Certified Tech Developer
The Ultimate Degree

DigitalHouse> Coding School

Reporte Final de Testing

Digirent

Integrantes:
Ariel Bertotto
Claudia Natalia Narvaez Alvarez
Leandro Occhiato
Luis Chavez
Maria Agustina Viruel
Romina Paola Colombo

Contenido

Certified Tech Developer
The Ultimate Degree

DigitalHouse> Coding School

Introducción

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

URL Digirent:
http://digirent.link/

Issues de testing:
https://gitlab.com/proyecto-integrador-0321/camada-2/grupo-2/-/issues?scope=all&state=all&label_name[]=Testing%20%2F%20QA

Nombre	Estado	Sprint
Planificación y ejecución de los tests (Issue 13)	Closed	1
Testear las API (Issue 14)	Closed	1
Implementar tests automatizados (Issue 32)	Closed	2
Test manuales (Issue 36)	Closed	2
Implementar tests manuales (Issue 57)	Closed	3
Implementar tests automatizados (Issue 58)	Closed	3
Test automatizados Postman (Issue 62)	Closed	3
Test automatizados JEST (Issue 63)	Closed	3
Implementar testing automatizado (Issue 70)	In Progress	4
Implementar test manuales (Issue 71)	In Progress	4
Confeccionar el reporte final de testing (Issue 72)	In Progress	4
Implementar testing automatizado Postman (Issue 75)	In Progress	4
Implementar testing automatizado Jest (Issue 76)	In Progress	4

Alcance

Dentro del Alcance

Las funcionalidades de Testing Manuales que se desarrollaron en Digirent fueron diversas. Las funcionalidades se testean dependiendo del tipo de usuario que navegaba la web.

Usuario (que no inició sesión)

- Navegar por la página de inicio
- Poder filtrar los productos por fecha y categoría
- Limpiar los filtros de los productos
- Ver los productos que se encuentran en Digirent
- Hacer click en algún producto
- En la página del producto poder ver las fotos mediante el uso de un carrusel de fotos
- Poder ver las fechas disponibles del producto
- No permitir iniciar reserva al no estar logueado
- Registrarse en el caso de no tener un registro previo

Usuario (que inició sesión)

- Debe poder realizar las mismas cosas que un usuario que no se inicio sesión
- Poder iniciar una reserva
- Poder elegir rango de fecha y hora de la reserva
- Poder ver las reservas realizadas cuando se hace click en el usuario

Usuario Administrador (que inició sesión)

- Al haber iniciado sesión se habilita un nuevo texto “administracion” que redirige a la pagina de administracion
- Poder crear producto
- Poder buscar producto por id
- Poder modificar productos

Por otro lado también se realizaron testeos automatizados de componentes, se realizaron testeos mediante jest para frontend y testeos de postman para corroborar las funcionalidades que estaban relacionadas con backend y base de datos.

Jest (Test automatizados)

- Renderización de componentes
- Pruebas de funcionalidades

Postman(Test automatizados)

- 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 y verificación de ID > 0).
- Obtener una categoría por ID. (Test de GET request exitoso y verificación de datos correctos).
- Crear una ciudad (Test de verificación de status = 200).
- Obtener un listado de todas las ciudades (Test de verificación de status = 200, verificación de ID > 0 y verificación de datos correctos).
- Crear una imagen (Test de verificación de status = 200 y verificación que el status code tenga un string).
- Obtener un listado de todas las imágenes. (Test de verificación de ID > 0 y verificación de datos correctos).
- Crear una característica. (Test de verificación de status = 200 y verificación de datos correctos).
- Obtener un listado de todas las características. (Test de verificación de status = 200 y verificación de ID > 0).
- 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 = 200, 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, verificación de ID > 0 y verificación del largo de la lista = 1).
- 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).
- Actualizar un producto (Test de verificación de status = 200 y verificación que el status code tenga un string).
- Eliminar un producto. (Test de verificación de status = 200).
- Crear una reserva (Test de verificación de status = 200).
- Obtener reservas por producto (Test de verificación de status = 200).
- 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.
- Obtener políticas por descripción.

Fuera de Alcance

- Despliegue de calendario al hacer click sobre el campo check-in check-out
- Movernos entre los meses del calendario al presionar el icono de flecha
- Poder seleccionar un rango de fechas en el calendario
- Despliegue de mapa desde el Home
- Poder visualizar el título del producto elegido al hacer click en el marker del mapa

Tipos de Pruebas Ejecutadas

SI/NO

	Sprint 1	***Sprint 2***	***Sprint 3***	***Sprint 4***
Prueba Estática	SI	SI	SI	SI
Prueba Exploratoria	SI	SI	SI	SI
Prueba de Humo	SI	SI	NO	NO
Prueba de Regresión	SI	SI	SI	SI
Prueba de Componente / Unidad	SI	SI	SI	SI
Prueba de Integración (Postman)	SI	SI	SI	SI

Enfoque de la Prueba

Se crearon diferentes tipos de testings.
Positivos vs Negativos, se crearon pruebas de ambos tipos, con el fin de poder probar tanto funcionalidades por un lado cómo requerimientos necesarios (por ejemplo qué sucedería en el caso de que alguien intentara ingresar una contraseña de menos de 5 caracteres).
Cada sprint se creó una nueva planilla en la cual se especificaba cuáles eran los casos de pruebas, si un caso de prueba no cumplía con lo requerido, se lo consideraba un defecto, se le asignaba un valor de defecto y se lo plasmaba en la planilla de defectos.
Fue importante antes de terminar el sprint ejecutar pruebas exploratorias y los defectos encontrados se cargaban sobre la misma planilla de defectos y se los asignaba cómo defecto para el próximo sprint.

Link Planilla de Casos de Prueba:
https://docs.google.com/spreadsheets/d/14t0kAF5qAm6qfpE_eHkw_Qx0KjVULuQw/edit#gid=1827996341
Link Planilla de Defectos:
https://docs.google.com/spreadsheets/d/14t0kAF5qAm6qfpE_eHkw_Qx0KjVULuQw/edit#gid=959351908

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, esto quiere decir que son los defectos que afectan la funcionalidad del sistema e impiden seguir trabajando

- No 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.

- De todas las declaraciones del programa probadas en jest se debe verificar un porcentaje mayor al 40%.
- Con respecto a la ejecución de los tests de postman, acordamos que el past rate debe ser mayor al 85%

Resumen de Resultados

Diseño de Pruebas

	Test Manuales	Test Automáticos (Jest y Postman)	Test Total
Login de Usuario	3	2	5
Login de Administrador	2	1	1
Crear Cuenta (Registro)	4	2	6
Filtrar producto (Buscador)	2	0	2
Ver Productos	2	0	5
Ver fotos del producto (carrousel)	3	0	3
Realizar Reserva	5	1	6
Administrar Producto	1	1	2
Volver al home desde logo	2	0	2
Volver al home con el lema	2	0	2
Sitio Responsive	20	0	20
Historial de reservas	2	0	2

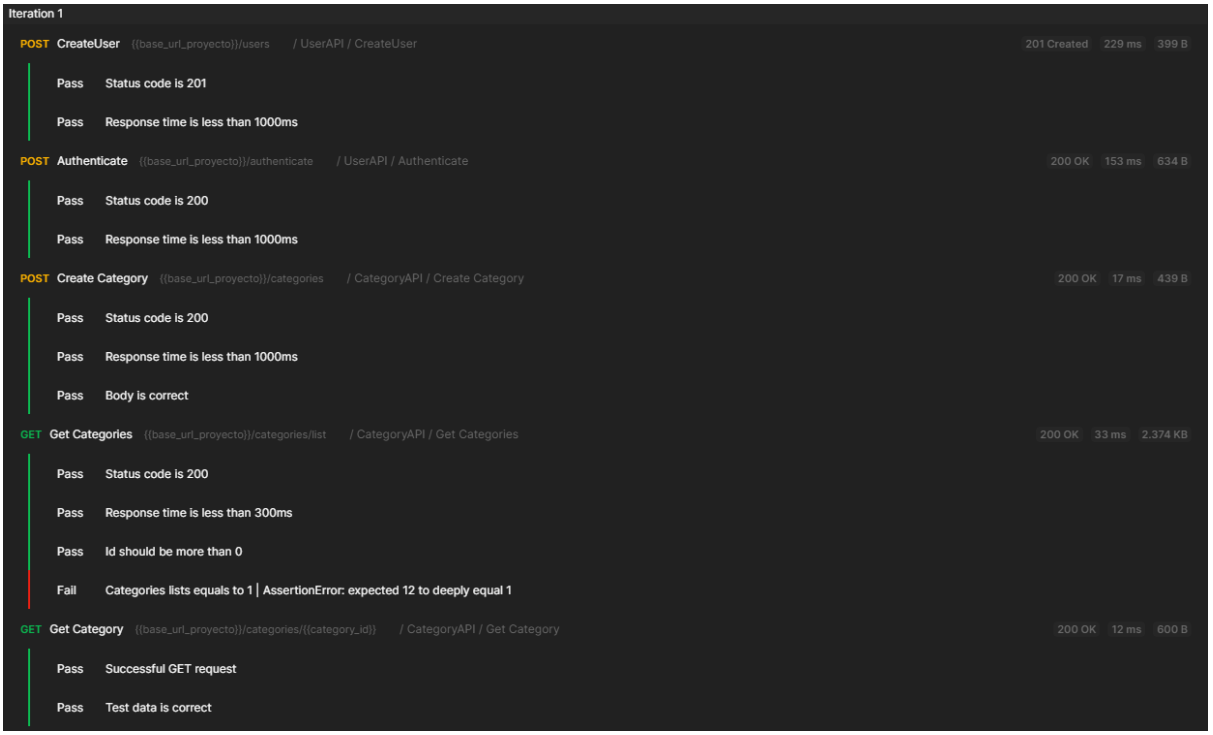
Ejecución de Pruebas

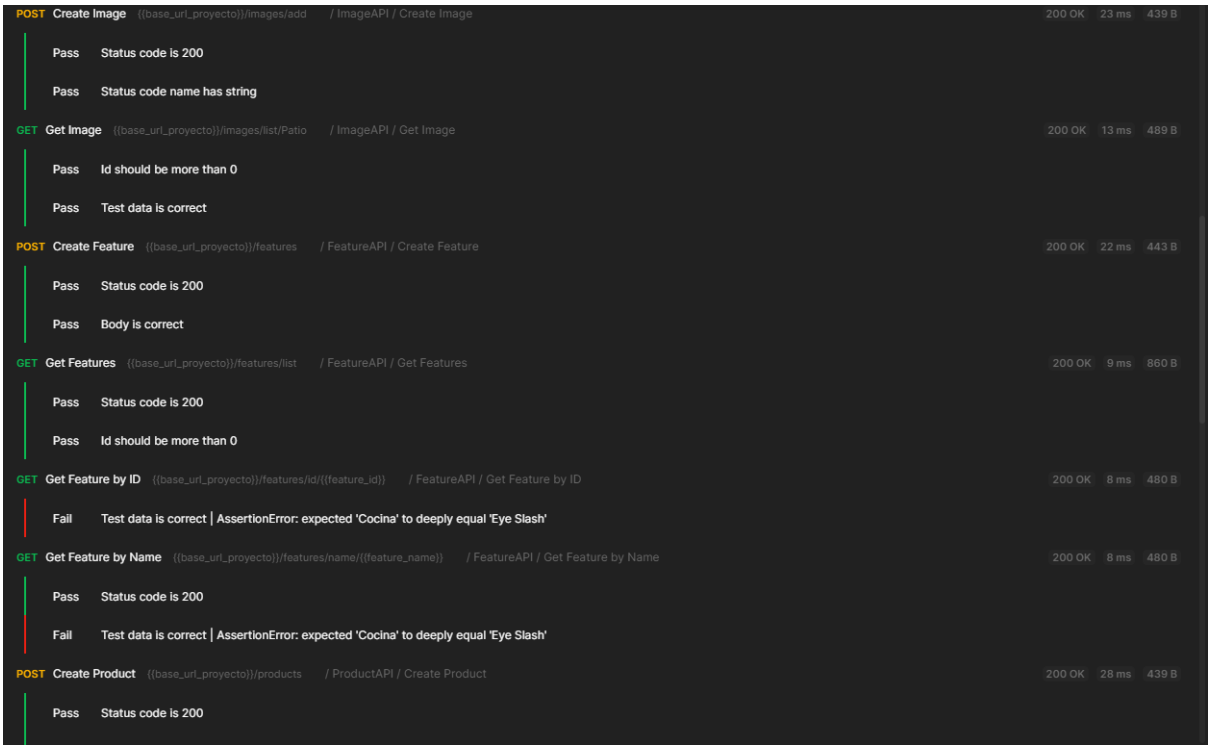
Ejecución Manual

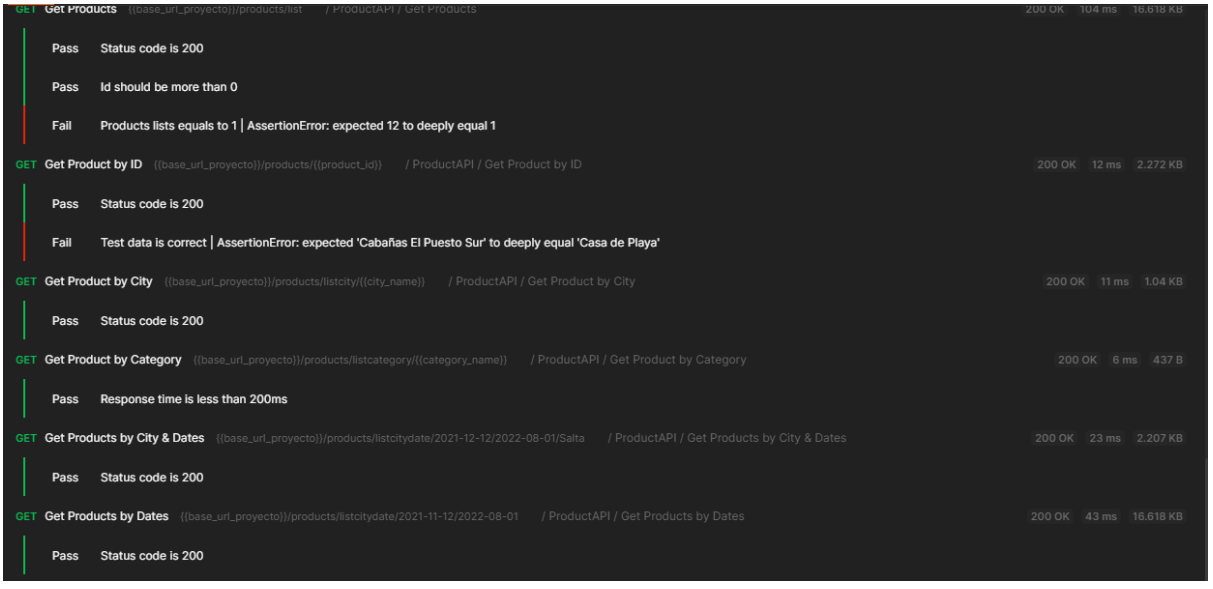
	Test Pasado	Test Fallados	Test no ejecutados	Test Total
Login de Usuario	2	1	0	3
Login de Administrador	1	1	0	2
Crear Cuenta (Registro)	3	1	0	4
Filtrar producto (Buscador)	2	0	0	2
Ver Productos	2	0	0	2
Ver fotos del producto (carrousel)	3	0	0	3
Realizar Reserva	4	1	0	5
Administrar Producto	1	0	0	1
Volver al home desde logo	2	0	0	2
Volver al home con lema	2	0	0	2
Sitio Responsive	18	2	0	20
Historial de reservas	2	0	0	2

Ejecución Automática

src		80%	4/5	100%	0/0	100%	1/1	80%	4/5
src/components/Home		100%	3/3	100%	0/0	100%	1/1	100%	3/3
src/components/auth		2.74%	2/73	0%	0/32	0%	0/10	2.82%	2/71
src/components/button		100%	2/2	0%	0/1	100%	1/1	100%	2/2
src/components/carDetails		16.67%	8/48	0%	0/6	0%	0/22	16.67%	8/48
src/components/card		66.67%	6/9	100%	0/0	40%	2/5	66.67%	6/9
src/components/category		72.73%	8/11	0%	0/2	60%	3/5	70%	7/10
src/components/context		100%	1/1	100%	0/0	100%	0/0	100%	1/1
src/components/footer		100%	2/2	100%	0/0	100%	1/1	100%	2/2
src/components/loading		50%	1/2	100%	0/0	0%	0/1	50%	1/2
src/components/navbar		71.43%	5/7	75%	6/8	66.67%	2/3	71.43%	5/7
src/components/policies		50%	1/2	100%	0/0	0%	0/1	50%	1/2
src/components/routers		100%	4/4	100%	0/0	100%	2/2	100%	4/4
src/components/search		88.89%	16/18	0%	0/2	85.71%	6/7	88.24%	15/17
src/components/socialMedia		100%	2/2	100%	0/0	100%	1/1	100%	2/2




POST Create Image {{base_url_proyecto}}/images/add / ImageAPI / Create Image 200 OK 23 ms 439 B
Pass Status code is 200
Pass Status code name has string
GET Get Image {{base_url_proyecto}}/images/list/Patio / ImageAPI / Get Image 200 OK 13 ms 489 B
Pass Id should be more than 0
Pass Test data is correct
POST Create Feature {{base_url_proyecto}}/features / FeatureAPI / Create Feature 200 OK 22 ms 443 B
Pass Status code is 200
Pass Body is correct
GET Get Features {{base_url_proyecto}}/features/list / FeatureAPI / Get Features 200 OK 9 ms 860 B
Pass Status code is 200
Pass Id should be more than 0
GET Get Feature by ID {{base_url_proyecto}}/features/id/{{feature_id}} / FeatureAPI / Get Feature by ID 200 OK 8 ms 480 B
Fail Test data is correct | AssertionError: expected 'Cocina' to deeply equal 'Eye Slash'
GET Get Feature by Name {{base_url_proyecto}}/features/name/{{feature_name}} / FeatureAPI / Get Feature by Name 200 OK 8 ms 480 B
Pass Status code is 200
Fail Test data is correct | AssertionError: expected 'Cocina' to deeply equal 'Eye Slash'
POST Create Product {{base_url_proyecto}}/products / ProductAPI / Create Product 200 OK 28 ms 439 B
Pass Status code is 200


GET Get Products {{base_url_proyecto}}/products/list / ProductAPI / Get Products 200 OK 104 ms 16.618 KB
Pass Status code is 200
Pass Id should be more than 0
Fail Products lists equals to 1 | AssertionError: expected 12 to deeply equal 1
GET Get Product by ID {{base_url_proyecto}}/products/{{product_id}} / ProductAPI / Get Product by ID 200 OK 12 ms 2.272 KB
Pass Status code is 200
Fail Test data is correct | AssertionError: expected 'Cabañas El Puesto Sur' to deeply equal 'Casa de Playa'
GET Get Product by City {{base_url_proyecto}}/products/listcity/{{city_name}} / ProductAPI / Get Product by City 200 OK 11 ms 1.04 KB
Pass Status code is 200
GET Get Product by Category {{base_url_proyecto}}/products/listcategory/{{category_name}} / ProductAPI / Get Product by Category 200 OK 6 ms 437 B
Pass Response time is less than 200ms
GET Get Products by City & Dates {{base_url_proyecto}}/products/listcitydate/2021-12-12/2022-08-01/Salta / ProductAPI / Get Products by City & Dates 200 OK 23 ms 2.207 KB
Pass Status code is 200
GET Get Products by Dates {{base_url_proyecto}}/products/listcitydate/2021-11-12/2022-08-01 / ProductAPI / Get Products by Dates 200 OK 43 ms 16.618 KB
Pass Status code is 200

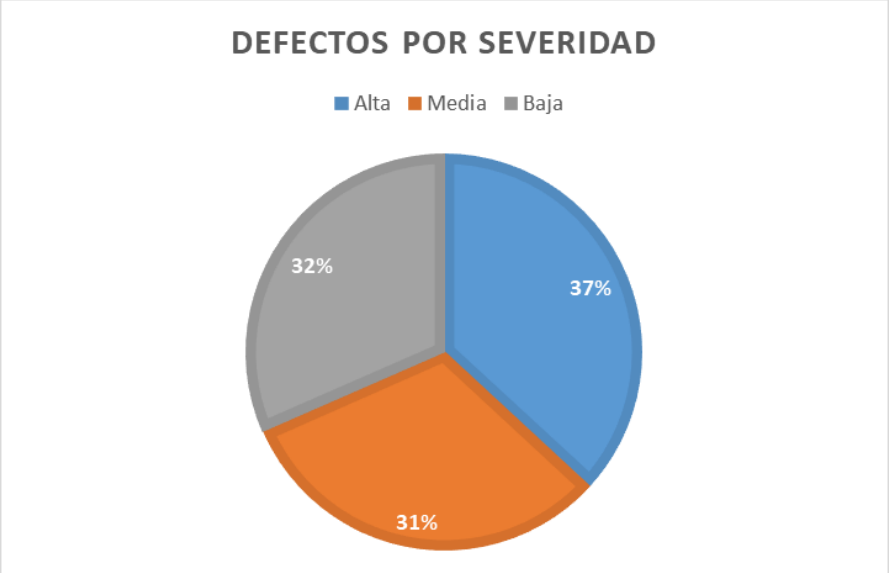
Reporte de Defectos

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.

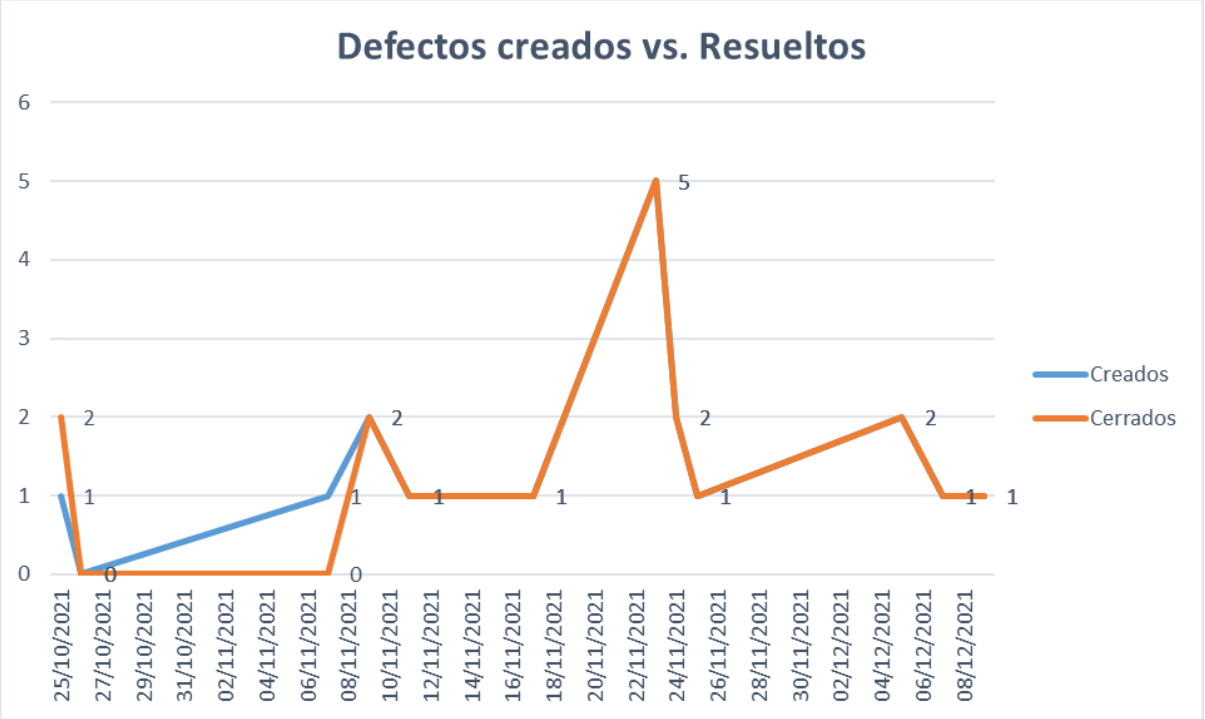
Defectos por prioridad

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.

Id	Título / Nombre	Descripción	Resultado Actual	Resultado Esperado	Estado	Prioridad	Severidad
D-19	Subir imagenes	Poder subir imagenes solamente mediante url, que el técnico haya subido ya a la base de datos	No se puede subir correctamente la imagen	Se debe poder subir imágenes al momento de crear el producto	EN PROGRESO	Media	Media
D-20	Botón de limpiar filtro	Limpiar el filtro de fecha y lugar	La funcionalidad del botón está. Se eliminan los filtros, aunque visualmente la fecha no desaparece de su contenedor	Se debería limpiar tanto el filtro de ciudad como el de fecha, para que el usuario pueda asegurarse de que se borraron sus selecciones	EN PROGRESO	Media	Baja

Lecciones Aprendidas / Conclusión

Existen diferentes puntos a lo largo del proceso de desarrollo de software en los que el error humano puede llevar a una aplicación o página a que no cumpla con los requisitos de los clientes. Es debido a esto que existe el rol de tester, quien se encarga de planificar y llevar a cabo todas las pruebas con el fin de comprobar que hay un correcto funcionamiento de todo el proyecto. Siempre hay cosas para testear, pero lo que hay que destacar es que mientras más casos de prueba existan, más se pondrá a prueba el sistema y se podrán encontrar lugares donde mejorarlo.

Las diferentes herramientas sirven para que el equipo de desarrollo pueda corregir errores previo a entregar un producto, para que este sea de buena calidad. A lo largo del nuestro proyecto integrador fue importante desarrollar diferentes casos de pruebas con el fin de detectar pequeños o grandes errores, estos ayudaron a que el producto pueda cumplir con todos los requisitos mínimos que el PO pedía. Es debido a el trabajo con los desarrolladores que se pudo llegar al punto donde estamos.