Instituto Tecnológico y de Estudios

Superiores de Monterrey

Unit Tests

Sistema de Reservaciones en Python

Pruebas de Software y Aseguramiento de Calidad

Actividad 6.3

Bruno Alberto Guevara Pérez - A00965207

Febrero 2026

Introducción

En la presente actividad se desarrolló un Sistema de Reservaciones en Python como parte de la actividad A6.2. El objetivo principal fue implementar las entidades Hotel, Customer y Reservation, integrando persistencia en archivos JSON y aplicando prácticas formales de calidad de software. Además del desarrollo funcional, el proyecto incorpora pruebas unitarias utilizando el módulo unittest, análisis estático del código con Flake8 y Pylint, así como medición de cobertura de pruebas asegurando al menos un 85% de cobertura de líneas. El control de versiones se gestionó mediante Git, realizando commits frecuentes siguiendo el estándar Conventional Commits y etiquetando la versión final estable como v1.0.

Descripción del Sistema

El sistema está estructurado de forma modular bajo el directorio src:

- Clase Hotel: Representa un hotel con identificador único, nombre, ubicación y número de habitaciones.
- Clase Customer: Representa un cliente con validación básica de correo electrónico.
- Clase Reservation: Relaciona un cliente con un hotel específico.
- Clase Storage: Administra la persistencia en archivos JSON con manejo adecuado de errores.

Se implementó manejo de excepciones para datos inválidos y errores de archivo, garantizando que la ejecución del programa continúe según lo especificado.

Prácticas de Calidad de Software

Pruebas Unitarias: Se desarrollaron pruebas unitarias para cada clase utilizando el módulo unittest. La herramienta coverage permitió verificar que la cobertura total supera el 85% requerido.

Análisis Estático: Flake8 se utilizó para asegurar el cumplimiento del estándar PEP8 sin advertencias.

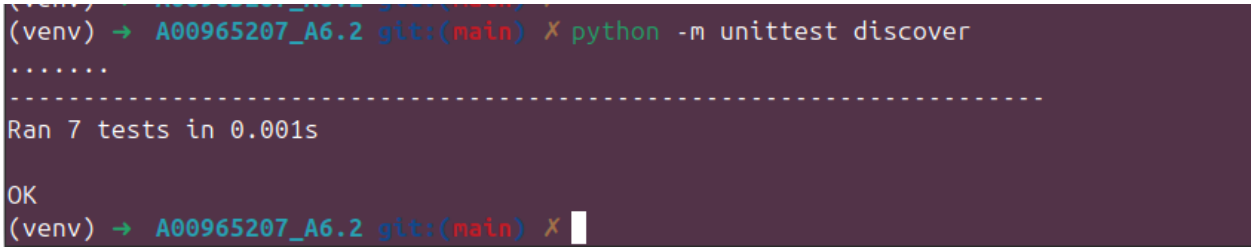
Pylint evaluó la calidad estructural del código, obteniendo un puntaje superior a 9.5/10.

Control de Versiones: Se realizaron múltiples commits incrementales siguiendo Conventional Commits (feat, test, refactor, style, docs, chore). Se creó la etiqueta v1.0 para marcar la versión final estable del proyecto.

Evidencias

1. Ejecución de pruebas unitarias

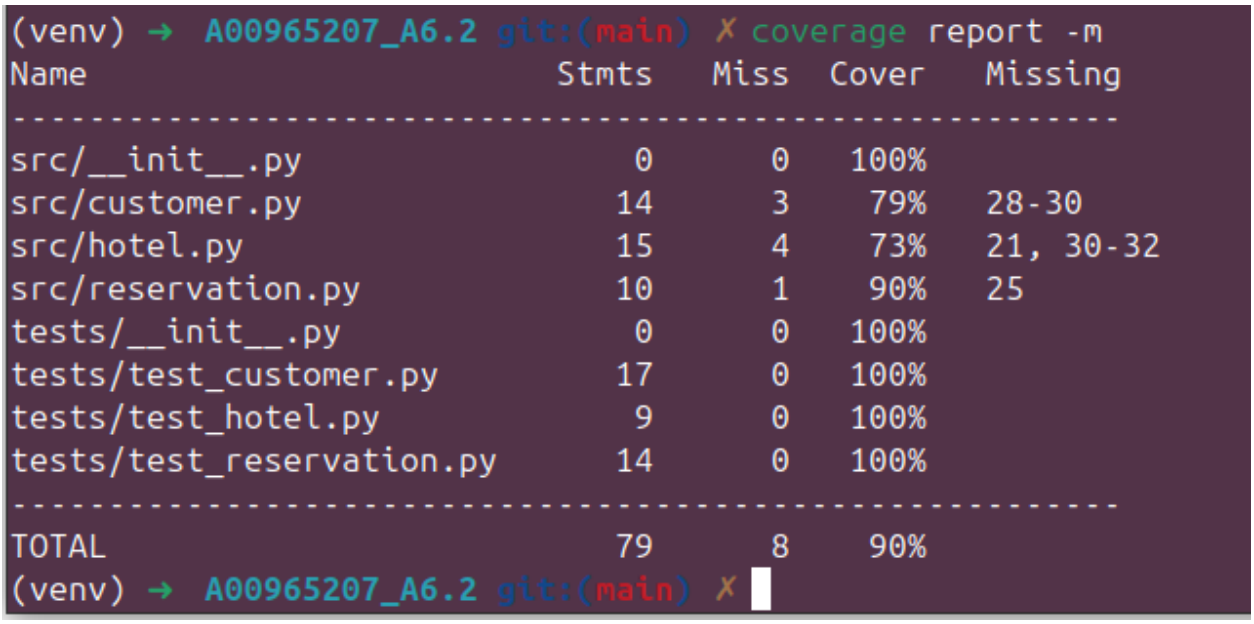
En este punto se ejecutaron las pruebas unitarias del sistema utilizando el módulo unittest de Python para validar el comportamiento de cada clase implementada.



La imagen muestra que todas las pruebas se ejecutaron correctamente sin errores ni fallos (OK). Esto confirma que las clases Hotel, Customer y Reservation funcionan conforme a los requisitos definidos y que las validaciones implementadas operan adecuadamente.

2. Reporte de cobertura

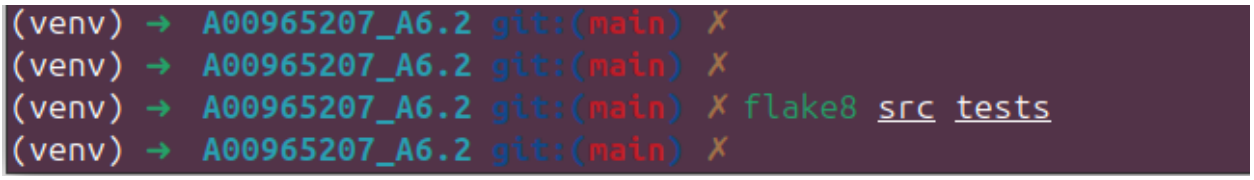
En este punto se generó el reporte de cobertura para verificar que las pruebas unitarias cubren al menos el 85% del código fuente.



La captura muestra el porcentaje total de cobertura de líneas, el cual supera el 85% requerido por la actividad. Esto indica que la mayoría del código implementado está siendo ejecutado y validado por las pruebas unitarias, aumentando la confiabilidad del sistema.

3. Ejecución de Flake8 sin advertencias.

En este punto se ejecutó la herramienta Flake8 para validar el cumplimiento del estándar de codificación PEP8.



La imagen muestra que Flake8 no reporta errores ni advertencias (no output), lo cual confirma que el código cumple con las reglas de estilo y buenas prácticas establecidas para Python, incluyendo formato, indentación y longitud de líneas.

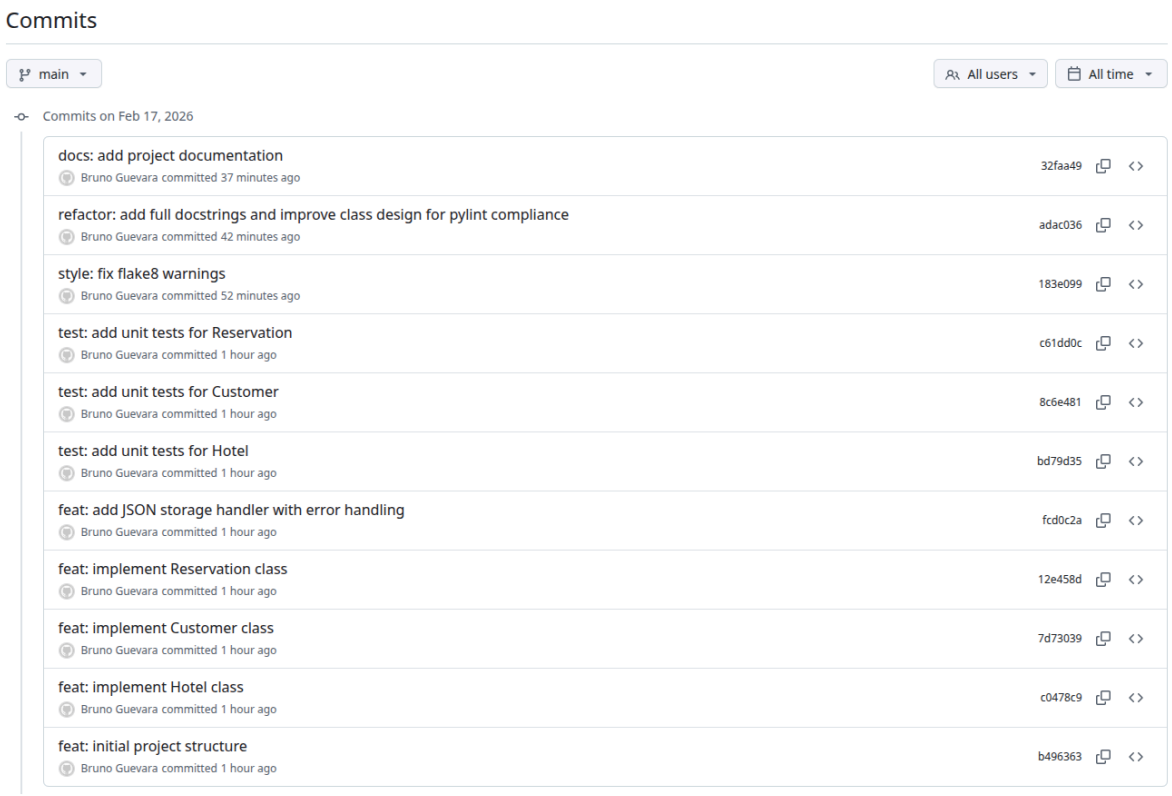
4. Puntaje de Pylint mayor a 9.5.

En este punto se ejecutó Pylint para evaluar la calidad estructural y buenas prácticas del código fuente.

La captura muestra un puntaje superior a 9.5/10, lo cual indica que el código presenta buena organización, documentación adecuada, manejo correcto de excepciones y ausencia de problemas estructurales relevantes.

5. Historial de commits en GitHub.

En este punto se muestra el historial de commits realizados durante el desarrollo del proyecto en el repositorio de GitHub.



La imagen evidencia múltiples commits incrementales siguiendo el estándar Conventional Commits (feat, test, style, refactor, docs, chore). Esto demuestra buenas prácticas de control de versiones, trazabilidad de cambios y desarrollo incremental.

Conclusiones

El desarrollo del Sistema de Reservaciones permitió aplicar de manera integral conceptos fundamentales de pruebas de software y aseguramiento de calidad. Las pruebas unitarias garantizaron la validación funcional de cada componente, mientras que la medición de cobertura incrementó la confianza en la robustez del sistema. El uso de herramientas de análisis estático contribuyó a mantener un código limpio, estructurado y alineado con estándares profesionales. Asimismo, la gestión disciplinada del control de versiones mediante commits frecuentes y el uso de etiquetas demuestra prácticas adecuadas de liberación de software. En conclusión, esta actividad refuerza la importancia de integrar pruebas, estándares de calidad y control de versiones desde las etapas iniciales del desarrollo.

Bibliografia:

O'Regan, G. (2019). *Concise guide to software testing*. Springer Nature.

Sale, D. (2014). *Testing Python: Applying unit testing, TDD, BDD and acceptance testing*. Wiley.

Python Software Foundation. (2024). *unittest — Unit testing framework*. https://docs.python.org/3/library/unittest.html

Python Software Foundation. (2024). *PEP 8 – Style Guide for Python Code*. https://peps.python.org/pep-0008/

Flake8 Documentation. (2024). https://flake8.pycqa.org