Certified Tech Developer
The Ultimate Degree

DigitalHouse>
Coding School

Reporte Final de Testing

EasyRides
GRUPO 11

Integrantes:
- Tomas Alegria
- Belen Risso
- Nicolás Santa María
- Stephania Martheyn
- Cristian Steven David Machado
- Octavio Egeo Faure

Contenido

Introducción

Este documento es el Informe Final de Pruebas del sistema Digital Booking del grupo 11 de la camada 1. El propósito de este documento es proveer evidencia de que el proceso de Testing se cumplió acorde a los parámetros de la industria y por lo tanto, puede
cerrarse. Los Issues de testing en el GitLab fueron implementados en los cuatro Sprints.

Resumen de las actividades de prueba

Nuestro sistema consiste en una web app de alquileres de vehículos. Para este sistema se realizaron pruebas unitarias de integración empleando postman para el backend. Para el frontend se usaron tanto test manuales tales como exploratorios, de sistema, regresión y de humo, para el test automatizado se usó Selenium 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:

http://easyrides.ddns.net/

http://easyridesdevelopment.s3-website.us-east-2.amazonaws.com/

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: registro, login home, producto, reservas y crear producto
    - Verificación de renderización de productos

- Verificación funcionamiento filtros: categoría, ciudad y calendario.
- Verificación formularios de ingreso y registro de usuario
- Verificación funcionamiento flecha para regresar
- 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 información del politicas de la empresa
- Verificación de páginas funcionando correctamente
- Verificación de botones funcionando correctamente
- Verificación de reserva funcionando correctamente
- Verificación funcionamiento del calendario en página de reserva
- Verificación funcionamiento reservas
- Verificación filtro de reservas por medio de flechas
- Verificación de visualización en mobile, tablet y desktop

● Backend

Entidades de las Apis evaluadas:

1. Categorías
2. Productos
3. Ciudades
4. Imágenes
5. Políticas
6. Reservas
7. Usuarios

- Verificación método agregar/guardar para cada una de las entidades de la API
- Verificación método listar para cada una de las entidades de la API
- Verificación método listar todos para cada una de las entidades de la API
- Verificación método actualizar para cada una de las entidades de la API
- Verificación de relaciones buscado por ID como clave foránea

● 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

- Verificación de la performance del S3

Las funcionales que se trataron con test automatizados (Postman y Selenium) fueron las siguientes:

- Postman:
  - Verificación CRUD de la API
  - Verificación security
  - Verificación integración del sistema

- 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 información del producto
  - Verificación de páginas funcionando correctamente
  - Verificación de botones funcionando correctamente
  - Verificación de reserva funcionando correctamente
  - Verificación de imagenes funcionando correctamente

Se usó el testing manual y automatizado para probar las funcionalidades. Cumpliendo las buenas prácticas, diseñamos casos de prueba y las llevamos a cabo en un excel. Allí determinamos la severidad y que tan crítico era el defecto.También usamos pruebas exploratorias basadas en las issues de gitlab y las user stories.

Abajo dejamos el link al excel
https://gitlab.ctd.academy/ctd/proyecto-integrador-0822/1021-ft/grupo-11/-/tree/main/Sprint%204#modal-upload-blob

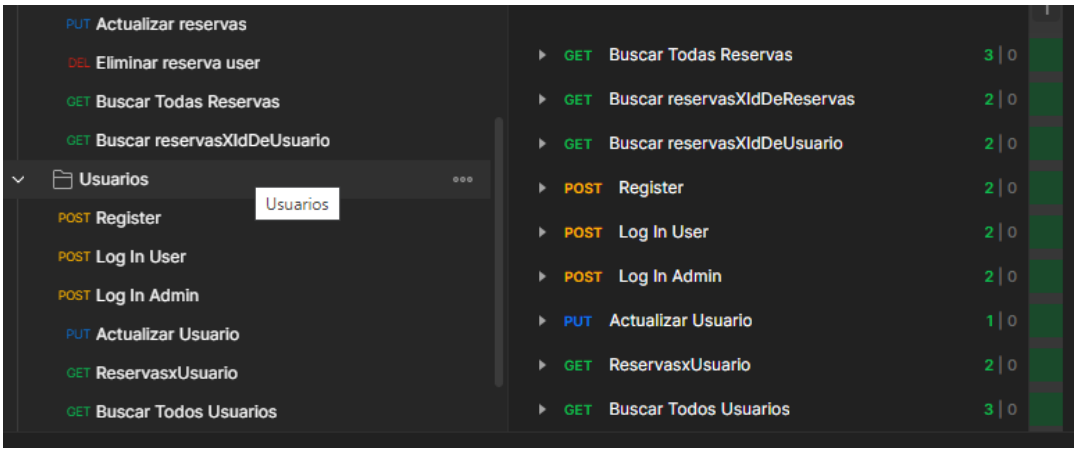
https://docs.google.com/spreadsheets/d/14q6Ji0Oh-pba_I_NXd-9Lsu29HSiK87XeQcFn0YYCUs/edit#gid=1671681234

Desde Postman se testeó con múltiples tests.Con envío de 200/201, tests negativos de 404, y el Requisito de token, también el contenido del response para que no sea null o contenga información errónea. Se testeo que se mantenga la persistencia y la consistencia evaluando que venga un mínimo de contenido, por ejemplo, las imágenes de los productos.

Aquí abajo dejamos el link a donde se exportaron los tests de postman (recuerden que hay que actualizar el token y el usuario creado, también es posible que en el ínterin se resetea la base de datos y no contenga reservas, con lo cual habría que crearlas)

https://gitlab.ctd.academy/ctd/proyecto-integrador-0822/1021-ft/grupo-11/-/tree/main/Sprint%204#modal-upload-blob


Digital Book-S4-
Categorias
GET Buscar Categoria
POST Guardar Categoria
PUT Actualizar Categoria
GET Buscar Todas Categorias
Productos
GET Buscar Producto
GET Buscar Todos Productos
POST Guardar Producto
PUT Actualizar Producto
GET Buscar ProductoxCiudad
GET filtroCiudadYFechas
Ciudades
GET Buscar Ciudad
POST Guardar Ciudad
GET Buscar Todas Ciudades
PUT Actualizar Ciudad
Imagenes
GET Buscar Imagenes
POST Guardar Imagenes
GET Buscar Todas Imagenes
PUT Actualizar Imagenes
Politicas
GET Buscar Todas Politicas
Reservas
POST Guardar Reserva user
PUT Actualizar reservas
RUN SUMMARY
GET Buscar Categoria 3 | 0
POST Guardar Categoria 3 | 0
PUT Actualizar Categoria 3 | 0
GET Buscar Todas Categorias 3 | 0
GET Buscar Producto 3 | 0
GET Buscar Todos Productos 3 | 0
POST Guardar Producto 1 | 0
PUT Actualizar Producto 2 | 0
GET Buscar ProductoxCiudad 2 | 0
GET filtroCiudadYFechas 2 | 0
GET Buscar Ciudad 3 | 0
POST Guardar Ciudad 3 | 0
GET Buscar Todas Ciudades 3 | 0
PUT Actualizar Ciudad 3 | 0
GET Buscar Imagenes 2 | 0
POST Guardar Imagenes 2 | 0
GET Buscar Todas Imagenes 2 | 0
PUT Actualizar Imagenes 2 | 0
GET Buscar Todas Politicas 2 | 0
POST Guardar Reserva user 2 | 0
PUT Actualizar reservas 2 | 0
GET Buscar Todas Reservas 3 | 0
GET Buscar reservasXIdDeReservas 2 | 0


PUT Actualizar reservas
DEL Eliminar reserva user
GET Buscar Todas Reservas
GET Buscar reservasXIdDeUsuario
Usuarios
Usuarios
POST Register
POST Log In User
POST Log In Admin
PUT Actualizar Usuario
GET ReservasxUsuario
GET Buscar Todos Usuarios
GET Buscar Todas Reservas 3 | 0
GET Buscar reservasXIdDeReservas 2 | 0
GET Buscar reservasXIdDeUsuario 2 | 0
POST Register 2 | 0
POST Log In User 2 | 0
POST Log In Admin 2 | 0
PUT Actualizar Usuario 1 | 0
GET ReservasxUsuario 2 | 0
GET Buscar Todos Usuarios 3 | 0

Fuera de Alcance

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

-Jest quedó descartado por el P.O.
-Eliminación de las reservas (un issue opcional).
- Creación de usuario Admin.
- Eliminación dentro de las entidades de las Apis en cuestión (un issue opcional).

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	Si	Si	Si	Si
Test Exploratorio	Si	Si	Si	No
Prueba Smoke	Si	Si	Si	Si
Regression Suite	No	Si	Si	Si
Prueba de Componente	Si	Si	Si	Si

Enfoque de la Prueba

La evolución del enfoque estuvo marcada por la Implementamos tests positivos y negativos desde el primer sprint. Inicialmente creamos varias ramas para cada área, Frontend, Backedn, Infra y testing. Usamos postman para las Apis y Selenium para el frontend. También tests exploratorios. El primer Sprint pudimos terminar con varios días de anticipación y nos dedicamos el resto a aprender nuevas tecnologías y a mejorar el trabajo en equipo. Como el primer Sprint fue relativamente sencillo pudimos realizar las pruebas cuando casi todo estaba terminado.

En el segundo sprint partimos con que Jest dejó de ser un issue obligatorio.
Al inicio del proyecto decidimos crear varias ramas para cada área: frontend, backend e infraestructura, pero rápidamente nos dimos cuenta que dada la interconexión del código era necesario reorganizarse en una rama Main y una Develop. La última siendo a donde subíamos el código cuando se libera alguna funcionalidad. Con respecto al testing, probamos todo en el local host y luego en la nube, implementando el testing manual y automatizado para verificar su funcionalidad. Selenium y Postman. Realizamos una prueba de humo para detectar defectos y reportarlos. Aqui ya empezamos con la filosofia DevOps, de integración continua y evaluamos a medida que los issues pasaban a la tabla de verificar en gitlab.

Antes de finalizar el sprint se unen las ramas principales y realizamos la integración antes de enviar todo a la rama main, durante y después de este proceso se realizaban tareas de testing (de regresión).

También probamos la performance de la infraestructura usando las métricas que nos brinda AWS.
Finalmente pudimos presentar todos los issues funcionales en tiempo y forma.

-En el tercer sprint tuvimos desafíos más importantes. Desde el Back: implementar la seguridad requirió capacitación, y usar Postman para testear las Apis fue fundamental. Desde El Front el Enrutado y los hooks de ract requirieron asistencia del TL, pero se pudo resolver en tiempo y forma. Ya con las ramas y el equipo creando sinergia, el departamento de Testing pudo determinar que la nube estaba lista para ser presentada en vivo al Product Owner y al Scrum Master.

Los testeos continuaron siendo con la filosofía DevOps de integración continua. Tests Automáticos y manuales con postman y selenium. Con mayor cantidad de tests de

regresión y humo que previamente, y con una alto porcentaje de éxito sobre todo los issues severos y críticos. Finalmente solo quedaron unos pocos carry over de issues para el cuarto Sprint. Desde el Back fue la creación de Roles. Y desde el front unos detalles de la página de reserva, los cuales ya estaban en progreso. Y el Calendario.

El cuarto Sprint
Arrancamos el cuarto Sprint habiendo mejorado la presentación grupal y la comunicación del equipo.
Las pruebas de humo automatizadas de postman se fueron ejecutando a medida que salía una nueva funcionalidad, en vez de varias a la vez. Mismo con las pruebas exploratorias y las de Selenium.
Creamos una rama extra, llamada postProductos para subir el Front en su propia rama y testearlo aisladamente. con lo cual quedamos en una rama de backend llamada Devlop, una de Front y la Main
Finalmente realizamos la suite completa de pruebas de regresión para identificar

defectos durante estas pruebas y corregirlos. Realizamos todas las pruebas de los sprint previos, salvo las que estaban incluidas en las nuevas pruebas de componente, regresión e integración. Todo esto llevaba a poder realizar pruebas de unidad completas

Exit Criteria

Se definió los siguientes criterios de aceptación para finalizar las pruebas:
- No debe tener defectos en estado abierto de severidad crítica y/o bloqueante.
- No se debe tener defectos en estado abierto de prioridad crítica.
- La ejecución de las pruebas de regresión deben tener mínimo un 90% de éxito.
- Cumplir con los requisitos asignados en las issues.
- Los tests de Postman de issues esenciales requieren 100% de éxito.

Resumen de Resultados

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 sobre las nuevas funcionalidades y sus efectos en las funcionalidades previamente desarrolladas.

Ejecución Automática

Tests postman
En el siguiente Link se encuentran los JSON usados para las validaciones del backend empleando postman dividido en sprints

https://gitlab.ctd.academy/ctd/proyecto-integrador-0822/1021-ft/grupo-11/-/tree/main/Sprint%204#modal-upload-blob

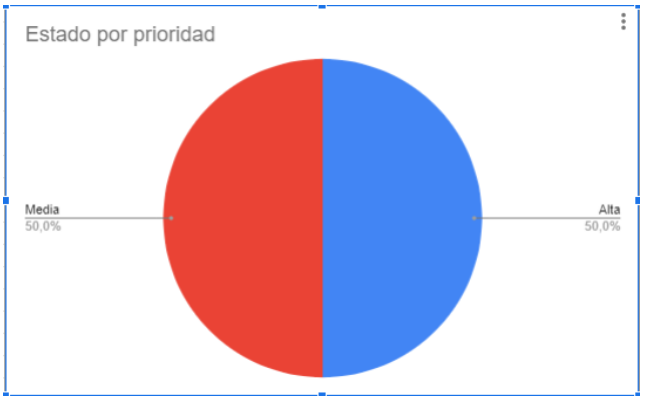
Tests Selenium
En el siguiente Link se encuentran los tests de selenium

https://gitlab.ctd.academy/ctd/proyecto-integrador-0822/1021-ft/grupo-11/-/tree/main/Sprint%204#modal-upload-blob

Reporte de Defectos

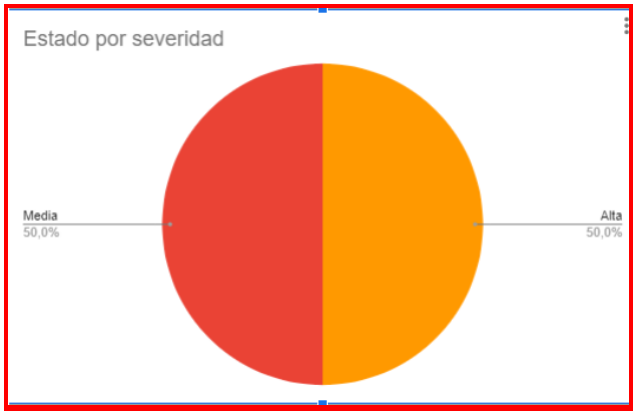
Defectos por prioridad

Definimos los defectos según prioridad Alta, media y baja.Y según Severidad, alta, Media y baja.



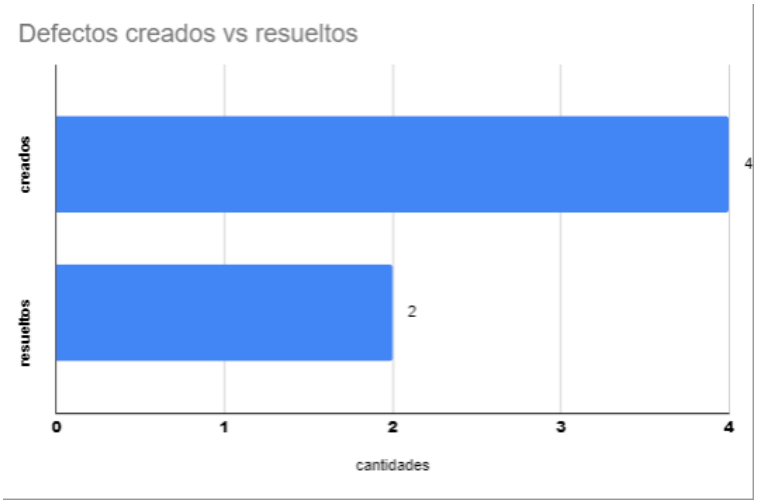
Defectos por Severidad

Para los defectos según su severidad se tiene la siguiente clasificación: Alta, Media y Baja. Esta categorización se clasificaron los defectos generados durante el desarrollo de los 4 sprints.



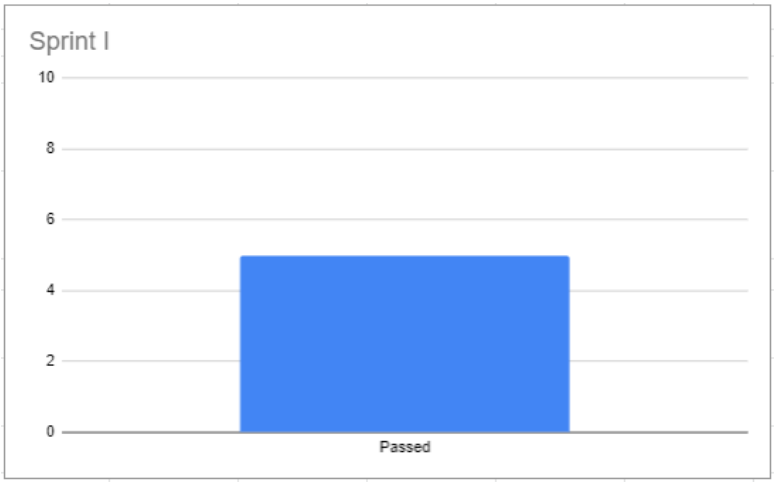
Defectos Creados vs Resueltos

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

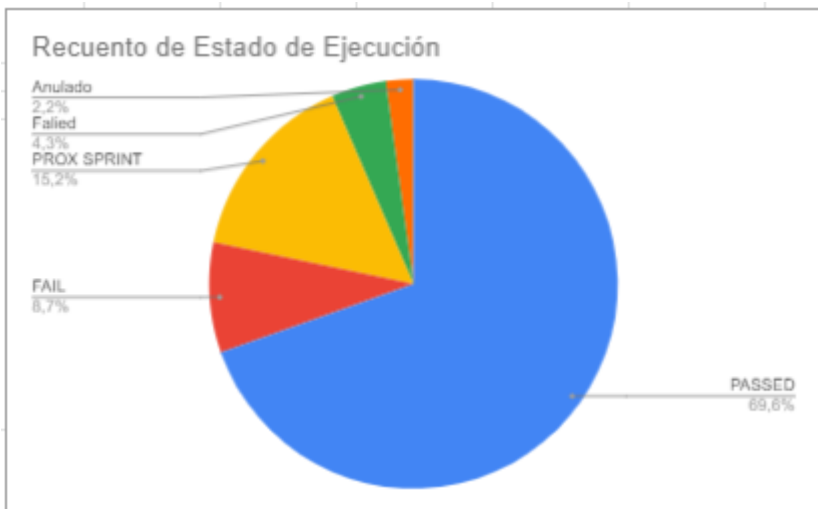


Sprints tests por estado

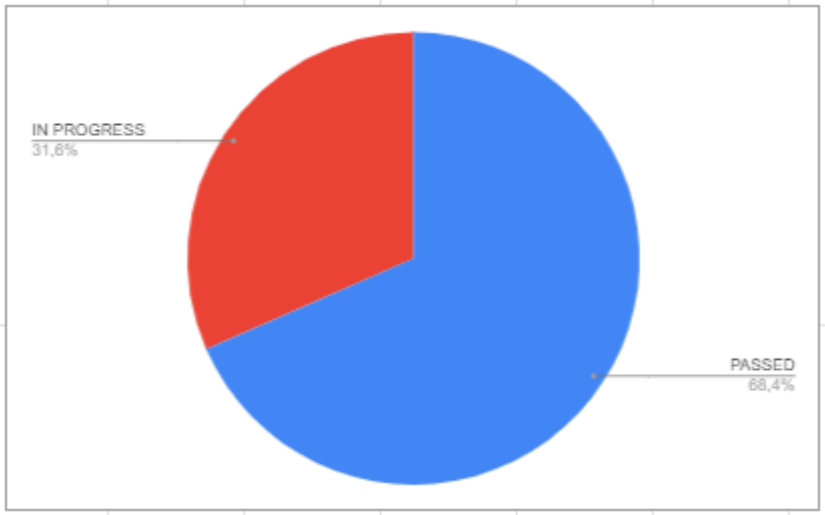
Sprint 1



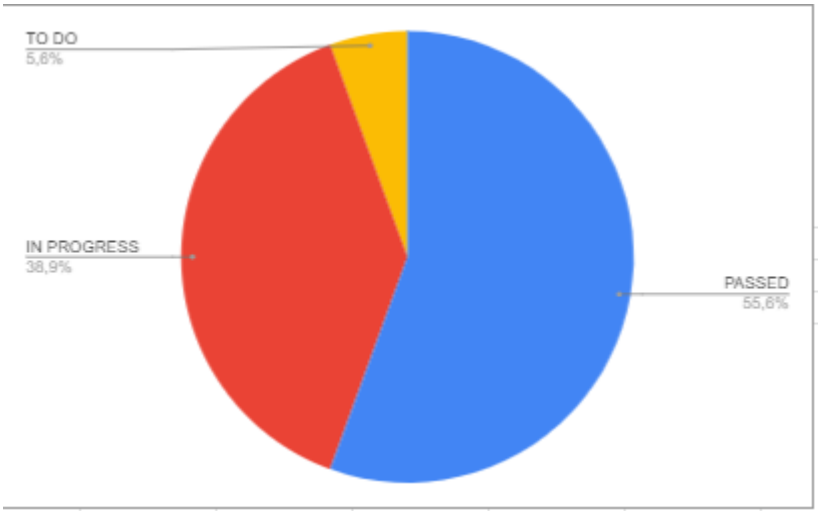
Sprint 2



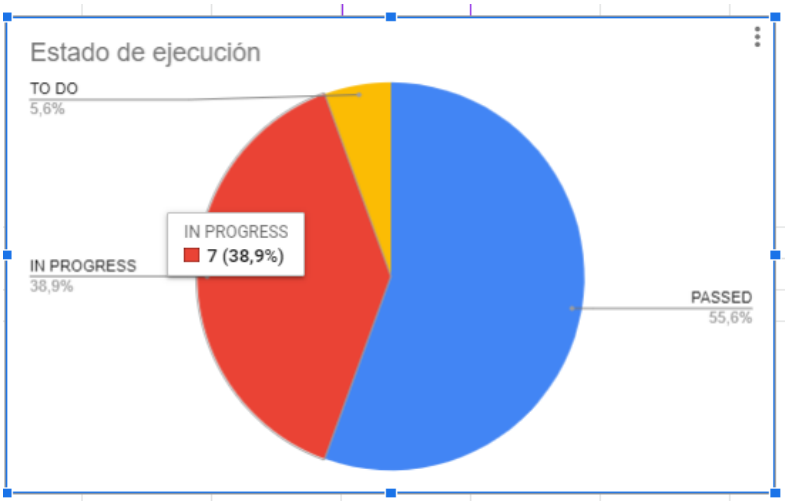
Sprint 3



Sprint 4



Sprint 4 por ejecución



Carry Over

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

- No logramos acceder a una reserva desde el front
- No se logró crear una página para ver las reservas de los usuarios.
- Método que nos permita filtrar productos por ciudad y dos fechas
- Crear la página de "creación de producto exitosa"

Lecciones Aprendidas / Conclusión

El equipo reconoció lo importante que es actuar y comunicarse de manera consistente. Llevar un buen sentido del orden a nivel personal y en las herramientas que usamos para organizarnos. Entendimos que coordinar un trabajo de esta envergadura es algo que requiere buenos manejos sociales.
Aceptamos que el aprendizaje constante es una realidad de la programación. Y que el correcto uso del tiempo de lo assentes técnicos es fundamental para destrabar issues
Finalmente La metodología de testing fue evolucionando a medida que solidificamos los aprendido durante la carrera. A mayor complejidad, mayor era el orden en el que procedemos. Y también la velocidad en la que encaramos una nueva funcionalidad, para asegurarnos que funcione bien lo antes posible