pgembed

Enquanto extensões como o pgvector preparam o PostgreSQL para armazenar e consultar dados vetoriais, o processo de gerar esses vetores — a vetorização (*embedding*) — ainda precisa ser executado separadamente.

Normalmente, esse processo exige que uma aplicação ou pipeline se comunique com um modelo de Machine Learning externo, como os oferecidos pela OpenAI, ou localmente, com modelos disponíveis no Ollama. O que pode exigir um conhecimento técnico mais amplo do time de dados ou o envolvimento de diversos profissionais com expertises variadas.

Em muitos dos casos, projetos focados na validação do *Minimum Viable Product (MVP)* optar por desenvolver funções no próprio PostgreSQL que consomem serviços de vetorização e facilitam essa atividade.

Contudo, é comum que as boas práticas, como o uso de dados em lotes para a redução do tráfego em rede, gestão do consumo de memória, configurações de segurança e mecanismo de resiliência, sejam negligenciados nessa abordagem.

Visando fornecer um conjunto de funções para vetorização de dados no PostgreSQL, com foco em desempenho, segurança e resiliência, a Tecnisys desenvolveu a extensão pgembed.

Funções de Vetorização

Após instalar e criar a extensão pgembed, serão disponibilizadas no esquema `pgembed` do banco de dados diversas funções para a vetorização de dados, organizadas pelo fornecedor da API de embedding e pelo tipo do dados de entrada:

Fornecedor	Linha de Texto Única	Conjunto de Linhas	Tabela
Ollama (local)	`embed_ollama`	`embed_batch_ollama`	`embed_table_ollama`
OpenAI	`embed_openai`	`embed_batch_openai`	`embed_table_openai`
Custom APIs	`embed_custom`	`embed_batch_custom`	`embed_table_custom`

Tipos de Funções:

- **Linha de Texto Única** (`embed_*`): Gera o embedding para uma única linha de texto.

  Por exemplo:

  ```
  SELECT pgembed.embed_openai('Hello, world!', 'text-embedding-3-small');
  ```

- **Conjunto de Linhas** (`embed_batch_*`): Processa múltiplas linhas de texto e retorna uma lista de vetores.

  Por exemplo:

  ```
  SELECT * FROM pgembed.embed_batch_openai(ARRAY['Hello, world!', 'Hello, PostgreSQL!'], 'text-embedding-3-small');
  ```

- **Tabela** (`embed_table_*`): Lê coluna de texto de uma tabela e atualiza automaticamente a coluna de embeddings correspondente, eliminando a necessidade de operações manuais de atualização.

  Por exemplo:

  ```
  SELECT * FROM pgembed.embed_table_openai(
      'public',       -- schema
      'tb_documents', -- table name
      'content',      -- content column
      'embedding',    -- embedding column
      FALSE,          -- regenerate: only update NULL embeddings (default: FALSE)
      1000,           -- batch_size
      'text-embedding-3-small'       -- model
  );
  ```

Parâmetros Avançados

Além dos parâmetros obrigatórios, as funções da extensão pgembed possuem diversos parâmetros avançados extremamente úteis, tais como:

- Timeout da requisição (`timeout` , default 60s)
- Se o certificado SSL deve ser verificado (`verify_ssl` , default TRUE)
- Se o texto deve ser truncado caso exceda o tamanho do contexto (`trucante` , default TRUE, para modelos do Ollama)
- JSON para opções avançadas de cada modelo (`options` , para modelos do Ollama)

- Número de dimensões dos embeddings gerados (`dimensions` , para modelos da OpenA
- Formato de codificação dos embeddings retornados (`encode_format` , para modelos da OpenAI)

Segurança e Resiliência

No quesito segurança e resiliência, a extensão pgembed disponibiliza parâmetros que poder ser configurados no nível do usuário, sessão e banco de dados, tais como:

- `pgembed.openai_api_key` ou `pgembed.custom_api_key` para definir a API key utilizada na requisição do serviço
- `pgembed.url_allowlist` para definir a lista de URL autorizadas. Por padrão: localhost, 127.0.0.1, 0.0.0.0 (qualquer porta), *.openai.com e api.openai.com
- `pgembed.max_retries` , `pgembed.initial_backoff_ms` , `pgembed.max_backoff_ms` para controle de tentativas
- `pgembed.circuit_breaker_threshold` e `pgembed.circuit_breaker_reset_timeout_s` para evitar que falhas em cascata bloqueiem temporariamente solicitações para serviços indisponíveis

Fonte: https://github.com/tecnisys/pgembed

Instalando o pgembed via PgSmart Cli

Comando

Terminal input

```
pgsmart install
```

1. Selecione a Opção Banco de Dados.
2. Selecione a Opção `Extensões` .
3. Selecione a Extensão `Integração com Text Embedding APIs (pgembed)` .
4. Informe a versão Majoritária do PostgreSQL.

   4.1 Informe a Release da versão Majoritária do PostgreSQL.
5. Confirme a instalação da extensão:

5.1. Confirme se deseja instalar/atualizar o repositório dos pacotes do PostgreSYS (esta instalação é necessária para dar continuidade, caso o repositório ainda não esteja configurado).

5.2. Caso tenha optado por instalar/atualizar o Repositório de Pacotes, informe a URL do Repositório de Pacotes.

> **NOTA**
>
> A opção de instalação do Repositório de Pacotes não é realizada se os repositórios do pgsys já estiverem presentes ou mapeados na máquina.

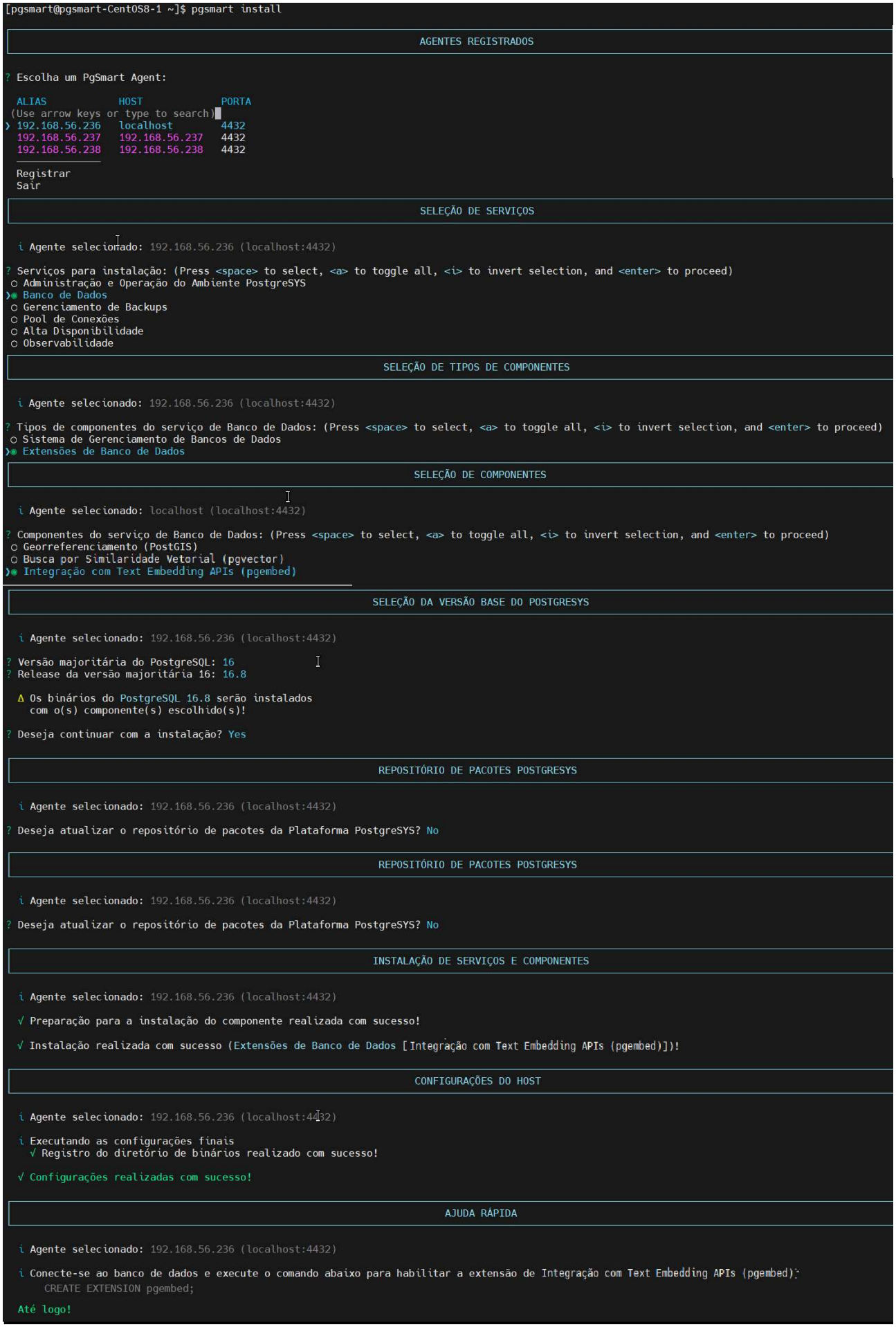

Figura 6 -Instalação extensão pgembed interativa

6. Conecte-se ao banco de dados e execute o comando abaixo :

Terminal input

```
CREATE EXTENSION pgembed;
```

Instalando o pgembed via PgSmart Web

1. Selecione a aba `Início/Gerenciar Ambientes/Serviços`.

   Serão apresentados todos os serviços instalados em todos os ambientes.

2. Clique em `Instalar`.

3. Selecione o `Ambiente` onde o Serviço será instalado.

4. Selecione o serviço `Banco de Dados`.

5. Selecione a `Versão Base` do PostgreSQL [Ex: 16].

6. Selecione a `Versão do PostgreSQL` [Ex: 16-2].

7. Clique em `Avançar`.

8. Selecione a Extensão `Integração com Text Embedding APIs (pgembed)`.

9. Informe se deseja inicializar a instância de banco de dados.

10. Selecione o `endereço IP` de um ou mais Agentes do PgSmart.

11. Clique em `Instalar`.

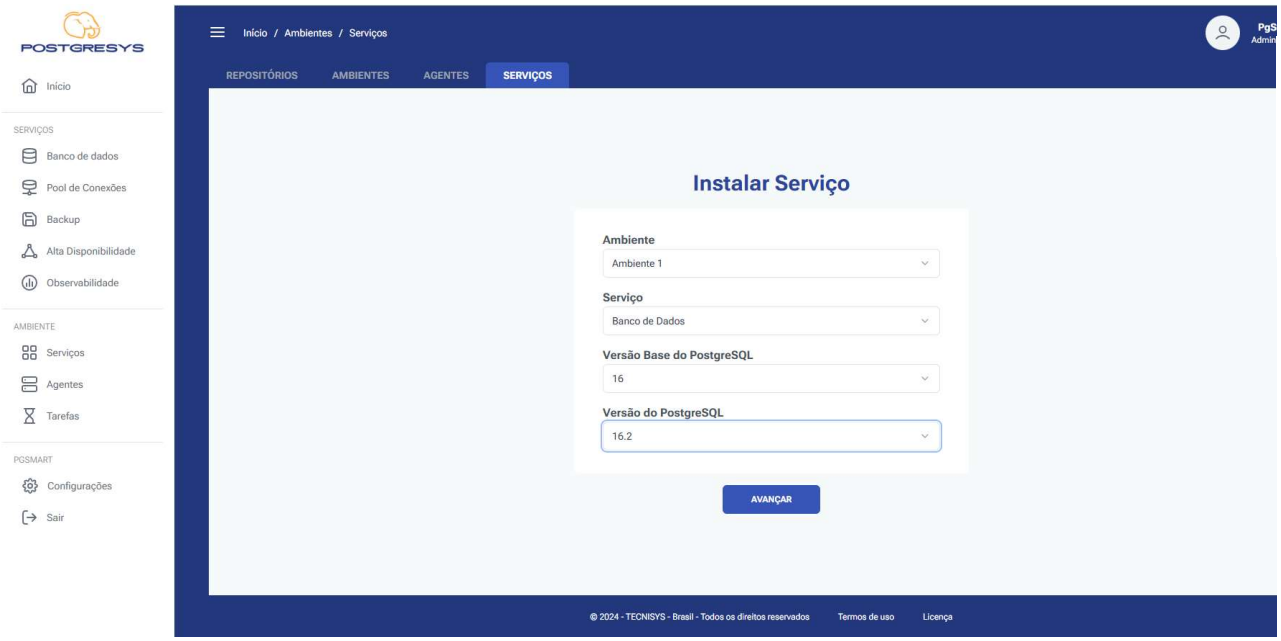

Figura 1 -Instalação do Serviço de Banco de Dados

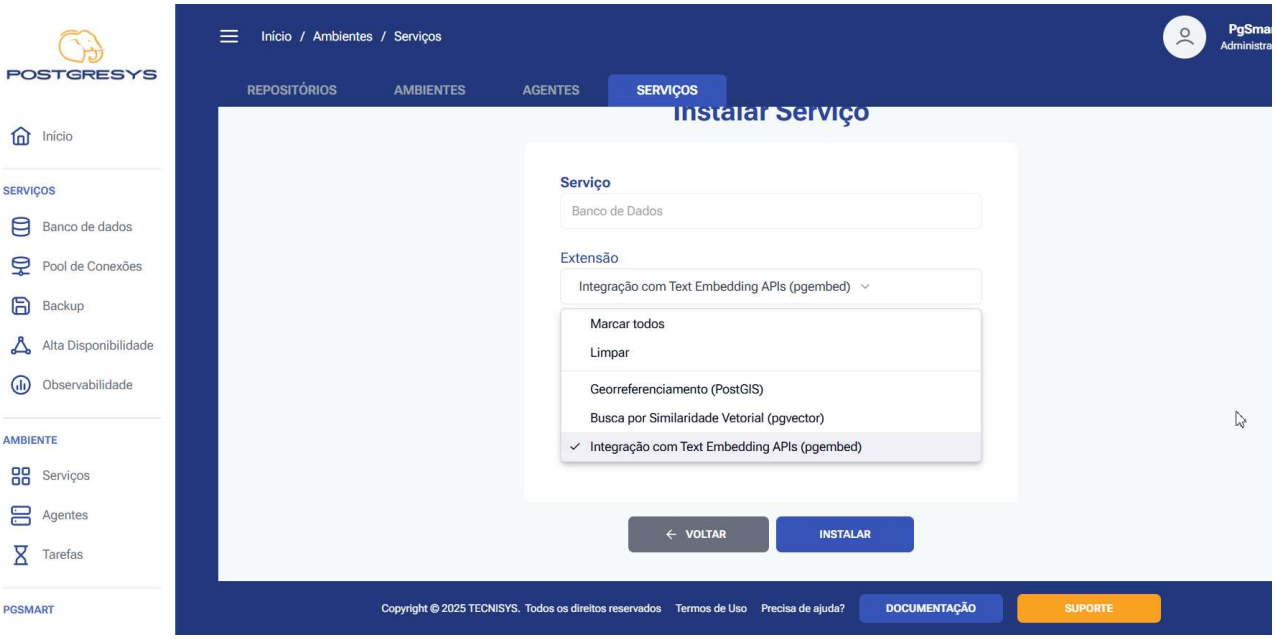

Figura 2 -Instalação do Serviço de Banco de Dados

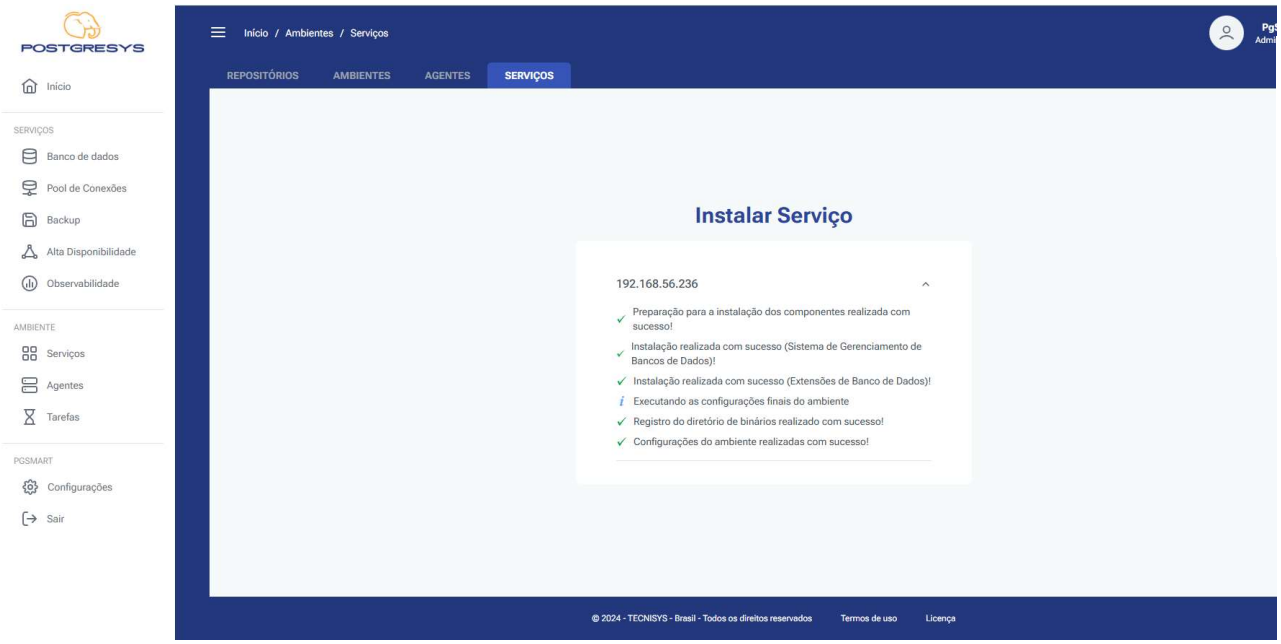

Figura 3 -Instalação do Serviço de Banco de Dados