INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

# Proyecto omega

# TurboMessage

Integrantes
Leopoldo Morante Castillo – 200821
Eduardo Velázquez Barragán – 201432

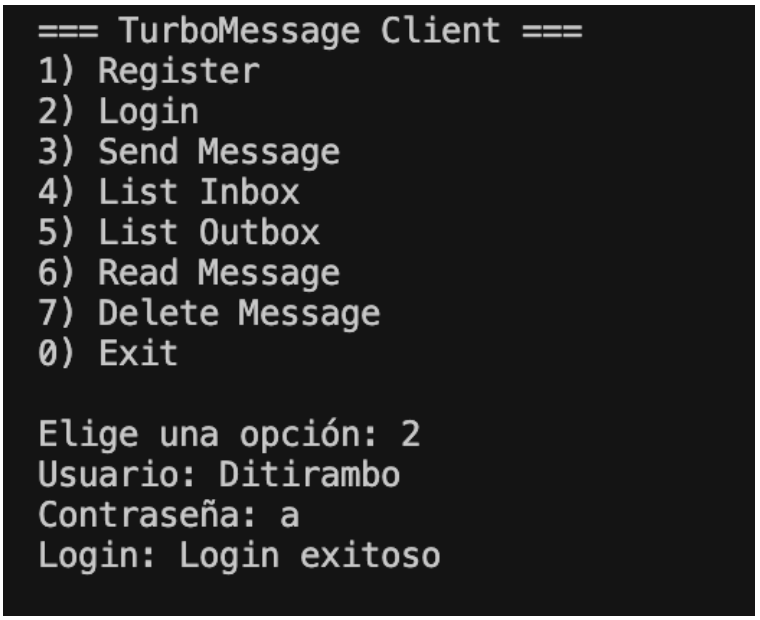
# Introducción

TurboMessage es un sistema de mensajería interna construido sobre gRPC y Protocol Buffers, diseñado para demostrar cómo exponer servicios de alto rendimiento y bajo acoplamiento en Python. El sitemap permite el registro y autenticación de usuarios mediante JSON Web Tokens (JWT), el envío de correos con límites configurables en bandejas de entrada y salida (en el caso del proyecto se fijaron con 5), así como la lectura y eliminación de mensajes, todo ello con una persistencia sencilla en archivos JSON. Gracias a HTTP y al streaming de servidor, las operaciones de listado de mensajes se sirven tan pronto como están disponibles, sin necesidad de esperar a recopilar toda la información.

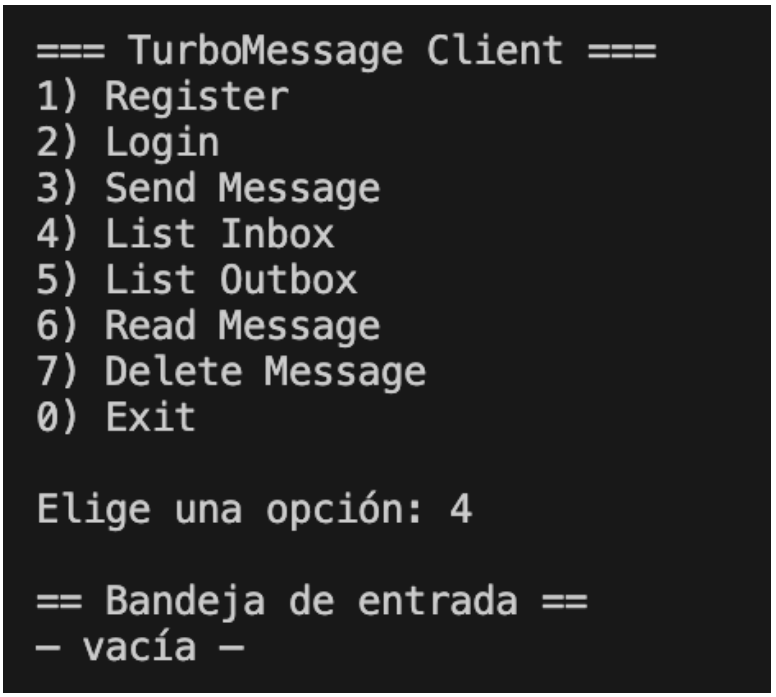
# Descripción de la arquitectura del sistema

El núcleo de TurboMessage es un servidor gRPC que ofrece los métodos Register, Login, SendMessage, ListInbox, ListOutbox, ReadMessage y DeleteMessage, implementados como RPCs unarios o de streaming cuando se devuelve más de un mensaje. Al arrancar, el servidor carga dos ficheros JSON —users.json y mails.json— que almacenan los usuarios (con su contraseña hasheada idealmente) y la colección de correos junto a las listas de IDs de cada bandeja de entrada y de salida. Cada usuario y cada correo recibe un identificador único (UUID v4) generado automáticamente en el momento de registro o envío, garantizando la unicidad de forma sencilla. Usar identificadores aleatorios generados con UUIDv4 es recomendable porque ofrecen unicidad prácticamente garantizada sin necesidad de coordinación central, ya que la probabilidad de colisión en un espacio de 128 bits es prácticamente nula.  La autenticación se procesa emitiendo un JWT firmado con HS256, que contiene el nombre de usuario como sujeto (sub) y un tiempo de expiración configurado (1 hora). Antes de ejecutar cualquier operación sensible, el servidor valida la firma y la validez temporal del token; en caso de error retorna el código UNAUTHENTICATED. Para mantener la consistencia frente a múltiples hilos el servidor usa un pool de threads todas las mutaciones de estado (registro, envío, marcado como leído, borrado) se protegen con un mutex de Python; en los métodos de streaming se hace primero una copia de la lista bajo lock y luego se liberan el cerrojo para emitir cada mensaje, minimizando las secciones críticas y evitando condiciones de carrera.

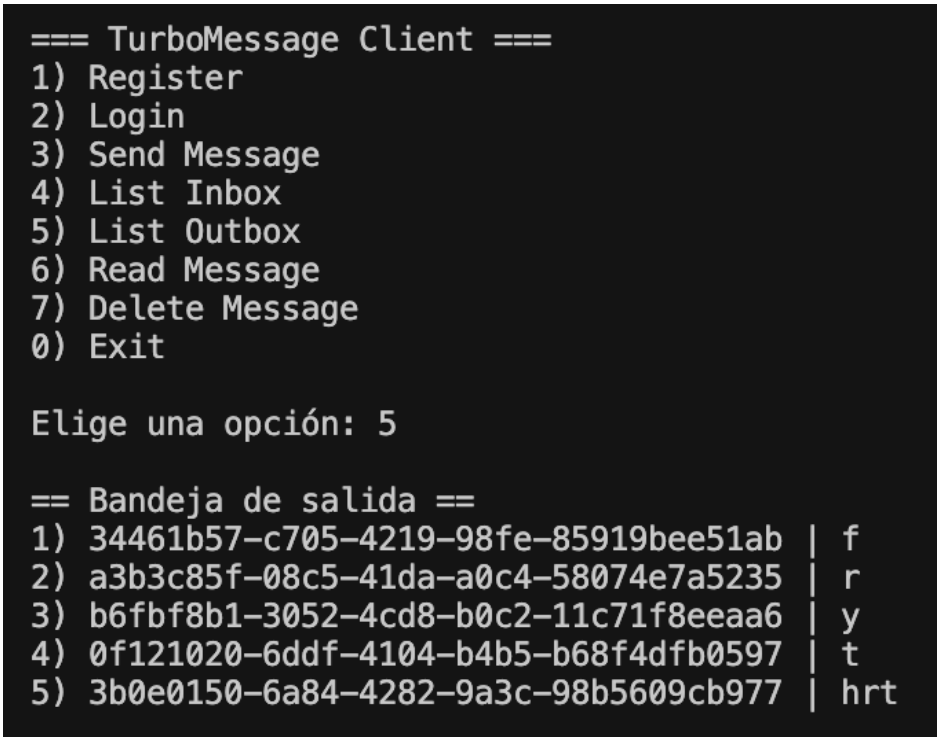
A continuación se ponen imágenes de la ejecución del cliente:



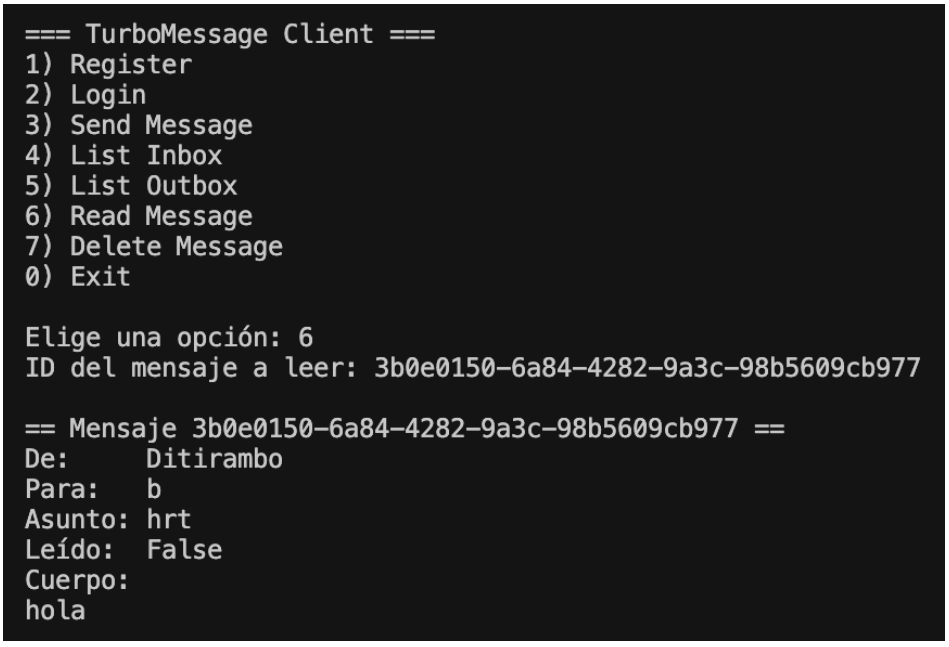

Login exitoso




Vista de la bandeja de entrada vacía




Vista de la bandeja de salida. Se muestra el identificador del correo y su asunto




Lectura de un mensaje existente en la bandeja de entrada o de salida.

# Conclusiones breves

El diseño de TurboMessage pone de manifiesto un patrón robusto para la construcción de APIs de mensajería: se combina la eficiencia de Protocol Buffers, la flexibilidad de gRPC y la seguridad básica de JWT con un mecanismo de persistencia sencillo y un esquema de

concurrencia controlado. Gracias a su arquitectura modular, es fácil evolucionar el sistema, por ejemplo migrando la persistencia a una base de datos transaccional, añadiendo cifrado TLS en el canal gRPC o reemplazando el cliente de consola por una interfaz gráfica. En su estado actual, TurboMessage soporta múltiples clientes concurrentes, respeta límites configurables de bandejas, mantiene integridad de datos y facilita la comprensión de buenas prácticas para proyectos basados en RPC.