🤖 Watson Assistant Log Persistence Service

Este serviço é responsável por coletar, processar e persistir logs do Watson Assistant em um banco de dados MongoDB. Ele opera como um serviço de cron job, executando periodicamente para manter os dados atualizados.

🗂️ Estrutura do Projeto

```
src/
├── config/         # Configurações do projeto
├── controllers/    # Controladores da aplicação
├── interfaces/     # Interfaces TypeScript
├── models/         # Modelos do MongoDB
├── schemas/        # Schemas de validação (Zod)
├── services/       # Serviços da aplicação
└── utils/          # Utilitários
```

🧩 Schemas e Tipos

Assistant Schema

Schema para validação dos dados do Watson Assistant.

```
// AssistantSkill
{
  type: string;      // Tipo da habilidade
  skill_id: string;  // ID único da habilidade
}

// AssistantEnvironment
{
  name: string;           // Nome do ambiente (ex: 'live', 'development')
  environment: string;    // Tipo do ambiente
  environment_id: string; // ID único do ambiente
}

// Assistant
{
  name: string;                    // Nome do assistente
  language: string;                // Idioma do assistente
  description: string;             // Descrição do assistente
  assistant_id: string;            // ID único do assistente
  assistant_skills: AssistantSkill[];      // Lista de habilidades
  assistant_environments: AssistantEnvironment[]; // Lista de ambientes
}
```

Logs Response Schema

Schema para validação da resposta de logs do Watson Assistant.

```
{
  startDate: string;   // Data de início do período em ISO 8601
  endDate: string;     // Data de fim do período em ISO 8601
  assistants: {        // Mapa de logs por assistente
    [assistantName: string]: LogCollection;
  }
}
```

Save Result Schema

Schema para validação do resultado do salvamento de logs.

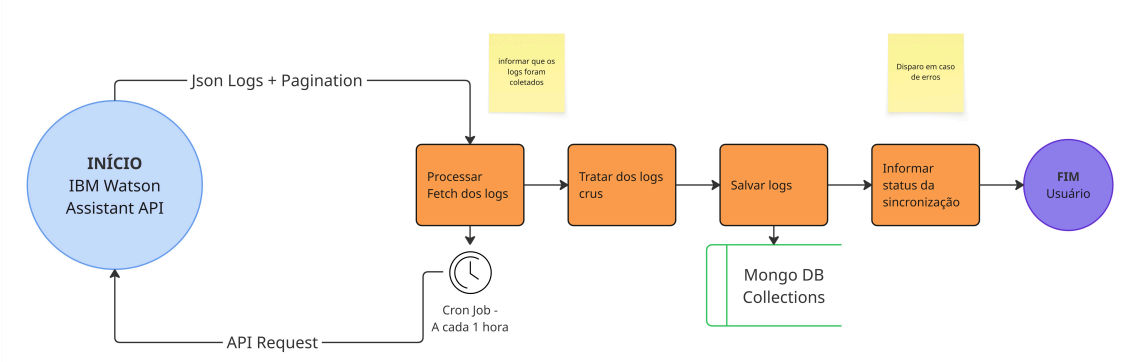
```
{
  success: boolean;           // Indica se o salvamento foi bem-sucedido
  count: number;              // Número de logs salvos com sucesso
  duplicates: number;         // Número de logs duplicados encontrados
  savedLogs?: StandardizedLog; // Logs que foram salvos (opcional)
  error?: string;             // Mensagem de erro, se houver
}
```

Sync Point Schema

Schema para validação dos pontos de sincronização.

```
{
  assistantId: string; // ID do assistente
  lastSyncTimestamp: Date; // Timestamp da última sincronização
  lastLogId: string; // ID do último log sincronizado
}
```

🔄 Fluxo de Dados



Fluxo de dados do serviço de persistência de logs do Watson Assistant

1. **Coleta de Logs**

- O sistema busca logs do Watson Assistant usando o `AssistantService`
- Os logs são validados usando o `LogsResponseSchema`

2. **Processamento**
   - Os logs brutos são transformados em um formato padronizado
   - A transformação é validada usando o `StandardizedLogSchema`
3. **Persistência**
   - Os logs padronizados são salvos no MongoDB
   - O resultado do salvamento é validado usando o `SaveResultSchema`
4. **Sincronização**
   - O sistema mantém um registro do último log sincronizado
   - O ponto de sincronização é validado usando o `SyncPointSchema`

✅ Validações

Todos os schemas utilizam o Zod para validação, garantindo:

- Tipos corretos para cada campo
- Campos obrigatórios preenchidos
- Formato correto de datas (ISO 8601)
- URLs válidas
- Strings não vazias
- Números não negativos

💡 Exemplos de Uso

Buscando Logs de um Assistente

```
const assistantService = AssistantService.getInstance();
const logs = await assistantService.getAssistantLogs(
  "assistant-id",
  new Date("2024-01-01"),
  new Date("2024-01-31")
);
```

Salvando Logs Padronizados

```
const persistanceService = PersistanceService.getInstance();
const result = await persistanceService.saveProcessedLogs(standardizedLogs);

if (result.success) {
  console.log(`Salvos ${result.count} logs com sucesso`);
  if (result.duplicates > 0) {
    console.log(`${result.duplicates} logs duplicados ignorados`);
  }
}
```

Atualizando Ponto de Sincronização

```
const syncPoint = {
  assistantId: "assistant-1",
  lastSyncTimestamp: new Date(),
  lastLogId: "log-123",
};

// Validação com Zod
const validatedSyncPoint = SyncPointSchema.parse(syncPoint);
```

⚙️ Configuração

O serviço utiliza variáveis de ambiente para configuração. Crie um arquivo `.env` na raiz do projeto com as seguintes variáveis:

```
MONGODB_URI=sua_uri_do_mongodb
WATSON_API_KEY=sua_chave_api_do_watson
WATSON_URL=sua_url_do_watson
```

📦 Instalação

```
npm install
```

▶️ Execução

```
npm start
```

🛠️ Desenvolvimento

```
npm run dev
```