Documentação do arquivo `training.cls`

Introdução

O arquivo `training.cls` contém uma classe Apex chamada `training`, que é usada para buscar contatos no Salesforce com base em um nome fornecido. A classe utiliza a anotação `@AuraEnabled`, permitindo que o método seja acessado a partir de componentes Lightning ou outras interfaces que suportem chamadas de métodos Apex.

Descrição

A classe `training` possui um método estático chamado `getContact`, que realiza uma consulta SOQL na tabela de contatos (`Contact`) para buscar registros cujo nome contenha uma string específica. O método retorna uma lista de contatos que correspondem ao critério de busca.

Estrutura

A estrutura do arquivo é composta por:

- Uma classe pública chamada `training`.
- Um método estático público chamado `getContact`, que é habilitado para uso em componentes Lightning.

Dependências

A classe depende do seguinte:

- Objeto padrão do Salesforce: `Contact`.
- Linguagem de consulta SOQL para buscar registros no banco de dados do Salesforce.

Imports

Não há imports explícitos no código, pois o Salesforce Apex não utiliza uma sintaxe de importação como outras linguagens. No entanto, ele depende implicitamente do objeto padrão `Contact`.

Variáveis

- **`searchContact`**: Uma variável local do tipo `String` que não é utilizada no código atual. Pode ser removida ou utilizada em futuras implementações.
- **`contact`**: Uma lista de objetos `Contact` que armazena os resultados da consulta SOQL.

Métodos

`getContact(String name)`

- **Descrição**: Este método realiza uma consulta SOQL para buscar contatos cujo nome contenha a string fornecida como parâmetro.
- **Parâmetros**:
  - `name` (String): O nome ou parte do nome do contato que será usado como critério de busca.
- **Retorno**: Retorna uma lista de objetos `Contact` que correspondem ao critério de busca.
- **Anotações**:
  - `@AuraEnabled`: Permite que o método seja chamado a partir de componentes Lightning ou outras interfaces externas.
  - `public static`: O método é público e estático, ou seja, pode ser acessado sem a necessidade de instanciar a classe.
- **Lógica**:
  1. Realiza uma consulta SOQL no objeto `Contact` para buscar registros cujo nome contenha a string fornecida.
  2. Exibe os resultados no log do sistema usando `System.debug`.

3. Retorna a lista de contatos encontrados.

Exemplo

Aqui está um exemplo de como o método `getContact` pode ser chamado a partir de um componente Lightning:

```
// JavaScript Controller em um componente Lightning
import { LightningElement, wire } from 'lwc';
import getContact from '@salesforce/apex/training.getContact';

export default class ContactSearch extends LightningElement {
    searchKey = '';
    contacts;

    handleSearch(event) {
        this.searchKey = event.target.value;
        getContact({ name: this.searchKey })
            .then(result => {
                this.contacts = result;
            })
            .catch(error => {
                console.error('Erro ao buscar contatos:', error);
            });
    }
}
```

Diagrama de Dependência

O diagrama abaixo ilustra a relação entre a classe `training` e o objeto `Contact`:

```
classDiagram
    class training {
        +List~Contact~ getContact(String name)
    }
    class Contact {
        +String Name
        +String Phone
        +String Email
        +String Description
    }
    training --> Contact : Consulta SOQL
```

Notas

- A variável `searchContact` não está sendo utilizada no código e pode ser removida para evitar confusão.
- O método utiliza a consulta SOQL com o operador `LIKE` para realizar buscas parciais. Certifique-se de que o índice do campo `Name` esteja otimizado para evitar problemas de desempenho em grandes volumes de dados.

Vulnerabilidades

- **Injeção SOQL**: O código está protegido contra injeção SOQL, pois utiliza a vinculação de variáveis (`:`) para inserir o parâmetro `name` na consulta.
- **Desempenho**: Consultas que utilizam o operador `LIKE` podem ser lentas em tabelas com muitos registros, especialmente se o campo `Name` não estiver indexado.
- **Limite de Governança**: Certifique-se de que o número de registros retornados pela consulta não exceda os limites de governança do Salesforce. Caso contrário, pode ser necessário implementar paginação.