Introdução:

Nesta sprint, foram implementadas melhorias em diversas classes do pacote **unicap.br.unimpact.service.auxiliary**. As alterações incluem adição de comentários Javadoc, utilização de padrões de nomenclatura mais descritivos, simplificação de código e outras práticas recomendadas.

EntityValidator.java

1. Comentários Adicionados:

- Adicionei um comentário Javadoc explicando a função do método **validate**.

2. Utilização de Collectors.toList():

- Substituí a chamada de **toList()** por **collect(Collectors.toList())**, tornando o código mais claro e aderente ao estilo funcional.

3. Tratamento de Exceção Melhorado:

- Adicionei chaves **{}** ao redor do bloco do if para melhorar a legibilidade e manter consistência no estilo de codificação.

4. Reformatação de Linhas:

- Reformatei algumas linhas para garantir um código mais legível e consistente.

ModelMap.java

1. Comentários Adicionados:

- Adicionei um comentário Javadoc explicando a função do método **map**.

2. Utilização de Nomes Descritivos:

- Renomeei os parâmetros de **src** e **dest** para **source** e **destiny**, tornando os nomes mais descritivos.

3. Declaração de ModelMapper Simplificada:

- Simplifiquei a declaração do ModelMapper, removendo o pacote redundante **org.modelmapper**.

4. Reformatação de Linhas:

- Reformatei algumas linhas para garantir um código mais legível e consistente.

NotificationFactory.java

Principais Alterações:

1. Comentários Adicionados:

- Adicionei comentários Javadoc explicando a função dos métodos e suas responsabilidades.

2. Reformatação de Código:

- Reformatei o código para seguir convenções de estilo e melhorar a legibilidade.

3. Ajuste de Nomes:

- Padronizei os nomes dos métodos para seguir convenções de nomenclatura mais comuns.

4. Melhoria na Manipulação de Mensagens:

- Ajustei o método **findMessageByKey** para ser mais claro e refatorei os métodos de criação de notificações para utilizar mensagens diretamente, tornando o código mais conciso.

ProjectFlowControl.java

1. Refatoração dos Métodos de Estado:

- Substituí os blocos de código longos por métodos privados mais específicos (**setNextStatusAndStateDraft**, **setNextStatusAndStateProposal**, **setNextStatusAndStateProject**) para melhorar a legibilidade.

2. Padronização dos Métodos:

- Padronizei o método **setNextStateAndStatus** para evitar repetição de código.

3. Simplificação do Método **setNextStatusAndState**:

- Simplifiquei o método **setNextStatusAndState** para chamar diretamente os métodos de estado correspondentes, melhorando a legibilidade.

4. Uso de switch no Método **getAuthorizedUsers**:

- Utilizei a instrução switch para melhorar a legibilidade do método **getAuthorizedUsers**.

5. Simplificação dos Métodos **returnAuthorizedUsers***:

- Simplifiquei os métodos **returnAuthorizedUsers*** para utilizar switch, tornando-os mais concisos.