone": "123-456-7890", "birthDate": "1500-01-01", "password": "123456", "role": "MANAGER"}	{ "status": 400, "message": "Idade inválida”, "timestamp":1721223568272 }	{ "status": 201, "message": "Usuario criado com sucesso", "timestamp": 1721223568272 }



TC-BB-BOUNDARY-07: AGE OF EMPLOYEES OUTLIER MAX

VERSION 1.0

OBJECTIVE:

- To check whether it is possible to register an employee in the system who has not yet been born.

PRECONDITIONS:

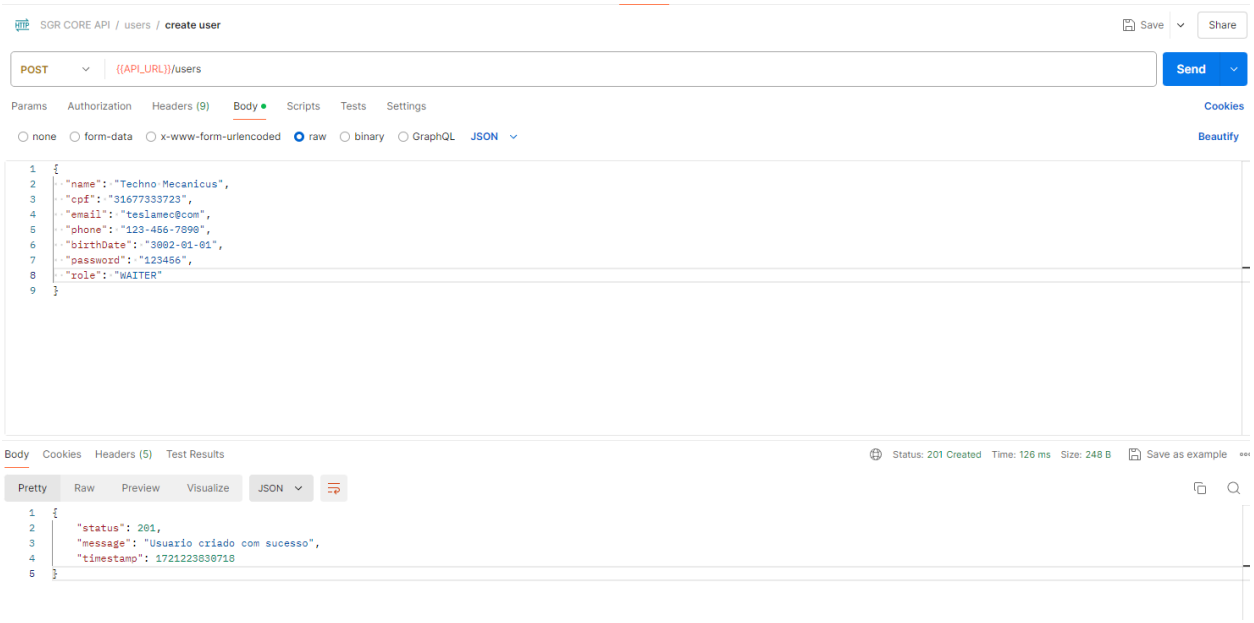
- The registered user must have a date of birth within one day

ACTIONS:

- Fill in the fields with valid data (age, cpf, email, password...)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

{ "name": "Techno Mecanicus",
"cpf": "31677333723",
"email": "teslamec@com",
"phone": "123-456-7890",
"birthDate": "3002-01-01",
"password": "123456",
"role": "WAITER"
}
	{ "status": 400,
	"message": "Idade inválida”,
	"timestamp":1721223830718
	}
		{
		"status": 201,
		"message": "Usuario criado com sucesso",
		"timestamp": 1721223830718
		}



CONCLUSION ABOUT THE BOUNDARY TESTING

From these tests, we found that our system does not yet have an integrated age verification for database registration. This means that all individuals are registered regardless of age, which does not meet our system's requirements. We could repeat these tests using the Regression technique as long as we fix the age verification.

EQUIVALENCE CLASS TESTING

During the application of the “Equivalence Class Testing” technique, an input condition was defined which specified an interval [a, b] to which the data entered belonged. In this way, we evaluated the application's ability to deal with different types of input in its “add stock item” functionality, which is responsible for adding an item to the store's stock, requiring three fields to be filled in: “name”, “allowFractionalQuantity” and “measurementUnit” for the action to be carried out.

Our tested domain will be the “name” input variable, which we will divide into four subdomains: valid name, very short name, very long name and name with special

characters.

The “valid name” subgroup consists of valid entries and the other subgroups are considered invalid entries;

TC-BB-EQUIVALENCE-CLASS-01: VALID NAME (3 <= length <= 255 characters)

VERSION 1.0

OBJECTIVE:

- Check that the action of inserting an item into the application database is successful

PRECONDITIONS:

- The item must not already be present in the database, this must be its first insertion

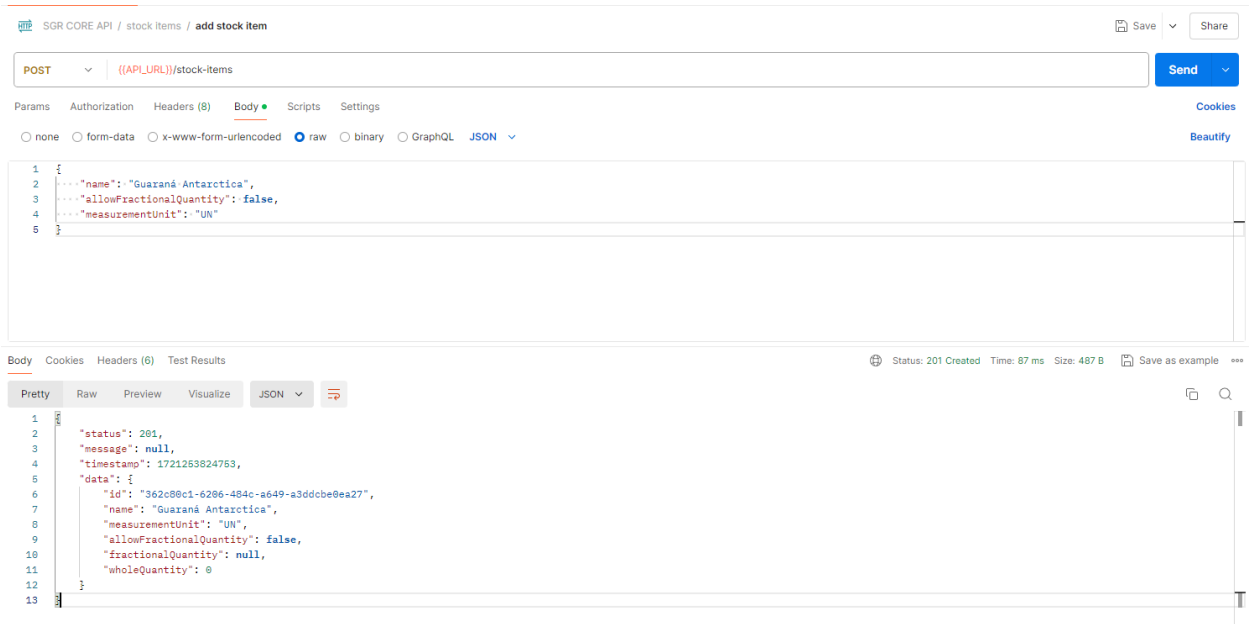
ACTIONS:

- Fill in the fields with valid data (name, allowFractionalQuantity, measurementUnit)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

<pre>{
"name": "Guaraná Antarctica",
"allowFractionalQuantity": false,
"measurementUnit": "UN"
}</pre>
	<pre>{
	"status": 201,
	"message": null,
	"timestamp": 1721253824753,
	"data": {
	"id": "362c80c1-6206-484c-a649-a</pre>
		<pre>{
		"status": 201,
		"message": null,
		"timestamp": 1721253824753,
		"data": {
		"id": "362c80c1-6206-484c-a649-a</pre>

	3ddcbe0ea27", "name": "Guaraná Antarctica", "measurementUnit": "UN", "allowFractionalQuantity": **false**, "fractionalQuantity": **null**, "wholeQuantity": 0 } }	3ddcbe0ea27", "name": "Guaraná Antarctica", "measurementUnit": "UN", "allowFractionalQuantity": **false**, "fractionalQuantity": **null**, "wholeQuantity": 0 } }



TC-BB-EQUIVALENCE-CLASS-02: VERY SHORT NAME (length < 3)

VERSION 1.0

OBJECTIVE:

- Check that the action of inserting an item into the application database is NOT successful

PRECONDITIONS:

- The item must not already be present in the database, this must be its first insertion

ACTIONS:

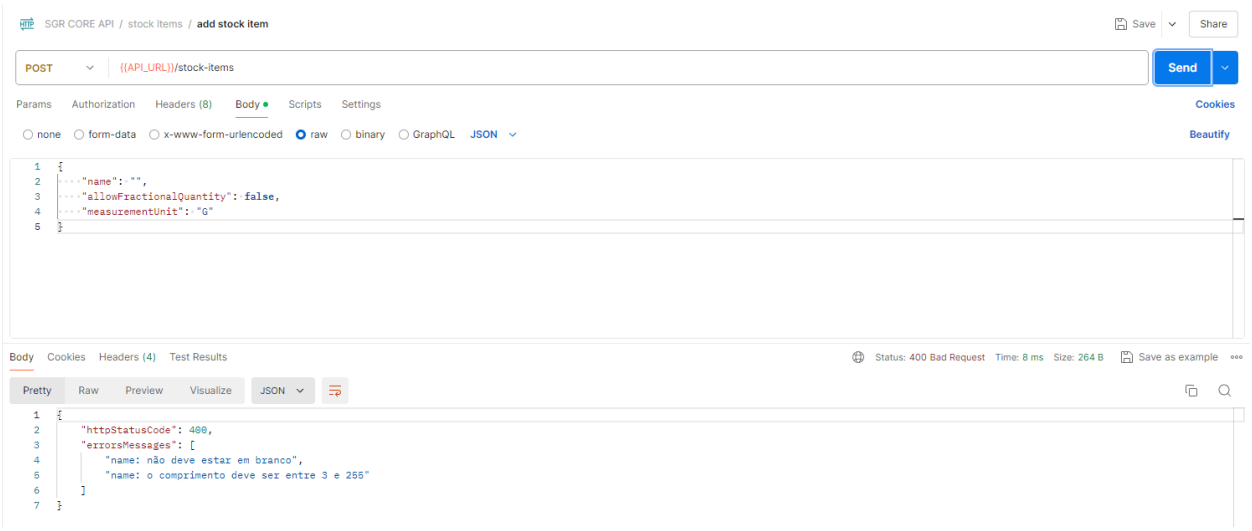
- Fill in the fields with valid data (name, allowFractionalQuantity, measurementUnit)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

{
"name": "",
"allowFractionalQuantity": false,
"measurementUnit": "G"
}
	{
	"httpStatusCode": 400,
	"errorsMessages": [
	"name: não deve estar em branco",
	"name: o comprimento deve ser entre 3 e 255"
	]
	}
		{
		"httpStatusCode": 400,
		"errorsMessages": [
		"name: não deve estar em branco",
		"name: o comprimento deve ser entre 3 e 255"
		]
		}

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "Ab",	{ "httpStatusCode": 400,	{ "httpStatusCode": 400,

"allowFractionalQuantity": false, "measurementUnit": "G" }	"errorsMessages": ["name: o comprimento deve ser entre 3 e 255"] }	"errorsMessages": ["name: o comprimento deve ser entre 3 e 255"] }



TC-BB-EQUIVALENCE-CLASS-03: VERY LONG NAME (length > 255)

VERSION 1.0

OBJECTIVE:

- Check that the action of inserting an item into the application database is NOT successful

PRECONDITIONS:

- The item must not already be present in the database, this must be its first insertion

ACTIONS:

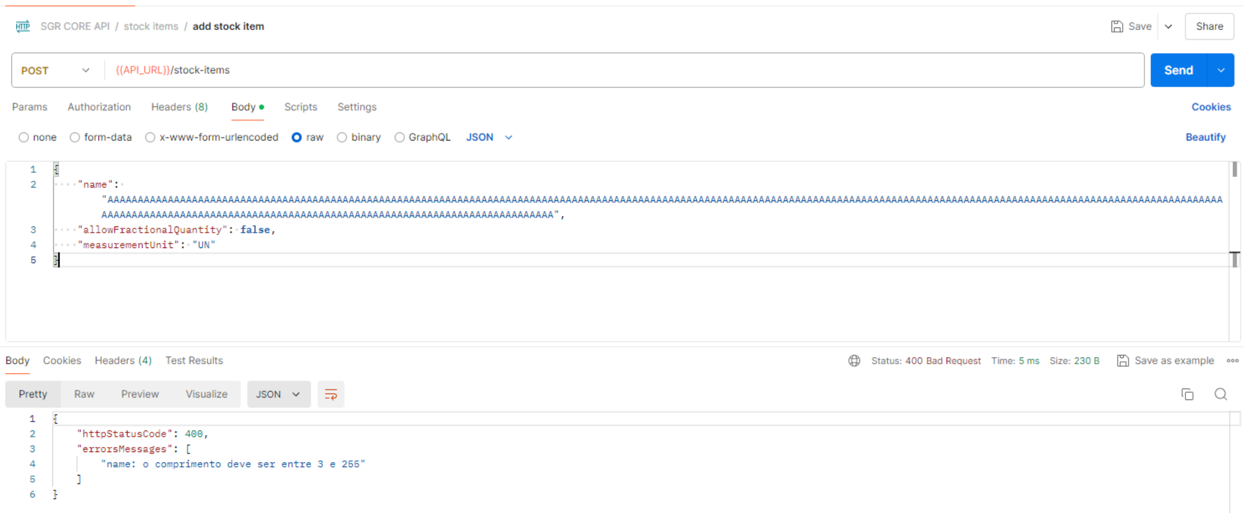
- Fill in the fields with valid data (name, allowFractionalQuantity,

measurementUnit)

- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT

{
 	
"name": "AAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA
A",
 	
"allowFractionalQuantity": false,
 	
"measurementUnit": "UN"
 	
}
	{
	
	"httpStatusCode": 400,
	
	"errorsMessages": [
	
	"name: o comprimento deve ser entre 3 e 255"
	
	]
	
	}
		{
		
		"httpStatusCode": 400,
		
		"errorsMessages": [
		
		"name: o comprimento deve ser entre 3 e 255"
		
		]
		
		}



TC-BB-EQUIVALENCE-CLASS-04: NAME WITH SPECIAL CHARACTERS

VERSION 1.0

OBJECTIVE:

- Check that the action of inserting an item into the application database is NOT successful

PRECONDITIONS:

- The item must not already be present in the database, this must be its first insertion

ACTIONS:

- Fill in the fields with valid data (name, allowFractionalQuantity, measurementUnit)
- Execute the request
- Check that the action has been carried out successfully

INPUT	EXPECTED OUTPUT	OUTPUT
{	{	{

<table>
<tr>
<td>

<pre>
    "name":
"Leonardo!@#$%¨&*}{PÇ^{`Ç`
ÇLKP::><",

"allowFractionalQuantity":
false,

    "measurementUnit":
"UN"

}
</pre>

</td>
<td>

<pre>
    "httpStatusCode": 400,

    "errorsMessages": [

        "name: o nome não
deve possuir caracteres
especiais"

    ]

}
</pre>

</td>
<td>

<pre>
    "status": 201,

    "message": null,

    "timestamp":
1721254112790,

    "data": {

        "id":
"d8373fd8-4ed1-496a-99ee-a
158f13f4228",

        "name":
"Leonardo!@#$%¨&*}{PÇ^{`Ç`
ÇLKP::><",

        "measurementUnit":
"UN",

"allowFractionalQuantity":
false,

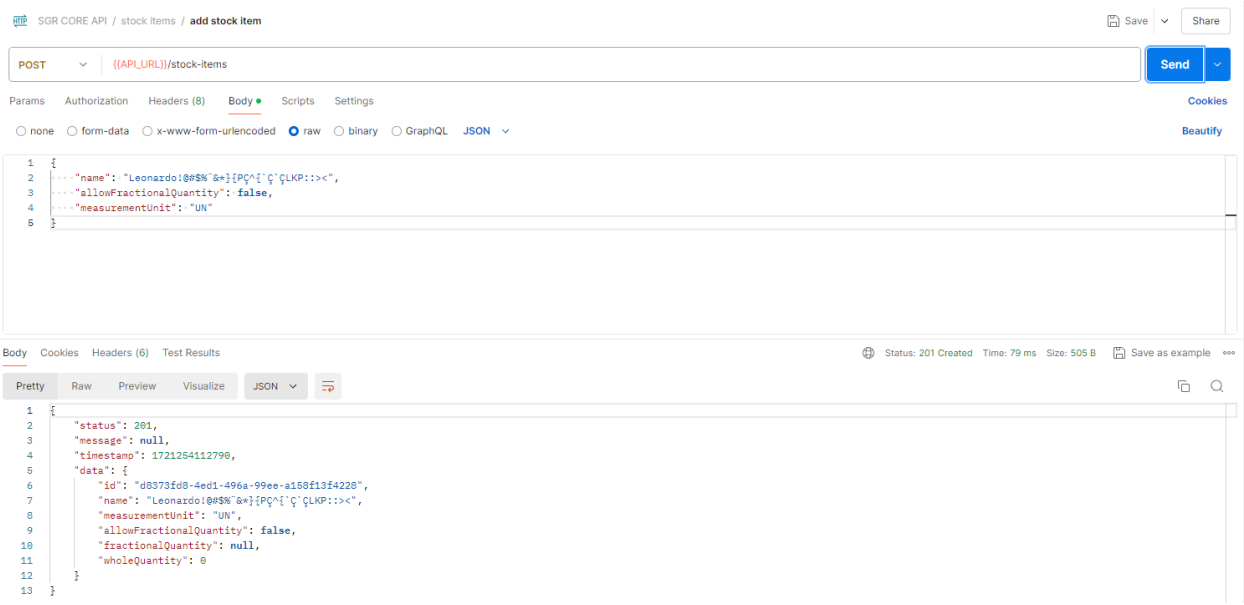
"fractionalQuantity":
null,

        "wholeQuantity": 0

    }

}
</pre>

</td>
</tr>
</table>



CONCLUSION ABOUT THE EQUIVALENCE CLASS TESTING

When we look at the resulting output from our tests, we can conclude that the system has partial filtering of the data that is recorded in the database, it normally handles texts that exceed the predefined size of the variable, but it is vulnerable to names that have special characters, something that should not be accepted by the system as it does not comply with the business rules.

DECISION TABLE – BASED TESTING

To apply the “Decision Table - Based Testing” method, we used the system's “table” object as the focus of our analysis. We evaluated its conditions of existence, how this affects the actions and the existing business rules that relate directly to this object.

CONDITIONS	RULES			
	1	2	3	4
Table is create	T	F	T	F
Table is available	T	F	F	T
ACTIONS				
Start a service	**T**	**F**	**F**	**F**
Make an order	**F**	**F**	**T**	**F**
Delivering an order	**F**	**F**	**T**	**F**
Close table service	**F**	**F**	**T**	**F**

CONCLUSION ABOUT THE DECISION TABLE – BASED TESTING

This methodology helped us visualize the communication between the system's objects and how they should relate to the requirements specified by the stakeholders.

The simplicity of visualizing the data and results makes it dynamic and facilitates the analysis and conclusion of the tests.

ALL-PAIRS TESTING

For the “All-Pairs Testing” method, we used the relationship and communication between the “Customer Table”, “Service” and “Order” objects in the system as a basis for analysis, using their possible state conditions as a basis: create , not create, available, not available, close , not close, delivery and not delivery.

Customer Table	Service	Order
create	create	create
create	not create	not create

create	close	delivery
create	not close	not delivery
not create	not create	delivery
not create	close	not delivery
not create	not close	create
not create	create	not create
available	close	create
available	not close	not create
available	create	delivery
available	not create	not delivery
not available	not close	delivery
not available	create	not delivery
not available	not create	create
not available	close	not create

CONCLUSION ABOUT THE ALL-PAIRS TESTING

All-Pairs Testing eliminates repetition and makes our tests focus on the main cases of combinations between related functionalities, covering a large part of the errors with reduced effort and expense. As a result, it is one of the main options for choosing a testing approach, being extremely efficient and affordable.

WHITE BOX

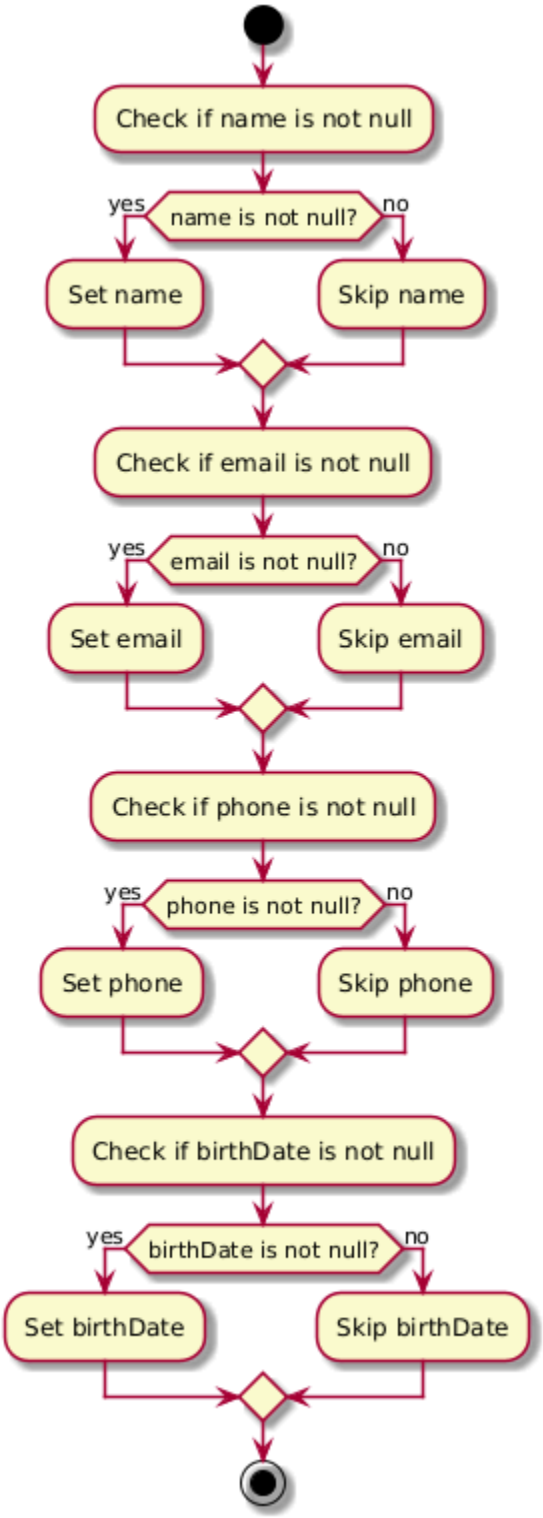
CONTROL FLOW TESTING

Control Flow Testing in the patchUser method aims to ensure that all branches and logical paths of the code are tested to verify the correctness of the control flow. The patchUser method updates the properties of a User object based on the values present in the UserUpdateRequest.

Graph Components:

- Null Check: For each attribute (name, email, phone, birthDate) in UserUpdateRequest, the method checks if the value is not null.
- Value Assignment: If the value is not null, the corresponding attribute in the User object is updated.
- Conditional Branching: If the value is null, the update for that attribute is skipped.

```
private void patchUser(User user,
UserUpdateRequest
userUpdateRequest) {
    if (userUpdateRequest.name()
!= null) {

user.setName(userUpdateRequest.na
me());
    }
    if (userUpdateRequest.email()
!= null) {

user.setEmail(userUpdateRequest.e
mail());
    }
    if (userUpdateRequest.phone()
!= null) {

user.setPhone(userUpdateRequest.p
hone());
    }
    if
(userUpdateRequest.birthDate() !=
null) {

user.setBirthDate(userUpdateReque
st.birthDate());
    }
}
```



EXPLORATORY TESTING

The exploratory text in the system was executed by creating test cases during test runtime. We performed a basic routine of registering a waiter, authenticating the registered waiter, adding items to the inventory, creating a sale item, creating a table, creating a service for this table with the waiter we created earlier, making two orders of the sale item we created, delivering the order, and finishing the service.

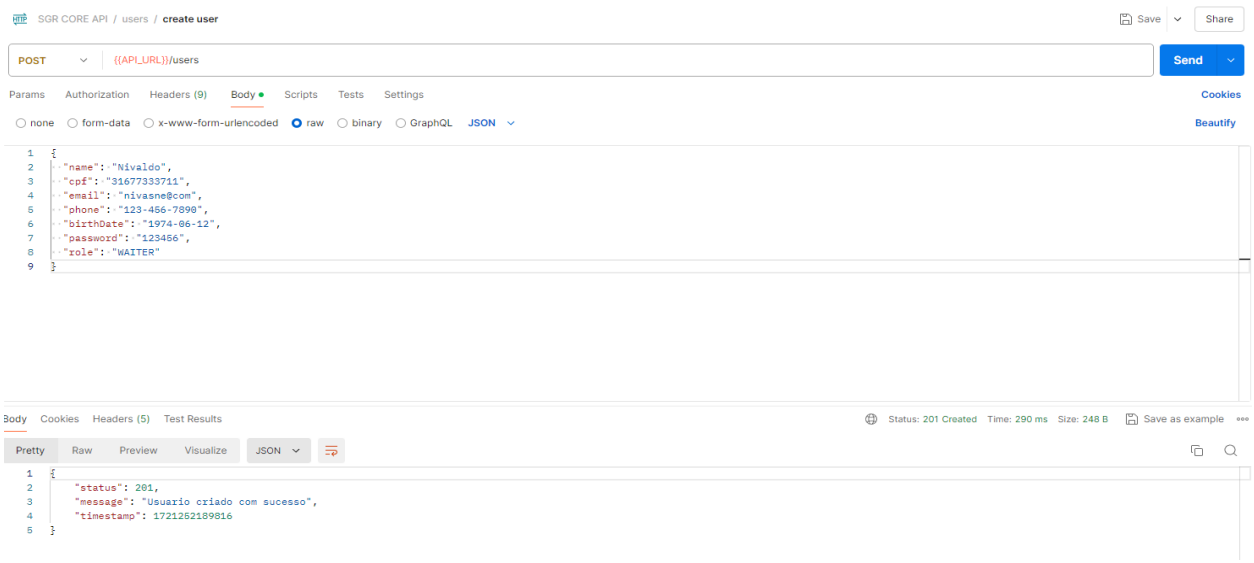
```
Steps
1. Create User
2. Authentication from the user created with wrong password
3. Authentication from the user created with correct correct password
4. Add a new stock item - Peanut (Amendoim)
5. Stock movement - Peanut, adding 3 kilograms
6. Stock movement - Amendoim, adding another 3 kilograms
7. Create a SaleItem Paçoca (Peanut Butter Snack) using Peanut (the stock
8. Create a customer table
9. Create a table service for the customer table
10. Order for a Peanut Butter Snack.
11. Order for a Peanut Butter Snack.
12. Deliver the first order
13. Deliver the second order
14.  Close table service
```

1. **Create User**

INPUT	EXPECTED OUTPUT	OUTPUT

<pre>{
  "name": "Nivaldo",</pre>
	<pre>{
	  "status": 201,</pre>
		<pre>{
		  "status": 201,</pre>

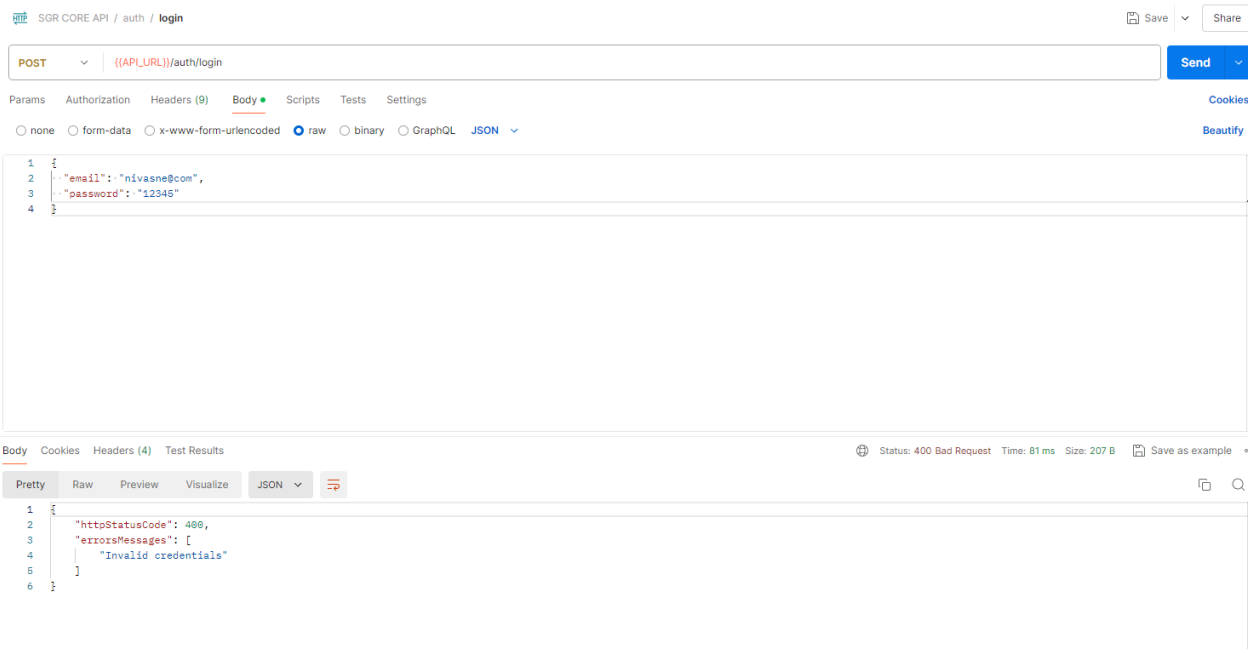
"cpf": "31677333711", "email": "nivasne@com", "phone": "123-456-7890", "birthDate": "1974-06-12", "password": "123456", "role": "WAITER"} }	"message": "Usuario criado com sucesso", "timestamp": 1721252189816 }	"message": "Usuario criado com sucesso", "timestamp": 1721252189816 }



2. Authentication from the user created with wrong password

INPUT	EXPECTED OUTPUT	OUTPUT
{ "email": "nivasne@com",	{ "httpStatusCode": 400,	{ "httpStatusCode": 400,

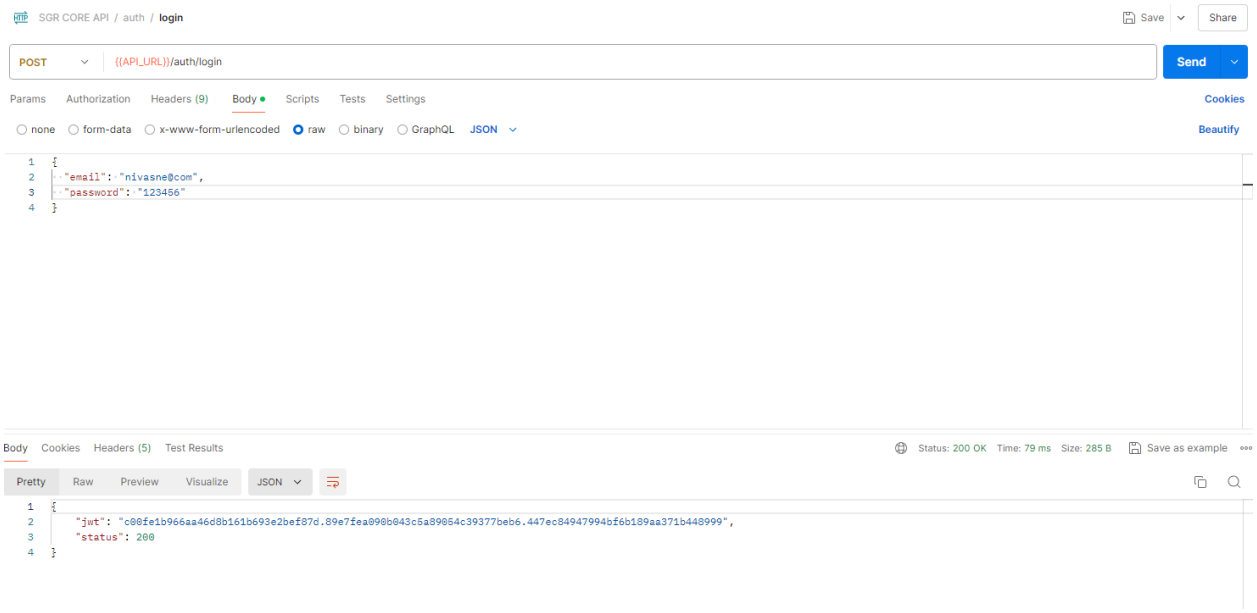
"password": "12345" }	"errorsMessages": ["Invalid credentials"] }	"errorsMessages": ["Invalid credentials"] }



3. Authentication from the user created with correct correct password

INPUT	EXPECTED OUTPUT	OUTPUT
{ "email": "nivasne@com", "password": "123456"	{ "jwt": "c00fe1b966aa46d8b161b693e	{ "jwt": "c00fe1b966aa46d8b161b693e

}	2bef87d.89e7fea090b043c5a89054c39377beb6.447ec84947994bf6b189aa371b448999", "status": 200 }	2bef87d.89e7fea090b043c5a89054c39377beb6.447ec84947994bf6b189aa371b448999", "status": 200 }

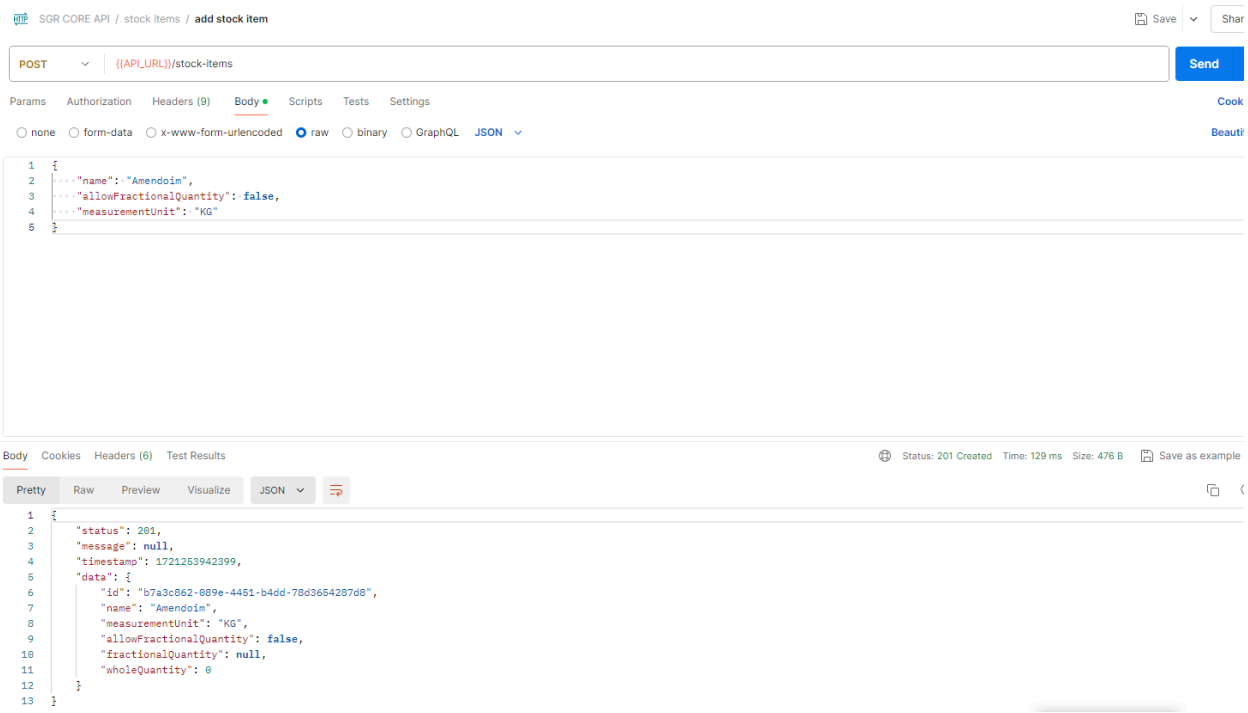


4. Add a new stock item - Peanut (Amendoim)

INPUT	EXPECTED OUTPUT	OUTPUT

{
"name": "Amendoim",
"allowFractionalQuantity": false,
"measurementUnit": "KG"
	{
	"status": 201,
	"message": null,
	"timestamp": 1721253942399,
	"data": {
		{
		"status": 201,
		"message": null,
		"timestamp": 1721253942399,
		"data": {

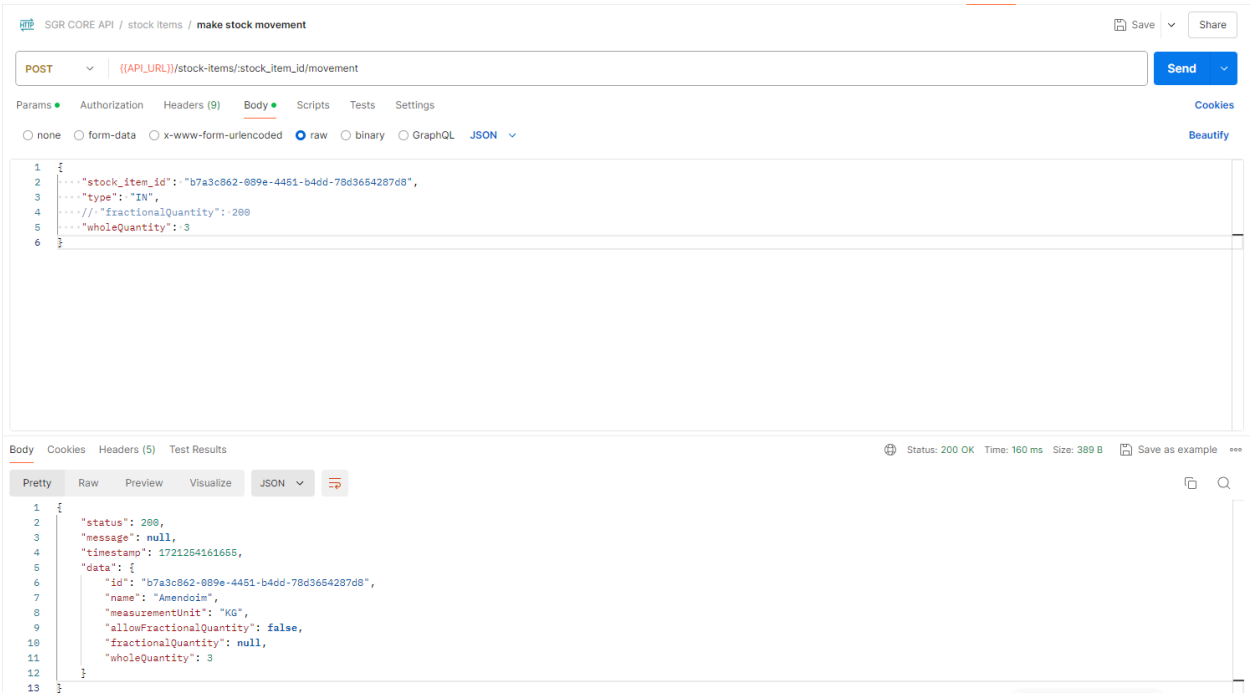
}	"id": "b7a3c862-089e-4451-b4dd-78d3654287d8", "name": "Amendoim", "measurementUnit": "KG", "allowFractionalQuantity": **false**, "fractionalQuantity": **null**, "wholeQuantity": 0 } }	"id": "b7a3c862-089e-4451-b4dd-78d3654287d8", "name": "Amendoim", "measurementUnit": "KG", "allowFractionalQuantity": **false**, "fractionalQuantity": **null**, "wholeQuantity": 0 } }



5. Stock movement - Peanut, adding 3 kilograms

INPUT	EXPECTED OUTPUT	OUTPUT
{ "stock_item_id": "b7a3c862-089e-4451-b4dd-78d3654287d8", "type": "IN", // "fractionalQuantity": 200 "wholeQuantity": 3	{ "status": 200, "message": null, "timestamp": 1721254161655, "data": { "id":	{ "status": 200, "message": null, "timestamp": 1721254161655, "data": { "id":

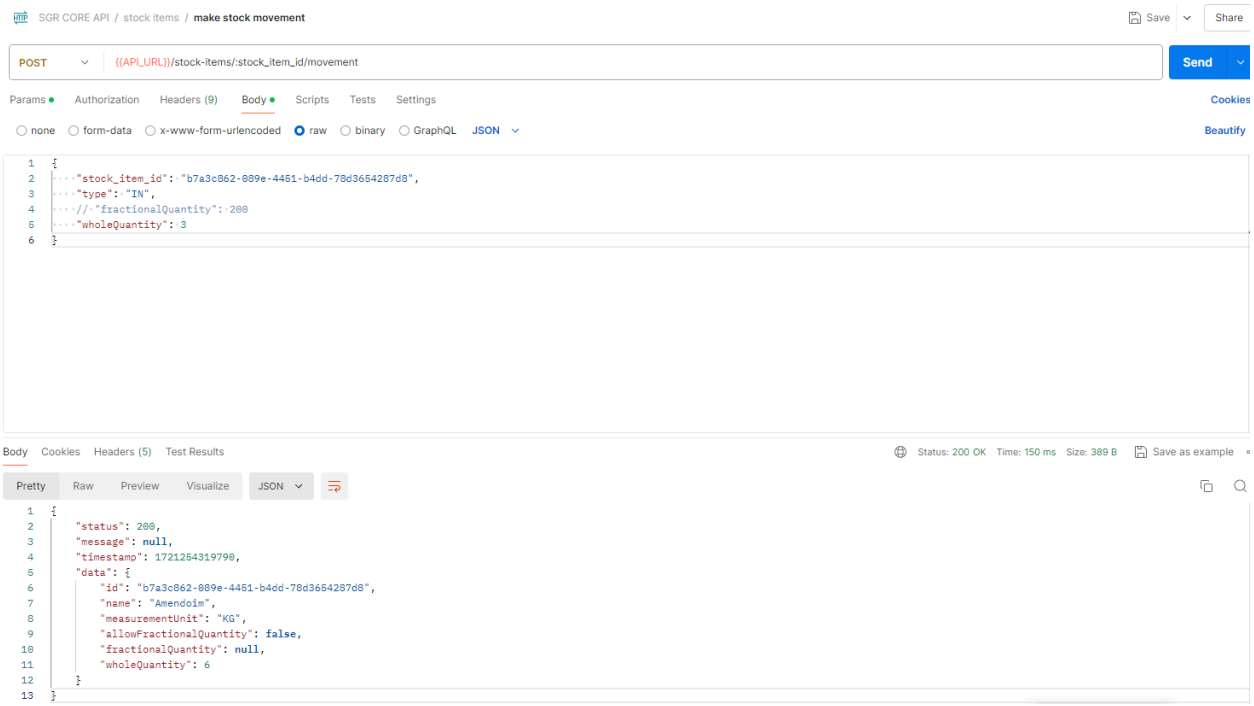
}	"b7a3c862-089e-4451-b4dd-78d3654287d8", "name": "Amendoim", "measurementUnit": "KG", "allowFractionalQuantity": false, "fractionalQuantity": null, "wholeQuantity": 3 } }	"b7a3c862-089e-4451-b4dd-78d3654287d8", "name": "Amendoim", "measurementUnit": "KG", "allowFractionalQuantity": false, "fractionalQuantity": null, "wholeQuantity": 3 } }



6. **Stock movement - Amendoim, adding another 3 kilograms**

INPUT	EXPECTED OUTPUT	OUTPUT

<pre>{
"stock_item_id": "b7a3c862-089e-4451-b4dd-78d3654287d8",
"type": "IN",
// "fractionalQuantity": 200
"wholeQuantity": 3
}</pre>
	<pre>{
	"status": 200,
	"message": null,
	"timestamp": 1721254319790,
	"data": {
	"id": "b7a3c862-089e-4451-b4dd-78d3654287d8",
	"name": "Amendoim",
	"measurementUnit": "KG",
	"allowFractionalQuantity": false,
	"fractionalQuantity": null,
	"wholeQuantity": 6
	}
	}</pre>
		<pre>{
		"status": 200,
		"message": null,
		"timestamp": 1721254319790,
		"data": {
		"id": "b7a3c862-089e-4451-b4dd-78d3654287d8",
		"name": "Amendoim",
		"measurementUnit": "KG",
		"allowFractionalQuantity": false,
		"fractionalQuantity": null,
		"wholeQuantity": 6
		}
		}</pre>



7. **Create a SaleItem Paçoca (Peanut Butter Snack) using Peanut (the stock item that we add) + Sugar (item that already exists in our database)**

INPUT	EXPECTED OUTPUT	OUTPUT
{ "name": "Paçoca", "price": 1.00, "priceCurrency": "BRL", "isSaleAvailable": true, "stockItems": [{	{ "status": 201, "message": null, "timestamp": 1721256030205, "data": { "id": "836eb68f-df12-46fd-bc0a-2366b760ffae",	{ "status": 201, "message": null, "timestamp": 1721256030205, "data": { "id": "836eb68f-df12-46fd-bc0a-2366b760ffae",

<table>
<tr>
<td>

<pre>
            "stockItemId": "b7a3c862-089e-4451-b4dd-78d3654287d8",

"wholeQuantity": 0.2

        },

        {

            "stockItemId": "4590ffa2-a597-4b25-b6c1-4504f274628b",

"wholeQuantity": 0.3

        }

    ]

}
</pre>

</td>
<td>

<pre>
        "name": "Paçoca",

        "price": 1.0,

        "priceCurrency": "BRL",

        "isSaleAvailable": true,

        "stockItems": [

            {

                "id": "ba62b07c-95db-4261-accf-63162ac8e69d",

"stockItem": {

                    "id": "b7a3c862-089e-4451-b4dd-78d3654287d8",

                    "name": "Amendoim",

"measurementUnit": "KG",

"allowFractionalQuantity": false,

"fractionalQuantity": null,

"wholeQuantity": 6

                },
</pre>

</td>
<td>

<pre>
        "name": "Paçoca",

        "price": 1.0,

        "priceCurrency": "BRL",

"isSaleAvailable": true,

        "stockItems": [

            {

                "id": "ba62b07c-95db-4261-accf-63162ac8e69d",

"stockItem": {

                    "id": "b7a3c862-089e-4451-b4dd-78d3654287d8",

"name": "Amendoim",

"measurementUnit": "KG",

"allowFractionalQuantity": false,

"fractionalQuantity": null,

"wholeQuantity": 6

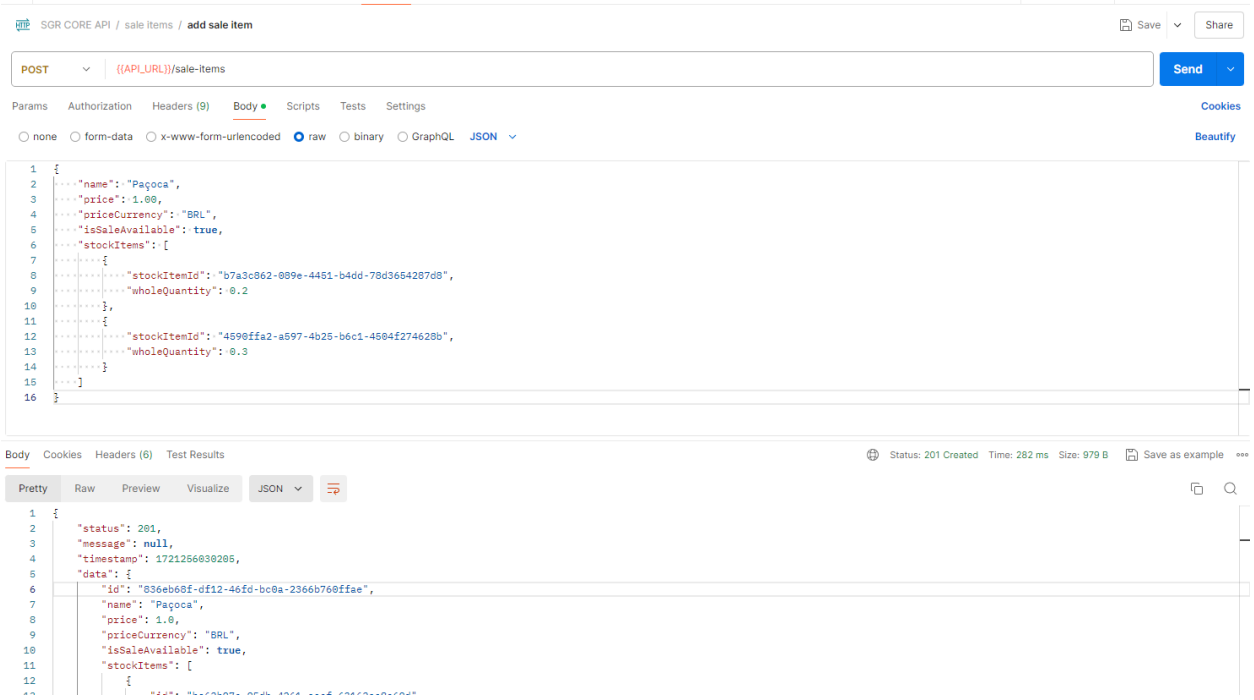
                },
</pre>

</td>
</tr>
</table>

```
"fractionalQuantity": null,

"wholeQuantity": 0

            },

            {

                "id":
"516c558a-ba5d-4a95-95f0-1f
da2cb5d7ee",

"stockItem": {

                    "id":
"4590ffa2-a597-4b25-b6c1-45
04f274628b",

                    "name":
"Açúcar",

"measurementUnit": "KG",

"allowFractionalQuantity":
false,

"fractionalQuantity": null,

"wholeQuantity": 10

                },

"fractionalQuantity": null,

"wholeQuantity": 0
```

```
"fractionalQuantity":
null,

"wholeQuantity": 0

            },

            {

                "id":
"516c558a-ba5d-4a95-95f0-
1fda2cb5d7ee",

"stockItem": {

                    "id":
"4590ffa2-a597-4b25-b6c1-
4504f274628b",

"name": "Açúcar",

"measurementUnit": "KG",

"allowFractionalQuantity"
: false,

"fractionalQuantity":
null,

"wholeQuantity": 10

                },

"fractionalQuantity":
```

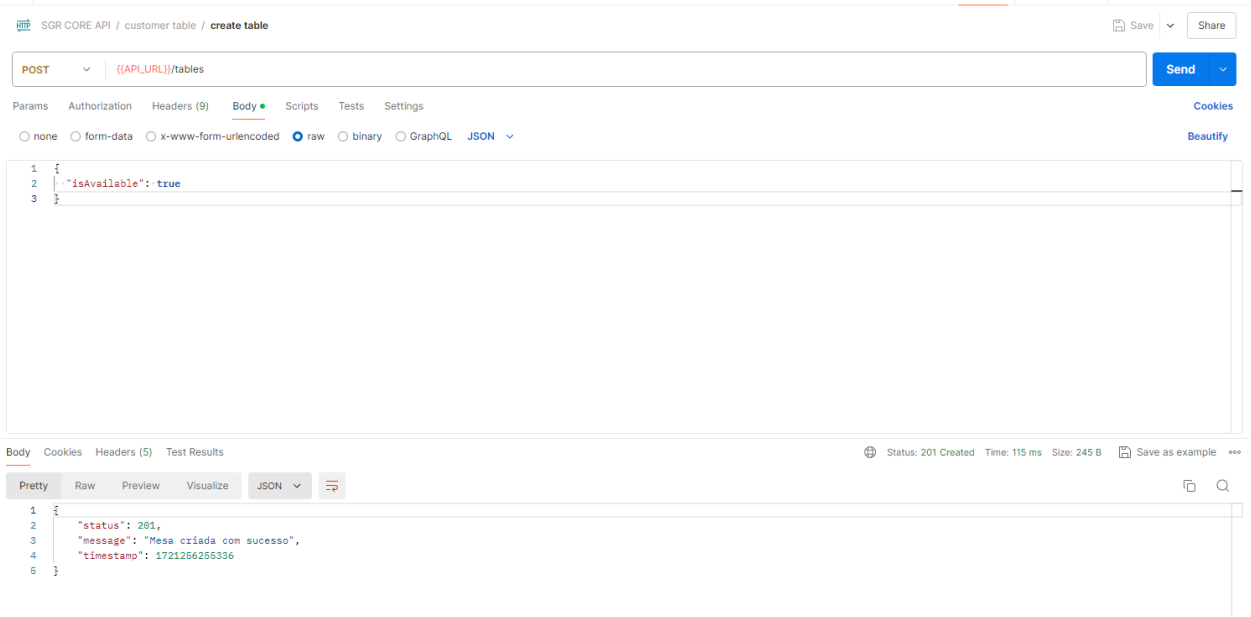
	}] } }	null, "wholeQuantity": 0 }] } }



8. Create a customer table

INPUT	EXPECTED OUTPUT	OUTPUT
{ "isAvailable": true	{ "status": 201,	{ "status": 201,

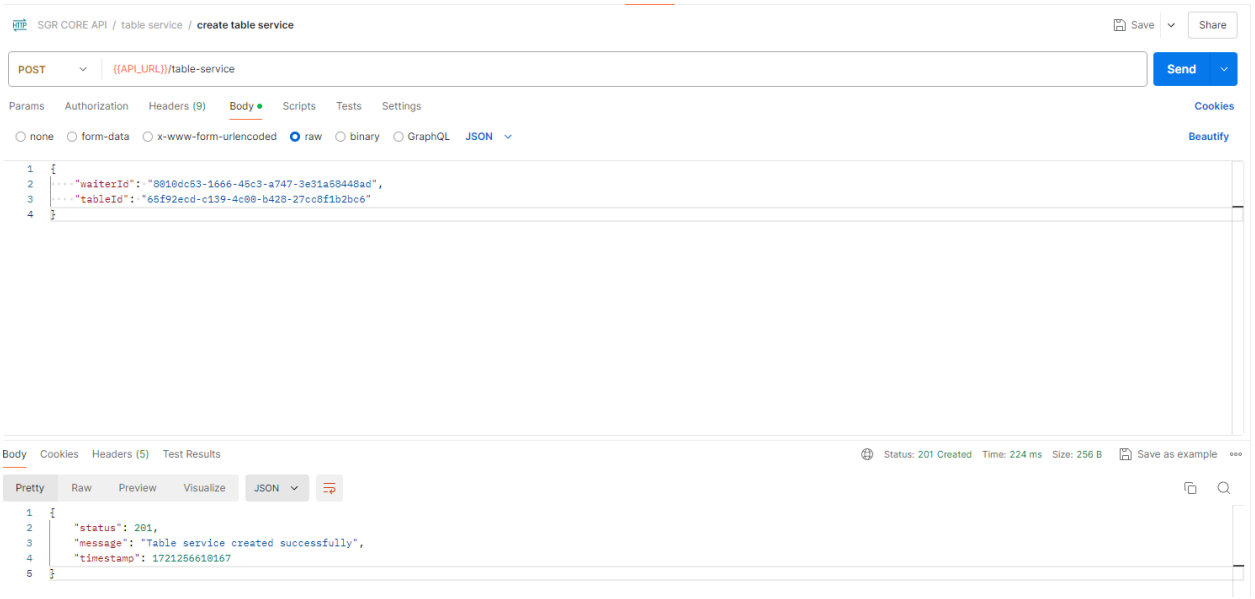
}	"message": "Mesa criada com sucesso", "timestamp": 1721256255336 }	"message": "Mesa criada com sucesso", "timestamp": 1721256255336 }



9. Create a table service for the customer table

INPUT	EXPECTED OUTPUT	OUTPUT
{ "waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad", "tableId":	{ "status": 201, "message": "Table service created successfully",	{ "status": 201, "message": "Table service created successfully",

"65f92ecd-c139-4c00-b428-27cc8f1b2bc6" }	"timestamp": 1721256610167 }	"timestamp": 1721256610167 }

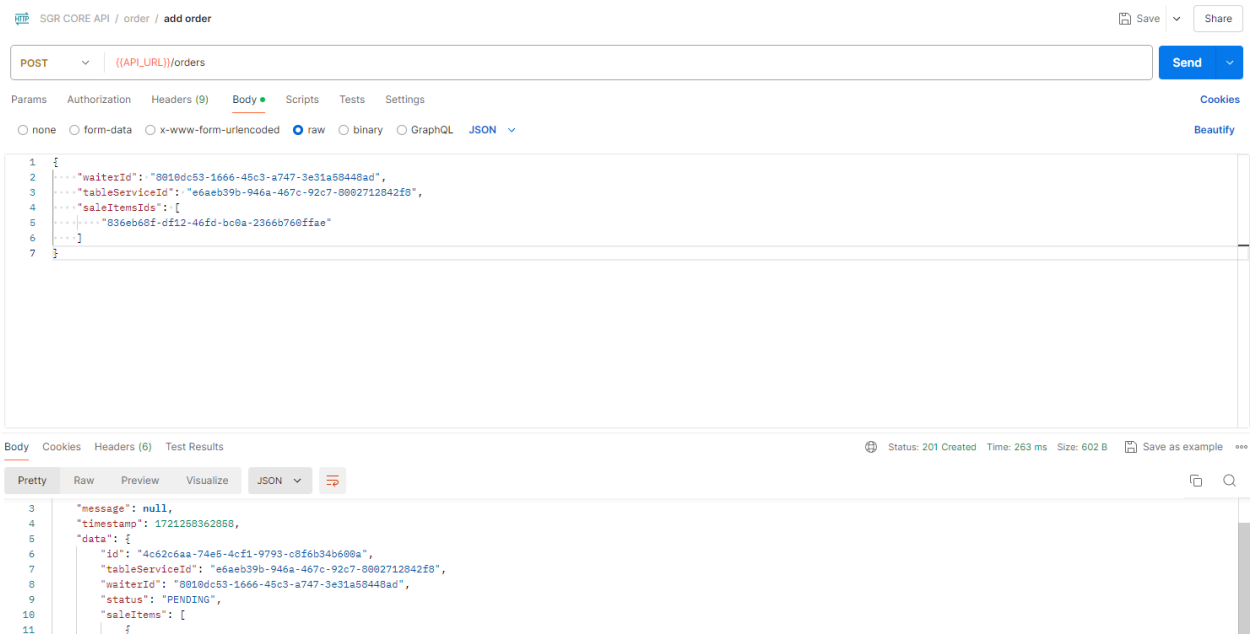


10. Order for a Peanut Butter Snack.

INPUT	EXPECTED OUTPUT	OUTPUT

{
 	
"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
 	
"tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8",
 	
"saleItemsIds": [
	{
	
	"status": 201,
	
	"message": **null**,
	
	"timestamp": 1721256793003,
	
	"data": {
	
	"id": "288e1992-f9da-47df-b3e3-a
		{
		
		"status": 201,
		
		"message": **null**,
		
		"timestamp": 1721256793003,
		
		"data": {
		
		"id": "288e1992-f9da-47df-b3e3-a

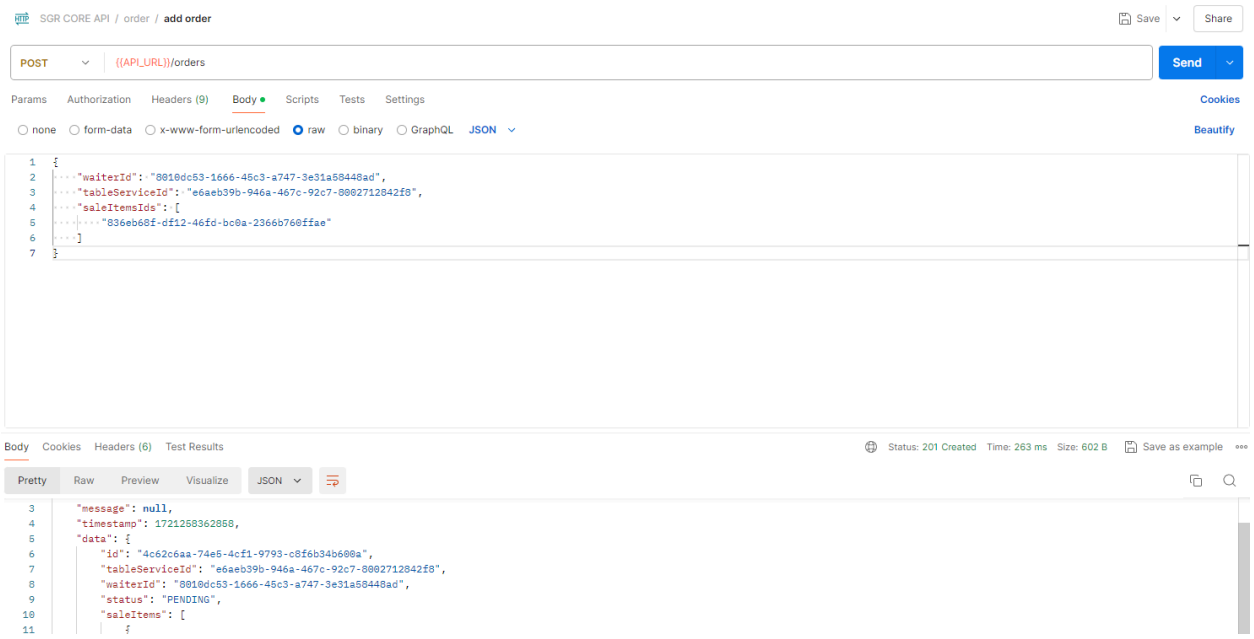
<table>
<tr>
<td>"836eb68f-df12-46fd-bc0a-2366b760ffae"
]
}</td>
<td>ffe0cd7ce97",
"tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8",
"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
"status": "PENDING",
"saleItems": [
{
"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
"name": "Paçoca",
"price": 1.0
}
],
"price": 1.0,
"priceCurrency": "BRL"
}
}</td>
<td>ffe0cd7ce97",
"tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8",
"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
"status": "PENDING",
"saleItems": [
{
"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
"name": "Paçoca",
"price": 1.0
}
],
"price": 1.0,
"priceCurrency": "BRL"
}
}</td>
</tr>
</table>



11. Order for a Peanut Butter Snack.

INPUT	EXPECTED OUTPUT	OUTPUT
{ "waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad", "tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8", "saleItemsIds": ["836eb68f-df12-46fd-bc0a-2366b760ffae"]	{ "status": 201, "message": null, "timestamp": 1721258362858, "data": { "id": "4c62c6aa-74e5-4cf1-9793-c8f6b34b600a", "tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8",	{ "status": 201, "message": null, "timestamp": 1721258362858, "data": { "id": "4c62c6aa-74e5-4cf1-9793-c8f6b34b600a", "tableServiceId": "e6aeb39b-946a-467c-92c7-8002712842f8",

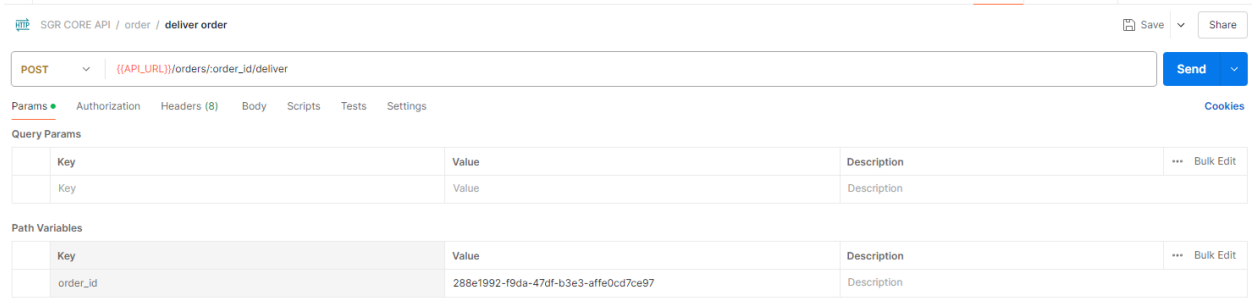
}	"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
	"status": "PENDING",
	"saleItems": [
	{
	"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
	"name": "Paçoca",
	"price": 1.0
	}
	],
	"price": 1.0,
	"priceCurrency": "BRL"
	}
	}
		"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
		"status": "PENDING",
		"saleItems": [
		{
		"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
		"name": "Paçoca",
		"price": 1.0
		}
		],
		"price": 1.0,
		"priceCurrency": "BRL"
		}
		}

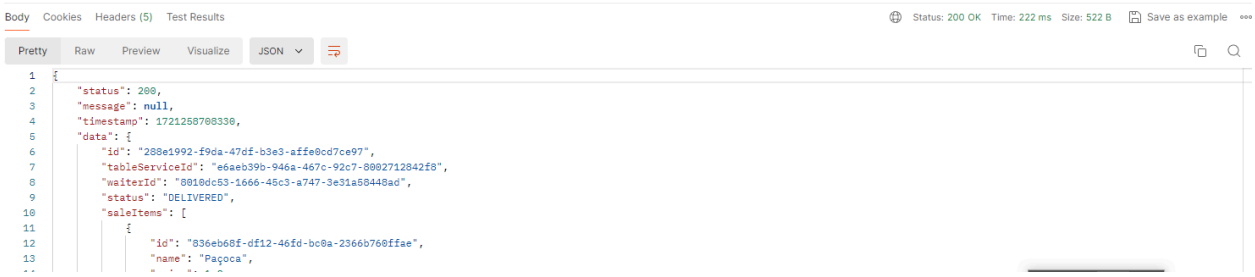


12. Deliver the first order

INPUT	EXPECTED OUTPUT	OUTPUT
{ "order_id": "288e1992-f9da-47df-b3e3-affe0cd7ce97" }	{ "status": 200, "message": null, "timestamp": 1721258708330, "data": { "id": "288e1992-f9da-47df-b3e3-affe0cd7ce97", "tableServiceId":	{ "status": 200, "message": null, "timestamp": 1721258708330, "data": { "id": "288e1992-f9da-47df-b3e3-affe0cd7ce97", "tableServiceId":

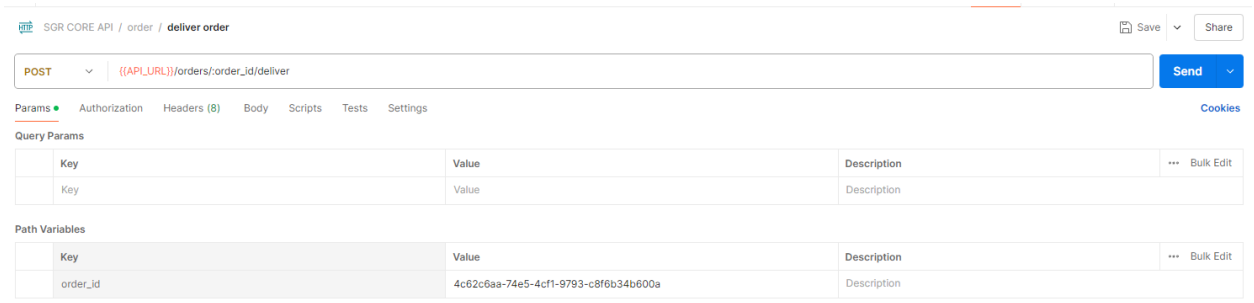
	"e6aeb39b-946a-467c-92c7-8002712842f8",
	"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
	"status": "DELIVERED",
	"saleItems": [
	{
	"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
	"name": "Paçoca",
	"price": 1.0
	}
	],
	"price": 1.0,
	"priceCurrency": "BRL"
	}
	}
		"e6aeb39b-946a-467c-92c7-8002712842f8",
		"waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad",
		"status": "DELIVERED",
		"saleItems": [
		{
		"id": "836eb68f-df12-46fd-bc0a-2366b760ffae",
		"name": "Paçoca",
		"price": 1.0
		}
		],
		"price": 1.0,
		"priceCurrency": "BRL"
		}
		}

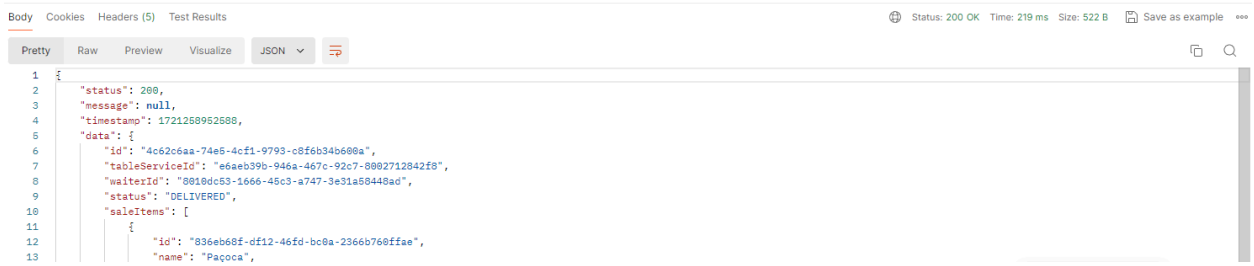




13. Deliver the second order

INPUT	EXPECTED OUTPUT	OUTPUT
{ "waiterId": "8010dc53-1666-45c3-a747-3e31a58448ad", "tableId": "65f92ecd-c139-4c00-b428-27cc8f1b2bc6" }	{ "status": 201, "message": "Table service created successfully", "timestamp": 1721256610167 }	{ "status": 201, "message": "Table service created successfully", "timestamp": 1721256610167 }





14. Close table service

INPUT	EXPECTED OUTPUT	OUTPUT
{ "tableId": "e6aeb39b-946a-467c-92c7-8002712842f8" }	{ "status": 200, "message": "Atendimento finalizado", "timestamp": 1721263792812 }	{ "status": 200, "message": "Atendimento finalizado", "timestamp": 1721263792812 }

FINAL CONCLUSION

Based on the results obtained, we can conclude that the use of advanced testing and quality techniques is extremely important and facilitates the systems development process. This work contributes to our academic and professional training by giving us the chance to have in-depth and detailed contact with an area that is so important but so little valued in technology today.

During our testing process, among the many techniques available, we appreciated exploratory testing for its practicality. It allowed us to execute our tests quickly and automatically prompted us to ask: “what if we did this?” This intuition guided us to explore other classes and develop a kind of instinct for finding bugs. On the other hand, the decision table was the most challenging in the context of our system because we had to identify the actions and conditions in complete isolation, disregarding events involving third-party objects.

We would again like to thank Professor Mohamad Kassab for teaching the classes and giving us some of his time. We would also like to thank Professor Valdemar Graciano for his guidance and for taking the initiative to enable us to have this contact with people from all over the world - it was a valuable experience that we will certainly carry in our hearts forever.