Alquifiestas: Sensacional Eventos
Sprint 6

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.
    - Pruebas postman collections pedidos
    - Diseño de base de datos de automatización
    - Creación de base de datos de automatización
    - Prueba automatizada de buscar pedido
    - Prueba automatizada de eliminar producto
    - Prueba automatizada de crear pedido
    - Prueba automatizada de eliminar pedido

Prueba

Master plan de unit test:

1. Configuración Inicial

- Frameworks y Herramientas:
  - Postman para pruebas de API.
  - Selenium para automatización de pruebas de interfaz de usuario.
  - Pytest para organizar y ejecutar las pruebas.
  - Flask-Testing para facilitar las pruebas en Flask.
  - PostgreSQL como base de datos.
  - Docker para ambientes de prueba aislados (opcional pero recomendado).

2. Plan de Pruebas

Clientes (CRUD)

1. 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.
   - Prueba de fallo:
     - Enviar una solicitud POST con datos inválidos (e.g., campos obligatorios faltantes).
     - Verificar que se recibe una respuesta de error adecuada.

2. Read Cliente:
   - Prueba de éxito:
     - Enviar una solicitud GET para recuperar un cliente existente.
     - Verificar que los datos del cliente sean correctos.
   - Prueba de fallo:
     - Enviar una solicitud GET para recuperar un cliente inexistente.
     - Verificar que se recibe una respuesta de error adecuada.

3. 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.
   - Prueba de fallo:
     - Enviar una solicitud PUT con datos inválidos.
     - Verificar que se recibe una respuesta de error adecuada.

4. 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.
   - Prueba de fallo:
     - Enviar una solicitud DELETE para eliminar un cliente inexistente.
     - Verificar que se recibe una respuesta de error adecuada.

Inventario (CRUD)

1. 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.
   - Prueba de fallo:
     - Enviar una solicitud POST con datos inválidos.

- Verificar que se recibe una respuesta de error adecuada.

2. Read Inventario:
  - Prueba de éxito:
    - Enviar una solicitud GET para recuperar un ítem de inventario existente.
    - Verificar que los datos del ítem sean correctos.
  - Prueba de fallo:
    - Enviar una solicitud GET para recuperar un ítem inexistente.
    - Verificar que se recibe una respuesta de error adecuada.

3. 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.
  - Prueba de fallo:
    - Enviar una solicitud PUT con datos inválidos.
    - Verificar que se recibe una respuesta de error adecuada.

4. 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.
  - Prueba de fallo:
    - Enviar una solicitud DELETE para eliminar un ítem inexistente.
    - Verificar que se recibe una respuesta de error adecuada.

Pedidos

1. 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.
  - Prueba de fallo:
    - Enviar una solicitud POST con datos inválidos.
    - Verificar que se recibe una respuesta de error adecuada.

2. Read Pedido:
  - Prueba de éxito:
    - Enviar una solicitud GET para recuperar un pedido existente.
    - Verificar que los datos del pedido sean correctos.
  - Prueba de fallo:
    - Enviar una solicitud GET para recuperar un pedido inexistente.
    - Verificar que se recibe una respuesta de error adecuada.

3. 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.
  - Prueba de fallo:
    - Enviar una solicitud PUT con datos inválidos.
    - Verificar que se recibe una respuesta de error adecuada.

4. 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.
  - Prueba de fallo:
    - Enviar una solicitud DELETE para eliminar un pedido inexistente.
    - Verificar que se recibe una respuesta de error adecuada.

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.

- Navegación:
  - Verificar que los usuarios puedan navegar entre las diferentes páginas (Clientes, Inventario, Pedidos) correctamente.

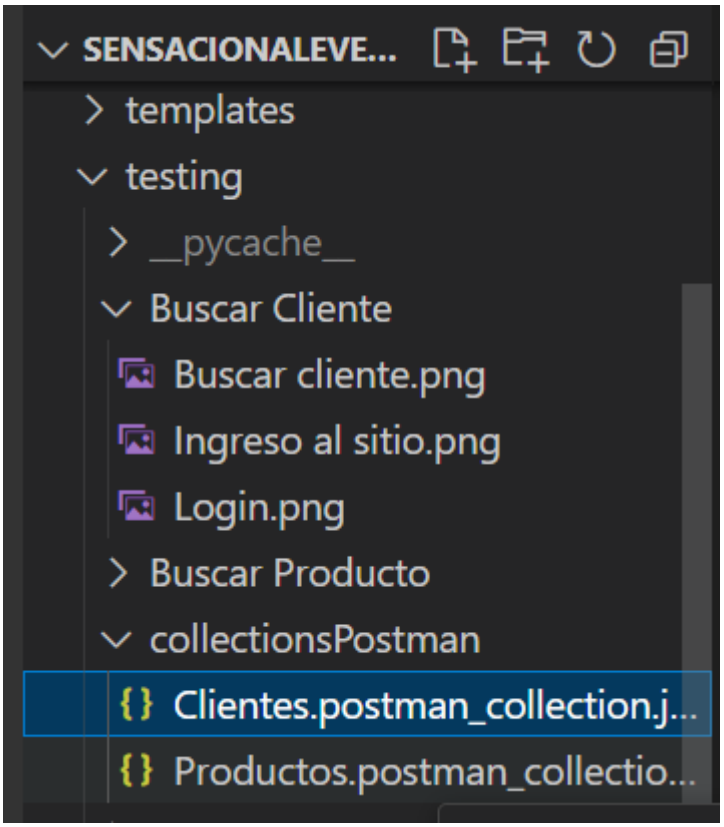
- Formularios:
  - Verificar que los formularios de creación, actualización y eliminación funcionen correctamente para cada módulo (Clientes, Inventario, Pedidos).

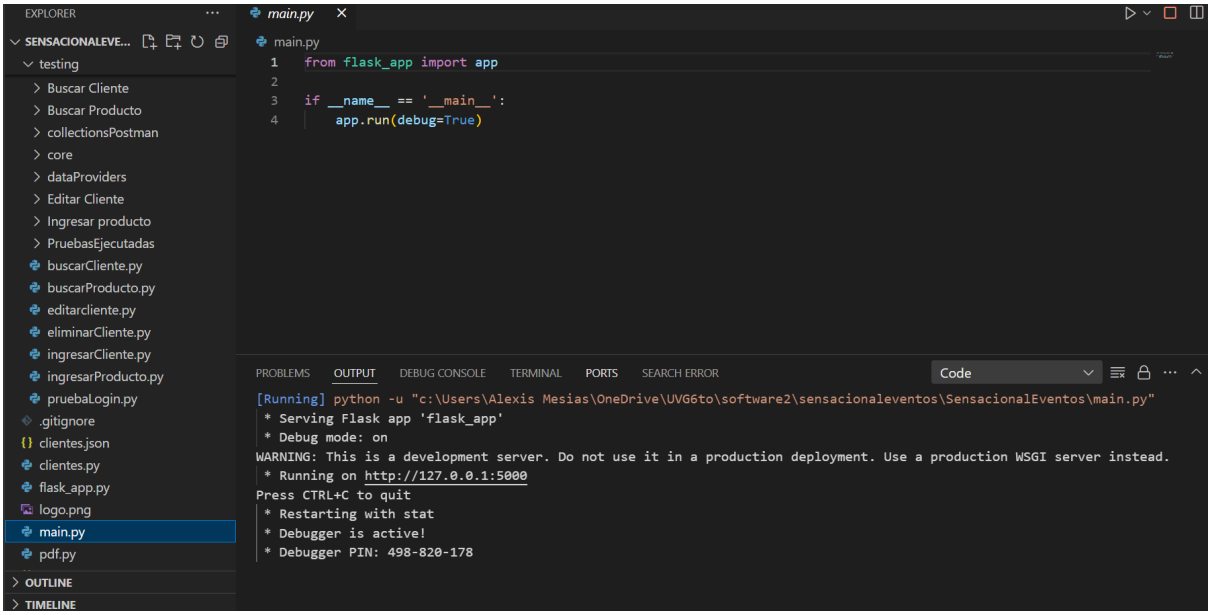
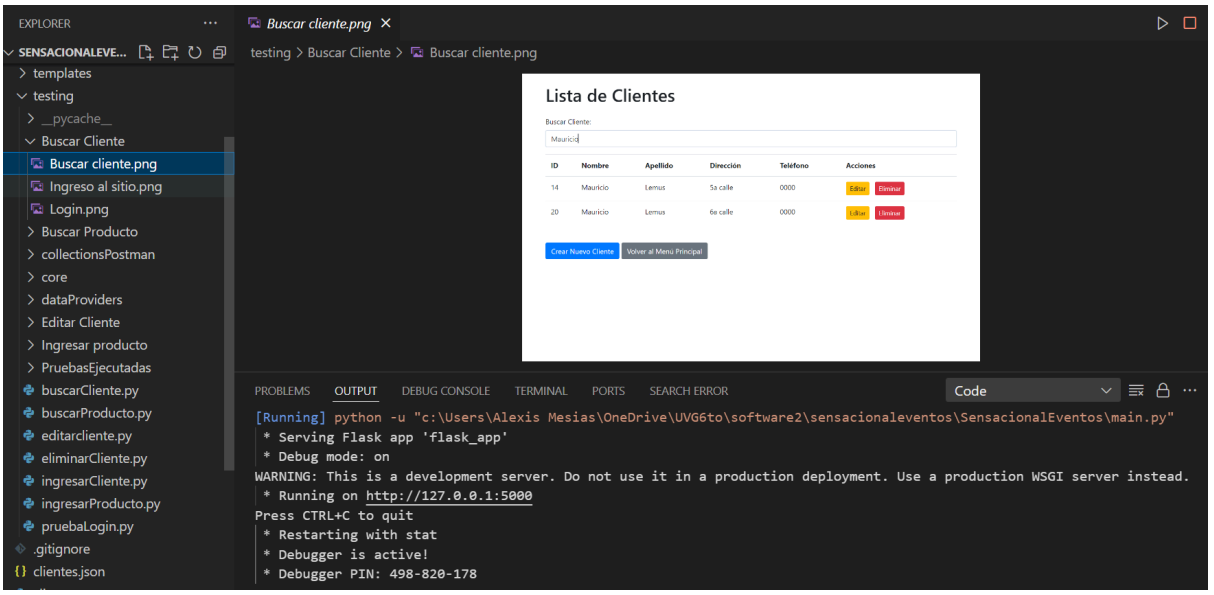
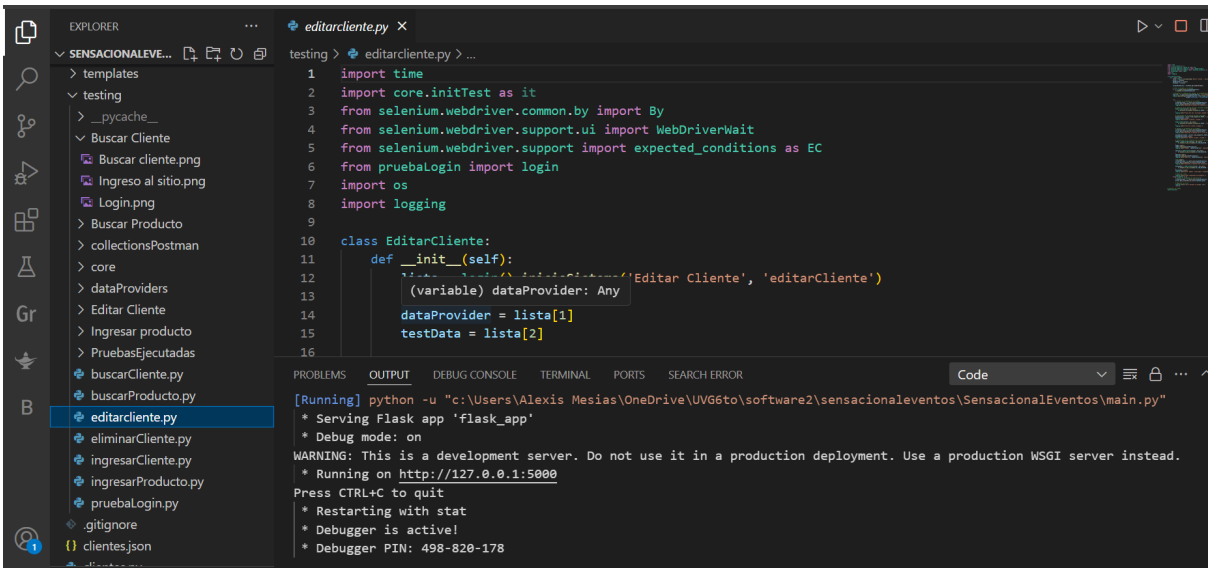
4. Ejecución de Pruebas

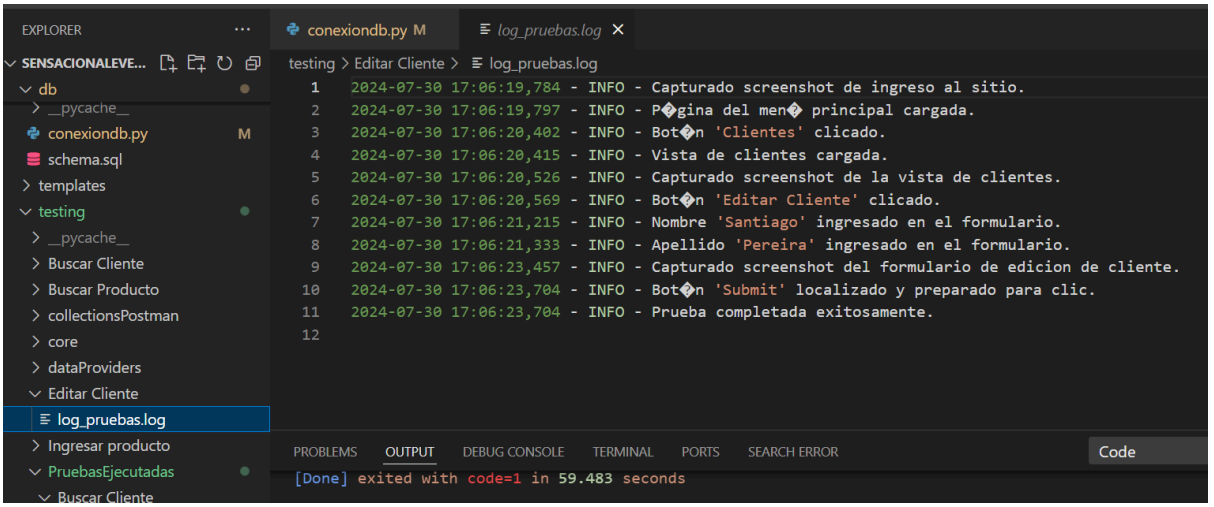
- Crear scripts de Postman para todas las pruebas de API.
- Utilizar Pytest para organizar y ejecutar las pruebas.
- Configurar Selenium para automatizar las pruebas de interfaz de usuario.
- Ejecutar las pruebas en un entorno de CI/CD para garantizar que las pruebas se ejecuten automáticamente con cada 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.
- Revisar y corregir los errores identificados durante las pruebas.

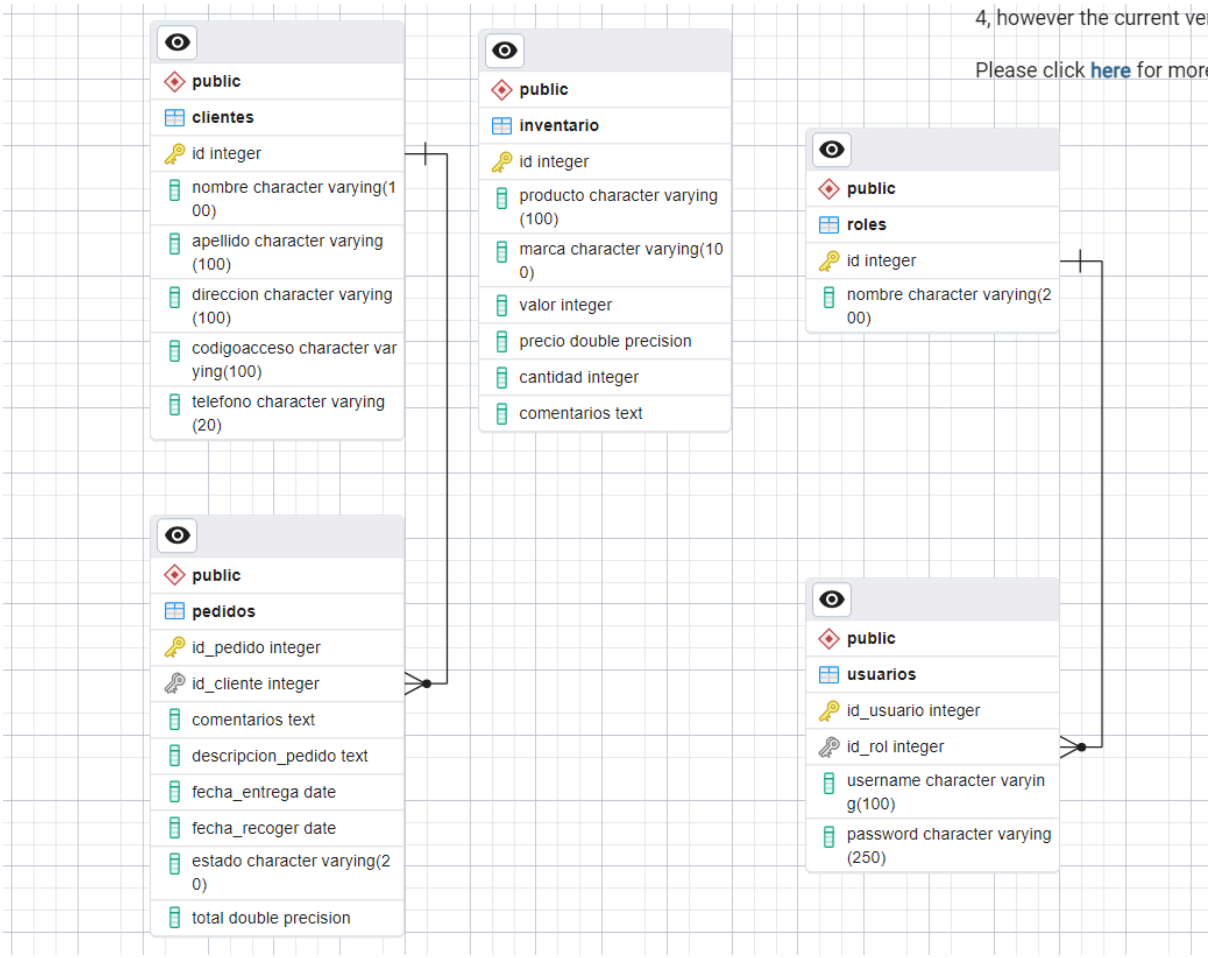
- Evidencias de implementación de pruebas unitarias en la herramienta automatizada y resultados obtenidos al ejecutarlas.
- 

- 
EXPLORER
main.py
SENSACIONALEVE...
testing
Buscar Cliente
Buscar Producto
collectionsPostman
core
dataProviders
Editar Cliente
Ingresar producto
PruebasEjecutadas
buscarCliente.py
buscarProducto.py
editarcliente.py
eliminarCliente.py
ingresarCliente.py
ingresarProducto.py
pruebaLogin.py
.gitignore
clientes.json
clientes.py
flask_app.py
logo.png
main.py
pdf.py
OUTLINE
TIMELINE
from flask_app import app
if __name__ == '__main__':
app.run(debug=True)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
Code
[Running] python -u "c:\Users\Alexis Mesias\OneDrive\UVG6to\software2\sensacionaleventos\SensacionalEventos\main.py"
* Serving Flask app 'flask_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 498-820-178
- 
EXPLORER
Buscar cliente.png
testing > Buscar Cliente > Buscar cliente.png
SENSACIONALEVE...
templates
testing
__pycache__
Buscar Cliente
Buscar cliente.png
Ingreso al sitio.png
Login.png
Buscar Producto
collectionsPostman
core
dataProviders
Editar Cliente
Ingresar producto
PruebasEjecutadas
buscarCliente.py
buscarProducto.py
editarcliente.py
eliminarCliente.py
ingresarCliente.py
ingresarProducto.py
pruebaLogin.py
.gitignore
clientes.json
Lista de Clientes
Buscar Cliente:
Mauricio
ID
Nombre
Apellido
Dirección
Teléfono
Acciones
14
Mauricio
Lemus
5a calle
0000
Editar
Eliminar
20
Mauricio
Lemus
6a calle
0000
Editar
Eliminar
Crear Nuevo Cliente
Volver al Menú Principal
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
Code
[Running] python -u "c:\Users\Alexis Mesias\OneDrive\UVG6to\software2\sensacionaleventos\SensacionalEventos\main.py"
* Serving Flask app 'flask_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 498-820-178
- 
EXPLORER
editarcliente.py
testing > editarcliente.py > ...
SENSACIONALEVE...
templates
testing
__pycache__
Buscar Cliente
Buscar cliente.png
Ingreso al sitio.png
Login.png
Buscar Producto
collectionsPostman
core
dataProviders
Editar Cliente
Ingresar producto
PruebasEjecutadas
buscarCliente.py
buscarProducto.py
editarcliente.py
eliminarCliente.py
ingresarCliente.py
ingresarProducto.py
pruebaLogin.py
.gitignore
clientes.json
import time
import core.initTest as it
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from pruebaLogin import login
import os
import logging
class EditarCliente:
def __init__(self):
(variable) dataProvider: Any
'Editar Cliente', 'editarCliente')
dataProvider = lista[1]
testData = lista[2]
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
Code
[Running] python -u "c:\Users\Alexis Mesias\OneDrive\UVG6to\software2\sensacionaleventos\SensacionalEventos\main.py"
* Serving Flask app 'flask_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 498-820-178



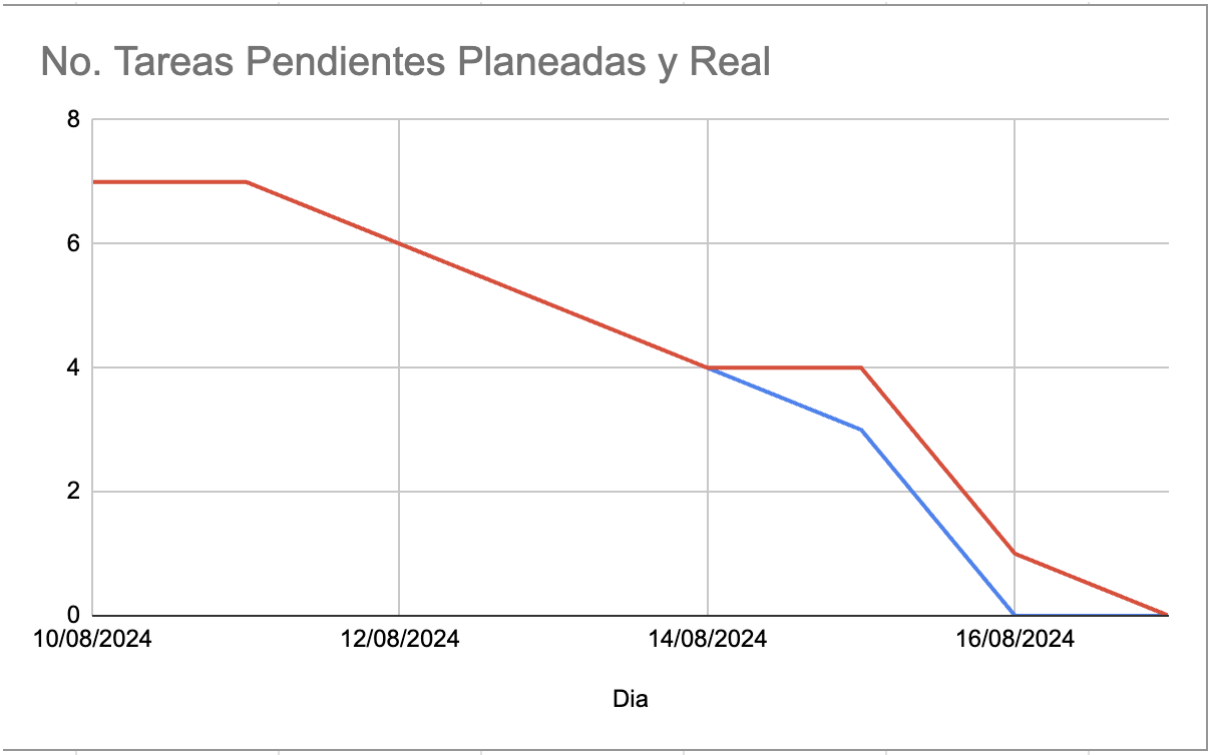
- Se realizaron pruebas unitarias para este sprint utilizando Postman. Luego, se llevaron a cabo pruebas automatizadas mediante Selenium, el cual, al ejecutar la prueba, guarda una serie de capturas de pantalla que se toman automáticamente en cada paso de la prueba. También se guarda un registro del paso a paso de cada prueba, y si se genera algún tipo de error, se describe en qué paso sucedió.

ERD DATABASE:



Resultados

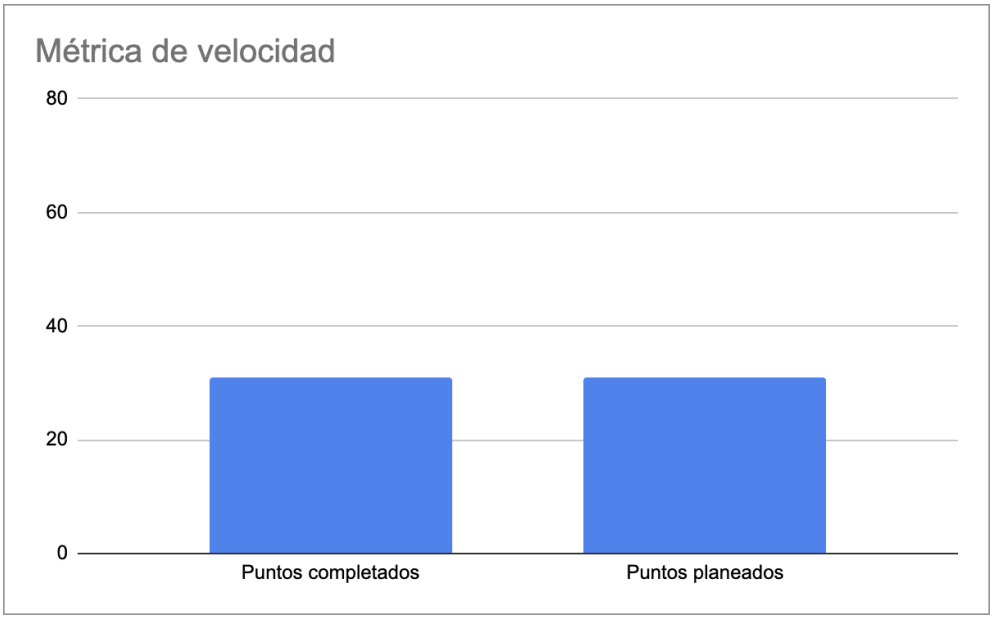
- **Gráfico burndown**



- **Métrica de velocidad**

  Puntos de historia planificados: 31

  Puntos de historia terminados: 31



Velocidad del equipo en el sprint $= \frac{31}{31} - 0 = 100\%$

- 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 diez tareas y se completaron las diez 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, aunque también enfrentamos algunos retrasos en ciertas áreas. Sin embargo, estos no impidieron que cumpliéramos con nuestros objetivos. Esto 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. A pesar de los retrasos, 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. Esta anticipación fue clave para evitar más retrasos y asegurar que cada miembro del equipo pudiera contribuir sin sentirse sobrecargado. Gracias a estas mejoras, mantuvimos un ritmo constante y productivo durante todo el sprint.

En resumen, la combinación de una mejor organización, una distribución más efectiva del trabajo, y una planificación anticipada fue crucial para el éxito del equipo en este sprint, incluso con algunos retrasos. Estamos en una buena posición para continuar este progreso y asegurar una entrega exitosa al final del sprint, aprendiendo de nuestras experiencias y aplicando estas lecciones en futuros proyectos.

Conclusiones:

- Las pruebas realizadas para este sprint, a pesar de tener cierta complejidad, nos ayudaron a tener más seguridad sobre el proyecto.
- A medida que avanzamos en las entregas de sprint, la efectividad incrementa y se entregan tareas con mayor organización.
- Realizar el sprint con un enfoque a entregar una variedad de tareas, aumenta los avances realizados.

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**
13-08-24	12:00	12:30	0 minutos	30 min	Sprint 6	Empecé la prueba automatizada para eliminar un producto
14-08-24	8:00	8:30	0 minutos	30 minutos	Sprint 6	Finalización de la prueba automatizada
15-08-24	2:00	3:30	30 minutos	1 hora	Sprint 6	Creacion de carpeta de evidencias
16-08-24	1:00	1:05	0 minutos	5 minutos	Sprint 6	Creacion de data provider

Nombre: Nancy Gabriela Mazariegos Molina
Carné: 22513

Fecha	**Inicio**	**Fin**	**Tiempo Interrupción**	**Delta tiempo**	**Fase**	**Comentarios**
11/08	20:00	20:45	0 min	45 min	Sprint 6	Creación de la prueba de crear pedido
12/08	21:15	22:00	15 min	45 min	Sprint 6	Modificación en la prueba de crear pedido
13/08	20:30	21:00	15 min	15 min	Sprint 6	Impresión de las screenshots de la prueba crear pedido
14/08	15:30	16:30	10 min	50 min	Sprint 6	Revisión y finalización de la prueba de crear pedido

Nombre: Hugo Eduardo Rivas Fajardo
Carné: 22500

Fecha	Inicio	Fin	Tiempo Interrupción	Delta tiempo	Fase	Comentarios
9/08	10:00	11:00	5 mins	45 mins	Sprint 6	Diseño de base de datos de automatización
12/08	9:00	9:30	0 mins	30 mins	Sprint 6	Diseño de base de datos de automatización
15/08	10:00	11:00	10 minutos	50 mins	Sprint 6	Creación de base de datos de automatización
16/08	12:00	12:20	0 mins	20 mins	Sprint 6	Creación de base de datos de automatización

Nombre: Giovanni Alejandro Santos Hernández
Carné: 22523

Fecha	Inicio	Fin	Tiempo Interrupción	Delta tiempo	Fase	Comentarios
14/08	22:00	22:45	0	45 minutos	Sprint 6	Empezar prueba eliminar producto
15/08	13:00	13:35	5	30 minutos	Sprint 6	Continuar prueba eliminar producto
16-08	18:00	18:50	10	40 minutos	Sprint 6	Continuar prueba eliminar producto

Nombre: Alexis Mesías Flores

Carné: 22562

Fecha	Inicio	Fin	Tiempo Interrupción	Delta tiempo	Fase	Comentarios
25/07/24	13:00	14:30	20 min	1h 10m	Sprint 6	Prueba editar cliente inicio (no funcional aun)
26/07/24	8:30	10:00	45min	45min	Sprint 6	Prueba editar cliente funcional pero solo con un cliente en especifico.
29/07/24	11:00	1:00	55 min	1h 5m	Sprint 6	Prueba Editar cliente funcional con cualquier cliente.
29/07/24	1:00	1:45	25min	20 min	Sprint 6	Se realizo la prueba repetidas veces comprobando su funcionamiento con cualquier cliente.

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/**