
# Alquifiestas: Sensacional Eventos
# Sprint 7

Hugo Eduardo Rivas Fajardo - 22500
José Santiago Pereira Alvarado - 22318
Nancy Gabriela Mazariegos Molina - 22513
Giovanni Alejandro Santos Hernández - 22523
Mauricio Julio Rodrigo Lemus Guzmán- 22461
Alexis Mesías Flores - 22562

**Product Backlog**

**Pila del producto**
https://docs.google.com/spreadsheets/d/1pHFa_t3TDiU_gIoxtiY6VaneJB60KE6gEEqGTWLpKHg/edit?gid=1739314590#gid=1739314590

**Gestión de tareas**

https://docs.google.com/spreadsheets/d/1pHFa_t3TDiU_gIoxtiY6VaneJB60KE6gEEqGTWLpKHg/edit?gid=0#gid=0

**Sprint Backlog**

https://docs.google.com/spreadsheets/d/1pHFa_t3TDiU_gIoxtiY6VaneJB60KE6gEEqGTWLpKHg/edit?gid=1378010058#gid=1378010058

**Incremento**

- Código desarrollado
    - Vínculo al repositorio: https://github.com/Rodlemus03/SensacionalEventos
- Lista de funcionalidades planificadas que se terminaron completamente.
    - Modificación de pruebas automatizadas, implementación de guardado en base de datos para  la visualización del tablero de QA.

**Prueba**

**Master plan de unit test:**

1. Configuración Inicial

Se usa selenium para la automatización de las pruebas funcionales del sistema. Se crea el modelo propuesto en PostgreSql para almacenar los resultados de las pruebas, ya sean exitosas, fallidas, ignoradas, etc. Al finalizar cada prueba se guarda el resultado esperado vs el resultado real, para poder cotejar la calidad del software.

FrameWorks:

- Selenium
- PsycoPg2

Herramientas externas

- Postman
- PgAdmin
- GitHub actions

2. Plan de Pruebas

**Clientes (CRUD)**

Create Cliente:
Prueba de éxito:
- Enviar una solicitud POST con datos válidos para crear un cliente.
- Verificar que el cliente se creó correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud POST con datos inválidos (e.g., campos obligatorios faltantes).
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Read Cliente:
Prueba de éxito:
- Enviar una solicitud GET para recuperar un cliente existente.
- Verificar que los datos del cliente sean correctos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud GET para recuperar un cliente inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Update Cliente:
Prueba de éxito:
- Enviar una solicitud PUT con datos válidos para actualizar un cliente.
- Verificar que los datos del cliente se actualizaron correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud PUT con datos inválidos.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Delete Cliente:
Prueba de éxito:
- Enviar una solicitud DELETE para eliminar un cliente existente.
- Verificar que el cliente se eliminó correctamente de la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud DELETE para eliminar un cliente inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

**Inventario (CRUD)**

Create Inventario:
Prueba de éxito:
- Enviar una solicitud POST con datos válidos para crear un ítem de inventario.
- Verificar que el ítem se creó correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud POST con datos inválidos.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Read Inventario:
Prueba de éxito:
- Enviar una solicitud GET para recuperar un ítem de inventario existente.
- Verificar que los datos del ítem sean correctos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud GET para recuperar un ítem inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Update Inventario:
Prueba de éxito:
- Enviar una solicitud PUT con datos válidos para actualizar un ítem de inventario.
- Verificar que los datos del ítem se actualizaron correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud PUT con datos inválidos.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Delete Inventario:
Prueba de éxito:
- Enviar una solicitud DELETE para eliminar un ítem de inventario existente.
- Verificar que el ítem se eliminó correctamente de la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud DELETE para eliminar un ítem inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

**Pedidos (CRUD)**

Create Pedido:
Prueba de éxito:
- Enviar una solicitud POST con datos válidos para crear un pedido.
- Verificar que el pedido se creó correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud POST con datos inválidos.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Read Pedido:
Prueba de éxito:
- Enviar una solicitud GET para recuperar un pedido existente.
- Verificar que los datos del pedido sean correctos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud GET para recuperar un pedido inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Update Pedido:
Prueba de éxito:
- Enviar una solicitud PUT con datos válidos para actualizar un pedido.
- Verificar que los datos del pedido se actualizaron correctamente en la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud PUT con datos inválidos.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

Delete Pedido:
Prueba de éxito:
- Enviar una solicitud DELETE para eliminar un pedido existente.
- Verificar que el pedido se eliminó correctamente de la base de datos.
- Guardar el resultado de la prueba en la base de datos.

Prueba de fallo:
- Enviar una solicitud DELETE para eliminar un pedido inexistente.
- Verificar que se recibe una respuesta de error adecuada.
- Guardar el resultado de la prueba en la base de datos.

3. Automatización con Selenium

Login:

- Verificar que el usuario pueda iniciar sesión correctamente.
- Verificar que los intentos de inicio de sesión fallidos muestren mensajes de error adecuados.
- Guardar el resultado de cada prueba en la base de datos.

Navegación:

- Verificar que los usuarios puedan navegar entre las diferentes páginas (Clientes, Inventario, Pedidos) correctamente.
- Guardar el resultado de cada prueba en la base de datos.

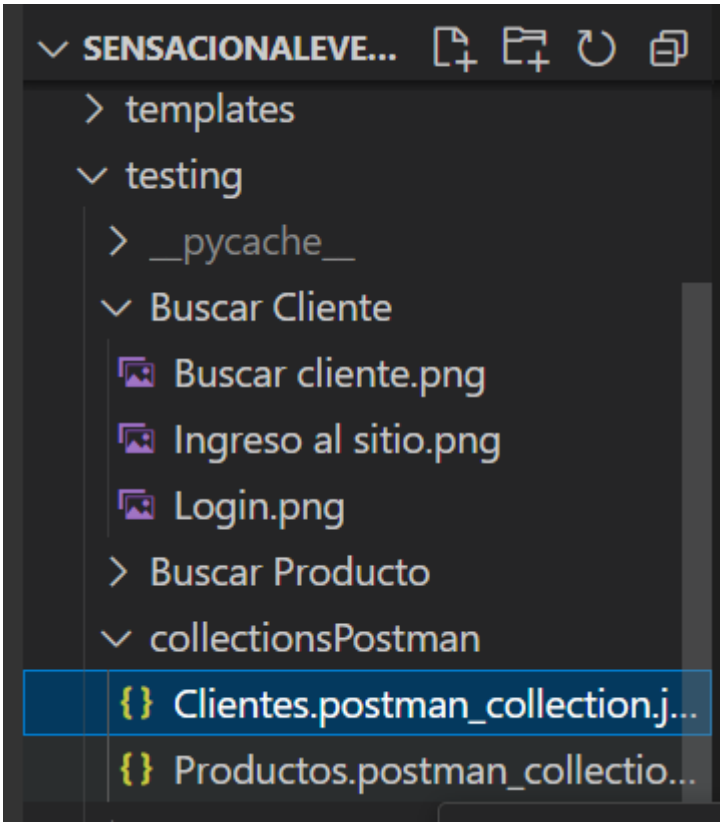
Formularios:

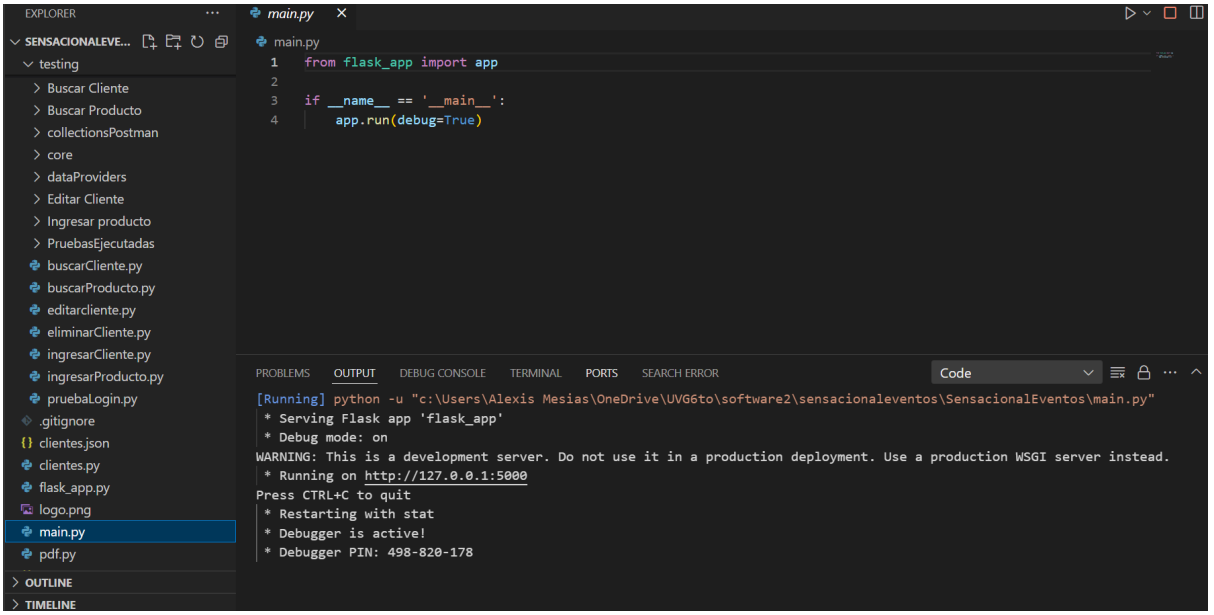
- Verificar que los formularios de creación, actualización y eliminación funcionen correctamente para cada módulo (Clientes, Inventario, Pedidos).
- Guardar el resultado de cada prueba en la base de datos.

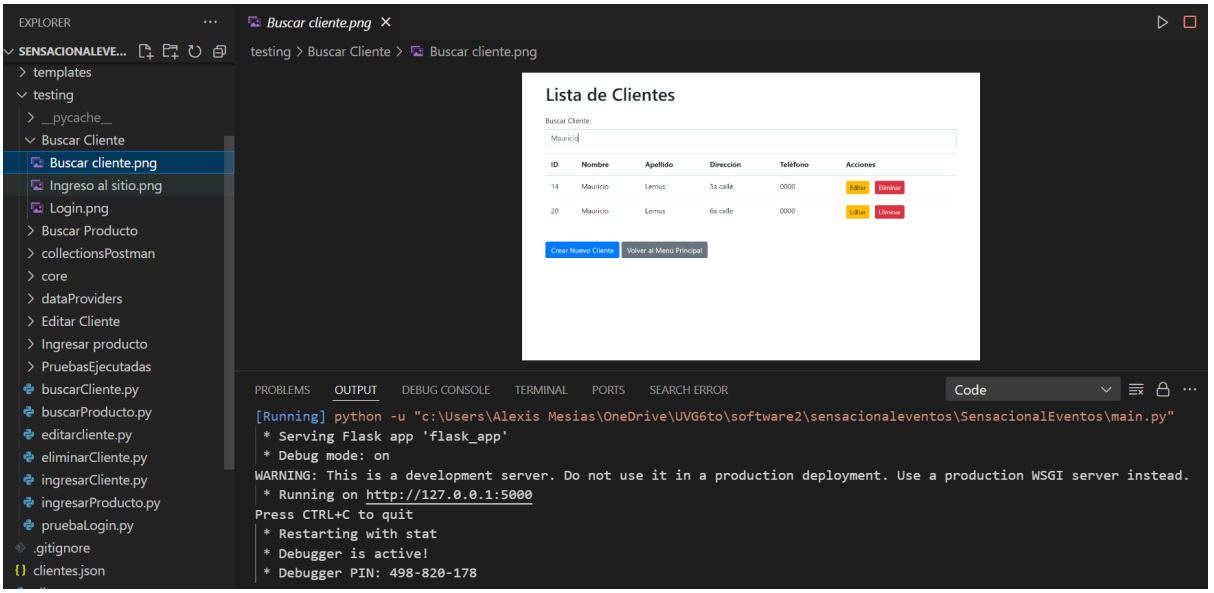
4. Ejecución de Pruebas
   - Crear scripts de Postman para todas las pruebas de API.
   - Utilizar Pytest para organizar y ejecutar las pruebas.
   - Configurar Selenium para automatizar las pruebas de interfaz de usuario.
   - Registrar los resultados de cada prueba en la base de datos.
   - Ejecutar las pruebas en un entorno de CI/CD para garantizar que las pruebas se ejecuten automáticamente con cada cambio en el código.

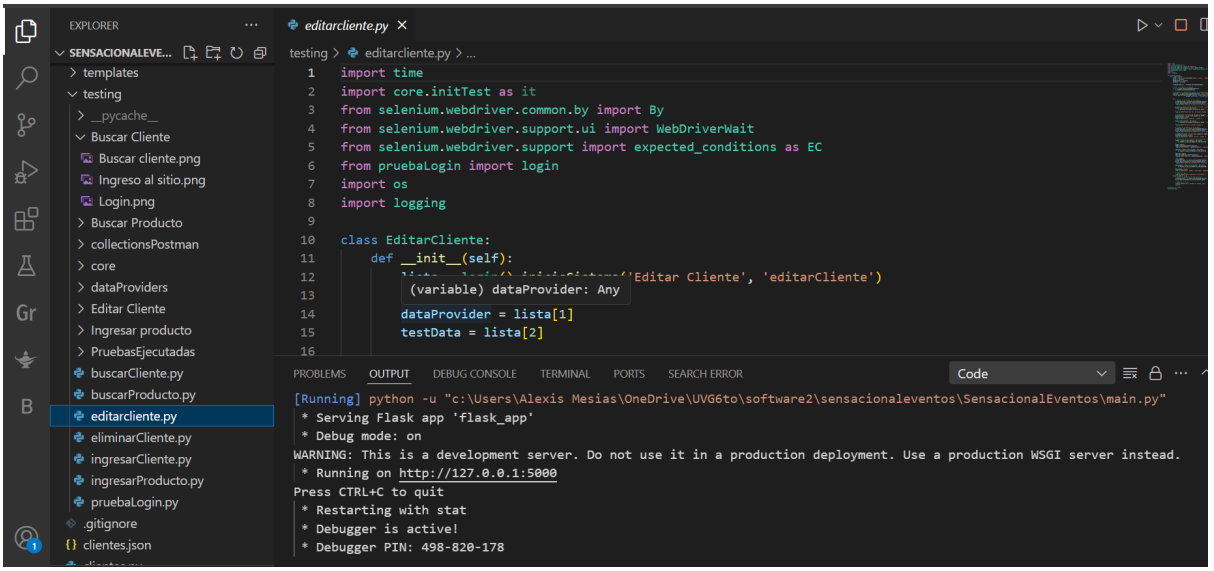
5. Informe de Resultados
   - Generar informes detallados con los resultados de las pruebas.
   - Incluir información sobre las pruebas que fallaron, junto con los detalles del error.
   - Incluir referencias a los registros almacenados en la base de datos.
   - Revisar y corregir los errores identificados durante las pruebas.

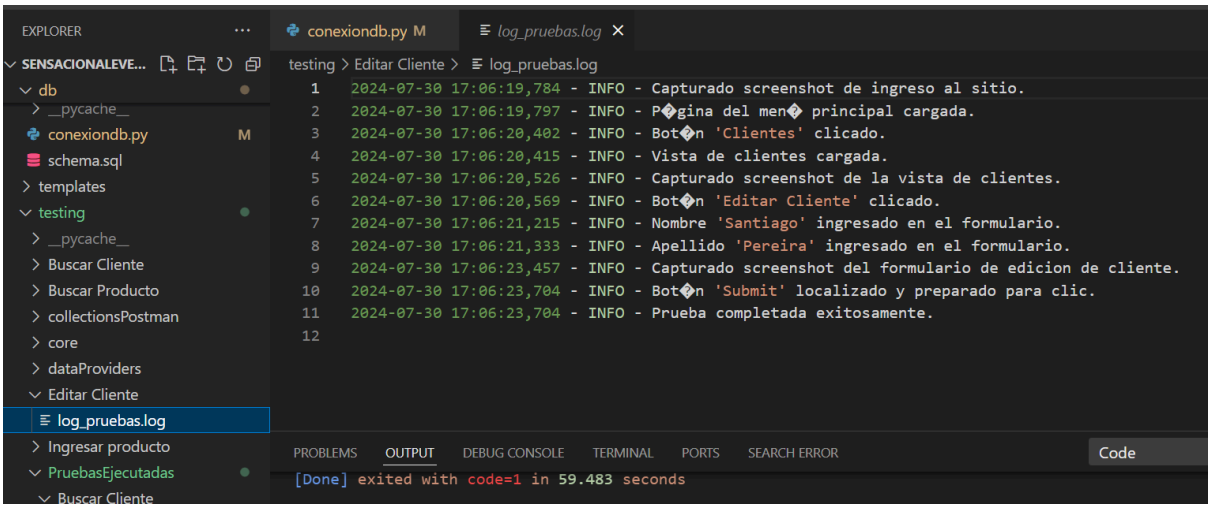
- Evidencias de implementación de pruebas unitarias en la herramienta automatizada y resultados obtenidos al ejecutarlas.



- 

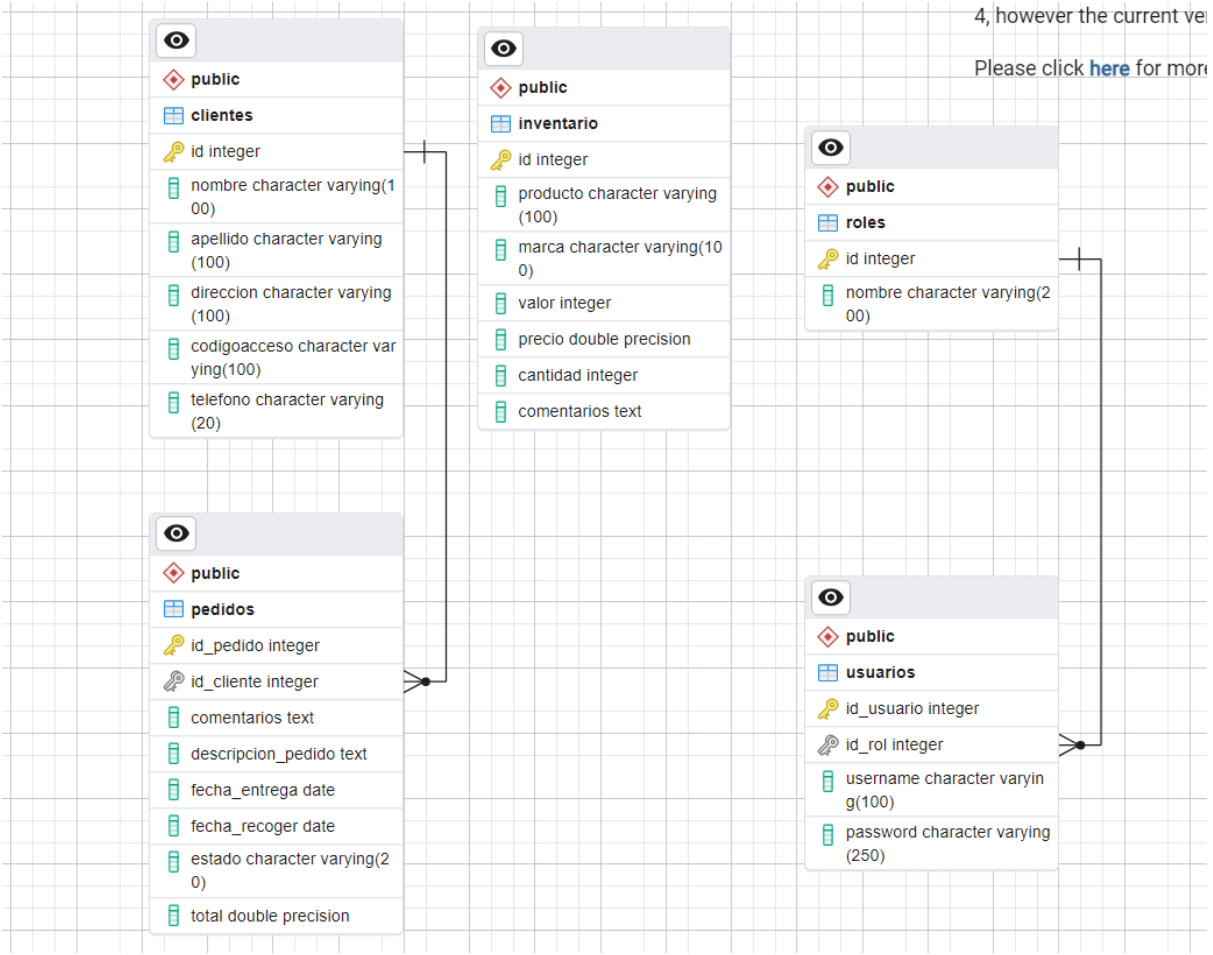
- 

- 

- Se realizaron pruebas unitarias para este sprint utilizando Postman. Luego, se llevaron a cabo pruebas automatizadas mediante Selenium, el cual, al ejecutar la prueba, guarda una serie de capturas de pantalla que se toman automáticamente en cada paso

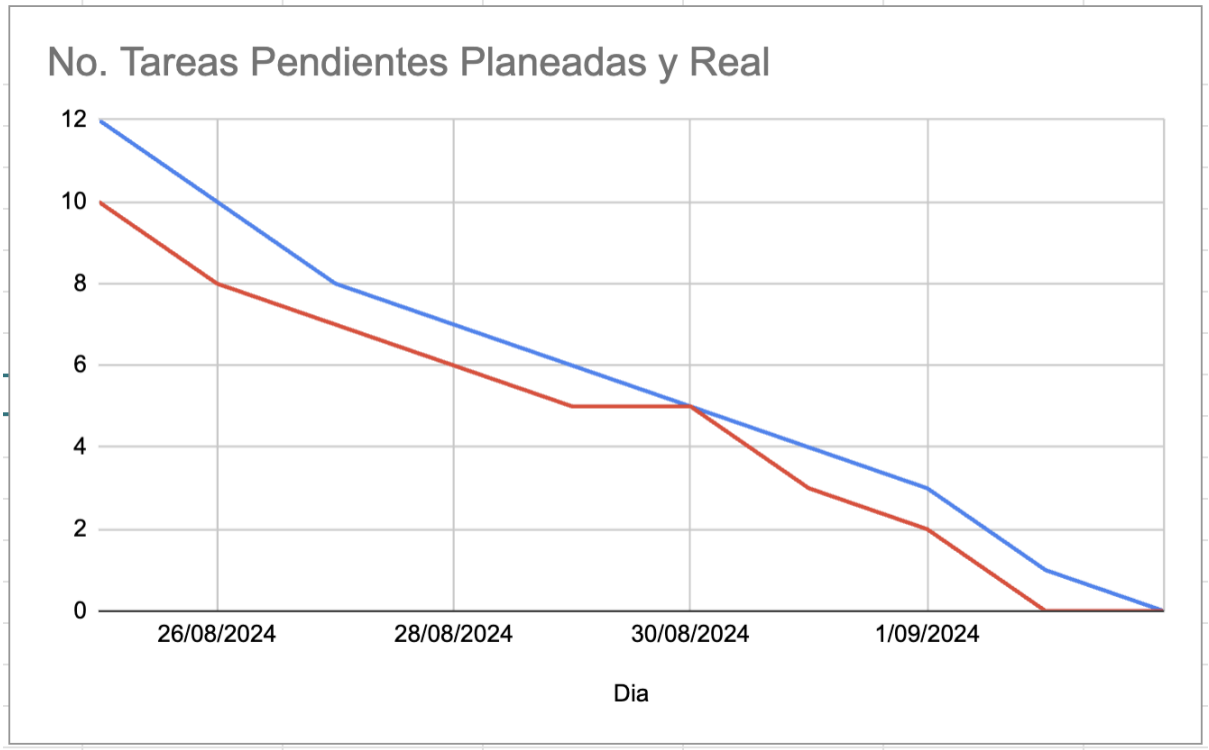
de la prueba. También se guarda un registro del paso a paso de cada prueba, y si se genera algún tipo de error, se describe en qué paso sucedió.

ERD DATABASE:




**Resultados**

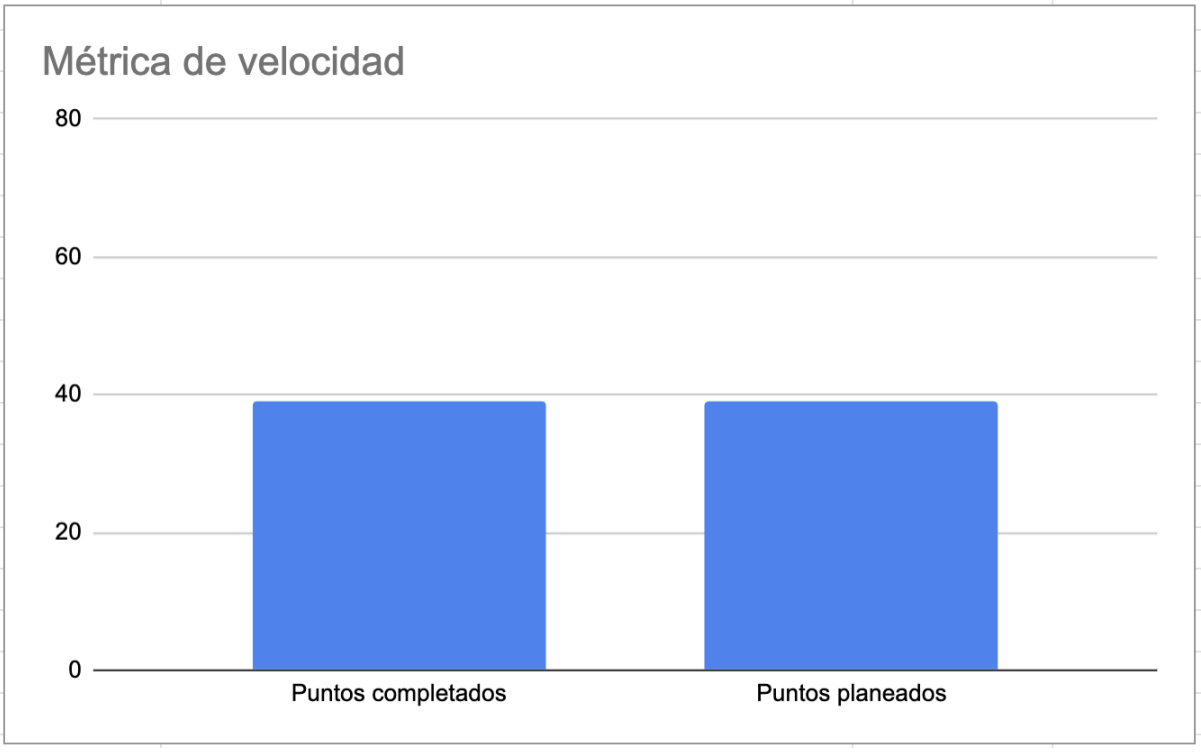
- **Gráfico burndown**




- **Métrica de velocidad**

  **Puntos de historia planificados: 43**

  **Puntos de historia terminados: 43**



**Velocidad del equipo en el sprint** $= \frac{43}{43} \; - \; 0 \; = \; 100\%$

- Como se puede observar en los gráficos que se presentaron anteriormente, se puede observar que el equipo pudo concluir con las metas propuestas para este sprint, ya que se tenían doce tareas y se completaron las doce en el tiempo estipulado.

- Evidencias de muestra del incremento desarrollado al product owner.


- **Retrospectiva del sprint.**

  Durante este Sprint, observamos un claro incremento en la eficiencia del equipo, y esta vez no enfrentamos retrasos en ninguna área. Cumplimos con todos nuestros objetivos, lo cual se refleja tanto en el avance del sprint como en el burndown chart, que muestran una notable mejora en nuestra productividad y efectividad.

  A diferencia de sprints anteriores, en este nos enfocamos en una variedad de tareas nuevas. Este cambio diversificó nuestro enfoque, permitiendo que el equipo desarrollara nuevas habilidades y enfrentara los desafíos con más eficacia. Logramos completar las pruebas planificadas dentro del tiempo previsto, lo que demuestra una mejora en la gestión del tiempo y los recursos.

La distribución del trabajo se ajustó significativamente, lo que nos ayudó a prever posibles problemas de tiempo y equilibrar mejor las cargas de trabajo. Gracias a estas mejoras, mantuvimos un ritmo constante y productivo durante todo el sprint.

**Conclusiones:**

- Las pruebas que hicimos en este sprint, aunque fueron un poco complicadas, nos dieron más confianza en el proyecto.
- Conforme avanzamos en los sprints, estamos trabajando de manera más efectiva y entregando tareas de forma más organizada.
- Enfocarnos en entregar diferentes tipos de tareas durante el sprint nos ha ayudado a avanzar más.

**Gestión del tiempo**
**Nombre:** Mauricio Julio Rodrigo Lemus Guzmán
**Carné:** 22461

| **Fecha** | **Inicio** | **Fin** | **Tiempo Interrupción** | **Delta tiempo** | **Fase** | **Comentarios** |
|---|---|---|---|---|---|---|
| 11-08-24 | 10:00 | 12:00 | 20 minutos | 1 hr 40 min | Sprint 6 | Empecé la coleccion de postman para crear pedido |
| 11-08-24\| | 2:00 | 2:30 | 0 minutos | 30 minutos | Sprint 6 | Terminé la colección de postman para crear un pedido |
| 12-08-24 | 2:00 | 3:30 | 30 minutos | 1 hora | Sprint 6 | Comencé test en scripts de postman |
| 13-08-24 | 2:00 | 3:30 | 30 minutos | 1 hora | Sprint 6 | Terminé test en scripts de postman |

**Nombre:** José Santiago Pereira Alvarado
**Carné:** 22318

| **Fecha** | **Inicio** | **Fin** | **Tiempo Interrupción** | **Delta tiempo** | **Fase** | **Comentarios** |
|---|---|---|---|---|---|---|
| 2-09-2024 | 9:00 | 9:10 | 0 minutos | 10 minutos | Sprint 7 | Realice cambios a la |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | prueba de eliminarCliente |
| 2-09-2024 | 11:30 | 11:35 | 0 minutos | 5 minutos | Sprint 7 | Realice cambios a la prueba de eliminarPedido |

**Nombre:** Nancy Gabriela Mazariegos Molina
**Carné:** 22513

| **Fecha** | **Inicio** | **Fin** | **Tiempo Interrupción** | **Delta tiempo** | **Fase** | **Comentarios** |
|---|---|---|---|---|---|---|
| 2/09 | 21:00 | 21:35 | 5 min | 30 min | Sprint 7 | Almacenamiento de prueba de buscar cliente en bd |
| 3/09 | 14:45 | 15:15 | 0 min | 30 min | Sprint 7 | Almacenamiento de prueba de buscar pedido en bd |

**Nombre:** Hugo Eduardo Rivas Fajardo
**Carné:** 22500

| **Fecha** | **Inicio** | **Fin** | **Tiempo Interrupción** | **Delta tiempo** | **Fase** | **Comentarios** |
|---|---|---|---|---|---|---|
| 30/08 | 10:00 | 11:00 | 20 mins | 40 mins | Sprint 7 | Prueba ingresar Clientes |
| 31/08 | 9:00 | 9:30 | 0 mins | 30 mins | Sprint 7 | Prueba ingresar Clientes |
| 31/08 | 10:00 | 11:00 | 20 minutos | 40 mins | Sprint 7 | Prueba eliminar Producto |
| 1/09 | 12:00 | 12:20 | 10 mins | 10 mins | Sprint 7 | Prueba eliminar Producto |

**Nombre:** Giovanni Alejandro Santos Hernández
**Carné:** 22523

| Fecha | Inicio | Fin | Tiempo Interrupción | Delta tiempo | Fase | Comentarios |
|---|---|---|---|---|---|---|
| 2/09 | 11:00 | 11:35 | 5 min | 30 min | Sprint 7 | Inicio update prueba de buscar producto |
| 2/09 | 18:00 | 19:30 | 5 min | 25 min | Sprint 7 | Finalizado el update prueba de buscar producto |
| 3/09 | 12:00 | 12:50 | 5 min | 45 min | Sprint 7 | Update de buscar producto |

**Nombre:** Alexis Mesías Flores
**Carné:** 22562

| Fecha | Inicio | Fin | Tiempo Interrupción | Delta tiempo | Fase | Comentarios |
|---|---|---|---|---|---|---|
| 2/09/2024 | 20:00 | 21:10 | 30min | 40 min | Sprint 7 | Inicio Almacenamiento de prueba de Editar cliente en bd |
| 3/09/24 | 8:00 | 8:40 | 20 min | 20 min | Sprint 7 | Finalizado el Almacenamiento de prueba de Editar cliente en bd |
| 3/09/24 | 12:00 | 13:20 | 20 min | 60 min | Sprint 7 | Almacenamiento de prueba de Ingresar Producto en bd |
| | | | | | | |

## Referencias


- *Postman Collections: Organize API Development and Testing*. (s. f.). Postman API Platform. https://www.postman.com/collection/

- García, B. (s. f.). *Boni García*. https://bonigarcia.dev/