Documentação Completa da API PHP com Redis

Sumário

1. Introdução

Esta documentação descreve detalhadamente a implementação de uma API PHP simples que utiliza Redis para armazenamento de dados. Redis é um banco de dados em memória altamente performático, frequentemente utilizado para caching, gerenciamento de sessões, filas de mensagens e operações em tempo real. Neste projeto, focamos em explorar diversas funcionalidades do Redis, como armazenamento de strings, listas, hashes, sets, e operações de publicação/assinatura (Pub/Sub).

2. Pré-requisitos

Antes de começar, certifique-se de ter os seguintes itens instalados no seu ambiente:

- **Ubuntu** (ou outro sistema baseado em Linux)

- **PHP** (versão 8 ou superior)
- **Composer** (para gerenciamento de dependências PHP)
- **Redis** (servidor Redis em execução)
- **Curl** ou um navegador para testar a API

3. Instalação e Configuração

3.1 Instalação do Redis

Para instalar o Redis no Ubuntu:

1. **Atualize o sistema:**

- **sudo apt update**

2. Instale o Redis:

- **sudo apt install redis-server**

3. Verifique se o Redis está em execução:

- **sudo systemctl status redis**

4. Teste a conexão ao Redis:

- **redis-cli ping**

3.2 Configuração do Ambiente PHP

1. **Instale o PHP e extensões necessárias:**

- **sudo apt install php php-cli php-fpm php-redis**

Verifique a instalação do PHP:

- **php -v**

3.3 Configuração do Composer

1. **Crie a pasta do projeto:**

- mkdir redis-api
- cd redis-api

Inicialize o projeto com Composer:

- composer init

Adicione a dependência Predis para trabalhar com Redis:

- composer require predis/predis

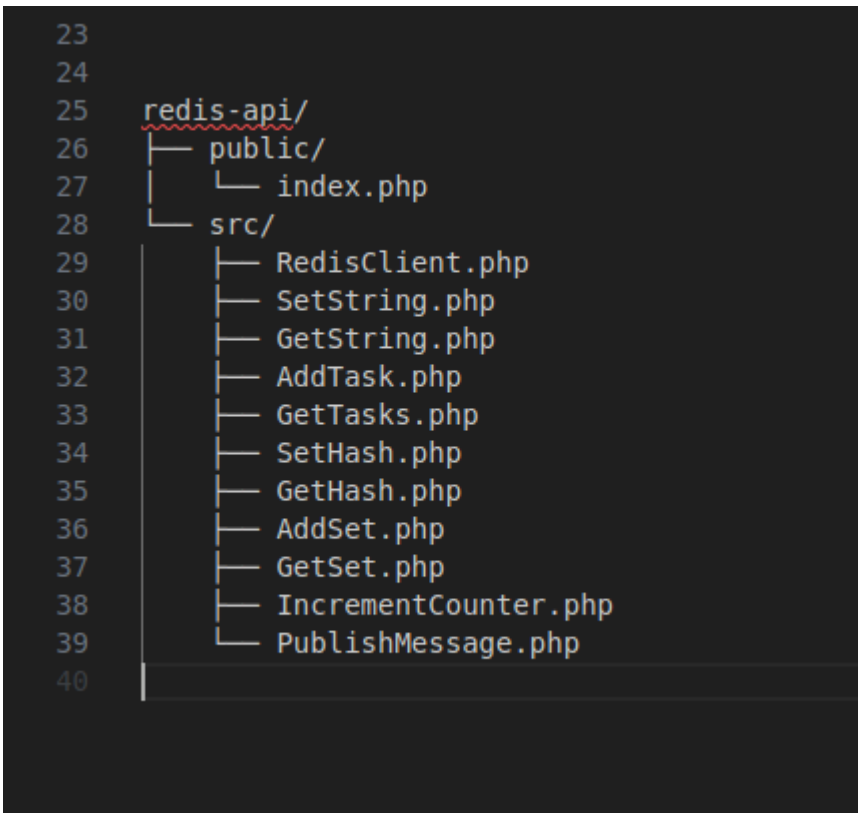
Estrutura básica do composer.json

Atualize o autoload:

composer dump-autoload

4. Estrutura do Projeto

O projeto segue a seguinte estrutura:



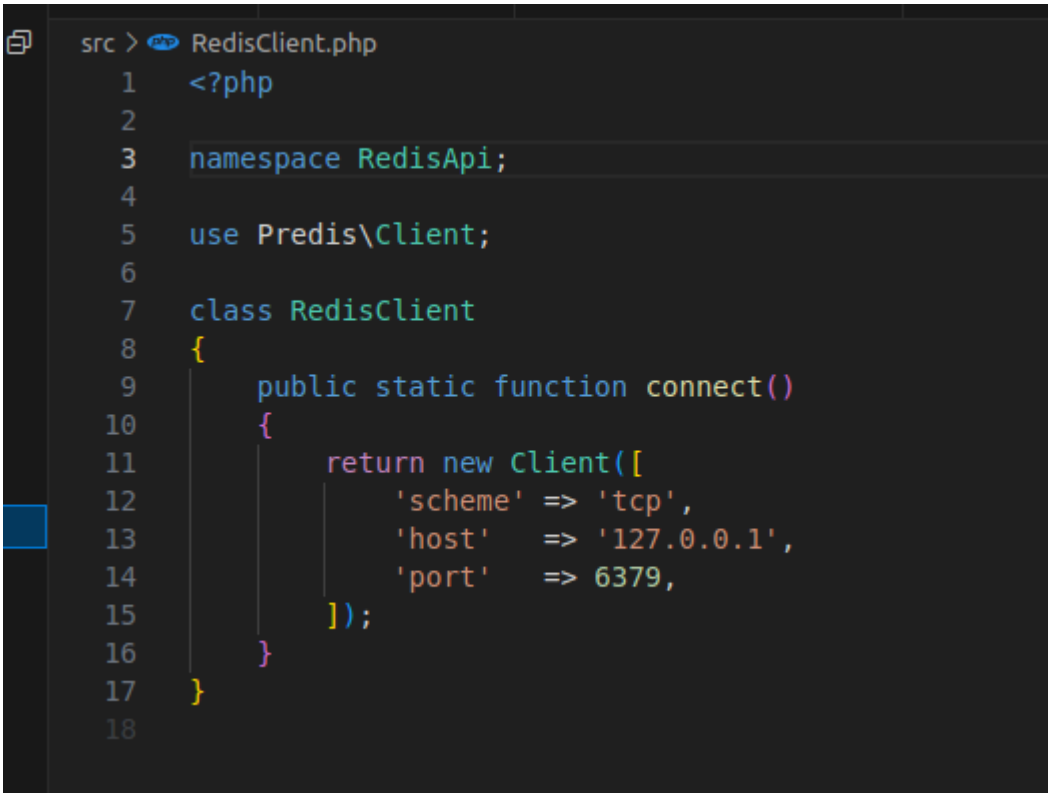
5. Explicação das Funcionalidades e Rotas

Cada rota da API foi projetada para demonstrar uma funcionalidade específica do Redis.

5.1 Conexão com o Redis

Arquivo: src\RedisClient.php

Este arquivo configura a conexão com o Redis.



Explicação:

- **Namespace:** Usamos \RedisApi para manter a organização do código em conformidade com o composer.json **Classe RedisClient:** Esta classe encapsula a lógica de conexão com o Redis, criando um novo cliente Predis.

5.2 Setar uma String

Arquivo: src\SetString.php

Esta rota permite armazenar uma string no Redis.

src > SetString.php

```php
<?php

namespace RedisApi;

class SetString
{
    public static function execute($key, $value)
    {
        $redis = RedisClient::connect();
        $redis->set($key, $value);
        return "String set successfully!";
    }
}
```

Explicação:

- **Função set:** O método setarmazena uma string no Redis. A chave (key) e o valor (value) são passados como parâmetros.
- **Armazenamento no Redis:** A string é armazenada na memória do Redis sob a chave especificada.

5.3 Obter uma String

Arquivo: src\GetString.php

Esta rota permite recuperar uma string armazenada no Redis.

src > GetString.php

```php
<?php

namespace RedisApi;

class GetString
{
    public static function execute($key)
    {
        $redis = RedisClient::connect();
        return $redis->get($key) ?? "Key not found.";
    }
}
```

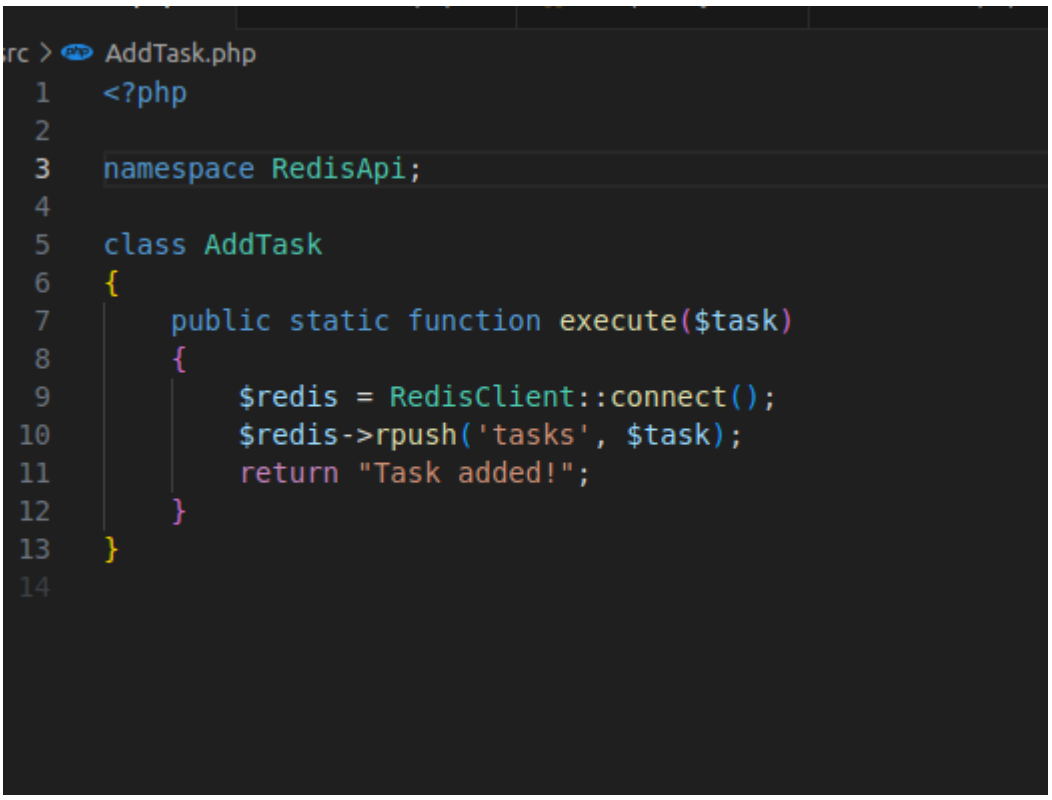
Explicação:

- **Função get:** O método get recupera o valor armazenado para a chave especificada no Redis. Se a chave não existir, retorna "Key not found.".
- **Armazenamento no Redis:** O valor é recuperado diretamente da memória onde o Redis armazena a string.

5.4 Adicionar Tarefa à Lista

Arquivo: src\AddTask.php

Esta rota permite adicionar uma tarefa a uma lista em Redis.



Explicação:

- **Função rpush:** rpushadiciona o item ao final da lista tasks. Se a lista não existir, ela é criada.
- **Armazenamento no Redis:** A lista tasks é armazenada como uma estrutura de dados em memória.

5.5 Obter Tarefas da Lista

Arquivo: src\GetTasks.php

Esta rota permite recuperar todas as tarefas de uma lista.

src > GetTasks.php

```php
<?php

namespace RedisApi;

class GetTasks
{
    public static function execute()
    {
        $redis = RedisClient::connect();
        $tasks = $redis->lrange('tasks', 0, -1);
        return $tasks ? "Tasks: " . implode(', ', $tasks) : "No tasks found.";
    }
}
```

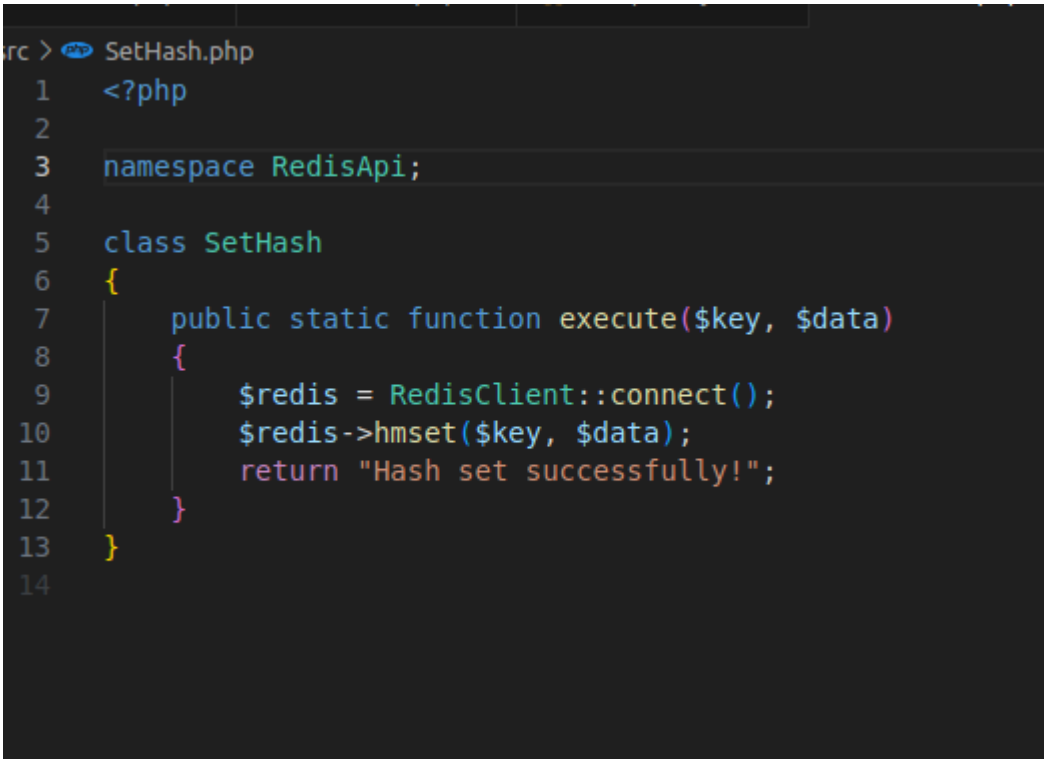
Explicação:

- **Função lrange:** lrangeretorna os elementos da lista tasksdo início (0) ao fim (-1).
- **Armazenamento no Redis:** A lista completa é mantida na memória e pode ser recuperada rapidamente.

5.6 Setar um Hash

Arquivo: src\SetHash.php

Esta rota permite armazenar um hash no Redis.



Explicação:

- **Função hmset: hmset** armazena um hash em Redis. Um hash é um mapa entre campos e valores.
- **Armazenamento no Redis:** O hash é armazenado como uma estrutura de dados complexa em Redis.

5.7 Obter um Hash

Arquivo: src\GetHash.php

Esta rota permite recuperar um hash do Redis.

src > GetHash.php

```php
<?php

namespace RedisApi;

class GetHash
{
    public static function execute($key)
    {
        $redis = RedisClient::connect();
        $hash = $redis->hgetall($key);
        return $hash ? "Hash: " . implode(', ', $hash) : "Hash not found.";
    }
}

```

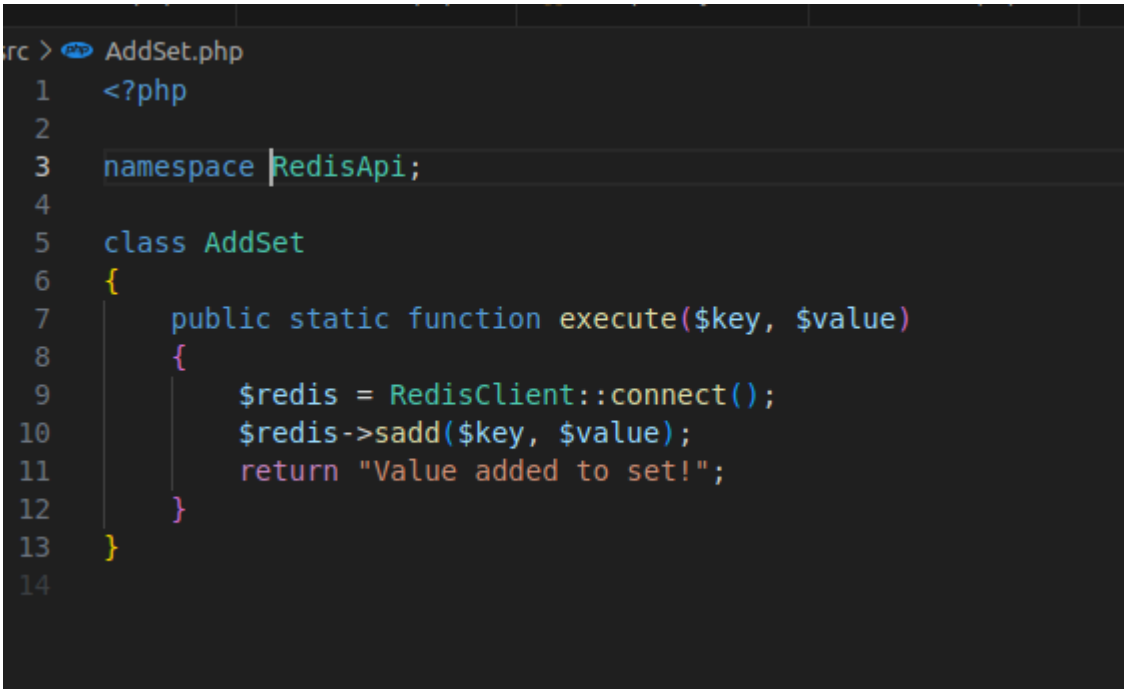
Explicação:

- **Função hgetall: hgetall** recupera todos os campos e valores associados a um hash.
- **Armazenamento no Redis:** A estrutura de hash é recuperada diretamente da memória.

5.8 Adicionar Valor a um Set

Arquivo: src\AddSet.php

Esta rota permite adicionar valores únicos a um set.



Explicação:

- **Função sadd: sadd** adiciona um valor a um set. Sets são coleções de valores únicos.
- **Armazenamento no Redis:** O set é armazenado na memória e garante que cada valor seja único.

5.9 Obter Valores de um Set

Arquivo: src\GetSet.php

Esta rota permite recuperar todos os valores de um set.

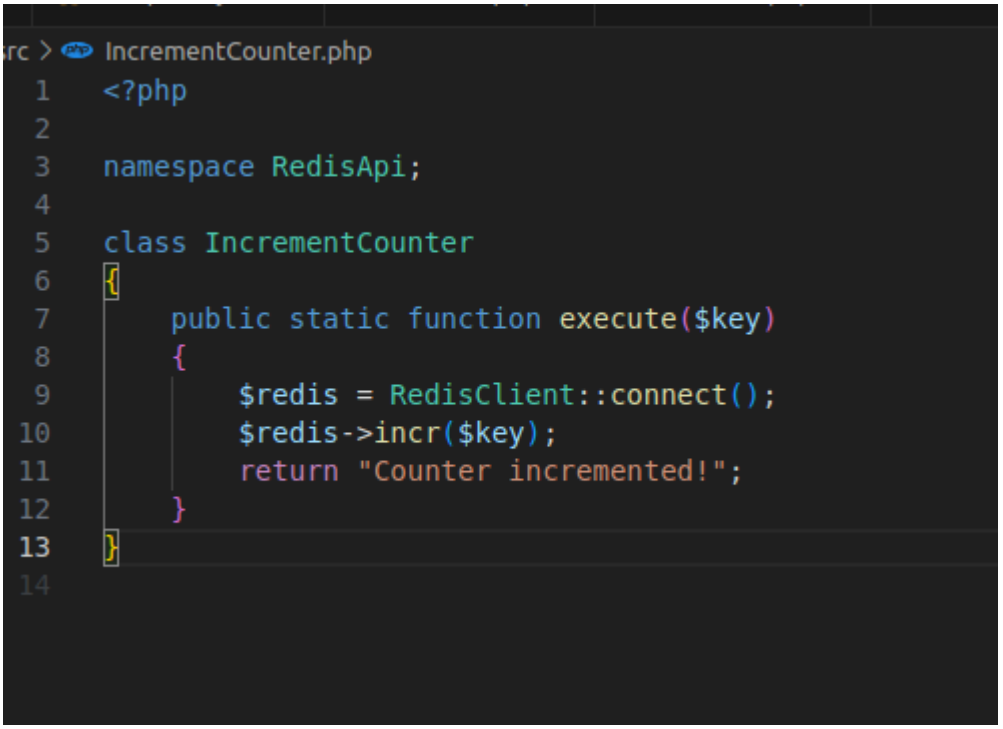
Explicação:

- **Função smembers: smembers** retorna todos os membros de um set.
- **Armazenamento no Redis:** O set é recuperado como uma coleção de valores únicos.

5.10 Incrementar um Contador

Arquivo: src\IncrementCounter.php

Esta rota permite incrementar um contador no Redis.



Explicação:

- **Função incr: incr** incrementa o valor armazenado para a chave especificada. Se a chave não existir, ela é criada e iniciada com 1.
- **Armazenamento no Redis:** O valor do contador é armazenado como uma string que representa um número inteiro.

5.11 Publicar uma Mensagem

Arquivo: src\PublishMessage.php

Esta rota permite publicar uma mensagem em um canal Redis.

src > PublishMessage.php

```php
<?php

namespace RedisApi;

class PublishMessage
{
    public static function execute($channel, $message)
    {
        $redis = RedisClient::connect();
        $redis->publish($channel, $message);
        return "Message published!";
    }
}
```

Explicação:

- **Função publish: publish** envia uma mensagem para todos os clientes inscritos no canal especificado.
- **Armazenamento no Redis:** A mensagem não é armazenada; é transmitida para os assinantes em tempo real.

6. Como o Redis Armazena os Dados

O Redis armazena os dados em memória RAM, o que permite acesso extremamente rápido. Cada tipo de dado (string, list, set, hash) tem sua própria estrutura de armazenamento:

- **Strings:** Armazenadas como pares chave-valor simples.
- **Lists:** Armazenadas como listas ligadas (linked lists).
- **Hashes:** Armazenados como mapas de hash que mapeiam campos para valores.
- **Sets:** Armazenados como coleções de valores únicos, sem ordem.
- **Contadores:** Implementados como strings que representam números inteiros.
- **Pub/Sub:** Mensagens não são armazenadas; são transmitidas diretamente para os assinantes.

7. Benefícios de Usar Redis

Redis oferece vantagens significativas para certos tipos de aplicações:

- **Alta performance:** Como os dados são armazenados em memória, o Redis é extremamente rápido.
- **Versatilidade:** Suporta diferentes tipos de dados (strings, hashes, lists, sets).
- **Simplicidade:** Fácil de usar, com uma interface simples para armazenamento e recuperação de dados.
- **Escalabilidade:** Redis pode ser escalado horizontalmente com clusters.

7.1 Quando Usar Redis

- **Caching:** Para armazenar respostas de API ou resultados de consultas frequentemente acessadas.
- **Sessões:** Para gerenciamento de sessões em aplicações web.
- **Filas de Tarefas:** Para enfileirar e processar tarefas em background.
- **Contagem e Monitoramento:** Para contagem de eventos em tempo real, como visualizações de página.

7.2 Quando Não Usar Redis

- **Persistência de Dados Críticos:** Redis não é ideal para armazenamento de dados que requerem persistência robusta e garantias de consistência ACID.
- **Grandes Volumes de Dados:** Como o Redis armazena dados em memória, grandes volumes de dados podem ser caros e ineficientes.
- **Transações Complexas:** Redis não é uma escolha ideal para sistemas que exigem transações complexas entre múltiplos tipos de dados.

8. Conclusão

Este projeto demonstrou como implementar uma API PHP simples que utiliza Redis para diversas operações de armazenamento e manipulação de dados. Redis é uma ferramenta poderosa que pode melhorar significativamente a performance de suas aplicações quando usado adequadamente. No entanto, é importante avaliar quando usar Redis e quando uma solução de armazenamento mais tradicional pode ser mais apropriada.