Soporte a la Gestión de Datos con Programación Visual

Trabajo Práctico Integrador

Profesores:

- Mariano Castagnino
- Juan Torres

Fecha de Entrega: 11/11/2025

Grupo: 12

Integrantes:

Nombre y Apellido	Legajo
Santino Cataldi	50384
Lucio Cosentino	50785
Gaspar Martinez	47857
Tomás Wardoloff	51543

Índice

Narrativa

Q-Sec es un sistema web con fines educativos que simula el protocolo de Distribución Cuántica de Claves (QKD) BB84. El objetivo principal es permitir a los usuarios ejecutar una simulación paso a paso de cómo funciona el protocolo BB84, permitiendo visualizar los resultados y así poder entender cómo la mecánica cuántica hace posible una comunicación segura y la detección de intrusos.

Abstract

El Trabajo Práctico Integrador (TPI) consiste en el desarrollo de una aplicación que aborda un problema real mediante la implementación de capas de datos, negocio y presentación. El proyecto aplica conceptos fundamentales de ingeniería de software, como diseño modular, encapsulamiento en Programación Orientada a Objetos (POO), validación de reglas de negocio y persistencia de información. Además, integra pruebas automáticas para garantizar el correcto funcionamiento de cada módulo y el cumplimiento de los requerimientos definidos. El objetivo principal del proyecto es consolidar habilidades de análisis, diseño e implementación de software utilizando buenas prácticas de desarrollo en Python.

Requerimientos Funcionales

De que en adelante utilizaremos los nombres de "Alice", "Bob" y "Eve" para referinos al emesior, receptor y espía del mensaje respectivamente.

- **RF01**: El sistema debe permitir el registro y autenticación de usuarios.
- **RF02**: Un usuario autenticado debe poder iniciar una nueva "Sesión de Intercambio de Clave".
- **RF03**: Al iniciar una sesión, el usuario debe poder configurar sus parámetros: la longitud de la clave inicial y si desea incluir un espía en la simulación.
- **RF04**: El sistema debe ejecutar la simulación del protocolo BB84 completo:
    - Generación de bits y bases aleatorias por parte de Alice.
    - Codificación de los bits en qubits.
    - Interceptación y reenvío de qubits por parte de Eve.
    - Generación de bases aleatorias y medición de qubits por parte de Bob.
    - Comparación pública de bases para filtrar la clave.
    - Verificación de la tasa de error para detectar a Eve.
- **RF05**: El sistema debe almacenar en la base de datos el resultado de cada sesión: la configuración, si la clave fue segura o comprometida, y la clave final (si es segura).
- **RF06**: El usuario debe poder ver un historial de sus sesiones de intercambio pasadas con sus resultados.

Requerimientos No Funcionales

Portability

- El sistema debe funcionar correctamente en múltiples navegadores (Sólo Web).

Security

- Todas las contraseñas deben guardarse con encriptado criptográfico (SHA o equivalente).
- Todas los Tokens / API Keys o similares no deben exponerse de manera pública.

Maintainability

- El sistema debe diseñarse con la arquitectura en 3 capas.
- El sistema debe utilizar control de versiones mediante GIT.
- El sistema debe estar programado en Python 3.9 o superior.

Scalability

- El sistema debe funcionar desde una ventana normal y una de incógnito de manera independiente.
    - Aclaración: No se debe guardar el usuario en una variable local, deben usarse Tokens, Cookies o similares.

Performance

- El sistema debe funcionar en un equipo hogareño estándar.

Flexibility

- El sistema debe utilizar una base de datos SQL o NoSQL.

Stack Tecnológico

Capa de Datos

- **Base de Datos**: SQLite
    - Se eligió SQLite por su simplicidad y portabilidad. Al ser una base de datos serverless que se almacena en un único archivo, es ideal para un proyecto de esta escala, eliminando la necesidad de configurar un servidor de base de datos separado.
- **ORM**: SQLAlchemy
    - SQLAlchemy es un ORM que permite interactuar con la base de datos utilizando objetos de Python, lo que abstrae las consultas SQL y facilita la gestión de los datos de manera orientada a objetos.

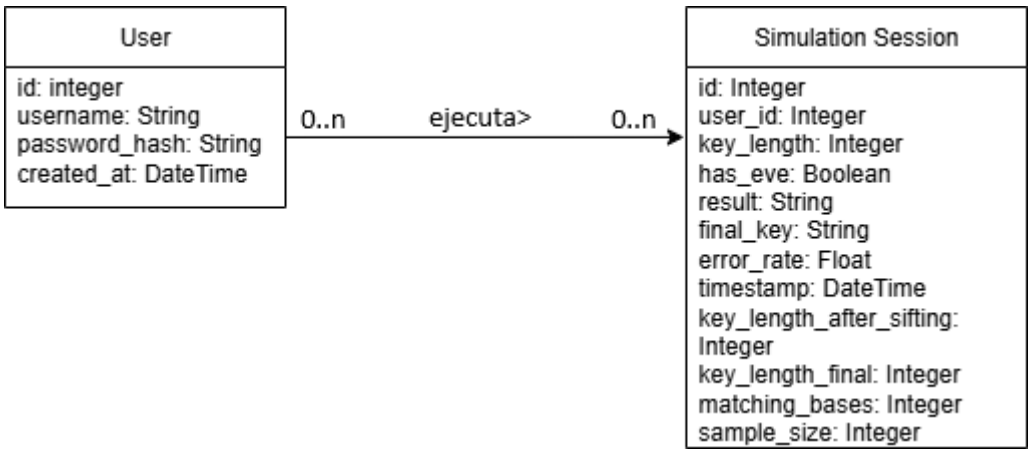
Capa de Negocio

- **Simulación Cuántica**: IBM Qiskit
    - Qiskit es el framework de código abierto para la computación cuántica que proporciona las herramientas necesarias para crear y manipular circuitos cuánticos, lo que lo hace perfecto para simular el protocolo BB84.
- **Testing**: Pytest
    - Se seleccionó Pytest para validar las reglas de negocio, como el cálculo de la tasa de error y la detección de espionaje.

Capa de Presentación

- **Framework Web**: Flask
    - Se eligió Flask por ser un microframework web ligero, modular y fácil de aprender. Lo cual lo hace ideal para este proyecto, permitiendo un desarrollo rápido de la interfaz de usuario.

Modelo de dominio



Documentación de Librerías

- Qiskit: https://quantum.cloud.ibm.com/docs/en/guides/install-qiskit
- SQLAlchemy: https://docs.sqlalchemy.org/en/20/
- Flask: https://flask.palletsprojects.com/en/stable/
- WTForms: https://wtforms.readthedocs.io/en/3.2.x/
- Werkzeug: https://werkzeug.palletsprojects.com/en/stable/
- Scipy: https://docs.scipy.org/doc/scipy/
- RustworkX: https://www.rustworkx.org
- Numpy: https://numpy.org/doc/
- Jinja: https://jinja.palletsprojects.com/en/stable/
- MarkupSafe: https://markupsafe.palletsprojects.com/en/stable/

Bibliografía

- https://pypi.org/project/qiskit/
- https://quantum.cloud.ibm.com/docs/en/guides/install-qiskit
- https://en.wikipedia.org/wiki/Qiskit
- https://github.com/qiskit

Link Código fuente

- https://github.com/Tomas-Wardoloff/frro-python-2025-12/tree/TPI