



BandCare  
Technologies

# POSTGRESQL JSONB





# INTRODUÇÃO

O que é PostgreSQL?

- PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, conhecido por sua robustez, flexibilidade e conformidade com os padrões SQL.
- Suporta uma ampla gama de tipos de dados e extensões, incluindo tipos de dados JSON e JSONB, que facilitam o armazenamento e manipulação de dados semi-estruturados.



**{JSON}**

PostgreSQL



# INTRODUÇÃO

A adoção de bancos de dados não-relacionais (NoSQL) como o MongoDB e o Redis vem crescendo a cada dia, porém se você utiliza o Postgres, saiba que desde a versão 9.4 é possível armazenar e manipular dados em JSON de forma eficiente com os JSONB Data Types.

## *Vantagens de Usar JSON/JSONB*

- Flexibilidade: JSON permite armazenar dados semi-estruturados que podem variar em formato e conteúdo.
- Eficiência: O tipo JSONB armazena dados em um formato binário eficiente, permitindo consultas rápidas e compactas.
- Integração: JSON facilita a integração com APIs web e outras fontes de dados que utilizam o formato JSON



**{JSON}**

PostgreSQL

# Quando e porque utilizar?



## 1 — QUERO ARMAZENAR UM ARRAY DE VALORES DENTRO DO MEU REGISTRO.

Um bom exemplo seria uma tabela Pessoa que teria uma coluna Interesses onde podemos gravar um array desta forma: “[‘programação’, ‘música’, ‘esportes’]”



## 2 — QUERO ARMAZENAR DADOS QUE VARIAM PARA CADA REGISTRO

Com SQL teríamos que criar uma coluna para cada um desses dados, fazendo com que ela fique muito extensa. Uma solução seria usar somente uma coluna do tipo jsonb onde gravaríamos um JSON com todos esses campos em forma de chave e valor. Exemplo: “{ “cor-favorita”: “azul”, “idade”: 32, “aprovado”: true }”

Continuar



# O QUE É JSON?

Na prática, .json é um arquivo que contém uma série de dados estruturados em formato texto e é utilizado para transferir informações entre sistemas. É importante dizer que, apesar de sua origem ser por meio da linguagem JavaScript, JSON não é uma linguagem de programação.

Em vez disso, é uma notação para a transferência de dados que segue um padrão específico. Por isso, pode ser amplamente utilizada em diferentes linguagens de programação e sistemas.

```
{  
  "nome": "João da Silva",  
  "idade": 35,  
  "cidade": "São Paulo",  
  "telefone": "(11) 1234-5678",  
  "email": "joao.silva@email.com"  
}
```

# INSERINDO DADOS EM JSONB

**CRIANDO UMA TABELA COM COLUNAS DO TIPO JSONB:**

```
CREATE TABLE pessoa (  
  id serial NOT NULL,  
  nome text,  
  interesses jsonb,  
  informacoes_adicionais jsonb  
);
```

**INSERIR ALGUNS REGISTROS NESTA TABELA**

```
INSERT INTO pessoa VALUES (1, 'João', '["futebol", "natação"]',  
  '{"idade": 28, "time": "Chapecoense"}');  
  
INSERT INTO pessoa VALUES (2, 'Maria', '["leitura", "programação",  
  "dança"]', '{"idade": 39, "trabalha-com-programacao": true, "area":  
  "back-end"}');  
  
INSERT INTO pessoa VALUES (3, 'Ana', '["programação"]', '{"idade":  
  29, "trabalha-com-programacao": false, "area": "front-end", "areas-  
  de-interesse": ["mobile", "design"]}');
```

**BUSCANDO DADOS**

```
SELECT nome, interesses, informacoes_adicionais->'idade' FROM pessoa
```

[Voltar ao índice](#)

# NOSSO PROJETO

Nosso projeto consiste em Um projeto de CRUD Gerenciador de Usuários envolve o desenvolvimento de um sistema capaz de realizar operações básicas de Create, Read, Update e Delete em um conjunto de dados de usuários armazenados em um banco de dados. Esse tipo de aplicação é fundamental em diversos cenários. Além disso, aprimoramos o armazenamento dos dados das consultas, salvando-os em um banco de dados PostgreSQL no formato JSON, permitindo uma gestão e análise mais eficiente dos dados dos jogadores.



CREATE



READ



UPDATE



DELETE

---

C

R

U

D





# NOSSO PROJETO

Utilização de um Script feito em Python, para a geração de dados de usuários em formato JSON. Que também já conecta ao Banco de Dados e faz a inserção.

```
geradorPESSOA.py > ...
1  from faker import Faker
2  import json
3
4  fake = Faker('pt_BR')
5
6  num_dados = 10000000
7
8  dados = []
9  n = 1
10 for _ in range(num_dados):
11     nome = fake.name()
12     telefone = fake.phone_number()
13     endereco = fake.address()
14     data_nascimento = fake.date_of_birth(tzinfo=None, minimum_age=18, maximum_age=90)
15     print(f'{n} criados')
16     dados.append({
17         'nome': nome,
18         'telefone': telefone,
19         'endereco': endereco,
20         'data_nascimento': data_nascimento.strftime('%Y-%m-%d')
21     })
22     n = n + 1
23
24 with open('dados_exemplo_10m.json', 'w', encoding='utf-8') as f:
25     json.dump(dados, f, ensure_ascii=False, indent=4)
26
27 print("Dados gerados e salvos em 'dados_exemplo.json'.")
28
```



# NOSSO PROJETO 2

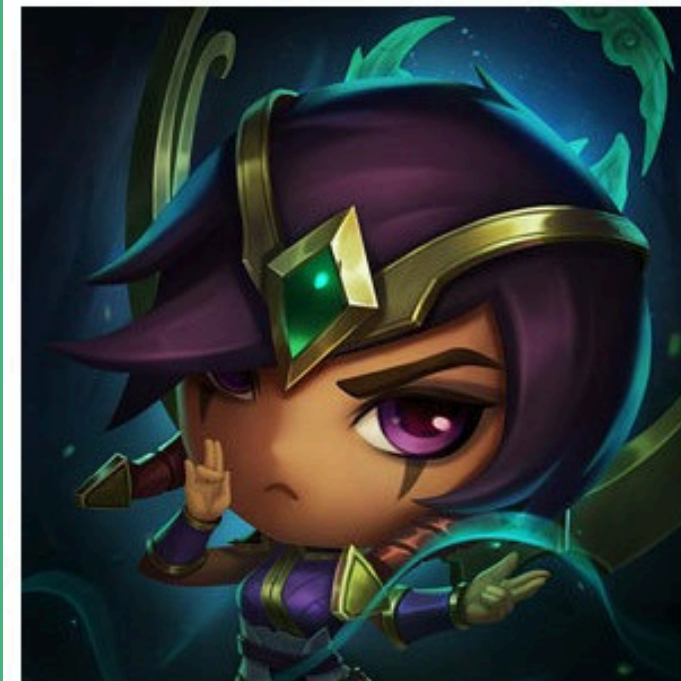
Nosso projeto consiste em um bot desenvolvido em Python que se conecta à API do jogo League of Legends para obter dados das contas dos jogadores. Originalmente, o bot foi implementado para funcionar no Discord, permitindo que os usuários consultassem informações diretamente na plataforma. Recentemente, adaptamos o bot para operar também via web utilizando o framework Django, proporcionando uma interface web para as consultas. Além disso, aprimoramos o armazenamento dos dados das consultas, salvando-os em um banco de dados PostgreSQL no formato JSON, permitindo uma gestão e análise mais eficiente dos dados dos jogadores.

## Perfil do Jogador

Nome: Rei Ayanami#NERV9

ID: 1zVg6DHi7DKezchVEcm7XuA2rGMGwG32XZ-bCtehR3h\_JMxruXUAugHLItrPptaaGGawihy6h9PFq

NIVEL: 960



## Dados de Liga

- Tipo de Fila: RANKED\_FLEX\_SR

Tier: DIAMOND II

Rank: II



# INSERÇÃO DE DADOS EM JSONB DO NOSSO PROJETO

```
-- Criação da tabela
CREATE TABLE jogadores (
    id SERIAL PRIMARY KEY,
    jogador_info JSONB
);

-- Inserção de dados
INSERT INTO jogadores (jogador_info) VALUES
('{
    "id": "0PxdrGVKXfrE6nSLO40xEfWDhKc04RGZmWnwNIZfT3s32_k",
    "puuid": "Jp4yXs6bUrqdpz4M80Y0wPzQvMhceNwF7NJ00rfCPupa5Ifq7yNg4P9zrXIK-ppBCTLpIY2DUkW3
    "accountId": "kdMOjXZxqAU4P5cwZmNcvSP98h4llknBTe3nobmZipNkuiM",
    "revisionDate": 1719877228346,
    "profileIconId": 6622,
    "summonerLevel": 1093
}');
```

# TEMPOS DE RESPOTAS APÓS 50 MILHÕES DE DADOS

```
1 -- Selecionar pessoas pela data de nascimento '1991-05-11'
2 SELECT * FROM select_pessoas_by_data_nascimento('1991-05-11');
3
```

36 SEGUNDOS

Data Output Messages Notifications



	nome text	telefone text	endereço text	data_nascimento date
1	Luiz Gustavo Macedo	+55 (061) 7908 7806	Viela Martins, 50	1991-05-11
2	Josué Pastor	31 5527 6426	Rodovia de Novaes, 29	1991-05-11
3	Gustavo Machado	+55 (011) 4186 8520	Ladeira Garcia, 99	1991-05-11
4	Gael Cavalcante	(051) 4055-4291	Parque Campos, 485	1991-05-11
5	Maria Sophia da Mata	+55 (084) 9888-0134	Quadra Montenegro, 49	1991-05-11
6	Eloá Cirino	+55 71 7788 2948	Rodovia de Pires, 618	1991-05-11
7	Renan Ramos	(061) 8758 2947	Jardim de Sá, 50	1991-05-11
8	Esther Cavalcante	31 9055-3571	Viaduto Nascimento, 181	1991-05-11
9	Davi Luiz Cirino	(051) 4691 2311	Jardim Liam Pimenta, 35	1991-05-11
10	Ian Cirino	(031) 6834 2746	Recanto Rezende, 927	1991-05-11
11	Ana Julia Cassiano	(084) 3982 8519	Rua Olívia Rios, 86	1991-05-11
12	Bento Oliveira	+55 21 9619 7682	Feira Cunha, 542	1991-05-11
13	Evelyn Fernandes	+55 (051) 0337-7400	Parque Araújo	1991-05-11
14	Guilherme Rocha	+55 84 8064-2101	Conjunto Ian Teixeira, 83	1991-05-11

Total rows: 1000 of 1868 Query complete 00:00:36.127

Ln 3, Col 1

[Voltar ao índice](#)

# TEMPOS DE RESPOTAS APÓS 50 MILHÕES DE DADOS

## TEMPO DA GERAÇÃO DOS DADOS

```
Windows@LAPTOP-K5PPB93G MINGW64 ~/Desktop/json
```



```
$ C:/Users/Windows/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/Windows/Desktop/json/inserirdados - Cópia.py"
```

```
Todos os dados do arquivo dados_exemplo.json foram inseridos com sucesso.
```

```
Tempo de execução: 3732.38 segundos.
```

# TEMPOS DE RESPOTAS APÓS 50 MILHÕES DE DADOS

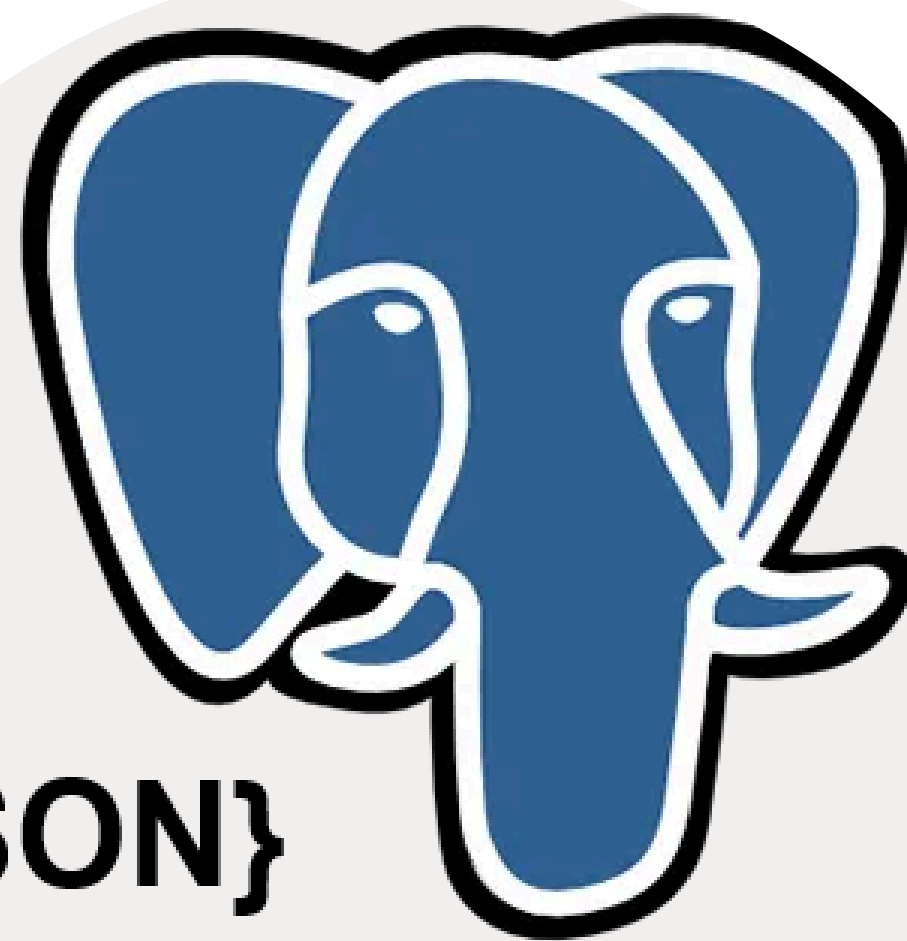
## TEMPO DA SELEÇÃO DOS DADOS

<input type="checkbox"/>			10380	Restore	PostgreSQL 16 (loca...	postgres	12/07/2024, 16:57:42	Finished	815.34
--------------------------	---	---	-------	---------	------------------------	----------	----------------------	----------	--------



# Conclusão

O Postgres é sem dúvida um dos bancos de dados mais poderosos do mercado, e a possibilidade de trabalhar com dados json nos dá muita flexibilidade, onde podemos mesclar a modelagem de nossas tabelas de forma relacional e também não-relacional graças aos JSONB Data Types.



**{JSON}**

PostgreSQL