

MEMORY BOOK

Um projeto que transforma lembranças em
pontos interativos no mapa

Autores:
Alberto Pontiery
Diogo Nascimento
Guilherme Pança

OBJETIVO:

Registrar momentos especiais (texto, fotos, descrições e localizações) e guardá-los em um espaço visual e afetivo.

FUNCIONALIDADES PRINCIPAIS:

- 📍 Marcar lugares importantes no mapa interativo (React-Leaflet)
- 📝 CRUD de memórias (criar, visualizar, editar, excluir)
 - 📷 Upload de fotos e cores personalizadas
- 🎵 Integração com Spotify para adicionar trilha sonora
- 👤 Sistema de autenticação e perfis de usuário
 - 🎨 Temas e gradientes personalizáveis
- 📱 Interface responsiva para desktop e mobile

PLANO DE TESTES

Cenários de teste

1. Funcionalidades
2. Cenários De Funcionalidade
3. Cenários de Desempenho

Relatório de Consolidação de Teste:

1. Escopo de Teste
2. Resultados Obtidos

UPLOAD DE VIDEO (ALBERTO)

NOVA FUNCIONALIDADE

Todos (1) Fotos (0) Vídeos (1)



Uma das novas funcionalidades foi o upload de vídeo:

- Vídeo de até 30s podem ser adicionados a memória
- O objetivo é poder tornar as memórias mais completas
- Há também a possibilidade de filtrar na memória por vídeos

UPLOAD DE VIDEO (ALBERTO)

REFATORAÇÃO

Objetivos:

- Centralizar validações e reduzir duplicidade
 - Aumentar segurança e confiabilidade
- Permitir organização opcional por usuário/memória sem quebrar clientes existentes

Antes

- Duração de vídeo validada em dois lugares distintos
- Upload salvava direto com nome original, sem logs
- Separação de fotos/vídeos limitada a data URLs

Depois

- Validação de duração centralizada (uma única função)
- Upload com nome sanitizado, validação antes de mover e logs
- Serialização mais robusta (prefixo/extensão/pasta) e organização opcional por usuário/memória

INTEGRAÇÃO SPOTIFY (GUILHERME)

REFATORAÇÃO

Objetivos:

- DESACOPLAR INTEGRAÇÃO DO CONTROLLER PARA MELHOR SEGURANÇA E RESILIÊNCIA
- IMPLEMENTAR FAIL-FAST CONTROLADO COM LOGS ESTRUTURADOS
- MANTER 100% COMPATIBILIDADE COM TESTES EXISTENTES

Antes:

- CONTROLLER FAZIA TUDO
- ERROS SILENCIOSOS
- SEGURANÇA FRÁGIL
- ZERO RESILIÊNCIA
- ALTO ACOPLAMENTO

Depois:

- CLIENTE CENTRALIZADO ERROS
- IDENTIFICAR E CORRIGE ERROS
- RESILIÊNCIA
- COMPATIBILIDADE TOTAL

TESTES (DIOGO)

ONDE FICAM OS TESTES

Os testes ficam na pasta tests/ e usamos pytest para rodá-los.

O arquivo run_tests.bat executa tudo de uma vez e mostra a cobertura de código. O conftest.py prepara o ambiente de teste com a aplicação Flask e um cliente HTTP para simular requisições reais.

EXEMPLOS DO QUE É TESTADO:

- test_auth.py: registro, login e logout.
- test_memories.py: CRUD de memórias.
- test_spotify.py: busca de músicas e armazenamento/edição de link.
- test_videos.py: upload ≤30s, bloqueio >30s, exclusão de mídia.

TESTES (DIOGO)

EXEMPLOS DIRETOS

```
## Login (Sucesso)
```python
Fluxo de login com usuário válido
def test_login_success(client, create_test_user, login_user):
 create_test_user(client, email="user@example.com", password="segredo123")
 res, data, _ = login_user(client, "user@example.com", "segredo123")
 assert res.status_code == 200
 assert "access_token" in data # token emitido
```

```

```
## Health Check - Backend Flask (OK)
```python
Testa se a API está saudável
def test_health_check_success(client):
 res = client.get("/api/health") # chama o endpoint de saúde
 assert res.status_code == 200 # valida status
 data = res.get_json() # extrai JSON
 assert data.get("status") == "OK" # valida conteúdo
```

```

MUITO
OBRIGADO!
