Alquifiestas: Sensacional Eventos
Sprint 8

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.
    - Creación de página de notificaciones de pedidos pendientes.
    - Creación de tablas en la BD de las notificaciones.
    - Adecuación de servicios en backend.
    - Creación de disparador .bat

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

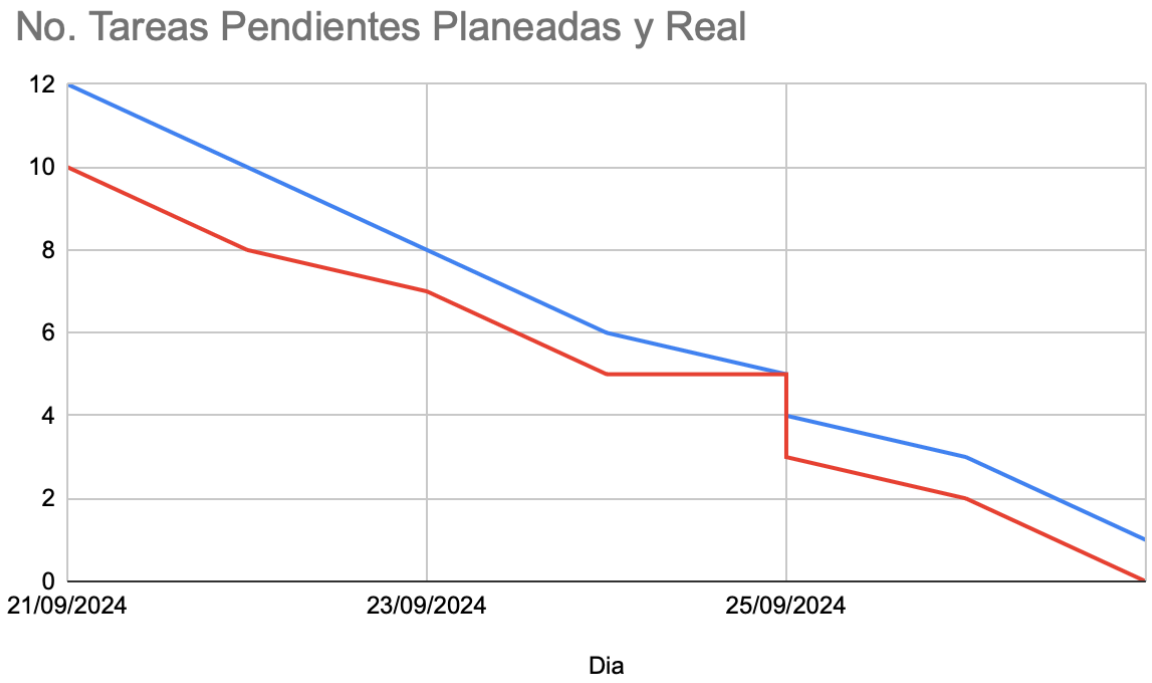
- Crear scripts de Postman para todas las pruebas de API.
- Utilizar Pytest para organizar y ejecutar las pruebas.
- Configurar Selenium para automatizar las pruebas de interfaz de usuario.
- Registrar los resultados de cada prueba en la base de datos.
- Ejecutar las pruebas en un entorno de CI/CD para 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.

Resultados

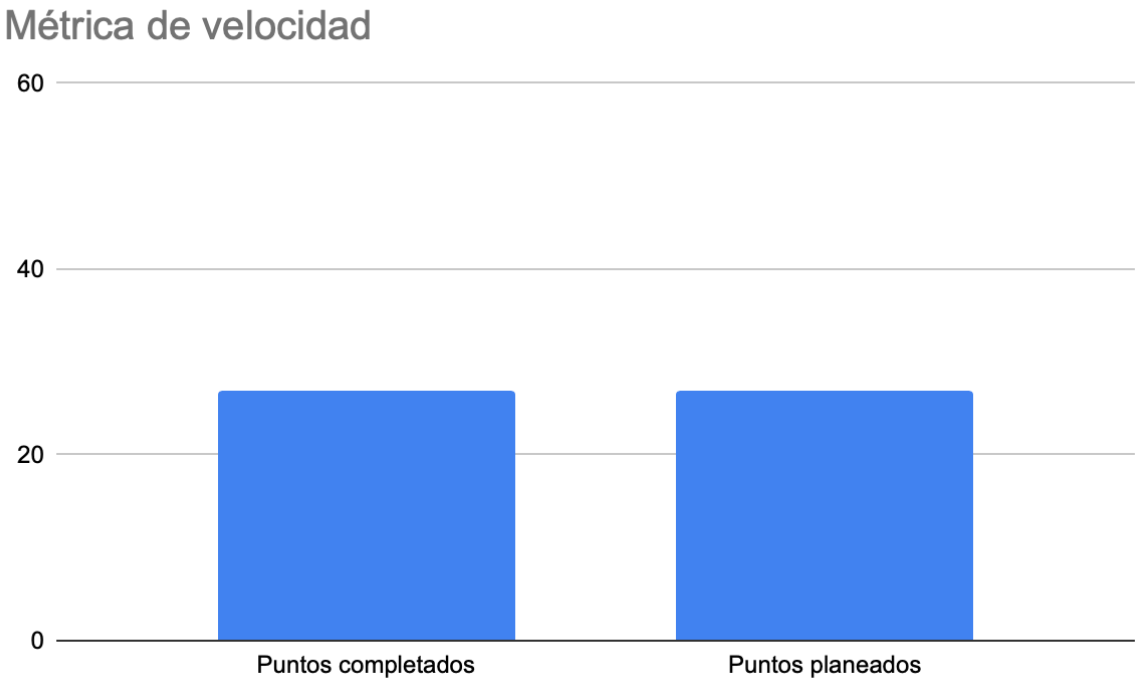
- **Gráfico burndown**



- **Métrica de velocidad**

  Puntos de historia planificados: 24

  Puntos de historia terminados: 24



Velocidad del equipo en el sprint $= \frac{27}{27} \; - \; 0 \; = \; 100\%$

- 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.

- **Presupuesto**

Descripción	Monto
Server EC2	$ 20.00
Dominio	$ 1.40
Salarios (6 empleados)	$ 2,240.00
Sistema Declaraguate	$ 20.00
Renta de 6 Computadoras	$ 1,200.00
Backup Físico	$ 50.00
Internet	$ 50.00
Total	$ 3,581.40

- **Retrospectiva del sprint.**

  Para este Sprint, pudimos realizar nuevas implementaciones más que todo con el objetivo de seguir facilitando los procesos que conlleva el tomar y realizar pedidos. También hemos visto que el ritmo de trabajo ha sido bastante bueno ya que se han logrado las tareas propuestas, lo que nos ha funcionado bastante bien es la división de trabajo equitativo, cabe resaltar que no sólo el hecho de que se divida en partes iguales el trabajo será más fácil si no que ver el tipo de trabajo a realizar para poder asignarle a cada colaborador dependiendo de sus conocimientos y habilidad con la herramienta.

  La distribución del trabajo y el tiempo asignado fue lo que nos ayudó a que no nos atrasarnos ni lo hiciéramos a la carrera. Logramos cumplir todos los objetivos que se tenían para este sprint, no tuvimos ningún percance gracias a todo lo que fue mencionado anteriormente. Si nos ponemos a analizar el progreso de los sprints anteriores nos damos cuenta que cada vez vamos mejorando o en dado caso nos quedamos igual lo cuál es bueno por que no hemos bajado en el rendimiento.

Conclusiones:

- Las funciones que implementamos para este sprint serán de gran ayuda para facilitar el trabajo de tomar y realizar los pedidos.
- La organización tanto de tiempo como de tareas fue bastante buena ya que no tuvimos inconvenientes en la realización de las mismas.
- El enfoque no estuvo en las pruebas si no que en agregar nuevas cosas para que se vea un poco más complejo.

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 8	Empecé la coleccion de postman para crear pedido
11-08-24|	2:00	2:30	0 minutos	30 minutos	Sprint 8	Terminé la colección de postman para crear un pedido
12-08-24	2:00	3:30	30 minutos	1 hora	Sprint 8	Comencé test en scripts de postman
13-08-24	2:00	3:30	30 minutos	1 hora	Sprint 8	Terminé test en scripts de postman

Nombre: José Santiago Pereira Alvarado
Carné: 22318

Fecha	Inicio	Fin	Tiempo Interrupción	Delta tiempo	Fase	Comentarios
24-09-2024	10:00	10:15	0 minutos	15 minutos	Sprint 8	Creacion de la pagina HMTL de notificaciones
25-09-2024	1:30	2:00	10 minutos	20 minutos	Sprint 8	Implementacion de pagina de notificaciones

Nombre: Nancy Gabriela Mazariegos Molina
Carné: 22513

Fecha	Inicio	Fin	Tiempo Interrupción	Delta tiempo	Fase	Comentarios
25/09	20:00	21:20	20 min	1 hora	Sprint 8	Función de aceptar notificaciones.
25/09	23:15	23:45	0 min	45 min	Sprint 8	Función de eliminar

						notificaciones.
26/09	18:00	18:30	15 min	15 min	Sprint 8	Renderizado.

Nombre: Hugo Eduardo Rivas Fajardo
Carné: 22500

Fecha	**Inicio**	**Fin**	**Tiempo Interrupción**	**Delta tiempo**	**Fase**	**Comentarios**
23/09	14:00	15:40	10 mins	30 mins	Sprint 8	Crear tabla de notificaciones
24/09	16:45	17:15	0 mins	30 mins	Sprint 8	Index notificaciones
25/09	20:00	21:00	20 minutos	40 mins	Sprint 8	Index notificaciones mensajes
25/09	21:30	22:20	10 mins	40 mins	Sprint 8	Index notificaciones aceptado

Nombre: Giovanni Alejandro Santos Hernández
Carné: 22523

Fecha	**Inicio**	**Fin**	**Tiempo Interrupción**	**Delta tiempo**	**Fase**	**Comentarios**
25/09	12:15	1:30	15 min	1 hora	Sprint 8	Inicio update prueba de buscar producto

Nombre: Alexis Mesías Flores
Carné: 22562

Fecha	**Inicio**	**Fin**	**Tiempo Interrupción**	**Delta tiempo**	**Fase**	**Comentarios**
23/09	18:05	19:20	25 min	50 min	Sprint 8	Tabla notificaciones BDD

24/09	21:00	22:00	20 min	40 min	Sprint 8	Index notificaciones ID BDD
25/09	20:15	20:40	5 min	30 min	Sprint 8	Index notificaciones mensajes BDD
26/09	22:05	22:45	10 min	30 min	Sprint 8	Index notificaciones aceptado BDD

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/