Certified Tech Developer
The Ultimate Degree

DigitalHouse> Coding School

# Reporte Final de Testing
# Sixvago

**Integrantes:**
*Guido Michel Oudin*
*Angel Estrada*
*Magali Ayelen Paiva Franco*
*Gabriel Franke*
*Flavio Bernachia*
*Jenny Patricia Vargas Naranjo*

# Contenido

# Introducción

Este documento es el Informe Final de Pruebas del sistema Sixvago. 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

El sistema que hemos desarrollado es una aplicación de página única (o SPA por sus siglas en inglés) que ofrece la posibilidad de reservar hoteles. Sixvago permite a sus usuarios filtrar los hoteles disponibles por fecha de estadía, categorías específicas o ambas. Los usuarios también pueden tomar otro rol y poner sus propios hoteles a reservar completando un sencillo formulario que informará al cliente de todo lo que necesite saber para su estadía.

URL: http://g1-sixvago-web.s3-website-us-east-1.amazonaws.com/

| | |
|---|---|
| SPRINT 1 | Planificación de los tests<br>Ejecución de los tests<br>Testeo de las APIS con Postman<br>Tabla de defectos para informarlos |
| SPRINT 2 | Tests unitarios con Jest<br>Cobertura mayor al %50 testeando unitariamente los componentes<br>Automatización de tests con la API en Postman |

| | |
|---|---|
| | Planificación de test con casos de prueba<br>Ejecución de esos mismos casos de prueba |
| SPRINT 3 | Mas tests con Jest donde la cobertura siguió en un %50 a pesar de añadir mas componentes a testear<br>Se utilizó Selenium IDE para testear la reserva de un producto<br>Se siguió automatizando tests de la API por Postman<br>Planificación de test con casos de prueba<br>Ejecución de esos mismos casos de prueba |
| SPRINT 4 | Se utilizó Selenium IDE para testear la creación de un producto<br>Se siguió automatizando tests de la API por Postman<br>Planificación de test con casos de prueba<br>Ejecución de esos mismos casos de prueba<br>Se implementó Selenium Web Driver |

# Alcance

## Dentro del Alcance

| Funcionalidades del sistema | Testing Manual | Testing Automatizado |
|---|---|---|
| Login de usuario | X | X |
| Creacion de usuario | X | X |
| Filtrar por Categorias | X | X |
| Filtrar por Fechas | X | X |
| Filtrar por Categorias y Fechas | X | |
| Filtrar por Ciudades | X | X |
| Filtrar por Ciudades y Fechas | X | |
| Calendario para seleccionar fechas | X | |
| Visualizar productos Recomendados | X | X |
| Productos recomendados son mostrados al azar al azar | X | |
| Pagina de Producto | X | X |
| El carrusel de imagenes de un producto cambia de imagenes cada pocos segundos | X | |
| El producto muestra una puntuacion promedio | X | X |
| Pagina de reserva | X | X |
| Solo se puede hacer una reserva en caso de estar logeado | X | X |
| La pagina de reserva tiene un calendario interactivo | X | X |
| Se puede hacer una reserva seleccionando hora de llegada, check-in y check-out | X | |

| | | |
|---|---|---|
| El usuario tiene un token de seguridad | | X |
| Al crear un usuario este recibe un mail para verificar | X | |
| Al hacer una reserva se envia un mail para verificar la reserva | X | |

## Fuera de Alcance

- Iconos
- Librerias
- Pentesting
- Pruebas al mapa de producto

## Tipos de Pruebas Ejecutadas

| | 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, Selenium Web Driver) | NO | NO | SI | SI |
| Prueba de Humo | SI | SI | SI | SI |
| Prueba de Regresión | NO | NO | SI | SI |
| Prueba de Componente / Unidad (JEST) | NO | SI | SI | SI |

| | | | | |
|---|---|---|---|---|
| Prueba de Integración (Postman) | SI | SI | SI | SI |

## Enfoque de la Prueba

En todos los sprints se hicieron planes de prueba y su posterior ejecución. Se crearon tanto pruebas de humo como regresivas para comprobar las funcionalidades de la aplicación basados en las historias de usuario. Además, ejecutamos pruebas exploratorias. En el transcurso de los sprints se crearon pruebas unitarios en distintos componentes para verificarlos. Se utilizo Postman para la verificación y correcto uso de la API y sus endpoints. Tambien se uso Selenium para test de componentes mas complejos.

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

**https://docs.google.com/spreadsheets/d/1367aiihCEk0j1q1xqQVI7xU2AENg337OSB0ONtXWbJA/edit?usp=sharing**

# Exit Criteria

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

- No se debe tener defectos en estado abierto de severidad crítica y/o bloqueante.
- Todos los endpoints testeados en Postman deben tener al menos un test que haya pasado correctamente
- Nuestro coverage con los componentes de frontend debe ser de al menos un %50
- Todos los casos de prueba deben haber sido ejecutados

# Resumen de Resultados

## Diseño de Pruebas

| | Test Manuales | Test Automáticos | Test de Integración (Postman) | Test Total |
|---|---|---|---|---|
| **Login de Usuario** | 2 | 0 | 1 | 3 |
| **Crear Cuenta** | 2 | 0 | 1 | 3 |
| **Seleccionar Producto** | 1 | 1 | 1 | 3 |
| **Realizar Reserva** | 5 | 1 | 1 | 7 |
| **Administrar Producto** | 1 | 1 | 1 | 3 |
| **Administrar puntuaciones** | 0 | 0 | 1 | 1 |
| **Agregar, listar o eliminar por ID de producto** | 0 | 0 | 1 | 1 |
| **Políticas** | 0 | 0 | 4 | 4 |
| **Calendario de Reservas** | 2 | 1 | 0 | 3 |
| **Bloque de Imágenes** | 2 | 0 | 1 | 3 |
| **Filtros** | 6 | 0 | 3 | 9 |

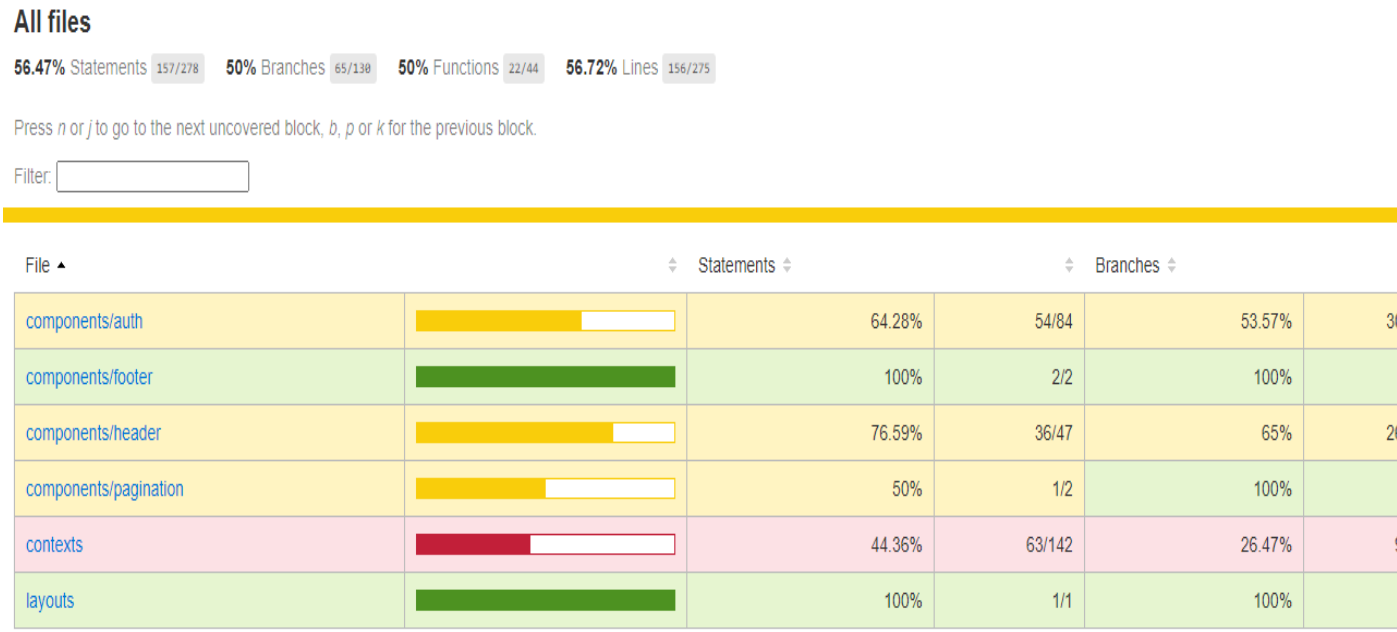
# Ejecución de Pruebas

## Ejecución Manual

| | Test Pasado | Test Fallados | Test no ejecutados | Test Total |
|---|---|---|---|---|
| **Login de Usuario** | 2 | 0 | 0 | 2 |
| **Crear Cuenta** | 1 | 1 | 0 | 2 |
| **Seleccionar Producto** | 1 | 0 | 0 | 1 |
| **Realizar Reserva** | 3 | 2 | 0 | 5 |
| **Administrar Producto** | 1 | 0 | 0 | 3 |
| **Administrar puntuaciones** | 0 | 0 | 0 | 0 |
| **Agregar, listar o eliminar por ID de producto** | 0 | 0 | 0 | 0 |
| **Calendario de Reservas** | 1 | 1 | 0 | 2 |
| **Bloque de Imágenes** | 2 | 0 | 0 | 2 |
| **Filtros** | 6 | 0 | 0 | 6 |

## Ejecución Automática

**Tests unitarios con Jest:**
Cubrimos con hasta un %50 de cobertura




**Tests automatizados con Postman:**

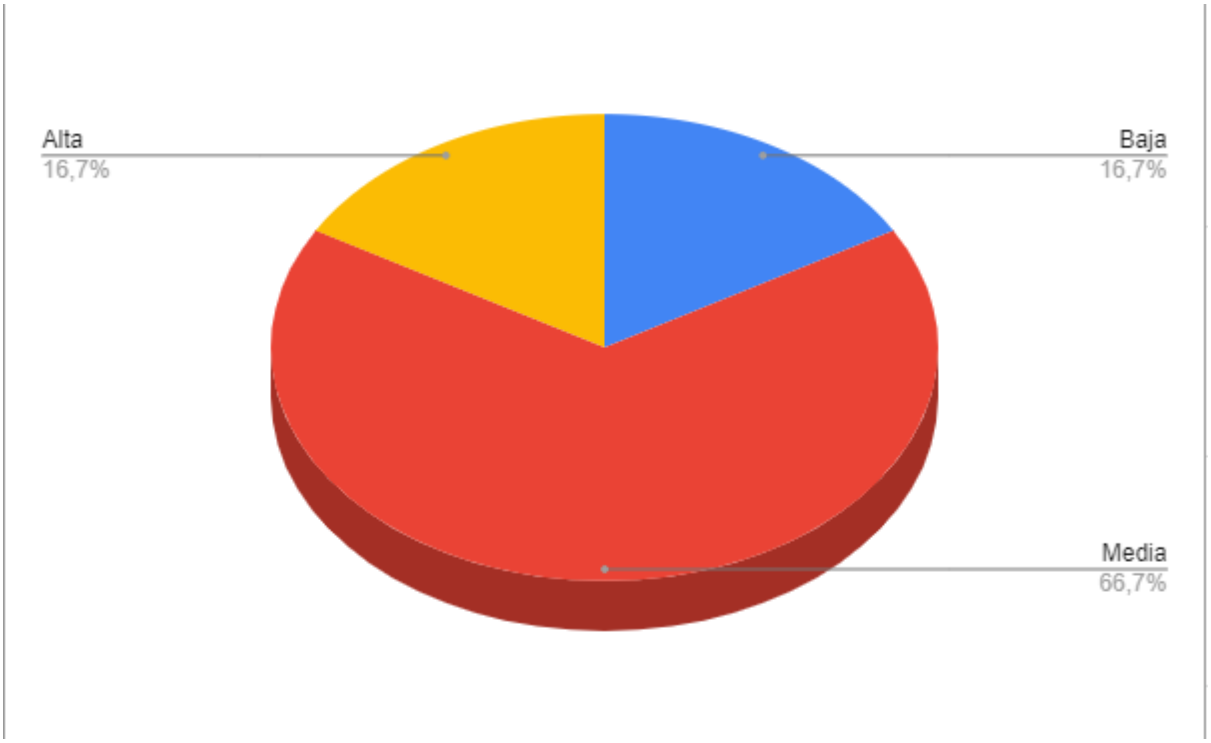
| API Testeada | Cantidad de tests | Resultado |
|---|---|---|
| Categorias | 4 | Passed |
| Authentication | 2 | Passed |
| Reservas | 7 | Passed |
| Producto | 7 | Passed |
| Ciudad | 4 | Passed |
| Características | 4 | Passed |
| Puntuación | 7 | Passed |
| Usuario | 1 | Passed |
| Políticas | 4 | Passed |

# Reporte de Defectos

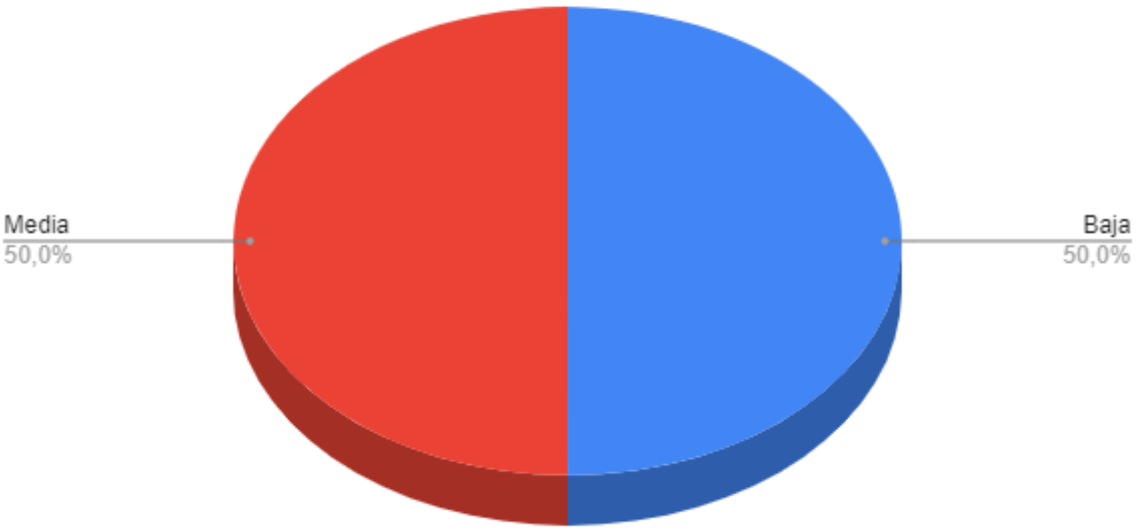
## Todos los defectos

La siguiente sección muestra información con respecto al número total de defectos que se han presentado durante la duración de la fase de prueba.

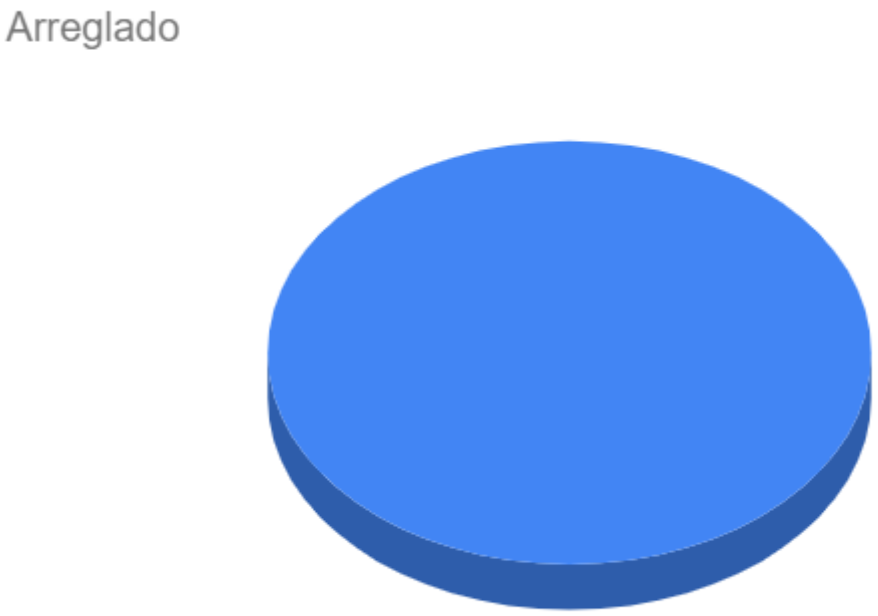
### Defectos por prioridad



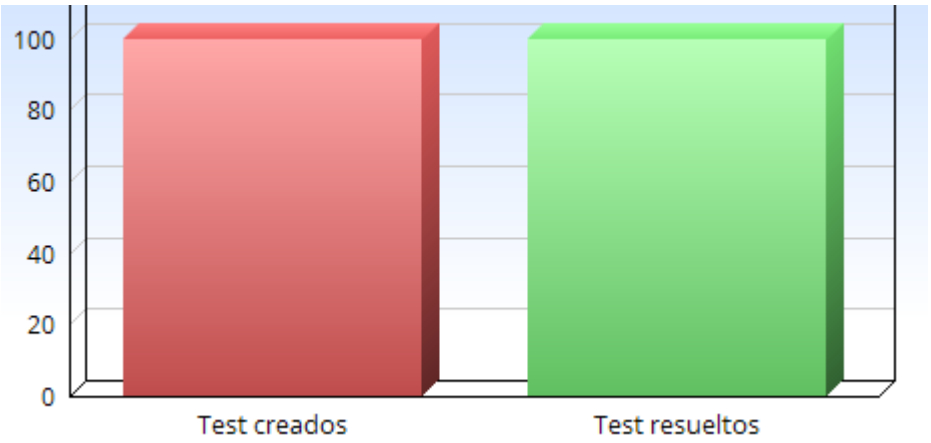
Defectos por Severidad




Defecto por Estado




Defectos Creados vs Resueltos




## Defectos Abiertos

La siguiente sección muestra información con respecto al número total de defectos que permanecen abiertos al final de la fase de pruebas.

Afortunadamente no existen defectos abiertos o por cerrar. Pudimos verificar y corregir todos los defectos con los que nos encontramos.

## Lecciones Aprendidas / Conclusión

Hemos tomado como equipo que nuestra conclusión ha sido que el presente proyecto fue de mucha ayuda para aplicar todos los conocimientos de testing aprendidos. En testing manual aprendimos sobre el diseño y planificación de los casos de pruebas, de suites humo y regresión, los cuales ayudaron a marcar un estándar mínimo de calidad del producto de Sixvago. Ejecutar estas pruebas fue fundamental ya que pudimos poner en práctica los tests diseñados anteriormente y logramos conocer que aspectos estaban fallando en estos.

Por la parte de testing automatizado hacer tests unitarios fue de gran ayuda para saber el funcionamiento de cada componente por separado. En el lado del backend pudimos usar Postman para verificar el correcto funcionamiento de la API, y pudimos automatizar esto para reducir los costos lo mas posible. Utilizamos Selenium WebDriver y IDE para verificar ciertos componentes mas complejos los cuales hicimos satisfactoriamente. Por último las pruebas exploratorias fueron de gran ayuda para identificar lo esencial guiándonos por las issues de gitlab.