

## PUNTO 1

- Plan de Pruebas

- ✓ Alcance

El alcance que tendremos dentro de las pruebas a realizar sobre la aplicación web <https://www.advantageonlineshopping.com/> se enfocará en el core del negocio en donde se realizan pruebas seleccionando productos y adicionándolos al carrito de compras, (específicamente se tendrán en cuenta los productos de la pestaña popular ítems) sobre los cuales realizaremos el proceso de compra y validaremos que lo que se muestre en el carrito de compra corresponda correctamente a los ítems adicionados, teniendo en cuenta las características de los mismos (Color y Cantidad).

Se implementarán pruebas automáticas end to end en donde se validará el proceso core del negocio, adicionando un item popular al carrito de compras, cambiándole el color y validando que lo que muestre el carrito de compras corresponda a lo ingresado.

Adicionalmente se revisarán los tiempos de respuesta de la aplicación al momento de realizar los siguientes procesos:

Login

Registro

Selección de Producto

Adición de un Producto al Carrito

Modificar características de un Producto

Comprar lo ingresado en el carrito

Escenarios o funcionalidades que no se encuentren relacionadas dentro del alcance no se tendrán en cuenta para su verificación en el proceso de pruebas que se está implementando.

- ✓ Objetivos

- Implementar pruebas funcionales de software en donde aseguremos la calidad del producto en las funcionalidades definidas
    - Desarrollar pruebas de software automáticas para asegurar la calidad en el funcionamiento core de la aplicación.
    - Garantizar el funcionamiento de la aplicación el cual sea acorde a lo esperado por el usuario

- ✓ Tipos de Pruebas

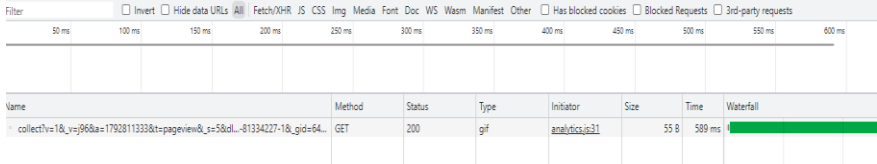
Teniendo en cuenta los Cuadrantes de pruebas, en esta prueba técnica emplearemos las siguientes pruebas:

- Q1 – No se implementarán pruebas en este cuadrante

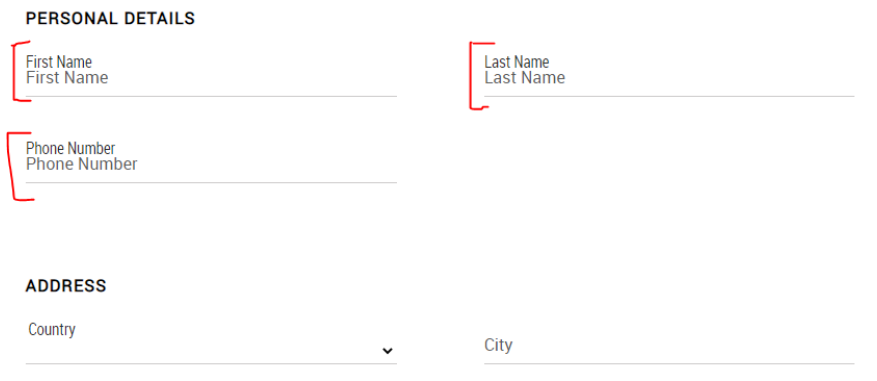
- Para el reporte de los defectos, se realizará directamente desde este documento, en donde se creará una plantilla con todos los elementos requeridos para reportar un defecto: (ID, Titulo, Descripcion, Resultados esperados, Resultados Obtenidos, Pasos para Reproducir el Defecto, Imágenes o Evidencias, Severidad). Para este tipo de reportes lo ideal seria usar una herramienta que permita hacer el control completo y llevar la trazabilidad como un Jira, Mantis o incluso un Bugtracker como tal.

ISSUE-001

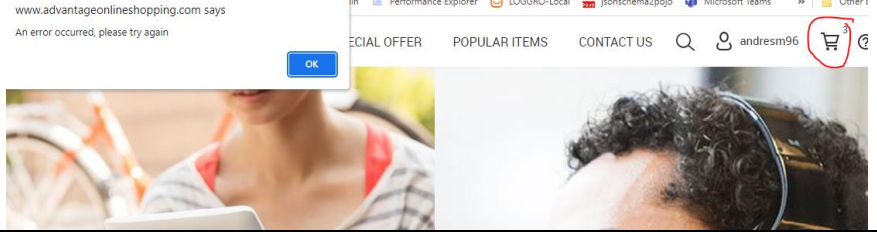
Título	Error de Performance al acceder a un Popular Item la Primera Vez																																								
Descripción	Al intentar acceder a un Popular item la primera vez la aplicación esta tardando mas de 25 segundos en cargar la información del producto, las siguientes veces que se accede carga la información en menos de 1 segundos.																																								
	<div>Primera Vez</div> <table><thead><tr><th>Name</th><th>Method</th><th>Status</th><th>Type</th><th>Initiator</th><th>Size</th><th>Time</th><th>Waterfall</th></tr></thead><tbody><tr><td> all_data</td><td>GET</td><td>200</td><td>xhr</td><td>main.min.js:94</td><td>9.2 kB</td><td>30.07 s</td><td></td></tr><tr><td> 10</td><td>GET</td><td>200</td><td>xhr</td><td>main.min.js:94</td><td>847 B</td><td>7.04 s</td><td></td></tr><tr><td> products</td><td>GET</td><td>200</td><td>xhr</td><td>main.min.js:94</td><td>3.5 kB</td><td>16.18 s</td><td></td></tr><tr><td> product-name.html</td><td>GET</td><td>104</td><td>xhr</td><td>main.min.js:94</td><td>178 B</td><td>584 ms</td><td></td></tr></tbody></table>	Name	Method	Status	Type	Initiator	Size	Time	Waterfall	all_data	GET	200	xhr	main.min.js:94	9.2 kB	30.07 s		10	GET	200	xhr	main.min.js:94	847 B	7.04 s		products	GET	200	xhr	main.min.js:94	3.5 kB	16.18 s		product-name.html	GET	104	xhr	main.min.js:94	178 B	584 ms	
Name	Method	Status	Type	Initiator	Size	Time	Waterfall																																		
all_data	GET	200	xhr	main.min.js:94	9.2 kB	30.07 s																																			
10	GET	200	xhr	main.min.js:94	847 B	7.04 s																																			
products	GET	200	xhr	main.min.js:94	3.5 kB	16.18 s																																			
product-name.html	GET	104	xhr	main.min.js:94	178 B	584 ms																																			

	<p>Segunda Vez</p> 
Severidad	Media

#### ISSUE-002

Titulo	Error Visual en los campos no obligatorios al momento del registro
Descripción	Al ubicar el cursor sobre los campos no obligatorios y hacer clic fuera de ellos en la pagina de registro, se esta duplicando el label del nombre del campo
	
Severidad	Baja

#### ISSUE-003

Titulo	Se presenta un error al intentar acceder al carrito de compras luego de loguearse
Descripción	Se realiza primero la adicion de los productos al carrito de compras y luego se intenta loguear en la aplicación, cuando se loguea y se hace clic en el carrito para finalizar la compra se presenta un mensaje de error que no deja avanzar con el proceso
	
Severidad	Alto

#### ISSUE-004

Titulo	Al refrescar la pagina se pierde el login realizado
Descripción	Se refresca la pagina luego de haberse logueado y al acceder al proceso de pago en el carrito, envia nuevamente a hacer login en la aplicación

	estando ya logueado (En la parte superior derecha de la imagen se muestra que el usuario ya esta logueado)
Severidad	Alto

- Casos de Prueba

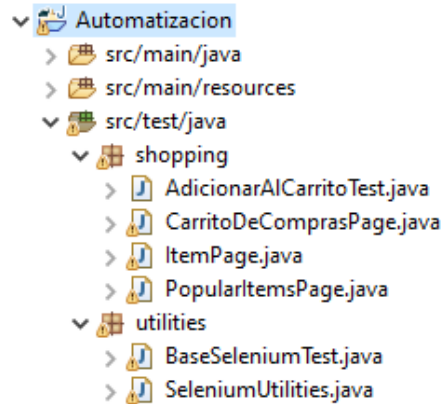
Para la creación de casos de prueba, se realizará directamente desde este documento, en donde se creará una plantilla con todos los elementos requeridos para crear un caso de prueba: (ID, Descripción, Precondiciones, Pasos a Ejecutar, Resultado Esperado).

Para estas actividades de diseño es ideal tener una herramienta que permita hacer el control completo de las ejecuciones de los mismos, se puede usar TestRail, Zhepyr, TestCaseLab entre otros.

- Automatización de Pruebas

Para la automatización de pruebas se tomaron en cuenta escenarios importantes relacionados con la selección y adición de productos al carrito de ventas y se realizó de la siguiente forma:

- Creación de un proyecto tipo Gradle
- Framework de automatización de pruebas UI “Selenium” (Utilizado Principalmente para realizar escenarios de regresion o criticos para el core del negocio)
- Framework de ejecución de pruebas JUnit
- Arquitectura del proyecto de pruebas POM “Page Object Model”, tambien se pudo utilizar una Screenplay ya que el core del negocio lo permite



Adicionalmente en estos test Automáticos estamos utilizando una patrón denominado AAA, en la cual se organiza el código de la siguiente forma:

- ✓ Arrange: Inicializa los objetos y preparamos los valores de los datos que vamos a utilizar en el Test que lo contiene.
- ✓ Act: Realizamos las acciones o llamados a métodos que realizan la prueba.
- ✓ Assert: Se realizan todas las validaciones que requerimos para garantizar que la prueba fue exitosa.

```
public class AdicionarAlCarritoTest extends BaseSeleniumTest {  
  
    @Test  
    /**  
     * Test que valida que se permita adicionar un item de otro color al carrito  
     */  
    public void adicionarTabletOtroColor() {  
        PopularItemsPage page = new PopularItemsPage(driver);  
        page.clickPopularItems();  
        page.clickDetallesPrimerItem();  
  
        ItemPage itemPage = new ItemPage(driver);  
        String nameItem = itemPage.getNameItem();  
        String price = itemPage.getPriceItem();  
        String color = "GRAY";  
        String cantidad = "1";  
        itemPage.setQuantityItem(cantidad);  
        itemPage.setColorItem(color);  
        itemPage.clickAddToCar();  
  
        page.irAlCarritoButton();  
  
        CarritoDeComprasPage carritoPage = new CarritoDeComprasPage(driver);  
        carritoPage.getPriceItem();  
  
        assertTrue(nameItem.equals(carritoPage.getNameItem()));  
        assertTrue(color.equals(carritoPage.getColorItem()));  
        assertTrue(cantidad.equals(carritoPage.getQuantityItem()));  
        assertTrue(price.equals(carritoPage.getPriceItem()));  
    }  
}
```

En nuestro caso el Arrange lo hacemos en la clase de la cual extendemos BaseSeleniumTest, en ella preparamos todo lo que tiene que ver con el navegador y navegamos a la pagina en la que realizaremos la prueba.

Dentro del método de prueba realizamos nuestro Act en donde consumimos los métodos que tenemos en los Pages interactuando con la aplicación, y finalmente en la parte inferior de los test tenemos nuestros métodos de validacion en donde cumplimos con la parte del Assert.

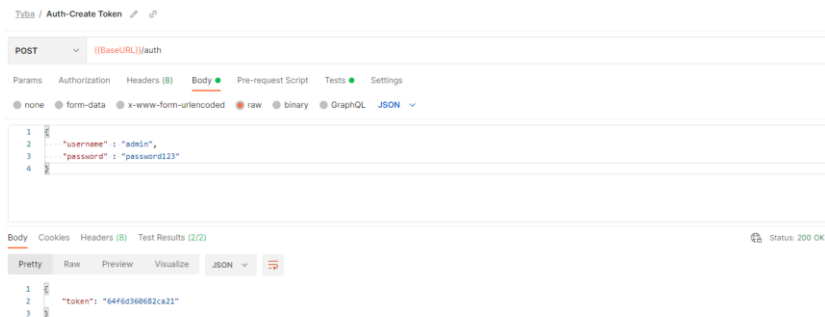
- Ejecución Casos

## PUNTO 2

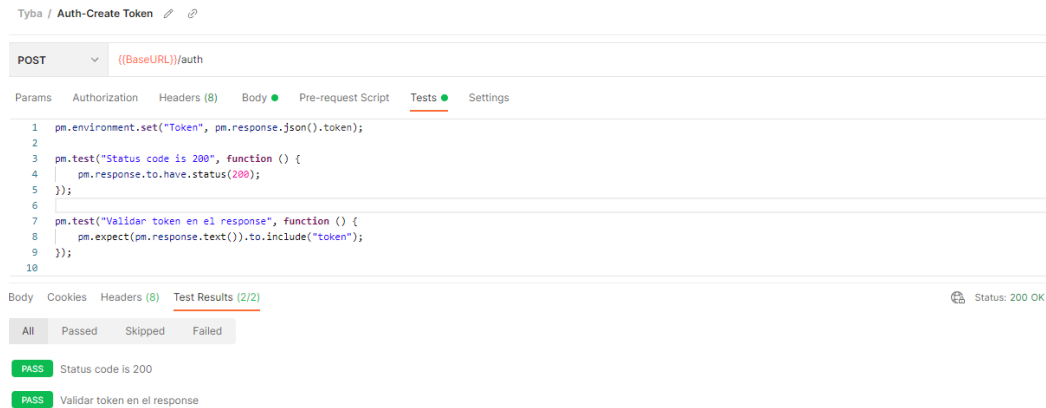
- Setup (Creación de variables de ambiente utilizadas durante los test, las cuales permiten que las pruebas se puedan ejecutar siempre que se deseen sin tener que modificar datos en los test manualmente) adicionalmente permiten facilidad de ejecución de pruebas en diferentes ambientes

Booking				
	VARIABLE	TYPE ①	INITIAL VALUE ①	CURRENT VALUE ①
<input checked="" type="checkbox"/>	BaseUrl	default	https://restful-booker.herokuapp.com	https://restful-booker.herokuapp.com
<input checked="" type="checkbox"/>	Token	default		64f6d360682ca21
<input checked="" type="checkbox"/>	firstname	default	Andres	Andres
<input checked="" type="checkbox"/>	lastname	default	Gonzalez	Gonzalez
<input checked="" type="checkbox"/>	totalprice	default	500	500
<input checked="" type="checkbox"/>	depositpaid	default	true	true
<input checked="" type="checkbox"/>	checkin	default	2018-01-01	2018-01-01
<input checked="" type="checkbox"/>	checkout	default	2018-01-30	2018-01-30
<input checked="" type="checkbox"/>	additionalneeds	default	Breakfast	Breakfast
<input checked="" type="checkbox"/>	bookingid	default		5815
	Add a new variable			

- Creación de token



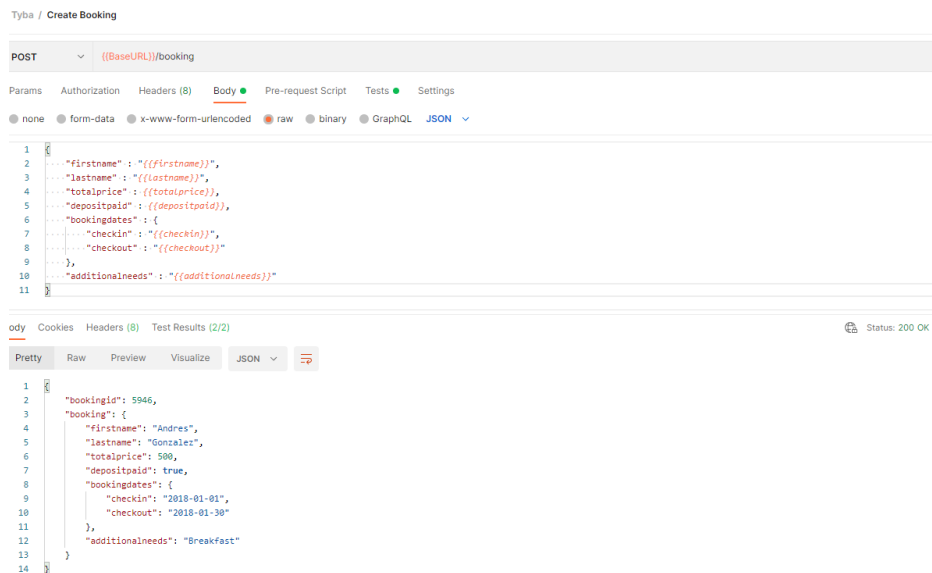
Se utiliza una variable de ambiente para guardar el token generado y usarlo automáticamente en los otros servicios.



Adicionalmente se hacen test para validar el status code y que exista un token en el response.

- Crear Booking

Se ejecuta el test de creación del Booking con los datos previamente almacenados en las variables de ambiente



En esta caso se utiliza el Bookingid para guardarlo en una variable y utilizarlo posteriormente en los servicios de consulta, actualización y eliminación de tal forma que permita ser reutilizable la collection.

POST ▼ {{BaseURL}}/booking

Params Authorization Headers (8) Body ● Pre-request Script Tests ● Settings

```

1 pm.environment.set("bookingid", pm.response.json().bookingid);
2
3 pm.test("Status code is 200", function () {
4   pm.response.to.have.status(200);
5 });
6
7 pm.test("Validar booking creado", function () {
8   pm.expect(pm.response.json().booking.firstname).to.eql(pm.environment.get("firstname"));
9   pm.expect(pm.response.json().booking.lastname).to.eql(pm.environment.get("lastname"));
10  pm.expect(pm.response.json().booking.totalprice.toString()).to.eql(pm.environment.get("totalprice"));
11  pm.expect(pm.response.json().booking.depositpaid.toString()).to.eql(pm.environment.get("depositpaid"));
12  pm.expect(pm.response.json().booking.bookingdates.checkin).to.eql(pm.environment.get("checkin"));
13  pm.expect(pm.response.json().booking.bookingdates.checkout).to.eql(pm.environment.get("checkout"));
14  pm.expect(pm.response.json().booking.additionalneeds).to.eql(pm.environment.get("additionalneeds"));
15 });

```

Body Cookies Headers (8) Test Results (2/2) 🔍 Status: 200 OK 1

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1 {
2   "bookingid": 5946,
3   "booking": {
4     "firstname": "Andres",
5     "lastname": "Gonzalez",
6     "totalprice": 500,
7     "depositpaid": true,
8     "bookingdates": {
9       "checkin": "2018-01-01",
10      "checkout": "2018-01-30"
11    },
12    "additionalneeds": "Breakfast"
13  }
14 }

```

En la creación se valida el response contra los datos ingresados, adicionalmente del status code de la respuesta del servicio.

Se encuentran los siguientes Issues:

- ✓ En el servicio de creación no se esta solicitando el token de seguridad, este tipo de servicios debe tener seguridad.

<https://restful-booker.herokuapp.com/booking>

- ✓ En el servicio de creación no se tiene validación de fechas, permite entonces tener una fecha de chekout inferior a la fecha del checkin. Hay escenarios que requieren tener una doble validación, back y front.

POST ▼ {{BaseURL}}/booking

Params Authorization Headers (8) Body ● Pre-request Script Tests ● Settings

none form-data x-www-form-urlencoded raw ● binary GraphQL JSON ▼

```

1 {
2   "firstname": "{{firstname}}",
3   "lastname": "{{lastname}}",
4   "totalprice": {{totalprice}},
5   "depositpaid": {{depositpaid}},
6   "bookingdates": {
7     "checkin": "{{checkin}}",
8     "checkout": "2017-01-30"
9   },
10   "additionalneeds": "{{additionalneeds}}"
11 }

```

Body Cookies Headers (8) Test Results (1/2) 🔍 Status: 200 OK

Pretty Raw Preview Visualize JSON ▼ 🔍

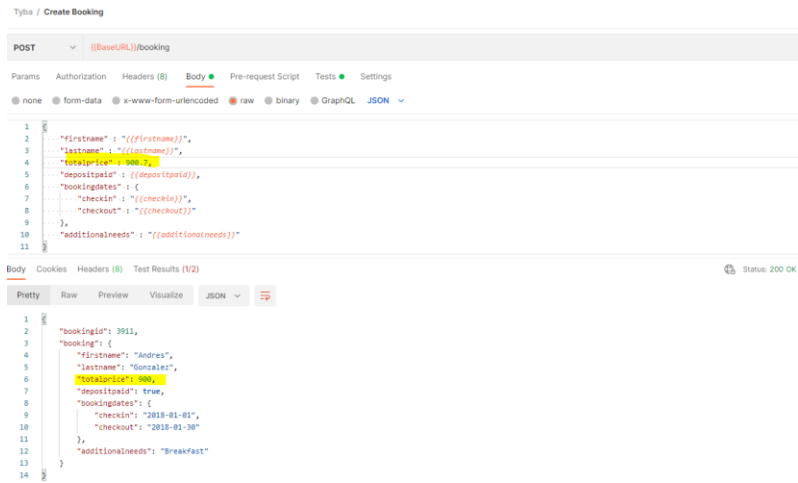
```

1 {
2   "bookingid": 3674,
3   "booking": {
4     "firstname": "Andres",
5     "lastname": "Gonzalez",
6     "totalprice": 500,
7     "depositpaid": true,
8     "bookingdates": {
9       "checkin": "2018-01-01",
10      "checkout": "2017-01-30"
11    },
12    "additionalneeds": "Breakfast"
13  }
14 }

```

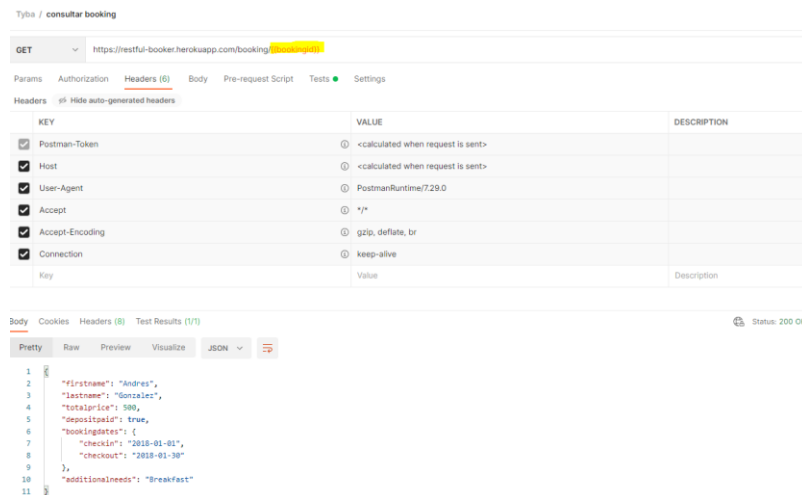


- ✓ En el servicio de creación cuando se envía un totalprice el cual contiene decimales, no respeta este valor sino que almacena en el booking un valor entero. Se debe revisar el requerimiento para validar la necesidad



- Consultar Booking

Se realiza consumo del servicio de consulta de booking en la cual utilizamos el Bookingid devuelto en el servicio de creación.



Se valida que la data que responda el servicio sea igual a los datos con los cuales se creó el booking, adicionalmente de validar el status code del response.

Tyba / consultar booking

GET https://restful-booker.herokuapp.com/booking/{bookingid}

Params Authorization Headers (6) Body Pre-request Script Tests Settings

```

1 pm.test("Validar booking creado previamente", function () {
2   pm.expect(pm.response.json().firstname).to.eql(pm.environment.get("firstname"));
3   pm.expect(pm.response.json().lastname).to.eql(pm.environment.get("lastname"));
4   pm.expect(pm.response.json().totalprice).to.eql(pm.environment.get("totalprice"));
5   pm.expect(pm.response.json().depositpaid).to.eql(pm.environment.get("depositpaid"));
6   pm.expect(pm.response.json().bookingdates.checkin).to.eql(pm.environment.get("checkin"));
7   pm.expect(pm.response.json().bookingdates.checkout).to.eql(pm.environment.get("checkout"));
8   pm.expect(pm.response.json().additionalneeds).to.eql(pm.environment.get("additionalneeds"));
9 });
10
11
12 pm.test("Status code is 200", function () {
13   pm.response.to.have.status(200);
14 });

```

Body Cookies Headers (8) Test Results (2/2) Status: 200 OK

All Passed Skipped Failed

PASS Validar booking creado previamente

PASS Status code is 200

Se valida que el servicio solo retorne valores cuando se consulta un booking existente

Tyba / consultar booking invalido

GET https://restful-booker.herokuapp.com/booking/ier

Params Authorization Headers (6) Body Pre-request Script Tests Settings

```

1 pm.test("Status code is 400, Invalid booking", function () {
2   pm.response.to.have.status(400);
3 });

```

Body Cookies Headers (8) Test Results (1/1) Status: 404 Not Found

All Passed Skipped Failed

PASS Status code is 400, Invalid booking

Se encuentran los siguientes Issues:

- ✓ En el servicio de consulta no se está solicitando el token de seguridad, este tipo de servicios debe tener seguridad.
- Se anexa servicio al reporte hecho previamente.

## • Actualizar Booking

Se realiza consumo del servicio de actualización de booking en la cual utilizamos el Bookingid devuelto en el servicio de creación y modificamos todos los datos para validar que efectivamente se actualice todo correctamente.

Tyba / update booking

PUT https://restful-booker.herokuapp.com/booking/{bookingid}

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "firstname": "James",
3   "lastname": "Brown",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2018-01-01",
8     "checkout": "2019-01-01"
9   },
10  "additionalneeds": "Lunch"
11 }

```

Body Cookies Headers (8) Test Results (1/1) Status: 200 OK

Pretty Raw Preview Visualize JSON

```

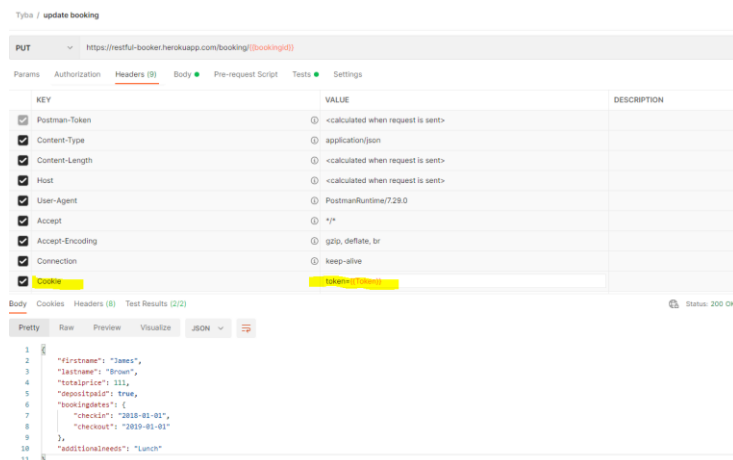
1 {
2   "firstname": "James",
3   "lastname": "Brown",
4   "totalprice": 111,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2018-01-01",
8     "checkout": "2019-01-01"
9   },
10  "additionalneeds": "Lunch"
11 }

```

Se crean test que validen que la información del response corresponda a la información modificada.



Se valida correctamente que valide el token de seguridad creado previamente



Se realizan validaciones del tamaño de los campos y el control sobre esto, es importante para no tener un costo muy alto en base de datos o validar que los servicios no respondan información relevante en la estructura (Se debe validar los valores limites de acuerdo al requerimiento o historia de usuario y validar que efectivamente cumplan con los tamaños permitidos)

Tyba / update booking

PUT https://restful-booker.herokuapp.com/booking/{bookingId}

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

3 "lastName": "Brown",

Body Cookies Headers (8) Test Results Status: 413 Payload Too Large

Pretty Raw Preview Visualize Text

1 Payload Too Large

- Eliminar Booking

Se realiza el consumo del servicio de eliminación de booking creado previamente, para esto se utiliza el bookingId devuelto en el servicio de creación.

Tyba / delete booking

DELETE https://restful-booker.herokuapp.com/booking/{bookingId}

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Headers Hide auto-generated headers

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.0	
<input checked="" type="checkbox"/> Accept	*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> Cookie	token={Token}	
Key	Value	Description

Jody Cookies Headers (8) Test Results Status: 201 Created

Pretty Raw Preview Visualize Text

1 Created

Se realiza prueba en la cual se intenta eliminar nuevamente el mismo booking que ya se había eliminado.

Tyba / delete booking No existente

DELETE ▼ https://restful-booker.herokuapp.com/booking/((bookingid))

Params Authorization Headers (7) Body Pre-request Script Tests ● Settings

```
1 pm.test("Status code is 405 - Eliminar booking que no exista", function () {
2   pm.response.to.have.status(405);
3 });
```

Body Cookies Headers (8) Test Results (1/1) 🔍 Status: 405 Method Not Allowed

Pretty Raw Preview Visualize Text ▼ 🔍

1 Method Not Allowed

Se valida que el servicio requiera el token para poder realizar la eliminación del booking

Tyba / delete booking sin token

DELETE ▼ https://restful-booker.herokuapp.com/booking/((bookingid))

Params Authorization Headers (7) Body Pre-request Script Tests ● Settings

```
1 pm.test("Status code is 403 - Eliminar booking sin usar token", function () {
2   pm.response.to.have.status(403);
3 });
```

Body Cookies Headers (8) Test Results (1/1) 🔍 Status: 403 Forbidden

Pretty Raw Preview Visualize Text ▼ 🔍

1 Forbidden

## EJECUCION DE LOS TEST

- ✓ Se realiza ejecución de toda la colección y se valida que corran todos los test correctamente.

RUN ORDER Deselect All Select All Reset

**Run Settings**

Iterations:

Delay:  ms

Data:

Advanced settings

- ☐ Save responses ⓘ
- ☒ Keep variable values ⓘ
- ☐ Run collection without using stored cookies
- ☒ Save cookies after collection run ⓘ

Tyba Booking, Today, 7:41 am

RUN SUMMARY

	1
▶ <b>POST</b> Auth-Create Token	2   0
▶ <b>POST</b> Create Booking	2   0
▶ <b>GET</b> consultar booking	2   0
▶ <b>GET</b> consultar booking invalido	1   0
▶ <b>PUT</b> update booking	2   0
▶ <b>PUT</b> update booking - tamaños largos	1   0
▶ <b>DELETE</b> delete booking	1   0
▶ <b>DELETE</b> delete booking No existente	1   0
▶ <b>DELETE</b> delete booking sin token	1   0

- ✓ Adicionalmente se valida la ejecución que corra en varias iteraciones para garantizar que la prueba si sea mantenible y reutilizable con todos los datos cargados en el setup

**Run Settings**

Iterations:

Delay:  ms

Data:

Advanced settings

- ☐ Save responses ⓘ
- ☒ Keep variable values ⓘ
- ☐ Run collection without using stored cookies
- ☒ Save cookies after collection run ⓘ

RUN SUMMARY

		1	2	3	4	5
▶ <b>POST</b> Auth-Create Token	10	0				
▶ <b>POST</b> Create Booking	11	0				
▶ <b>GET</b> consultar booking	10	0				
▶ <b>GET</b> consultar booking invalido	5	0				
▶ <b>PUT</b> update booking	10	0				
▶ <b>PUT</b> update booking - tamaños largos	5	0				
▶ <b>DELETE</b> delete booking	5	0				
▶ <b>DELETE</b> delete booking No existente	5	0				
▶ <b>DELETE</b> delete booking sin token	5	0				