**Reporte Final de Testing**

**DIGITAL BOOKING**

**GRUPO 6**

Integrantes:

* Edilberto Carvajal
* Emily Vera
* Gregorio García
* Georgiana Garino
* Matías Raspa
* Jenny Zambrano

Contenido

[**Introducción**](#_om2zx4natrlq) **3**

[Resumen de las actividades de prueba](#_7cfu97xxsofc) 3

[**Alcance**](#_hz0k763vai) **4**

[Dentro del Alcance](#_zebajytw3idt) 4

[Fuera de Alcance](#_6vamp9zhl86x) 5

[Tipos de Pruebas Ejecutadas](#_3j1cy6t0awyc) 6

[Enfoque de la Prueba](#_1vkiz8c4um2g) 6

[**Exit Criteria**](#_wgssa8plb62c) **7**

[**Resumen de Resultados**](#_nxs5ys7rn9h5) **7**

[Diseño de Pruebas](#_nssc69fe1xfq) 7

[Ejecución de Pruebas](#_vw6516yt7fbo) 7

[Ejecución Manual](#_atzu5bxivfaa) 7

[Ejecución Automática](#_mtfm9pfwfiux) 7

[Reporte de Defectos](#_fjlzrc9wi8le) 8

[Todos los defectos](#_me66g8gc2i05) 8

[Defectos por prioridad](#_7is4fsha8xhj) 8

[Defectos por Severidad](#_1uj7qfvqsjj) 8

[Defecto por Estado](#_advh3z8bxvj6) 8

[Defectos Creados vs Resueltos](#_9mudjmvzn28t) 8

[Defectos Abiertos](#_qhs3c8yw5pqs) 8

[**Lecciones Aprendidas / Conclusión**](#_vfztmbnkxrb7) **9**

# Introducción

Este documento es el Informe Final de Pruebas del sistema Digital Booking del grupo 6 de

la camada 1. El propósito de este documento es proveer evidencia de que 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

Nuestro sistema consiste en una web app de Booking y reservas de alojamientos. Para este

sistema se realizaron pruebas unitarias de integración empleando postman

para el backend. Para el frontend se usaron test manuales tales como exploratorios, de

sistema y de humo, para el test automatizado se usó Jest para validar el funcionamiento de

algunos componentes.

En el siguiente link se puede acceder a las pruebas realizadas para cada uno de los sprints del proyecto https://drive.google.com/drive/folders/1A5ssc4GsicInHfRL6IcVjOZppB6M01lp?usp=sharing

# Alcance

## Dentro del Alcance

Desde el testing manual se probaron funcionalidades de frontend, backend, base de datos e

infraestructura, se implementaron pruebas de regresión y humo. Las funcionalidades

probadas fueron las siguientes:

● Frontend:

○ Verificación de renderización header

○ Verificación de renderización footer

○ Verificación login success

○ Verificación de login failed

○ Verificación de pantallas como login, registro, home y producto

○ Verificación de renderización de productos

○ Verificación funcionamiento bloque buscador

○ Verificación formularios de ingreso y registro de usuario

○ Verificación funcionamiento flecha back

○ Verificación funcionamiento galería de imágenes de producto

○ Verificación pantalla detalles del producto

○ Verificación renderización del mapa para el producto

○ Verificación renderización características del producto

○ Verificación cerrar sesión

○ Verificación creación de productos

○ Verificación de urls amigables para el usuario

○ Verificación funcionamiento para seleccionar hora de llegada

○ Verificación funcionamiento del calendario

○ Verificación funcionamiento reservas

○ Verificación filtro de reservas

○ Verificación de visualización en mobile, tablet y desktop

● Backend

○ Verificación método agregar para cada una de las entidades de la API

○ Verificación método listar para cada una de las entidades de la API

○ Verificación método actualizar para cada una de las entidades de la API

○ Verificación método eliminar para cada una de las entidades de la API

○ Verificación de relaciones

● Bases de datos:

○ Verificación de creación del schema

○ Verificación de la creación de las tablas

○ Verificación de inserción de los datos

○ Verificación de lectura de la base de datos

○ Verificación de actualización en la base de datos

○ Verificación de borrado de registros en la base de datos

● Infraestructura:

○ Verificación del funcionamiento de la instancia EC2

○ Verificación de la creación del RDS

○ Verificación del funcionamiento del S3

Las funcionales que se trataron con test automatizados (Jest, Postman y Selenium) fueron

las siguientes:

● Jest:

○ Verificación renderización componentes funcionales

○ Verificación renderización textos y etiquetas HTML

○ Verificación consumo de API en productos y categorias

○ Verificación enrutamiento de los link

○ Verificación eventos click en botones

● Postman:

○ Verificación CRUD de la API

○ Verificación security y roles

○ Verificación integración del sistema

● Selenium:

○ Verificación creación de usuario success

○ Verificación creación de usuario failed

○ Verificación login usuario creado

○ Verificación bloque de búsqueda

Se usó el testing manual y automatizado como complemento entre ellas para probar la

mayoría de las funcionalidades de nuestro sistema.

## Fuera de Alcance

Dentro de las funcionalidades que no fueron probadas bajo ningún tipo de prueba se

encuentran las siguientes:

- Selección fechas lejanas en los calendarios

- Funcionamiento de los mapas

- Funcionamiento en el envío de emails

- Selección y actualización de la página de favoritos

## 

## Tipos de Pruebas Ejecutadas

Durante el desarrollo del sistema se emplearon diferentes tipos de pruebas para testear

cada una de las funcionalidades implementadas, en cada uno de los sprint se aplicaron las

pruebas que mejor se ajustaban a los requerimientos del sistema.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sprint 1 | Sprint 2 | Sprint 3 | Sprint 4 |
| Prueba Estática | x | x | x | x |
| Test Exploratorio | x | x | x | x |
| Prueba Smoke | x | x | x | x |
| Regression Suite |  | x | x | x |
| Prueba de Componente | x | x | x |  |
|  |  |  |  |  |

## Enfoque de la Prueba

En el desarrollo del testing para nuestro sistema, inicialmente se implementaron tests

funcionales positivos y un poco del negativo para el primer sprint, desde el segundo sprint se aumentó la cantidad de tests negativos pero la mayoría sigue siendo del tipo positivo. Decidimos crear desde el inicio del proyecto dos ramas principales para frontend y backend respectivamente y a partir de estas, generar nuevas ramas para funcionalidades en concreto. Cuando se libera alguna funcionalidad de las ramas mencionadas anteriormente se implementa el testing manual y automatizado para verificar su funcionalidad. A la par se realizaba una prueba de humo para detectar defectos y reportarlos en caso de ser necesario.

Antes de finalizar el sprint se unen las ramas principales con sus subtareas y se realizaba la

integración entre ramas antes de enviar todo a la rama main, durante este proceso se

realizaban tareas de testing al verificar el funcionamiento de las funcionalidades

implementadas y mediante pruebas de regresión identificar donde se debía realizar el ajuste

en el código.

Las pruebas automatizadas se desarrollaron sobre las funcionalidades mencionadas en el

alcance y se ejecutaban cada vez que se agregaba una funcionalidad a la rama principal de

frontend o backend y se corría de nuevo antes de generar el build previo a la review del

sprint.

Para el último sprint, se realizó la suite completa para pruebas de regresión para identificar

defectos durante estas pruebas y corregirlos para la entrega final, también se detectaron

aquellos defectos que no logramos ajustar por el tiempo y que no afectan al desempeño del

sistema en general para ser corregidos en una próxima iteración. Pruebas de regresión funcionales se encontraban automatizadas mediante la utilización de Selenium IDE, por lo tanto, se dejó la aclaración de las que no se ejecutaron manualmente durante el sprint IV (como en los sprints anteriores si se habían realizado). Mientras que si se realizó pruebas de regresión manuales para las que era de tipo visual.

En cuanto a la metodología de trabajo para las actividades de testing, cuando se trataba de un nuevo sprint, y correspondía realizar un nuevo build, desde desarrollo se ejecutaban pruebas de humo para las nuevas funcionalidades y pruebas de regresión para corroborar que lo desarrollado anteriormente no haya tenido ningún tipo de problema, y que continúe funcionando de acuerdo a lo establecido. Se realizaron además pruebas estáticas durante el sprint I y II

Antes de finalizar cada sprint se ejecutaban pruebas exploratorias, para que, en base a la experiencia del tester, se corrobora el funcionamiento correcto y, en el caso de existir defectos, fueran reportados en la planilla de defectos.

Se desarrollaron pruebas de unidad sobre las funcionalidades descritas en el cuadro “diseño de pruebas”. Las pruebas de unidad se ejecutaban cada vez que se generaba un nuevo build del sistema

# 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.

● No se debe tener defectos en estado abierto de prioridad crítica.

● Los tests automatizados con Jest deben tener una cobertura mínima del 40% para

todo el sistema.

● La ejecución de las pruebas de regresión deben tener mínimo un 90% de pass rate.

# 

# Resumen de Resultados

## Diseño de Pruebas

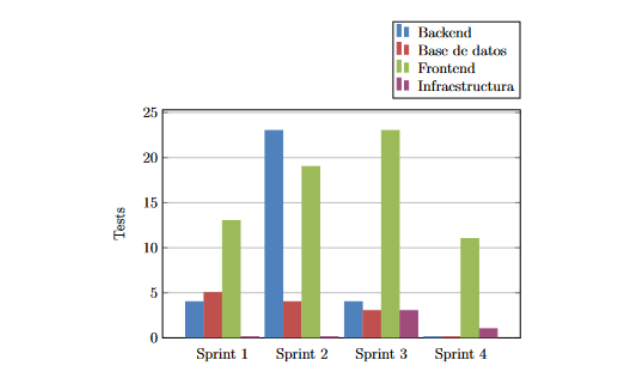
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Manuales | Test Automáticos | Test de Integración o Postman | Test total |
| Responsive | 11 | 1 | 0 | 12 |
| Register | 18 | 1 | 1 | 20 |
| Login | 2 | 2 | 1 | 5 |
| Categorias | 5 | 2 | 1 | 8 |
| Listado | 1 | 3 | 1 | 5 |
| Buscador | 3 | 3 | 2 | 8 |
| Evento Header | 2 | 0 | 2 | 4 |
| API-Categorias | 9 | 9 | 4 | 22 |
| Producto | 5 | 2 | 1 | 8 |
| Seleccionar ciudad | 6 | 2 | 1 | 9 |
| Seleccionar Fechas | 2 | 2 | 1 | 5 |
| Filtro de categorias | 4 | 2 | 1 | 7 |
| Borrar filtro | 2 | 2 | 1 | 5 |
| Ver mas | 8 | 2 | 1 | 11 |
| Mostrar Mapa | 6 | 2 | 2 | 10 |
| Paginación | 2 | 2 | 2 | 6 |
| Social Share | 2 | 1 | 1 | 4 |
| Favorite | 4 | 1 | 1 | 6 |
| Fechas Disponibles | 8 | 0 | 2 | 10 |
| Realizar reservas | 8 | 0 | 2 | 10 |
| Administrar Producto | 10 | 0 | 2 | 12 |
| Ver Mis Reservas | 1 | 1 | 2 | 4 |
| Elegir Fechas | 6 | 1 | 0 | 7 |
| Volver Al Home | 9 | 1 | 1 | 11 |
| Administrador | 1 | 1 | 1 | 3 |
|  |  |  |  |  |

## Ejecución de Pruebas

### Ejecución Manual

En el siguiente gráfico se detallan las pruebas de humo y regresión usadas para cada uno

de los sprint y según el tipo (Backend, Base de datos, Frontend e Infraestructura)



Para el sprint 4 se implementaron pruebas de regresión para verificar que las nuevas

funcionalidades integradas llegasen a afectar las funcionalidades que ya se encontraban

implementadas.

### Ejecución Automática

En la siguiente gráfica se presentan el resumen de las pruebas realizadas con Jest para los

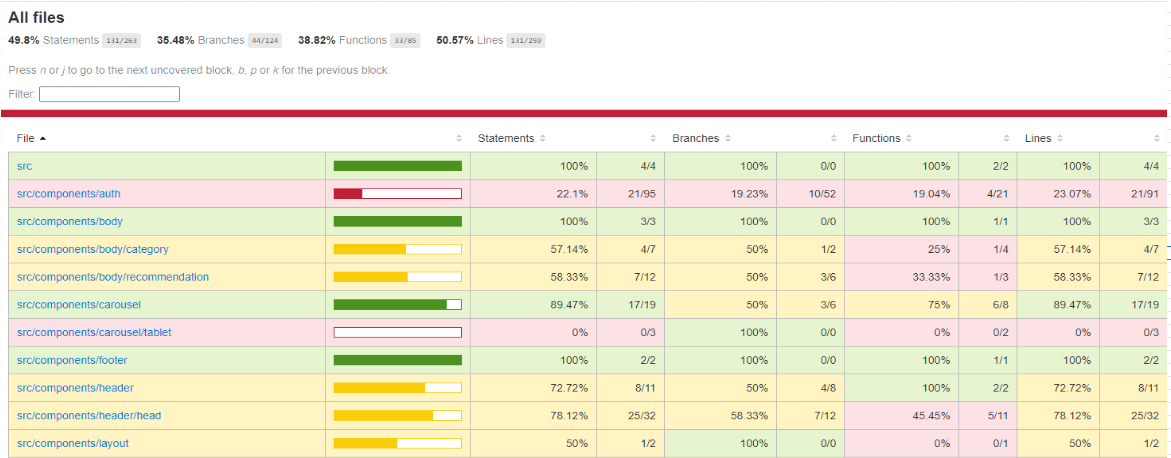
componentes implementados en el Frontend con React, para este tenemos una cobertura

mínima del 40%, en el caso de componentes sencillos se logró una cobertura del 100%

pero en los componentes de integración que tienen la renderización de varios componentes

y una lógica más compleja, se testeó que su renderización fuera correcta y eventos de link

para redireccionamiento a otras páginas.



Tests postman

En el siguiente Link se encuentran los JSON usados para las validaciones del backend

empleando postman dividido en sprints

https://drive.google.com/drive/folders/1A5ssc4GsicInHfRL6IcVjOZppB6M01lp?usp=sharing

## 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

Para los defectos según la prioridad del cliente definimos diferentes categorías, tales como

Prioridad Crítica, Alta, Media y Baja. Teniendo en cuenta esta categorización clasificamos

cada uno de los defectos de nuestro sistema y los resultados son los siguientes.

* 40% Alta
* 30% Media
* 20% Baja
* 10% Crítica

### Defectos por Severidad

Para los defectos según su severidad se tiene la siguiente clasificación: Alta, Media y Baja.

Bajo esta categorización se clasificaron los defectos generados durante el desarrollo de los

4 sprints.

* 25% Alta
* 65% Media
* 10% Baja

### Defecto por Estado

Para la clasificación según el estado, contamos con dos clasificaciones que son defectos

nuevos o creados durante las pruebas y los defectos corregidos durante el desarrollo del

sistema.

* 90% Corregido
* 10% Nuevos

### Defectos Creados vs Resueltos

Los defectos detectados durante los 4 sprint y la cantidad que se corrigió durante esta misma cantidad de tiempo fueron descubiertos a medida que avanzaban los sprints, pero como iban siendo detectados fueron corregidos durante el sprint o el siguiente

### 

### 

### Defectos Abiertos

La siguiente sección muestra información con respecto al número total de defectos que

permanecen abiertos al final de la fase de pruebas.

* Estilos en el bloque buscador
* Filtro limpiar todos
* Al ingresar un mail repetido en el input de la sección arroja un error incorrecto, debería ser más explicito
* Scroll no centralizados
* Botón de mis reservas que sea mas visible

# Lecciones Aprendidas / Conclusión

Como grupo nos llevamos la importancia de aplicar un control de calidad en las

funcionalidades para aumentar el valor agregado de nuestro sistema, evidenciamos los

beneficios de las diferentes pruebas que realizamos porque logramos detectar a tiempo

defectos y bugs que de forma individual no se identificaban, tanto del front como del back son necesarios, El testing de frontend cambio en estos últimos años para darle una perspectiva más abarcativa con cada vez más herramientas y lenguajes

.