Diagrama UML - APBIA

Assistente de Projetos para Bragantec Baseado em IA

Apresentação: < 4 minutos

🎯 SLIDE 1: Visão Geral do Sistema

O que é o APBIA?

Sistema que integra **IA (Gemini 2.5 Flash)** com **gestão de projetos** para a **Bragantec** (feira de ciências do IFSP Bragança Paulista).

Módulos Principais

1. 🔐 **Autenticação** (Usuario, TipoUsuario)
2. 💬 **Chat IA** (Chat, TipoIA, Mensagem, ArquivoChat)
3. 📊 **Projetos** (Projeto + associações)
4. 👁️ **Supervisão** (NotaOrientador, VisualizacaoOrientador)

Stack Tecnológica

- **Backend:** Python Flask
- **IA:** Google Gemini 2.5 Flash
- **Banco:** PostgreSQL (Supabase)
- **Frontend:** HTML/CSS/JS Puro

👤 SLIDE 2: Usuario + TipoUsuario

Usuario

- **Chave:** `email` e `numero_inscricao` (BP) únicos
- **Tipos:** Admin, Participante, Orientador
- **Apelido:** Personaliza interação com IA
- **Cardinalidade:** `Usuario 1 --> 1 TipoUsuario`

TipoUsuario

- Define permissões e comportamento
- Extensível (novos tipos via admin)

Decisão: BP obrigatório apenas para Participante/Orientador

💬 SLIDE 3: Chat + TipoIA + Mensagem

Chat

- Sessão de conversa usuário ↔ IA
- **Cardinalidades:**
  - `Usuario 1 --> * Chat` (um user, vários chats)
  - `Chat * --> 1 TipoIA` (cada chat usa UM tipo de IA)

TipoIA

- Personalidades: Geral, Participante, Orientador
- **Gerenciável pelo admin** (nova feature!)

Mensagem

- `role`: 'user' | 'model'
- `ferramenta_usada`: JSON com tools (Google Search, Code Execution, Modo Bragantec)
- `thinking_process`: Raciocínio da IA (opcional)
- **Cardinalidade:** `Chat 1 --> * Mensagem`

📎 SLIDE 4: ArquivoChat

Funcionalidades

- Upload de PDFs, imagens, vídeos, áudio
- Processamento via Gemini (multimodal)
- Integração temporária (48h) com `gemini_file_uri`

Cardinalidades

- `Chat 1 --> * ArquivoChat` (chat tem vários arquivos)
- `Mensagem 1 --> 0..* ArquivoChat` (mensagem pode ter anexos)

Decisão de Design

- Arquivos salvos no disco (`static/uploads/chat_files/`)
- URI do Gemini para análise temporária
- `mensagem_id` opcional (arquivo pode existir antes da mensagem)

📊 SLIDE 5: Projeto

Estrutura Completa

- **Dados:** Nome, categoria, resumo, palavras-chave
- **Metodologia:** Introdução, objetivos, metodologia, cronograma
- **Especiais:**
  - `objetivos_especificos`: ARRAY
  - `cronograma`: JSONB (flexível)
  - `gerado_por_ia`: Flag + prompt usado

Cardinalidades (N:M)

- `Usuario * --> * Projeto` via:
  - **orientadores_projetos** (orientação)
  - **participantes_projetos** (autoria)

Por quê N:M?

- Projeto pode ter múltiplos orientadores/participantes
- Participante pode ter múltiplos projetos

👁️ SLIDE 6: NotaOrientador + VisualizacaoOrientador

NotaOrientador

- Feedback contextualizado em mensagens específicas
- **Cardinalidades:**
  - `Mensagem 1 --> * NotaOrientador` (mensagem pode ter várias notas)
  - `Usuario 1 --> * NotaOrientador` (orientador escreve notas)
- Versionamento: `data_atualizacao` para edição

VisualizacaoOrientador

- Tracking de quando orientador acessa chats
- **Cardinalidades:**
  - `Chat 1 --> * VisualizacaoOrientador` (chat visualizado várias vezes)
  - `Usuario 1 --> * VisualizacaoOrientador` (orientador tem visualizações)

Uso: Analytics + notificações ("orientador viu sua mensagem")

🔧 SLIDE 7: Decisões Técnicas Críticas

1. Remoção de `ferramentas_chat`

- ❌ Tabela **nunca usada** no código
- ✅ Substituída por JSON em `mensagens.ferramenta_usada`

2. JSONB no `cronograma`

```json
{
  "etapas": [
    {"nome": "Revisão", "inicio": "2025-03", "fim": "2025-04"},
    {"nome": "Coleta", "inicio": "2025-05", "fim": "2025-07"}
  ]
}
```

Por quê? Flexibilidade + queries nativas no PostgreSQL

3. BIGINT vs INTEGER

- `mensagens.id`: BIGINT (pode crescer exponencialmente)
- `projetos.id`: INTEGER (crescimento moderado)

4. Modo Bragantec (consumo de tokens)

- Contexto COMPLETO: ~100k-200k tokens
- Opcional (usuário ativa quando precisar)

📐 SLIDE 8: Padrões de Design Identificados

Strategy Pattern (TipoIA)

Troca comportamento da IA sem alterar código

Composite Pattern (Chat → Mensagem → Arquivo)

Hierarquia de conteúdo estruturada

Observer Pattern (VisualizacaoOrientador)

Tracking de eventos de acesso

Audit Trail (Timestamps)

`data_criacao` + `data_atualizacao` em tudo

⚡ SLIDE 9: Escalabilidade

Pontos de Atenção

- **Mensagens:** Crescimento rápido → considerar particionamento
- **Arquivos:** URLs externas (S3/Cloudinary)
- **Limites Gemini FREE:**
  - 10 RPM (requests/minuto)
  - 250k TPM (tokens/minuto)
  - 250 RPD (requests/dia)

Índices Críticos

```sql
CREATE INDEX idx_chat_id ON mensagens(chat_id);
CREATE INDEX idx_usuario_id ON chats(usuario_id);
CREATE INDEX idx_projeto_id ON participantes_projetos(projeto_id);
```

✅ SLIDE 10: Conclusão

Pontos Fortes

✅ Modular e extensível
✅ Suporte multimodal (texto, imagem, vídeo, PDF)
✅ Rastreamento completo (auditoria)
✅ Flexível (JSONB, arrays)
✅ Escalável (arquitetura preparada)

Princípios Seguidos

- **Normalização:** Sem redundância
- **Atomicidade:** Uma responsabilidade por tabela
- **Rastreabilidade:** Timestamps everywhere
- **Segurança:** Constraints + hashing de senhas

Próximos Passos

1. Gestão de Tipos IA (✅ implementado agora!)
2. Sistema de notificações
3. Versionamento de projetos

4. Dashboard de analytics

🙏 Perguntas?

Tempo estimado: 3min 45s

Contato:

- Sistema: APBIA (Flask + Gemini 2.5 Flash)
- Banco: PostgreSQL (Supabase)
- 10 tabelas, 8 classes principais