

Relatório de Entrega: Semana 5-6

Milestone: Object Storage e Connectors Mock

Aluno: Bruno Evangelista Bertoldo, Augusto Arantes Chaves, Enzo Alvarez Dias, Matheus Pereira Figueredo

Data: 24/11/2025

1. Objetivos Alcançados

Expansão do sistema Chat4All v2 para suportar envio de arquivos binários (Object Storage) e simulação de integração com plataformas externas (WhatsApp e Instagram), completando o ciclo de vida da mensagem (SENT → DELIVERED → READ).

2. Arquitetura e Componentes

2.1.

Object Storage (MinIO)

Implementação de um servidor **MinIO** (compatível com S3) no Docker Compose.

- **Upload:** Endpoint `POST /v1/files/upload` recebe arquivos `multipart/form-data` e retorna um `file_id`.
- **Download:** Geração de URLs pré-assinadas (*Presigned URLs*) para acesso seguro e temporário aos arquivos.

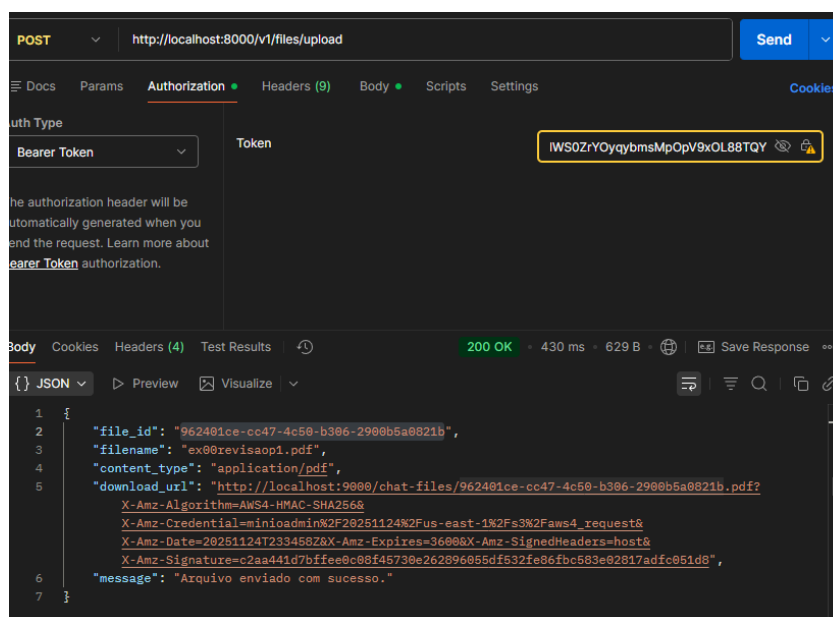


Figura 1: Resposta da API de Upload retornando `file_id` e URL pré-assinada

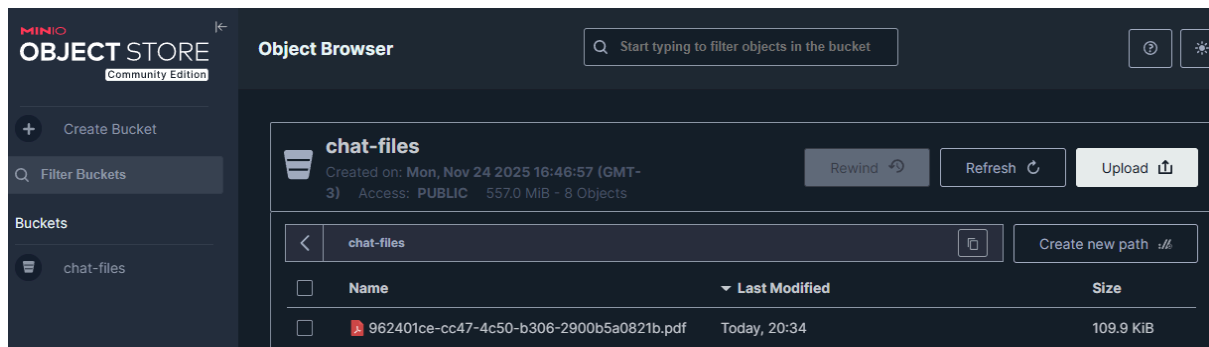


Figura 2: Painel do MinIO mostrando o Upload e retornando `file_id`

2.2.

Mensagens com Anexos

Atualização do schema da API e da tabela do Cassandra para suportar metadados de arquivos.

- Mensagens contendo `file_id` são automaticamente classificadas como `type: "file"`.

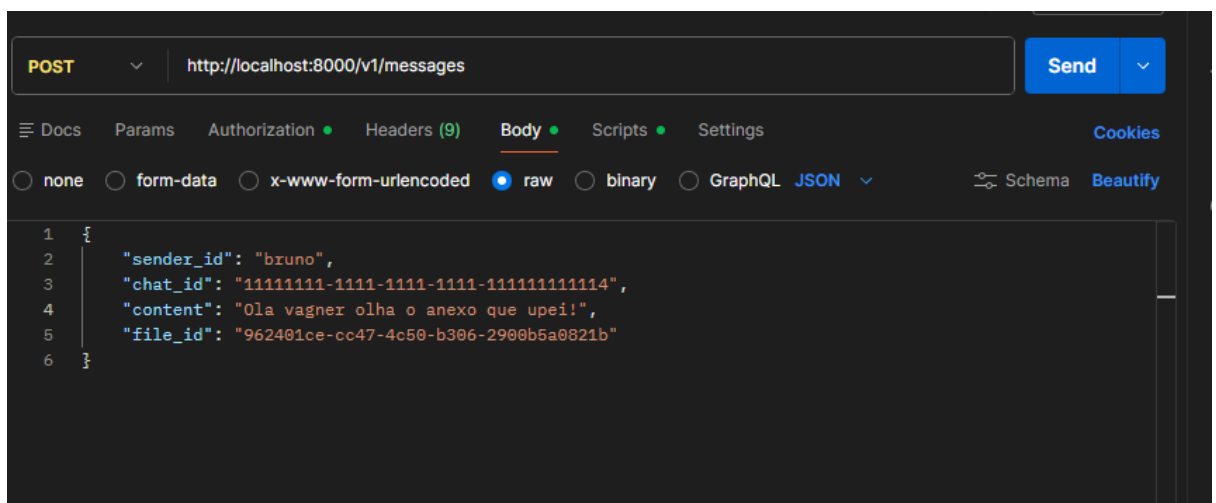


Figura 3: Mensagem sendo enviada com `file_id`

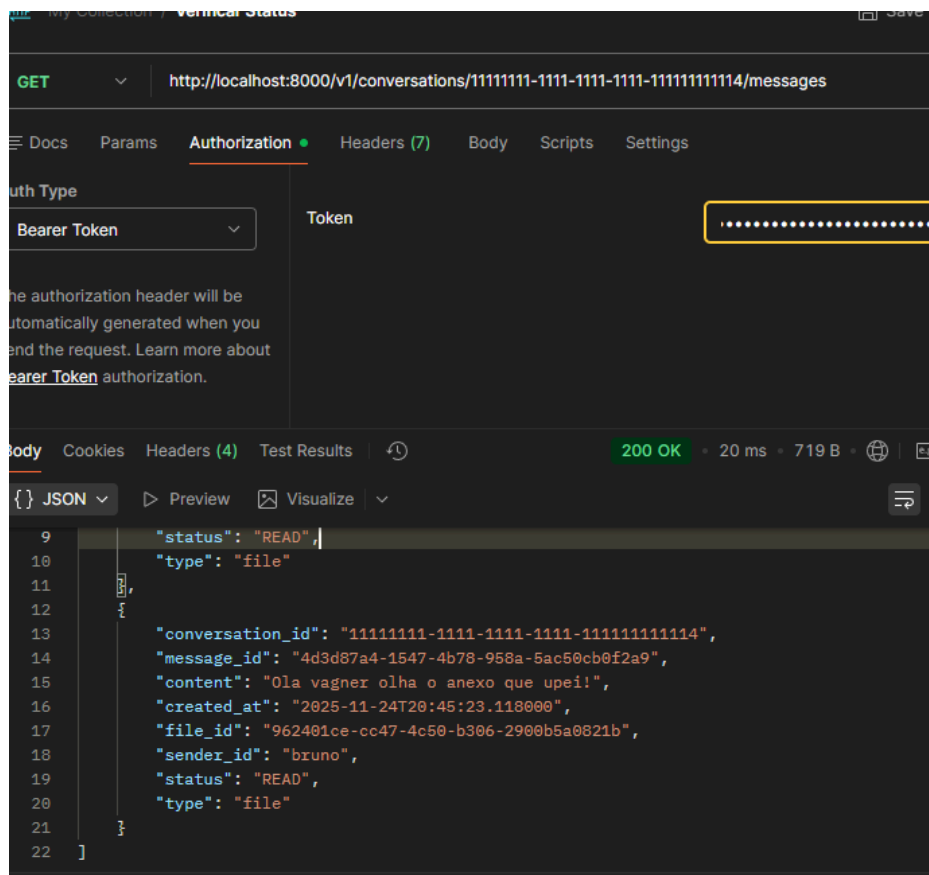


Figura 4: Teste de recebimento da mensagem com anexo e classificada com o tipo file

2.3.

Connectors Mock

Criação de dois novos microsserviços em Python:

1. **Connector WhatsApp:** Ouve o tópico `whatsapp_outbound`.
 2. **Connector Instagram:** Ouve o tópico `instagram_outbound`.
- **Roteamento Inteligente:** O `Router Worker` analisa o conteúdo. Mensagens iniciadas com `@` são roteadas para o Instagram; as demais para o WhatsApp.

```

connector_whatsapp_c | 2025-11-24 20:09:53,772 - [WHATSAPP MOCK] - 📡 [Callback] Enviado status READ para API (Msg: 2f6f364c-6d90-4
Bad-a7f9-d6a29ce5b472)
connector_whatsapp_c | 2025-11-24 20:45:23,193 - [WHATSAPP MOCK] - 📧 Recebida mensagem para +556299999999
connector_whatsapp_c | 2025-11-24 20:45:25,106 - [WHATSAPP MOCK] - ✅ [WhatsApp API] Entregue ao usuário +556299999999: Ola vagner
olha o an...
connector_whatsapp_c | 2025-11-24 20:45:26,714 - [WHATSAPP MOCK] - 📡 [Callback] Enviado status READ para API (Msg: 4d3d87a4-1547-4
b78-958a-5ac50cb0f2a9)

```

Figura 5: Connector Whatsapp recebendo mensagem do respectivo tópico mensagem de recebimento, simulação de envio e disparo de webhook de leitura

```
connector_instagram_c | 2025-11-24 20:32:47,764 - [INSTAGRAM MOCK] - 📄 Recebida DM para usuario_insta
connector_instagram_c | 2025-11-24 20:32:48,362 - [INSTAGRAM MOCK] - 💜 [Instagram API] DM enviada para usuario_insta: @professor ol
na esse...
connector_instagram_c | 2025-11-24 20:32:49,429 - [INSTAGRAM MOCK] - 🗨️ [Callback] Enviado status READ para API (Msg: 100623a0-82d4-
43e5-a84f-393e1300aa53)
```

Figura 6: Connector Instragram recebendo mensagem de recebimento, simulação de envio e disparo de webhook de leitura.

2.4.

Controle de Status (Ciclo Completo)

Implementação de **Webhooks (Callbacks)** para atualização de status em tempo real.

1. **SENT:** Definido pela API ao postar no Kafka.
2. **DELIVERED:** Definido pelo Worker ao salvar no Cassandra.
3. **READ:** Definido via Callback HTTP. O Connector Mock, após simular o envio, chama a API (`PATCH /messages/{id}/status`), que atualiza o registro no Cassandra.

3. Evidências de Execução

Cenário de Teste: Envio de mensagem com anexo de imagem direcionada ao Instagram (@maria).

Resultado (JSON do Histórico):

The screenshot shows a REST client interface with the following details:

- Method:** GET (though the request is a PATCH)
- URL:** `http://localhost:8000/v1/conversations/11111111-1111-1111-1111-111111111114/messages`
- Authorization:** Bearer Token (masked)
- Status:** 200 OK
- Response Body (JSON):**

```
[
  {
    "conversation_id": "11111111-1111-1111-1111-111111111114",
    "message_id": "c7eb53c1-2b71-47d6-9c49-bd0cb910f770",
    "content": "@maria olha o anexo que upei!",
    "created_at": "2025-11-24T20:35:27.253000",
    "file_id": "962401ce-cc47-4c50-b306-2900b5a0821b",
    "sender_id": "bruno",
    "status": "READ",
    "type": "file"
  }
]
```

4. Conclusão

A entrega da Semana 5-6 foi concluída com êxito, atendendo integralmente aos requisitos de expansão do sistema para suporte a multimídia e integração multiplataforma .

O sistema **Chat4All v2** evoluiu de uma API de texto simples para uma plataforma de mensageria robusta com as seguintes capacidades consolidadas:

1. **Gestão de Arquivos:** A integração com **Object Storage (MinIO)** permitiu o upload e download seguro de arquivos, desacoplando o armazenamento de binários do banco de dados principal .
2. **Simulação Realista:** A criação dos **Connectors Mock** (WhatsApp e Instagram) validou a arquitetura orientada a eventos, demonstrando que o sistema pode escalar para múltiplos canais de comunicação sem alterar o núcleo da aplicação .
3. **Rastreabilidade Total:** A implementação do fluxo de status completo (**SENT** → **DELIVERED** → **READ**) via Webhooks garantiu a consistência da informação entre os microsserviços assíncronos .

Os testes integrados demonstraram que a arquitetura suporta o fluxo completo de ponta a ponta, estando o sistema pronto para cenários de alta demanda e futuras implementações de interfaces de usuário.