



# Reporte Final de Testing

## *Morgam Rent*

### Integrantes:

- *Mateo Adan*
- *Estefani Gonzalez Cabezas*
- *Adrián Moncada*
- *María Laura Murga*
- *Dayana Otagri*
- *Diana Ramos*



## Contenido

<b>Introducción</b>	<b>3</b>
Resumen de las actividades de prueba	3
<b>Alcance</b>	<b>3</b>
Dentro del Alcance	3
Fuera de Alcance	3
Tipos de Pruebas Ejecutadas	3
Enfoque de la Prueba	4
<b>Exit Criteria</b>	<b>4</b>
<b>Resumen de Resultados</b>	<b>5</b>
Diseño de Pruebas	5
Ejecución de Pruebas	6
Ejecución Manual	6
Ejecución Automática	6
Reporte de Defectos	6
Todos los defectos	6
Defectos por prioridad	7
Defectos por Severidad	7
Defecto por Estado	7
Defectos Creados vs Resueltos	8
Defectos Abiertos	9
<b>Lecciones Aprendidas / Conclusión</b>	<b>9</b>

## Introducción

Este documento es el Informe Final de Pruebas del sistema Morgam Rent. 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 al 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 permite a los usuarios poder alquilar vehículos teniendo diferentes opciones de categorías, con la posibilidad de realizar búsquedas por diferentes criterios, seleccionar un vehículo para conocer más detalles sobre el mismo y poder realizar reservas de manera segura ingresando con un usuario y contraseña que previamente debe haber registrado. Por otra parte, también permite a los administradores dar de alta nuevos productos, indicando todas sus particularidades y datos distintivos de los demás vehículos.

### Sprint 1

<b>Testear la API</b> #13 · created 1 month ago by Santiago Vallazza · Sprint 1 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 1 month ago
<b>Planificación y ejecución de los tests</b> #12 · created 1 month ago by Santiago Vallazza · Sprint 1 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 1 month ago

### Sprint 2

<b>Testeo Exploratorio - Manual - Estático</b> #44 · created 1 month ago by Santiago Vallazza · Sprint 2 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 3 weeks ago
<b>Testeo Automatizado (Postman)</b> #43 · created 1 month ago by Santiago Vallazza · Sprint 2 · DONE · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 3 weeks ago
<b>Testeo Automatizado (Selenium)</b> #42 · created 1 month ago by Santiago Vallazza · Sprint 2 · Electiva · Testing / QA	CLOSED · 0 comments · updated 3 weeks ago

### Sprint 3

<b>Implementar tests automatizados</b> #67 · created 3 weeks ago by Santiago Vallazza · Sprint 3 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 1 week ago
<b>Implementar tests manuales</b> #66 · created 3 weeks ago by Santiago Vallazza · Sprint 3 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated 1 week ago

### Sprint 4

<b>Implementar testing manual</b> #75 · created 1 week ago by Santiago Vallazza · Sprint 4 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated just now
<b>Implementar testing automatizado</b> #74 · created 1 week ago by Santiago Vallazza · Sprint 4 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated just now
<b>Confeccionar el reporte final de testing</b> #76 · created 1 week ago by Santiago Vallazza · Sprint 4 · Obligatoria · Testing / QA	CLOSED · 0 comments · updated just now



## Alcance

### Dentro del Alcance

Para probar las funcionalidades del sistema desde el testing en el backend y el frontend de la aplicación, se trabajó con:

#### **Postman**

Se realizaron pruebas automatizadas para comprobar las altas, bajas, modificaciones y consultas a los endpoints establecidos en la API que tenían impacto en la base de datos. Para estas pruebas se utilizó Postman y se verificó a través de tests si los pedidos a la API daban la respuesta esperada (códigos de respuesta satisfactoria como los 200 o 201 y códigos de error como 403 o 500).

Postman nos permitió testear las APIs para, así, garantizar el correcto funcionamiento de manera externa de la aplicación y, a su vez, crear una estructura de información para el uso del equipo, con el fin de unificar los datos y su correcto despliegue en la aplicación, obteniendo la siguiente colección:



RUN SUMMARY			1
▶	POST	POST PRODUCT	1   0
▶	GET	GET ALL PRODUCTS	2   0
▶	GET	GET BY ID PRODUCTS	2   0
▶	GET	Get By City	2   0
▶	PUT	PUT PRODUCTS	2   0
▶	POST	POST CATEGORIES	1   0
▶	GET	GET ALL CATEGORIES	2   0
▶	GET	GET BY ID CATEGORIES	2   0
▶	PUT	PUT CATEGORIES	2   0
▶	POST	POST CHARACTERISTIC	1   0
▶	GET	GET ALL CHARACTERISTIC	2   0
▶	GET	GET BY ID CHARACTERISTIC	2   0
▶	PUT	PUT CHARACTERISTIC	2   0
▶	POST	POST IMAGE	1   0
▶	POST	CREATE RESERVATION	1   0
▶	GET	GET ALL RESERVATIONS	2   0
▶	GET	GET BY PRODUCT ID	2   0
▶	GET	Get All Users	2   0
▶	GET	Token Filter	1   0
▶	POST	New User Admin	1   0
▶	POST	New User	1   0
▶	POST	Auth	1   0

## Testing Automatizado – frontend

Se propuso utilizar Selenium para el test de automatización de frontend, ya que este mismo permite llevar a cabo la tarea de automatización de diferentes funcionalidades desde el frontend. Para cubrir las principales funcionalidades de la aplicación, hubo que tener en cuenta los siguientes test cases:

- **Selenium:**
  - Verificación creación de usuario success
  - Verificación creación de usuario failed



- Verificación login usuario creado
- Verificación login admin
- Verificación de failed login usuario creado
- Verificación de búsqueda por calendario
- Verificación de búsqueda por ciudad
- Verificación de búsqueda por ciudad y calendario
- Verificación de búsqueda de productos
- Verificación de información del producto
- Verificación de páginas funcionando correctamente
- Verificación de botones funcionando correctamente
- Verificación de reserva funcionando correctamente

## Testing Manual

A través del testing manual, primero se redactaron casos de prueba basados en los requerimientos establecidos en cada lectura de sprint, que abarcaron gran parte del testeo de la interfaz gráfica y funcionalidades de la página. La ejecución de los casos de prueba fue documentada en las planillas correspondientes a cada sprint. Por último, al finalizar cada sprint se realizaron las pruebas exploratorias basadas en las user stories y las issues de gitlab. Tanto por los casos de prueba como por el testeo exploratorio surgieron defectos que fueron documentados en la planilla de defectos.

📁 PI\_MORGAM\_Grupo4

### ● 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 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 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
  
- 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

## Fuera de Alcance

Las issues/actividades que no fueron probadas/realizadas son:

- Pruebas automatizadas con Jest



## Tipos de Pruebas Ejecutadas

	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Prueba Estática	SI	SI	SI	SI
Prueba Exploratoria	SI	SI	NO	NO
Prueba de Sistema (Selenium IDE, Selenium Web Driver)	NO	SI	SI	SI
Prueba de Humo	SI	SI	SI	SI
Prueba de Regresión	SI	SI	SI	SI
Prueba de Componente / Unidad (JEST)	NO	NO	NO	NO
Prueba de Integración (Postman)	SI	SI	SI	SI

## Enfoque de la Prueba

El enfoque del testing del proyecto se basó en cubrir test funcionales para probar el funcionamiento del sistema, lo realizamos a través de test manuales y exploratorios.

Para testear el front utilizamos Selenium y para testear el back utilizamos Postman.

**Link Planilla de Casos de Prueba:**

 **PI\_MORGAM\_Grupo4**



## 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.
- Cumplir con los requisitos asignados en las issues.
- Los tests de Postman requieren 100% de pass rate.
- 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
Header de la web	7	0	0	12
Body y footer de la web	7			
Registro	6	3	1	20
Login	2	2	1	5
Categorías	4	2	1	8
Listado	1	3	1	5
Buscador	3	3	2	8
Producto	5	2	1	8
Seleccionar ciudad	6	2	1	9
Seleccionar Fechas	2	2	1	5
Filtro de categorías	4	2	1	7
Ver mas	8	2	1	11
Social Share	2	1	1	4
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	2	1	1	3

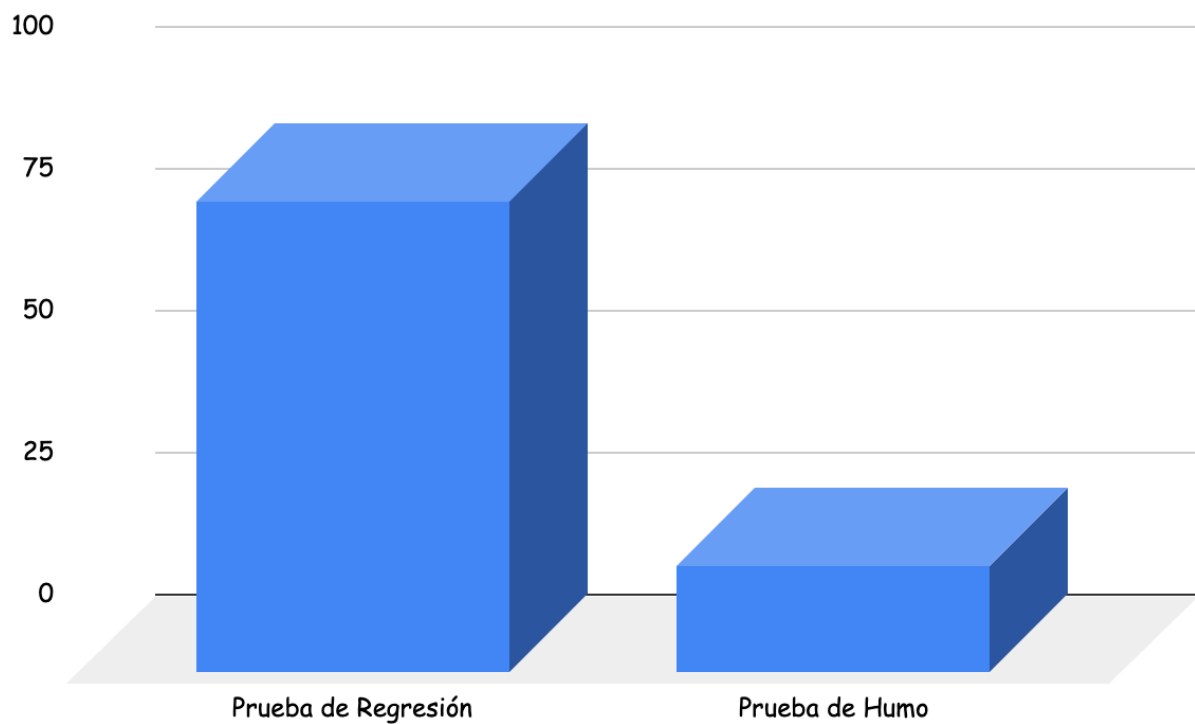


## Ejecución de Pruebas

### Ejecución Manual

En el siguiente gráfico se muestra las pruebas de regresión & las pruebas de humo con las que se llegaron al final del proyecto.

Para el sprint 4 se implementaron pruebas de regresión sobre las nuevas funcionalidades y sus efectos en las funcionalidades previamente desarrolladas.





## Ejecución Automática

### **Test Postman**

En el siguiente link se pueden ver los JSON utilizados en los diferentes sprints.

[https://drive.google.com/drive/folders/1UCnhoNk0wBvd\\_DIH0F4FXonW3PswHZex](https://drive.google.com/drive/folders/1UCnhoNk0wBvd_DIH0F4FXonW3PswHZex)

### **Selenium**

El siguiente link muestra las pruebas realizadas en selenium en cada sprint.

<https://drive.google.com/drive/folders/1Nn64rlw4tw7ghWdT-9vKRVsPOWysjJrd>

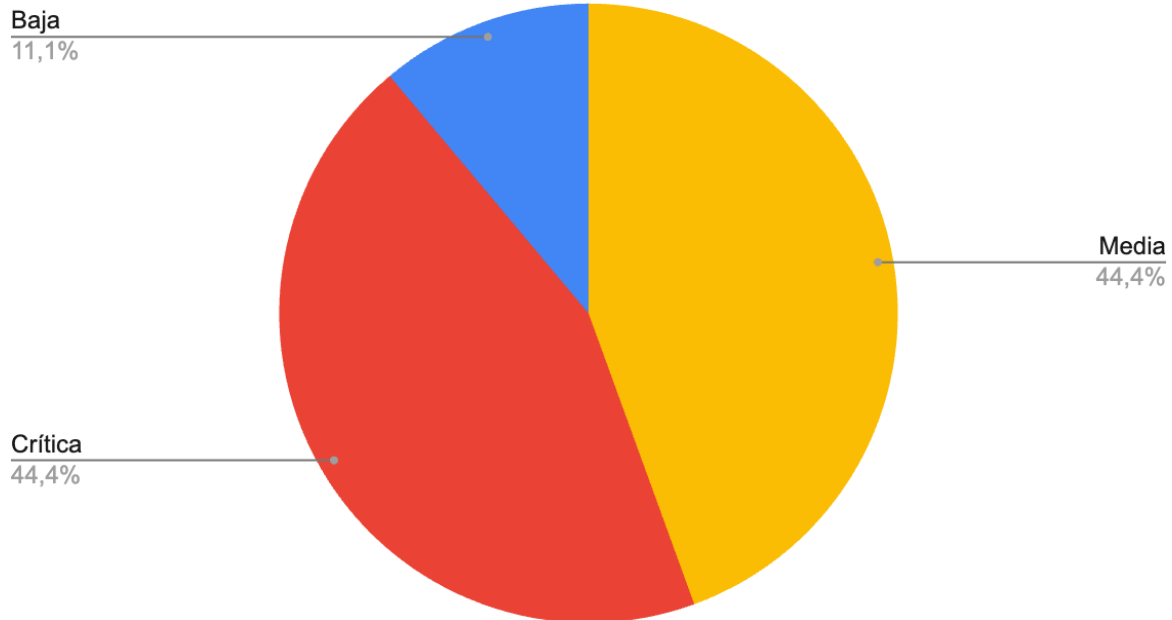
## Reporte de Defectos

### Todos los defectos

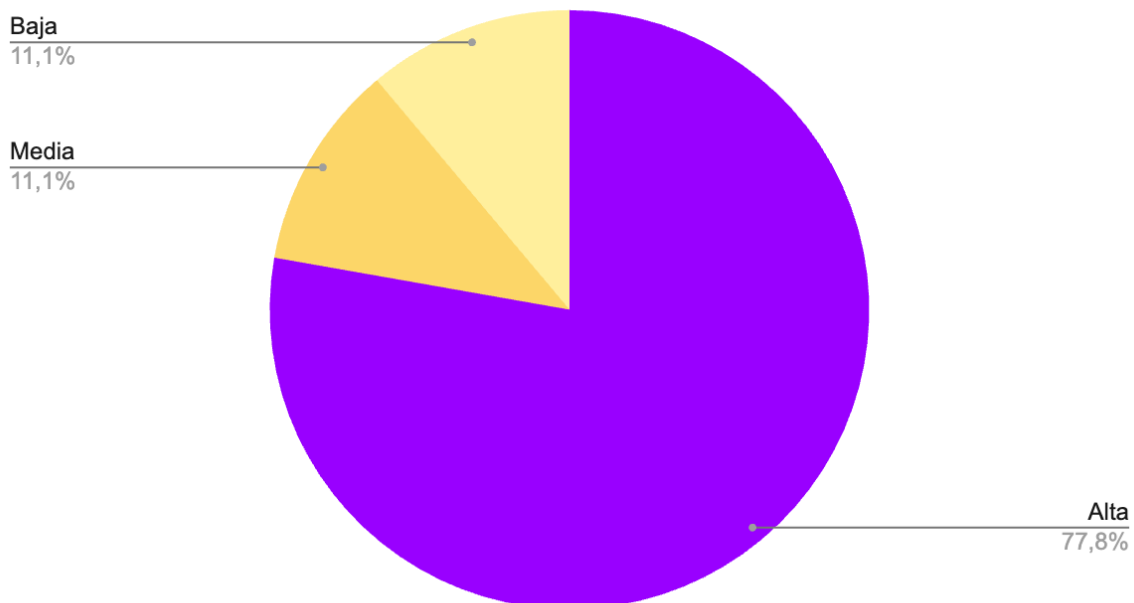
La siguiente sección muestra información con respecto al número total de defectos que se han presentado durante la duración de la fase de prueba.



## Defectos por prioridad

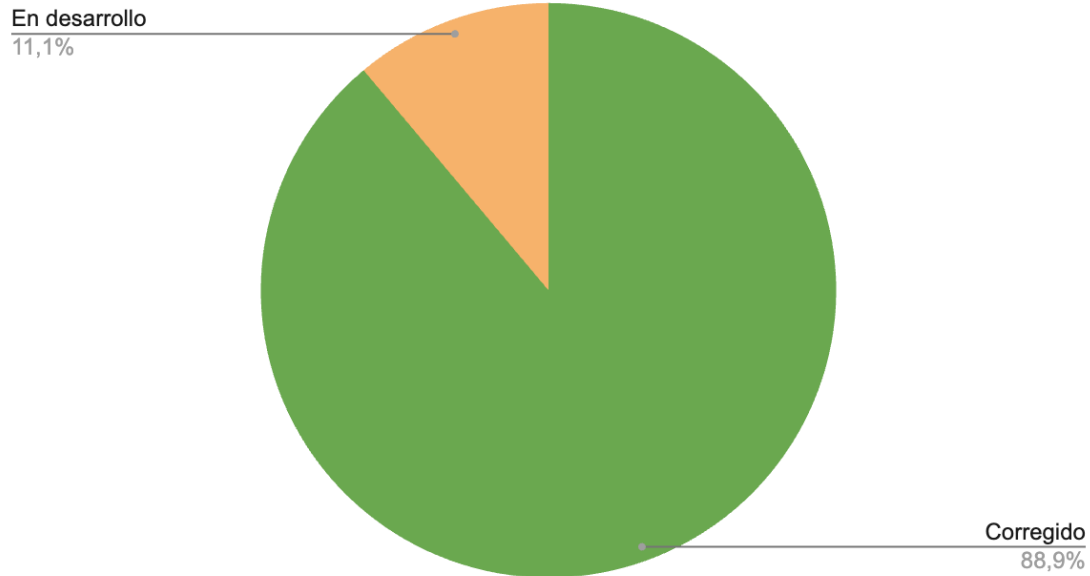


## Defectos por severidad



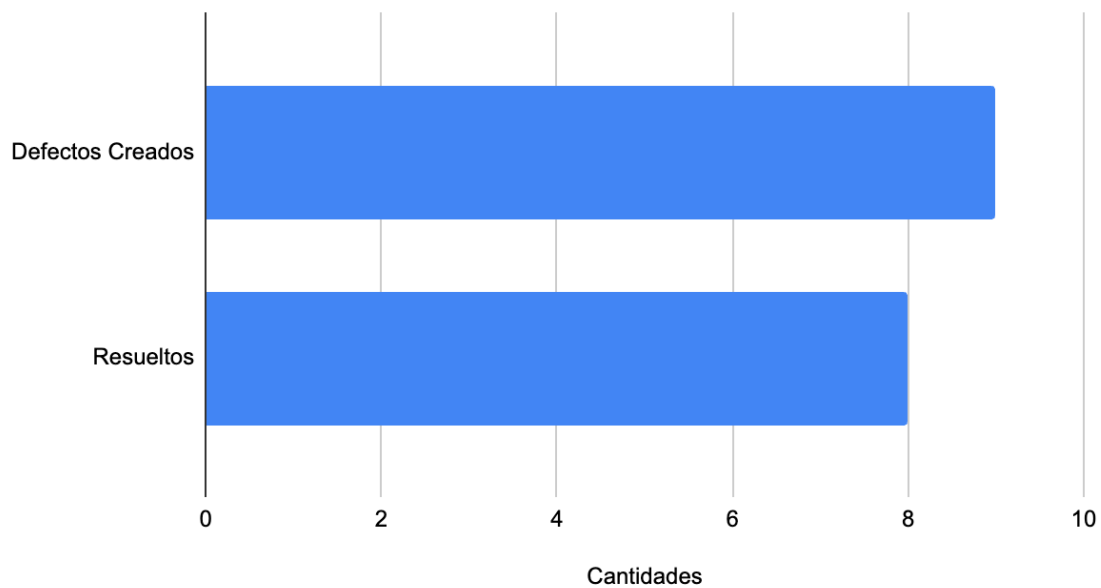


### Defectos por estado



### Defectos creados vs resueltos


#### 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.

- Queda abierto el defecto "Defecto\_pagReserva-009" (  PLMORGAM\_Grupo4 )

## 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.

Por último, nos facilitó la comunicación entre diversas áreas de trabajo, generando así un ambiente respetuoso y ameno para poder lograr una aplicación de calidad.