Nome: Gabriel Saque Marquez
Nome: Danilson Matsushita Junior
Nome: Pedro Henrique Silva Fernandes

Princípios de refatoração SOLID utilizados no sistema.

Princípio da Responsabilidade Única
Refatoração: Dividir a classe GerenciadorContatos em duas classes separadas: ListaContatos e ServiçoContato.

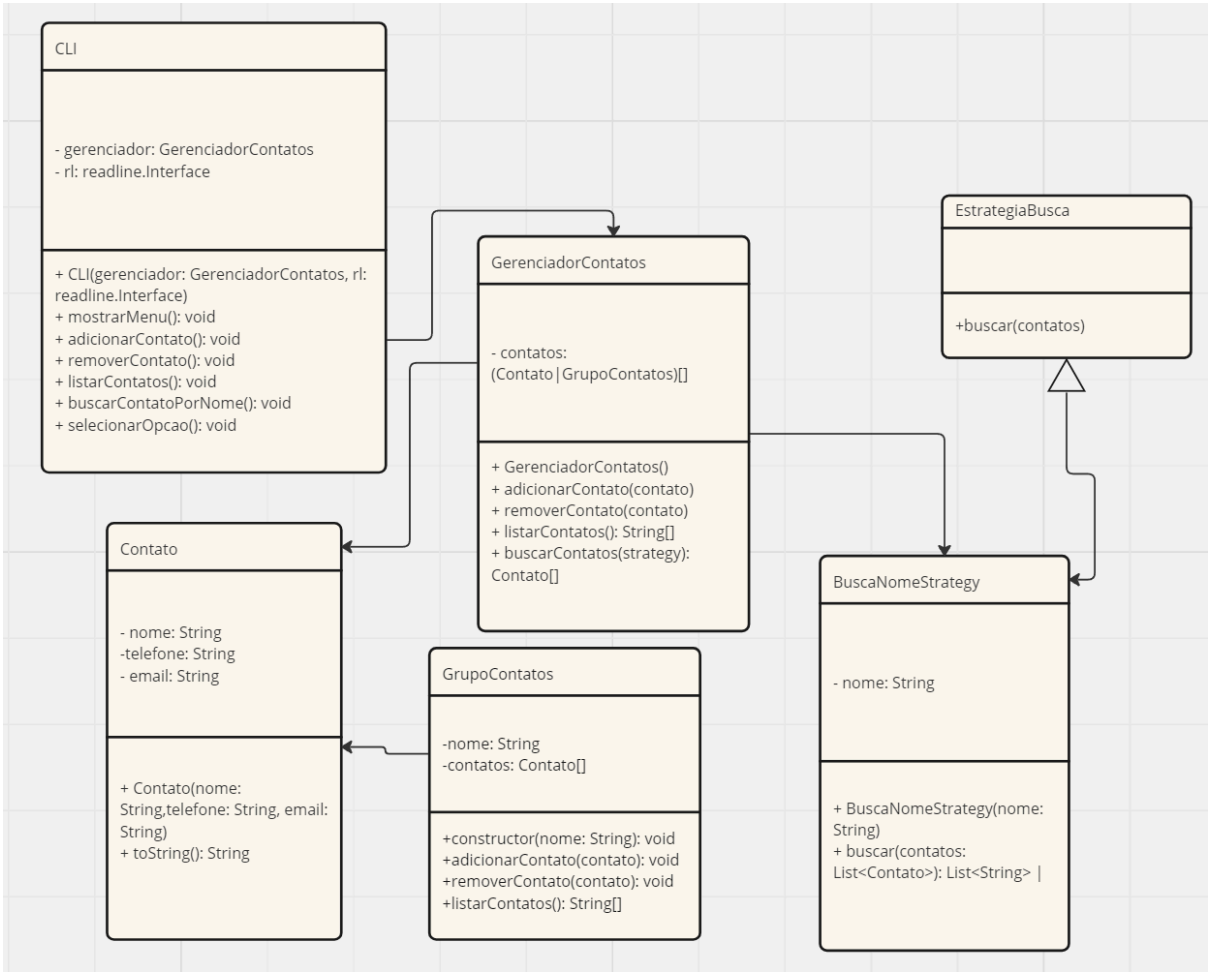
Justificativa: Isso separa a responsabilidade de gerenciar a lista de contatos (em ListaContatos) da responsabilidade de manipular e interagir com os contatos (em ServiçoContato), seguindo o SRP.

Princípio Aberto/Fechado
Refatoração: Criar uma interface IStorage e uma classe LocalStorage que implementa IStorage.

Justificativa: Isso permite que o sistema seja facilmente estendido para suportar diferentes tipos de armazenamento (por exemplo, armazenamento em nuvem) sem modificar o código existente, seguindo o OCP.

Princípio da Segregação de Interfaces
Refatoração: Extrair interfaces da classe ServiçoContato para separar as responsabilidades em interfaces menores e mais específicas, como AdicionarContato, RemoçãoContato, PesquisaContato, etc.

Justificativa: Isso garante que os clientes só precisam implementar as interfaces relevantes para eles, evitando a necessidade de implementar métodos desnecessários, seguindo o ISP.



GeneradorDeContratos: Esta classe parece ser responsável pela geração de contratos, o que é indicado pelo método “+generarContrato()”. Isso sugere que a classe tem a responsabilidade de criar novos contratos.
EstrategiaBusca: Esta classe é provavelmente uma interface ou uma classe abstrata, o que é comum em padrões de projeto como o Strategy. O método “+buscar()” indica que qualquer classe que implemente ou estenda “EstrategiaBusca” deve fornecer uma implementação para a operação de busca.
BuscaNombreStrategy: Esta classe implementa “EstrategiaBusca”, o que é indicado pela seta de herança/implementação. Isso sugere que “BuscaNombreStrategy” fornece uma implementação específica do método de busca, possivelmente buscando contratos por nome.
GrupoContratos: Esta classe tem um atributo “estrategiaBusca” do tipo “EstrategiaBusca”, o que sugere que ela usa a estratégia de busca para realizar operações. Isso é um exemplo do padrão de projeto Strategy, onde o comportamento de uma classe pode ser alterado em tempo de execução ao mudar a estratégia de busca.

Classe sem nome: A operação “+leerArchivo()” sugere que esta classe é responsável por ler arquivos. No entanto, sem mais informações, é difícil justificar mais sobre o propósito desta classe.
As relações entre as classes são justificadas pelas linhas que as conectam. Por exemplo, a linha entre “GeneradorDeContratos” e “GrupoContratos” sugere uma relação de associação, onde “GeneradorDeContratos” usa “GrupoContratos” para algum propósito, possivelmente para gerar contratos para um grupo.