Relatório sobre alterações nas exceptions

1. Introdução:

- Ao analisar as classes de exceção fornecidas, identifiquei uma redundância significativa no código.
- Propus a criação de uma nova classe base, chamada **BusinessException**, para consolidar o código comum e reduzir a duplicação.

2. Refatoração da Classe Base (BaseRuntimeException):

- Modifiquei a classe para incluir uma implementação padrão para o método **getExceptionKey()**.
- Tornei a classe abstrata para garantir que as subclasses forneçam suas próprias implementações.

3. Introdução da Classe BusinessException:

- Criei uma nova classe, **BusinessException**, que estende a classe base **BaseRuntimeException**.
- Introduzi um prefixo de chave para identificar exceções de negócios.
- Adicionei construtores sobrecarregados para lidar com diferentes tipos de dados e gerar mensagens de exceção automaticamente.
- Modifiquei o método **getExceptionKey()** para retornar a chave específica da exceção de negócios.

4. Exemplo de Uso na Subclasse AlreadyExistsException:

- Demonstrei como usar a nova classe **BusinessException** para a exceção de "já existente".
- Os parâmetros fornecidos ao construtor geram automaticamente as mensagens de exceção.

5. Resultados Esperados:

- Espero que as alterações propostas reduzam significativamente a duplicação de código.
- As subclasses de exceção agora podem ser simplificadas, utilizando a nova classe **BusinessException** para gerenciar a lógica comum.

6. Considerações Adicionais:

- Observei que, em algumas instâncias, a chamada **super.getExceptionKey()** na classe base pode ter causado confusão.
- Corrigi essa questão para garantir que as subclasses implementem diretamente o método **getExceptionKey()** sem depender da chamada à superclasse.