



# **Equipo 4**

## **Reporte Final de Testing**

### ***Digital Booking***

**Integrantes:**

- Ingrid Banguero
- Juan Pablo Riaño
- Tatiana Zanelli
- Daniel Orozco



## Contenido

<b>INTRODUCCIÓN .....</b>	<b>3</b>
Resumen de las actividades de prueba.....	3
<b>ALCANCE.....</b>	<b>5</b>
Dentro del Alcance.....	5
Fuera de Alcance .....	7
Tipos de Pruebas Ejecutadas .....	8
Enfoque de la Prueba .....	8
<b>EXIT CRITERIA .....</b>	<b>10</b>
<b>RESUMEN DE RESULTADOS.....</b>	<b>11</b>
Diseño de Pruebas .....	11
Ejecución de Pruebas.....	12
Ejecución Manual.....	12
Ejecución Automática.....	13
<b>REPORTE DE DEFECTOS .....</b>	<b>17</b>
Todos los defectos .....	17
<b>CONCLUSIÓN .....</b>	<b>18</b>



## Introducción

Este documento es el Informe Final de Pruebas del sistema Digital Booking. El propósito de este documento es proveer evidencia de que el Exit Criteria para el proceso de Testing se cumplió y, por lo tanto, se concluye la fase de pruebas y puede cerrarse. Se demuestra que los Issues de GitLab relacionados con testing fueron implementados desde los Sprint 1 a 4. Este documento va a ser utilizado como entrada para la revisión general de las actividades de prueba y para tomar la decisión si el sistema cumple con las expectativas.

## Resumen de las actividades de prueba

- **Resumen del sistema**

Digital Booking es un sitio web que nos permite realizar reservas de alojamientos categorizados en hoteles, hostels, departamentos y bed & breakfast en distintos lugares turísticos del mundo. El sitio puede ser utilizado por usuarios para buscar alojamientos y realizar reservas, y también por administradores, que además pueden publicar sus productos. En la página principal de este sitio podrás buscar hospedaje filtrando por categoría, ciudad y fecha. Puedes conocer cada uno de nuestros inmuebles leyendo la descripción de este y viendo sus fotos de alta calidad. Registrándote en la página lograrás confirmar tu reserva y planificar tus vacaciones.

- **URL de acceso** → <http://0521ptc8n2-grupo4-frontend.s3-website.us-east-2.amazonaws.com/>

- **Tabla con Issues de GitLab y estado**

Issue	Descripción	Sprint	Estado
#12	Planificación y ejecución de los tests	1	CERRADO
#13	Testear las APIs	1	CERRADO
#42	Testeo Automatizado (JEST)	2	CERRADO
#41	Testeo Automatizado (Selenium)	2	CERRADO
#44	Testeo Automatizado (Postman)	2	CERRADO
#45	Testeo Exploratorio - Manual - Estático	2	CERRADO



#67	Implementar tests manuales	3	CERRADO
#68	Implementar tests automatizados	3	CERRADO
#76	Implementar testing automatizado	4	CERRADO
#77	Implementar testing manual	4	CERRADO
#78	Confeccionar el reporte final de testing	4	CERRADO



# Alcance

## Dentro del Alcance

### ► TESTING MANUAL:

- Registrarse en el sitio de manera exitosa: pruebas de humo positivas y negativas.
- Loguearse en el sitio de manera exitosa, se realizaron pruebas de humo positivas y negativas, y de regresión positiva.
- Al loguearse, en el header debe aparecer un avatar con el nombre del usuario: prueba de humo positiva
- Al loguearse el administrador, debe aparecer en el header un icono de administrador: prueba de humo positiva
- Al clickear el ícono de administrador debe dirigirse a la página de administración
- Comprobar que el sitio sea Responsive: prueba de humo positiva
- Filtrar producto por ciudad: prueba de humo positiva
- Comprobar que el Bloque de imágenes sea Responsive
- Abrir carrusel de imágenes: prueba de humo positiva
- Poder pasar las fotos en el carrusel de manera manual
- Visualizar cards de productos en el home (conexión API): prueba de regresión positiva
- Vuelta al home desde detalle de producto presionando logo: prueba de humo positiva
- Al hacer una reserva sin loguearse debe mostrar un aviso que diga que debemos loguearnos: prueba de humo negativa
- Para realizar una reserva todos los campos deben ser completados correctamente: pruebas de humo negativas y positivas
- Vuelta al home desde reserva presionando logo: prueba de humo positiva
- Verificar que un nuevo producto creado exitosamente aparezca en la página: prueba de regresión positiva
- Poder agregar varios atributos al momento de crear un nuevo producto: prueba de humo positiva
- Verificar que las url que cargamos al momento de crear un nuevo producto, sean visibles al ver el producto cargado: prueba de regresión positiva
- Verificar que las políticas del producto se carguen correctamente: prueba de regresión
- Verificar que no se pueda crear un nuevo producto si no se completaron todos los campos: prueba de humo negativa
- Confirmación de creación exitosa de un producto: prueba de humo positiva



- Verificar que el icono de administración nos dirija a la página de administración: test de humo positivo
- Vuelta al home desde formulario de creación de producto presionando logo: prueba de humo positiva
- Que no se pueda ingresar a la página de administración si nos logueamos como usuario: prueba de humo negativa

#### ► TESTING AUTOMÁTICO:

##### JEST

- Renderizado de los componentes relacionados al detalle de producto, el home, el detalle de las reservas, el footer y la página del administrador.

##### POSTMAN

- Crear un usuario (Test de tiempo de respuesta < 1000ms y verificación de status = 201).
- Autenticar el usuario para obtener un token (Test de tiempo de respuesta < 1000ms y verificación de status = 200).
- Crear una categoría. (Test de tiempo de respuesta < 1000ms, verificación de status = 200 y un response en el body = OK).
- Obtener un listado de todas las categorías. (Test de tiempo de respuesta < 300ms, verificación de status = 200)
- Obtener una categoría por ID. (Test de GET request exitoso y verificación de datos correctos).
- Crear una ciudad (Test de verificación de status = 201).
- Obtener un listado de todas las ciudades (Test de verificación de status = 200 y verificación de datos correctos).
- Crear una imagen (Test de verificación de status = 201 y verificación que el status code tenga un string).
- Obtener un listado de todas las imágenes. (Test de verificación de ID y verificación de datos correctos).
- Crear una característica. (Test de verificación de status = 201 y verificación de datos correctos).
- Obtener un listado de todas las características. (Test de verificación de status = 200 y verificación de ID).
- Obtener una característica por ID (Test de verificación de datos correctos).
- Obtener una característica por nombre (Test de verificación de status = 200 y verificación de datos correctos).
- Crear un producto (Test de status code = 201, tiempo de respuesta < 500ms y verificación que el status code tenga un string).



- Obtener un listado de todos los productos (Test de verificación de status = 200)
- Obtener un producto por ID (Test de verificación de status = 200 y verificación de datos correctos).
- Obtener productos por ciudad (Test de verificación de status = 200).
- Obtener productos por categoría (Test de verificación de status = 200).
- Obtener productos por ciudad y fecha (Test de verificación de status = 200).
- Obtener productos por fecha (Test de verificación de status = 200).
- Eliminar un producto. (Test de verificación de status = 200).
- Crear una reserva (Test de verificación de status = 201).
- Obtener un listado de todas las reservas (del usuario logueado) (Test de verificación de status = 200).
- Crear una política de cancelación.
- Crear una política de salud.
- Crear una política de normas de la casa.

## Fuera de Alcance

- Galería de imágenes.
- Que el componente de la galería se adaptara al número de imágenes del producto.



## Tipos de Pruebas Ejecutadas

Funcionalidades	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Prueba Estática	Si	SI	Si	Si
Prueba Exploratoria	Si	Si	Si	Si
Prueba de Sistema / Selenium IDE	No	No	Si	SI
Prueba de Humo	Si	Si	Si	Si
Prueba de Regresión	SI	Si	SI	Si
Prueba de Componente / Unidad (JEST)	No	No	No	Si
Prueba de Integración (Postman)	Si	Si	Si	Si

## Enfoque de la Prueba

Desde el área de Testing, comenzamos evaluando conjuntamente los problemas escritos en Gitlab, las historias de usuarios, el diseño en Figma y las épicas de diseño para asegurarnos de que haya coherencia entre ellos. En caso de discrepancias, se registraban para que el propietario pudiera ser consultado y resuelto de la mejor manera posible. Una vez resuelto el paso anterior, se construyen los casos de prueba positivos y negativos, guiándonos para hacerlo desde problemas e historias de usuarios de GitLab.

Durante el desarrollo de las características, los defectos se marcaron antes, lo que se reflejó en la tabla de defectos. Cuando publicamos la herramienta, se realizaron pruebas de regresión de humo y humo positivas y negativas para los casos de prueba desarrollados anteriormente. Durante este proceso se reportaban los defectos detectados y si eran de gravedad media/alta, Sprint los resolvía. Finalmente, se realizaron pruebas exploratorias para garantizar que cada problema requerido se desarrolló correctamente, de lo contrario, se informó como un error.





En cada nuevo Sprint, todos los casos de prueba, incluidos los de Sprints anteriores, se vuelven a ejecutar para verificar que ninguna funcionalidad se vea afectada por las funciones recién integradas. El mismo proceso se realiza con las pruebas exploratorias.

Esto se vio reflejado en la implementación de pruebas automatizadas en Postman. Cada endpoint contiene todos sus métodos de control en un solo archivo, y también tiene pruebas de buen estado, latencia y datos para verificar que están funcionando.

**Link Planilla de Casos de Prueba y Link Planilla de Defectos:**

[https://gitlab.ctd.academy/ctd/proyecto-integrador-1022/0521-pt-c8/grupo-04/-/blob/develop/testing/Grupo\\_4\\_-\\_Testing\\_Proyecto\\_Final\\_-\\_Sprint\\_IV.xlsx](https://gitlab.ctd.academy/ctd/proyecto-integrador-1022/0521-pt-c8/grupo-04/-/blob/develop/testing/Grupo_4_-_Testing_Proyecto_Final_-_Sprint_IV.xlsx)



## Exit Criteria

Se definió los siguientes criterios de aceptación para finalizar las pruebas:

- No se debe tener defectos en estado abierto de severidad alta (crítica y/o bloqueante) implicando a los que afectan la funcionalidad del sistema e impiden seguir trabajando
- No se debe tener más de un 25% de defectos en estado abierto de severidad media implicando a los que afectan la funcionalidad del sistema, pero no impiden seguir trabajando.
- Se pueden tener defectos en estado abierto de severidad baja ya que no afectan la funcionalidad del sistema.
- Con respecto a la ejecución de los tests de postman, acordamos que el passing rate debe ser mayor al 70%.



## Resumen de Resultados

### Diseño de Pruebas

<b>Funcionalidades</b>	<b>Test Manuales</b>	<b>Test Automáticos (Jest, Postman y Selenium)</b>	<b>Test Total</b>
Login de Usuario	3	2	5
Login de Administrador	3	1	4
Crear Cuenta (Registro)	2	2	4
Filtrar producto (Buscador)	2	2	4
Ver Productos	2	3	5
Ver fotos del producto (carrousel)	3	0	3
Realizar Reserva	5	1	6
Administrar Producto	3	1	4
Volver al home desde logo	2	1	3
Sitio Responsive	2	0	2
Historial de reservas	3	0	3



## Ejecución de Pruebas

### Ejecución Manual

	<i>Test Pasado</i>	<i>Test Fallados</i>	<i>Test no ejecutados</i>	<i>Test Total</i>
<i>Login de Usuario</i>	3	0	0	3
<i>Login de Administrador</i>	3	0	0	3
<i>Crear Cuenta (Registro)</i>	2	0	0	2
<i>Filtrar producto (Buscador)</i>	2	0	0	2
<i>Ver Productos</i>	2	0	0	2
<i>Ver fotos del producto (carousel)</i>	3	0	0	3
<i>Realizar Reserva</i>	5	0	0	5
<i>Administrar Producto</i>	3	0	0	3
<i>Volver al home desde logo</i>	2	0	0	2
<i>Sitio Responsive</i>	2	0	0	2
<i>Historial de reservas</i>	3	0	0	3

## Ejecución Automática

### JEST

#### All files

51.45% Statements 186/286 38.63% Branches 17/44 30.76% Functions 24/78 53.26% Lines 186/199

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
components/AdminAttributes	<div><div></div></div>	47.82%	11/23	25%	1/4	33.33%	3/9	52.38%	11/21
components/AdminForm	<div><div></div></div>	51.06%	24/47	50%	4/8	21.05%	4/19	51.06%	24/47
components/AdminImages	<div><div></div></div>	36.36%	8/22	33.33%	2/6	14.28%	1/7	38.09%	8/21
components/AdminPolicies	<div><div></div></div>	62.5%	5/8	50%	1/2	25%	1/4	62.5%	5/8
components/Body	<div><div></div></div>	100%	2/2	100%	0/0	100%	1/1	100%	2/2
components/Calendar	<div><div></div></div>	69.56%	16/23	25%	1/4	50%	4/8	69.56%	16/23
components/CardCategory	<div><div></div></div>	66.66%	2/3	100%	0/0	50%	1/2	66.66%	2/3
components/Footer	<div><div></div></div>	100%	2/2	100%	0/0	100%	1/1	100%	2/2
components/FormLogin	<div><div></div></div>	35.71%	10/28	25%	2/8	25%	2/8	35.71%	10/28
components/Search	<div><div></div></div>	50%	15/30	50%	5/10	25%	3/12	53.57%	15/28
components/SocialNetwork	<div><div></div></div>	100%	2/2	100%	0/0	100%	1/1	100%	2/2
context	<div><div></div></div>	56.25%	9/16	50%	1/2	33.33%	2/6	64.28%	9/14

### JUNIT

Current scope: all classes

#### Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	64.7% (22/34)	40.5% (87/215)	30.3% (118/389)

#### Coverage Breakdown

Package	Class, %	Method, %	Line, %
com.example.Grupo4	0% (0/1)	0% (0/2)	0% (0/2)
com.example.Grupo4.config	0% (0/3)	0% (0/11)	0% (0/32)
com.example.Grupo4.controller	83.3% (5/6)	27% (10/37)	32.6% (31/95)
com.example.Grupo4.dto	40% (2/5)	12.5% (4/32)	12.5% (4/32)
com.example.Grupo4.exception	100% (1/1)	100% (4/4)	100% (4/4)
com.example.Grupo4.model	100% (8/8)	76.6% (58/77)	76.6% (58/77)
com.example.Grupo4.security	0% (0/1)	0% (0/2)	0% (0/21)
com.example.Grupo4.service	71.4% (5/7)	25% (10/40)	18.2% (20/110)
com.example.Grupo4.util	0% (0/1)	0% (0/10)	0% (0/16)

generated on 2022-12-13 11:15



## Postman (Runner)

**PI Grupo 4 - Sprint 1 - Run results** Run Again Automate Run + New Run Export Results

Run on 10 Nov, 2022 17:00:17 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	681ms	6	55 ms

**All Tests** Passed (6) Failed (0) Skipped (0) [View Summary](#)

Iteration 1

<b>POST Create</b>	http://localhost:8080/categorias / API Categorías / Create	201 Created	149 ms	363 B
Pass	Successful POST request			
<b>GET Find by id</b>	http://localhost:8080/categorias/1 / API Categorías / Find by id	200 OK	37 ms	356 B
Pass	Successful GET request			
<b>GET Find all</b>	http://localhost:8080/categorias / API Categorías / Find all	200 OK	62 ms	358 B
Pass	Categories load must be valid and have a body			
Pass	Categories schema is valid			
<b>PUT Update</b>	http://localhost:8080/categorias / API Categorías / Update	200 OK	16 ms	377 B
Pass	Successful UPDATE request			
<b>DELETE Delete</b>	http://localhost:8080/categorias/1 / API Categorías / Delete	204 No Content	11 ms	201 B
Pass	Successful DELETE request			



PI Grupo 4 - Sprint 2 - Run results

Run on 10 Nov, 2022 19:11:46 · [View all runs](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	957ms	18	43 ms

All Tests

Passed (18)

Failed (0)

Skipped (0)

[View Summary](#)

Iteration 1

POST

Create

http://localhost:8080/ciudades

/ API Ciudades / Create

201 Created

173 ms

307 B

Pass

Successful POST request

GET

Find by id

http://localhost:8080/ciudades/1

/ API Ciudades / Find by id

200 OK

28 ms

302 B

Pass

Successful GET request

Pass

Correct product returned

Pass

Body matches string

GET

Find all

http://localhost:8080/ciudades

/ API Ciudades / Find all

200 OK

72 ms

304 B

Pass

Successful GET request

Pass

Correct product returned

PUT

Update

http://localhost:8080/ciudades

/ API Ciudades / Update

200 OK

17 ms

295 B

Pass

Successful PUT request

Pass

Correct product updated

POST

Create

http://localhost:8080/productos

/ API Productos / Create

201 Created

26 ms

558 B

Pass

Successful POST request



### PI Grupo 4 - Sprint 3 - Run results

Run on 25 Nov, 2022 10:51:33 - [View all runs](#)

[Run Again](#) Automate Run ▾ + New Run [Export Results](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	643ms	14	52 ms

All Tests Passed (14) Failed (0) Skipped (0) [View Summary](#)

Iteration 1

POST Create	http://localhost:8080/usuarios	/ API Usuarios / Create	201 Created	112 ms	634 B
Pass	Successful POST request				
POST Authenticate	http://localhost:8080/usuarios/auth	/ API Usuarios / Authenticate	200 OK	173 ms	685 B
Pass	Successful POST request				
GET Find by id	http://localhost:8080/usuarios/1	/ API Usuarios / Find by id	200 OK	13 ms	631 B
Pass	Successful GET request				
Pass	Correct user returned				
Pass	Body matches string				
POST Create	http://localhost:8080/reservas	/ API Reservas / Create	201 Created	31 ms	748 B
Pass	Successful POST request				
GET Find by id	http://localhost:8080/reservas/1	/ API Reservas / Find by id	200 OK	34 ms	891 B
Pass	Successful GET request				
Pass	Correct reservation returned				
Pass	Body matches string				

### PI Grupo 4 - Sprint 4 - Run results

Run on Today, 17:36:31 - [View all runs](#)

[Run Again](#) Automate Run ▾ + New Run [Export Results](#)

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	3s 719ms	4	1741 ms

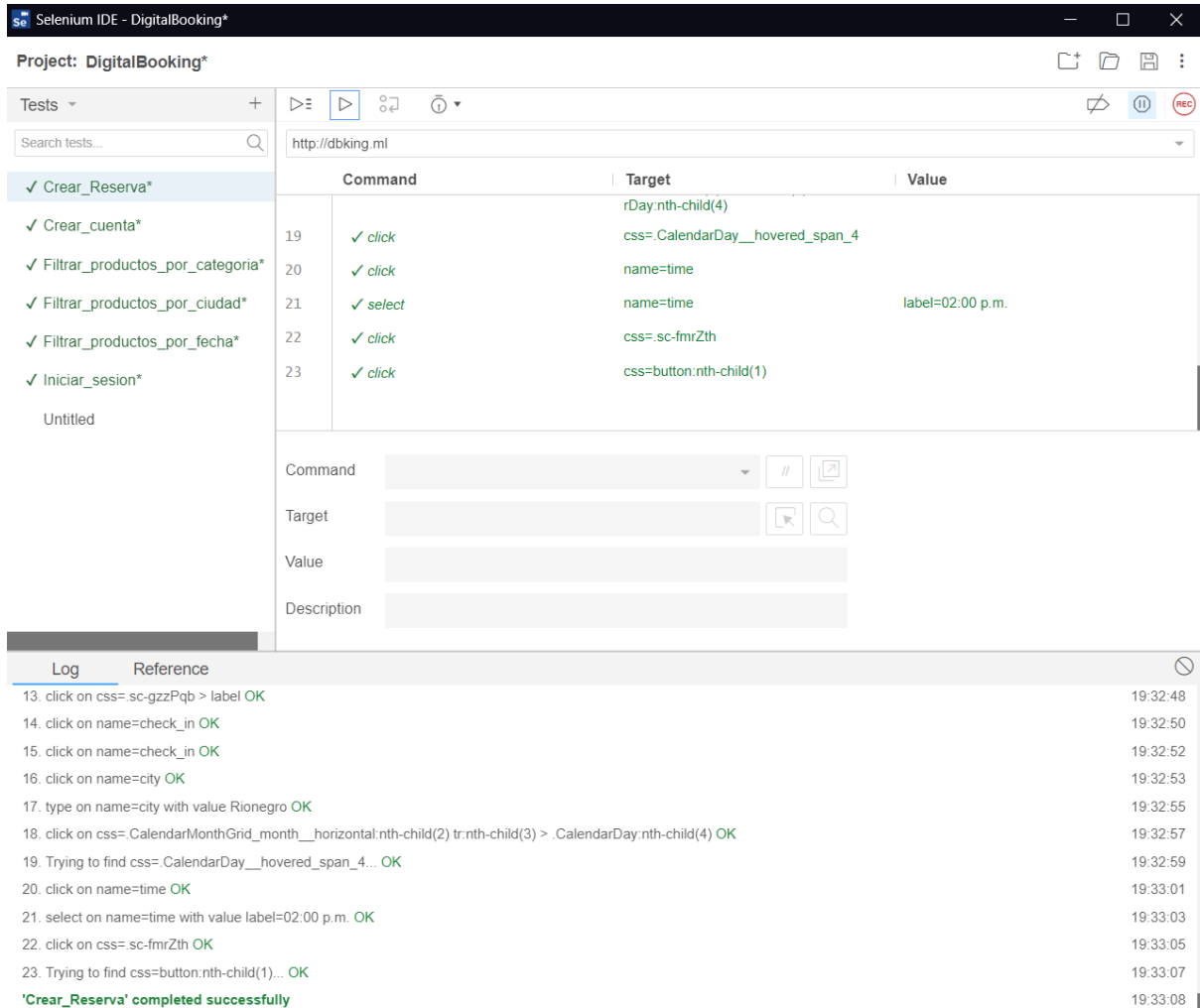
All Tests Passed (4) Failed (0) Skipped (0) [View Summary](#)

Iteration 1

GET Find all	http://localhost:8080/caracteristicas	/ API Caracteristicas / Find all	200 OK	865 ms	871 B
Pass	Features schema is valid				
Pass	Features load must be valid and have a body				
GET Find by city and date ...	http://localhost:8080/productos/ciudadYfechas?idCiudad=4&fechaInicial=2022-12-14&fecha...	/ API Pro...	200 OK	2617 ms	2.333 KB
Pass	Successful GET request				
Pass	Correct product returned				



## Selenium IDE



Project: DigitalBooking\*

Search tests...

Command	Target	Value
✓ click	rDay:nth-child(4)	
✓ click	css=CalendarDay__hovered_span_4	
✓ select	name=time	label=02:00 p.m.
✓ click	css=.sc-fmrZth	
✓ click	css=button:nth-child(1)	

Command: [input type="text"] [//] [icon]

Target: [input type="text"] [icon] [icon]

Value: [input type="text"]

Description: [input type="text"]

Log

Log	Reference
13. click on css=.sc-gzzPqb > label OK	19:32:48
14. click on name=check_in OK	19:32:50
15. click on name=check_in OK	19:32:52
16. click on name=city OK	19:32:53
17. type on name=city with value Rionegro OK	19:32:55
18. click on css=.CalendarMonthGrid_month__horizontal:nth-child(2) tr:nth-child(3) > .CalendarDay:nth-child(4) OK	19:32:57
19. Trying to find css=.CalendarDay__hovered_span_4... OK	19:32:59
20. click on name=time OK	19:33:01
21. select on name=time with value label=02:00 p.m. OK	19:33:03
22. click on css=.sc-fmrZth OK	19:33:05
23. Trying to find css=button:nth-child(1)... OK	19:33:07
'Crear_Reserva' completed successfully	19:33:08

## Reporte de Defectos

### Todos los defectos

El número total de defectos que se han presentado durante la duración de la fase de prueba fue de 2. A continuación se puede observar una representación gráfica de los mismos según su prioridad y severidad.

## Conclusión

Las pruebas realizadas en el área de Testing nos ayudaron a reducir la cantidad de errores que se pueden encontrar, lo que a su vez reduce el riesgo de mal funcionamiento en el sistema desarrollado.

No se pueden probar todas las combinaciones de entrada, priorizamos e implementamos los casos de prueba que consideramos más importantes. Detectar fallas con cada sprint nos ahorra el tiempo y el esfuerzo que nos costaría más corregir con el tiempo. Ponemos más énfasis en detectar errores en las unidades en las que trabajamos mucho. A medida que el sistema evolucionó, se cambió el diseño de los nuevos casos de prueba para encontrar fallas que fueran diferentes a las de sus predecesores. Las pruebas realizadas fueron desarrolladas teniendo en cuenta que nuestro proyecto responde a un sistema que permite realizar alquileres temporales de diferentes inmuebles (departamentos y hoteles, entre otros).

Finalmente, para que nuestro proyecto sea realmente exitoso, es muy importante considerar las necesidades de los usuarios finales e intentar obtener retroalimentación de ellos en cada MVP del desarrollador. Al final, son ellos quienes deben estar satisfechos con el producto resultante.