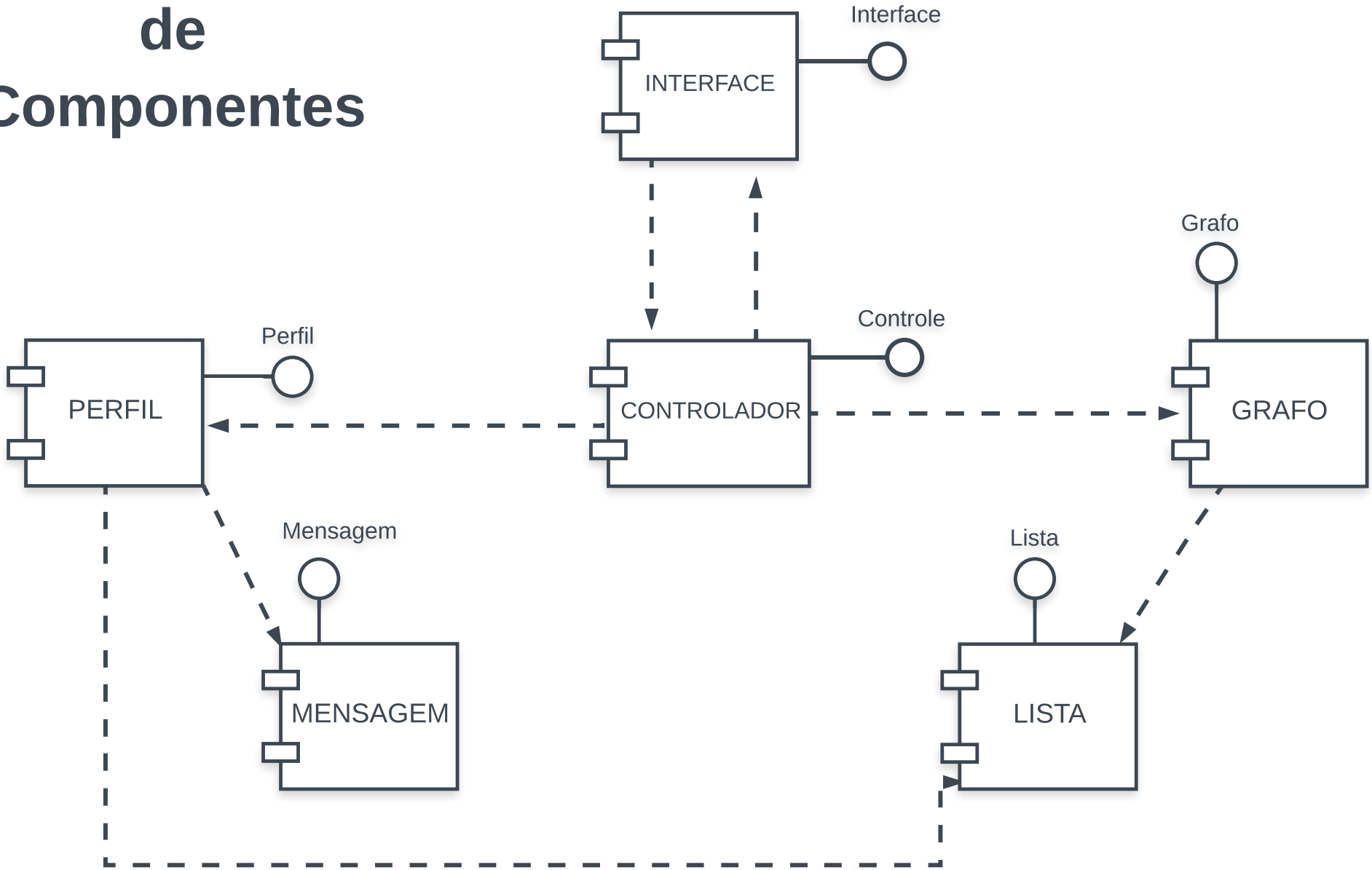
# Arquitetura de Módulos

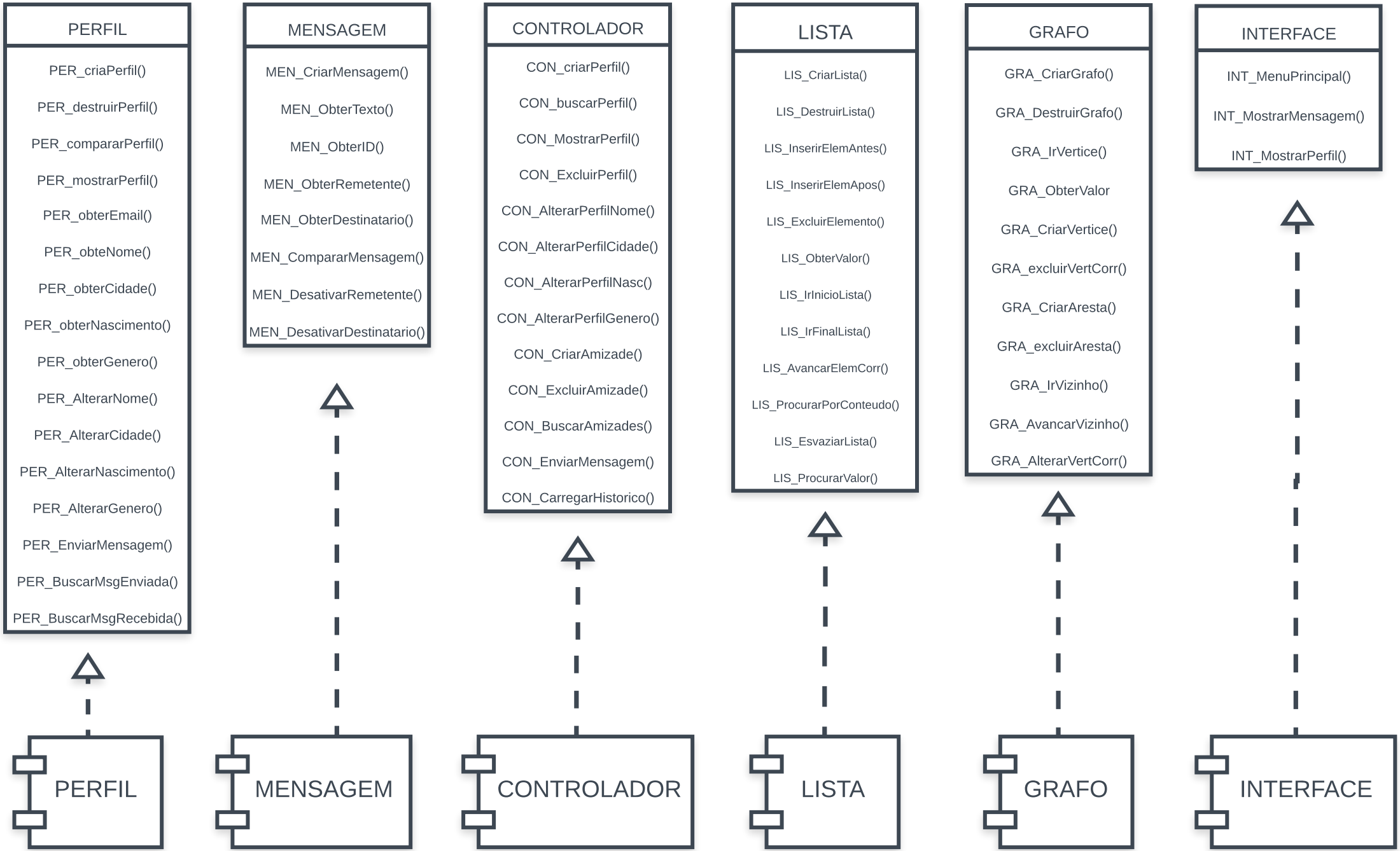
Trabalho 4 - Rede de Relacionamentos
INF 1301 - PUC-RIO

Grupo:
Gabriel Cantergiani
Gabrielle Bradenburg
Wellington Bezerra

# Modelo de Componentes



# Interfaces


PERFIL
PER_criaPerfil()
PER_destruirPerfil()
PER_compararPerfil()
PER_mostrarPerfil()
PER_obterEmail()
PER_obteNome()
PER_obterCidade()
PER_obterNascimento()
PER_obterGenero()
PER_AlterarNome()
PER_AlterarCidade()
PER_AlterarNascimento()
PER_AlterarGenero()
PER_EnviarMensagem()
PER_BuscarMsgEnviada()
PER_BuscarMsgRecebida()
MENSAGEM
MEN_CriarMensagem()
MEN_ObterTexto()
MEN_ObterID()
MEN_ObterRemetente()
MEN_ObterDestinatario()
MEN_CompararMensagem()
MEN_DesativarRemetente()
MEN_DesativarDestinatario()
CONTROLADOR
CON_criarPerfil()
CON_buscarPerfil()
CON_MostrarPerfil()
CON_ExcluirPerfil()
CON_AlterarPerfilNome()
CON_AlterarPerfilCidade()
CON_AlterarPerfilNasc()
CON_AlterarPerfilGenero()
CON_CriarAmizade()
CON_ExcluirAmizade()
CON_BuscarAmizades()
CON_EnviarMensagem()
CON_CarregarHistorico()
LISTA
LIS_CriarLista()
LIS_DestruirLista()
LIS_InserirElemAntes()
LIS_InserirElemApos()
LIS_ExcluirElemento()
LIS_ObterValor()
LIS_IrInicioLista()
LIS_IrFinalLista()
LIS_AvancarElemCorr()
LIS_ProcurarPorConteudo()
LIS_EsvaziarLista()
LIS_ProcurarValor()
GRAFO
GRA_CriarGrafo()
GRA_DestruirGrafo()
GRA_IrVertice()
GRA_ObterValor
GRA_CriarVertice()
GRA_excluirVertCorr()
GRA_CriarAresta()
GRA_excluirAresta()
GRA_IrVizinho()
GRA_AvancarVizinho()
GRA_AlterarVertCorr()
INTERFACE
INT_MenuPrincipal()
INT_MostrarMensagem()
INT_MostrarPerfil()
PERFIL
MENSAGEM
CONTROLADOR
LISTA
GRAFO
INTERFACE

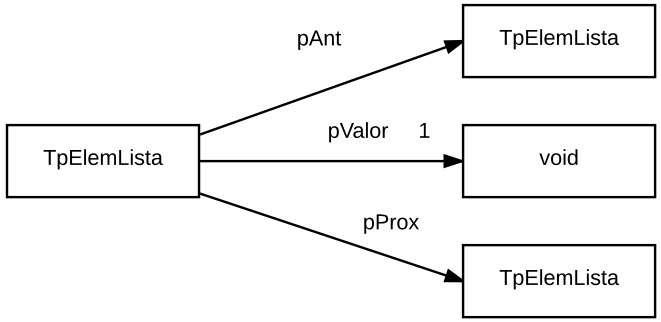
# Modelagem Física

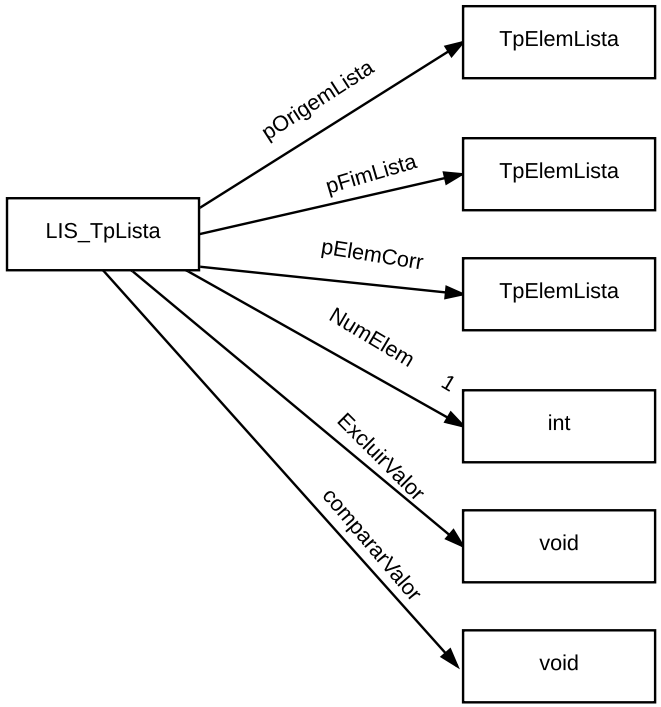
Trabalho 4 - Rede de Relacionamentos
INF 1301 - PUC-RIO

Grupo:
Gabriel Cantergiani
Gabrielle Bradenburg
Wellington Bezerra

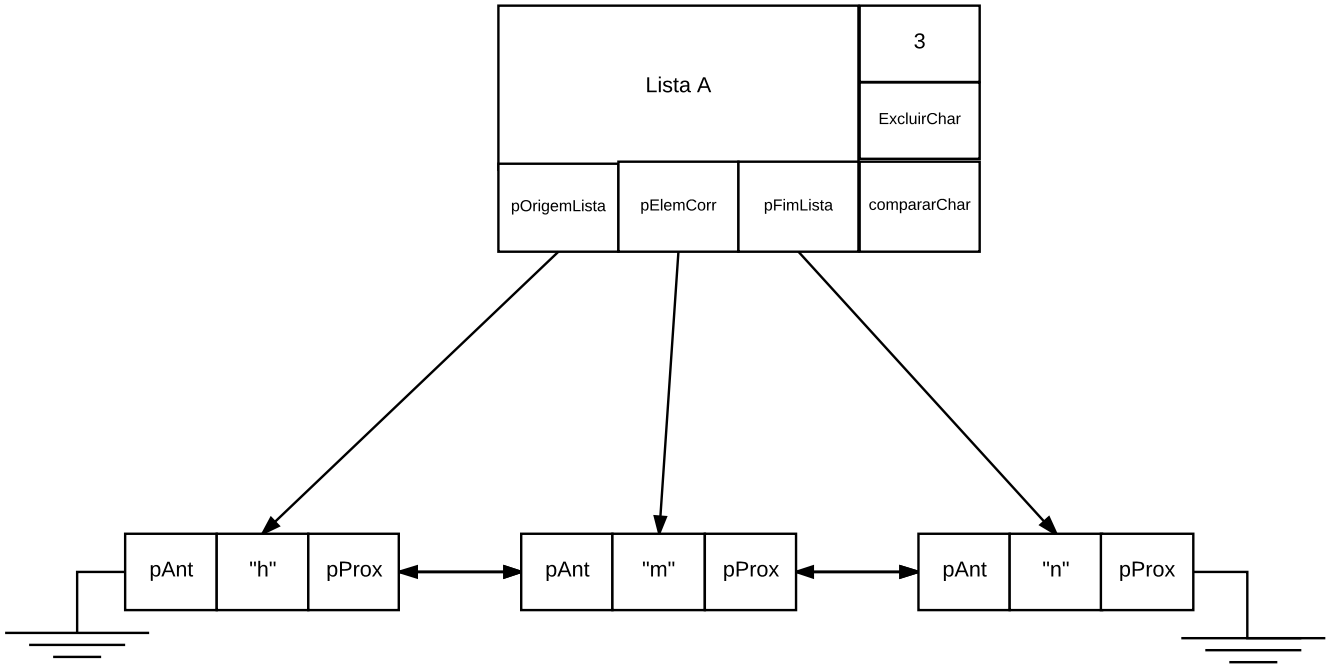
# LISTA DUPLAMENTE ENCADEADA

## Modelo Físico




## Exemplo Físico




## Assertivas

Para todo elemento Elem da lista:

- se Elem-> pAnt != NULL, então Elem -> pAnt -> pProx == Elem

- se Elem -> pProx != NULL, então Elem -> pProx -> pAnt == Elem

- se Elem é o primeiro da lista, então Elem -> pAnt == NULL

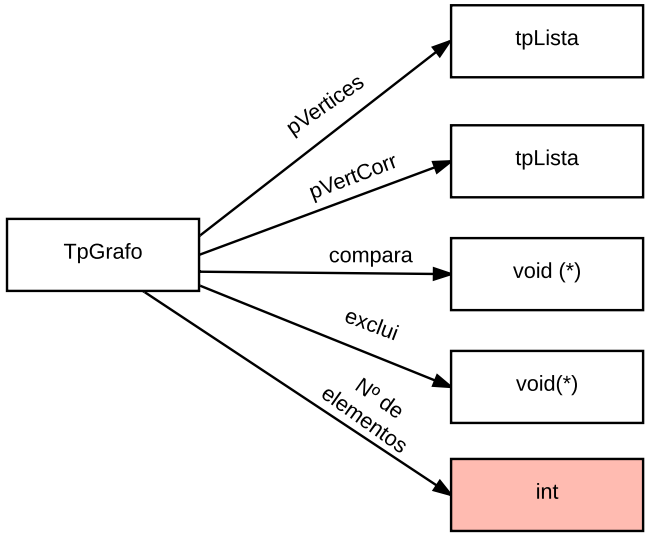
-se Elem é o último da lista, então Elem -> pProx == NULL

# GRAFO

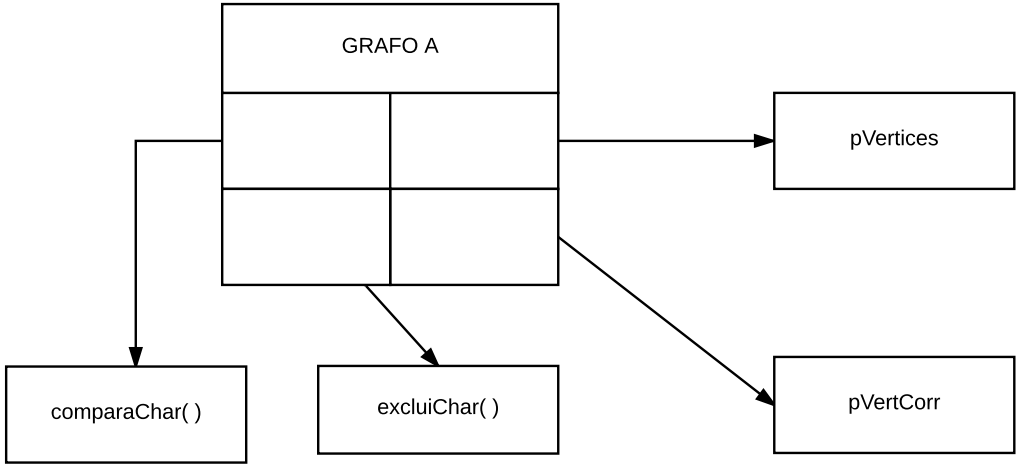
= Redundância

## Cabeça do Grafo

### Modelo Físico

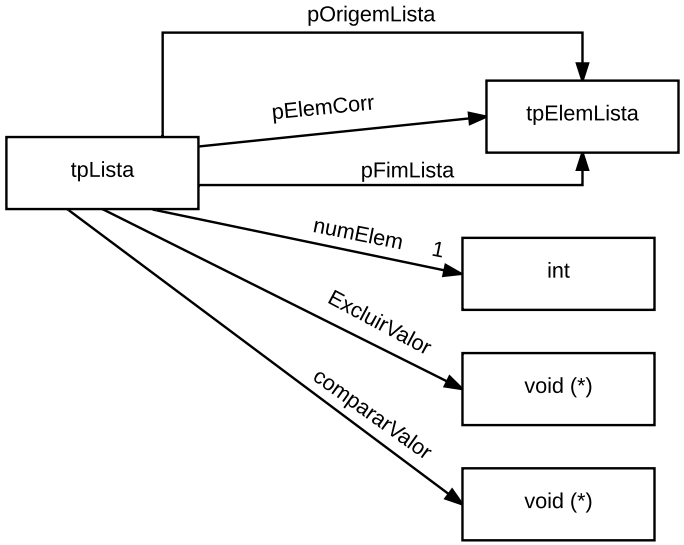


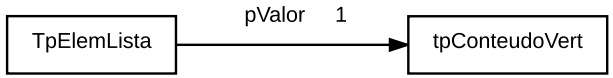

### Exemplo Físico



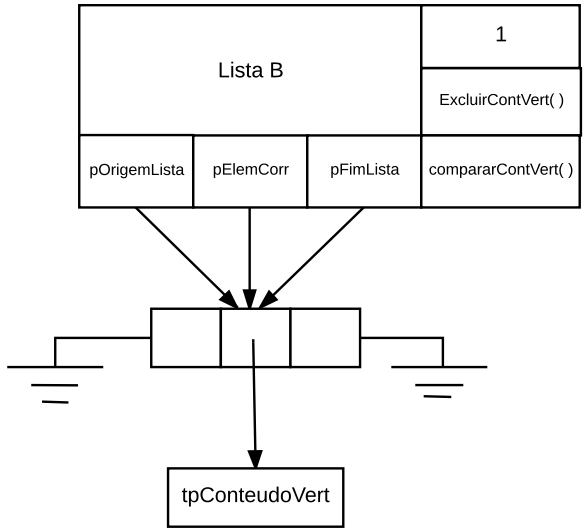

## Vértice (lista de 1 nó)

### Modelo Físico



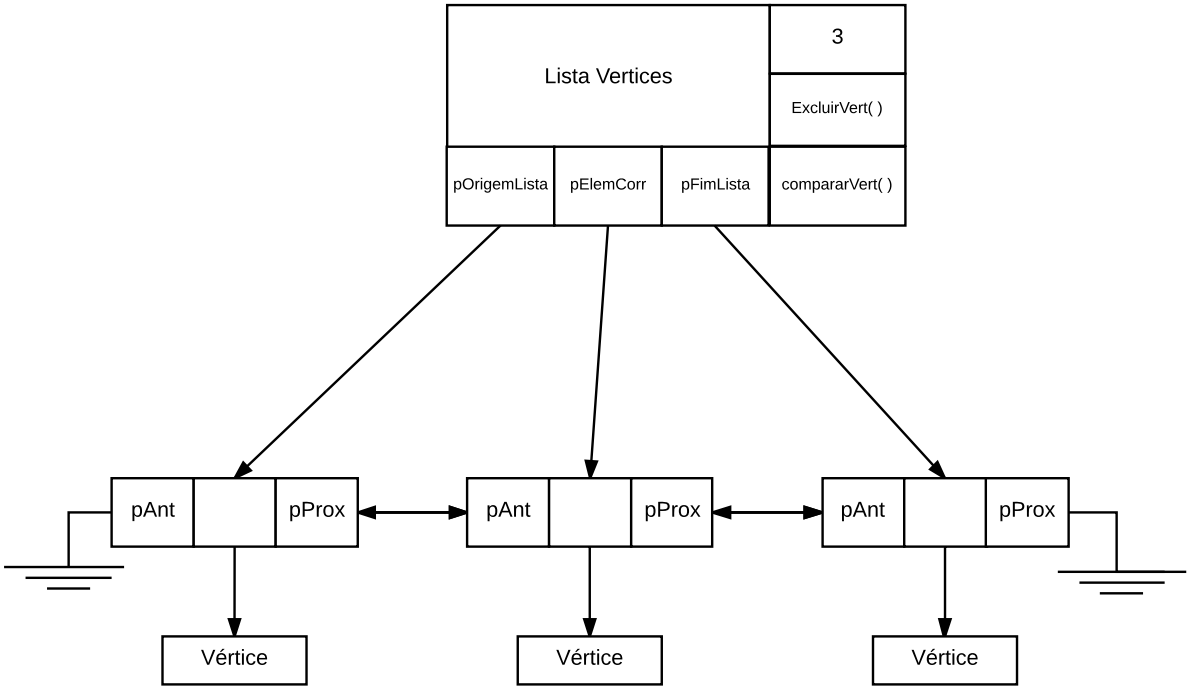

### Exemplo Físico



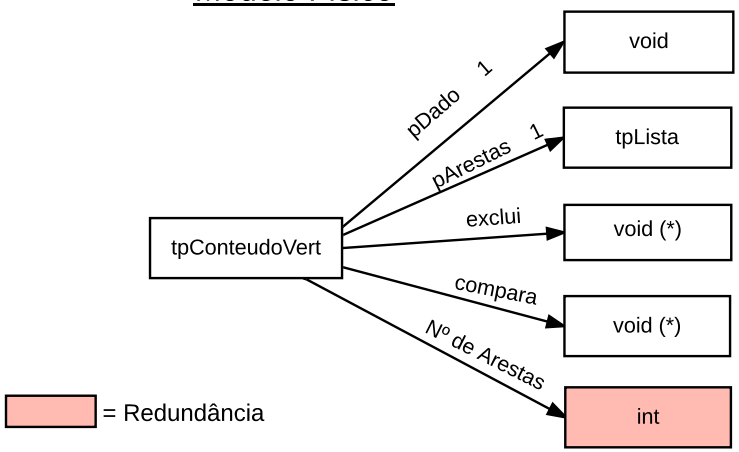
# GRAFO (continuação)

## Lista Vértices

### Exemplo Físico

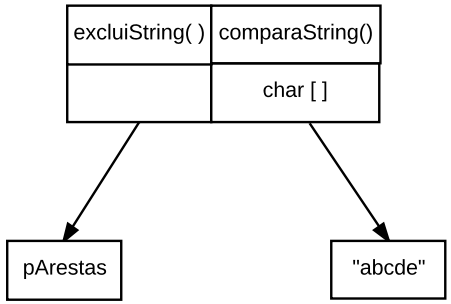



### Modelo Físico




## Conteúdo do Vértice

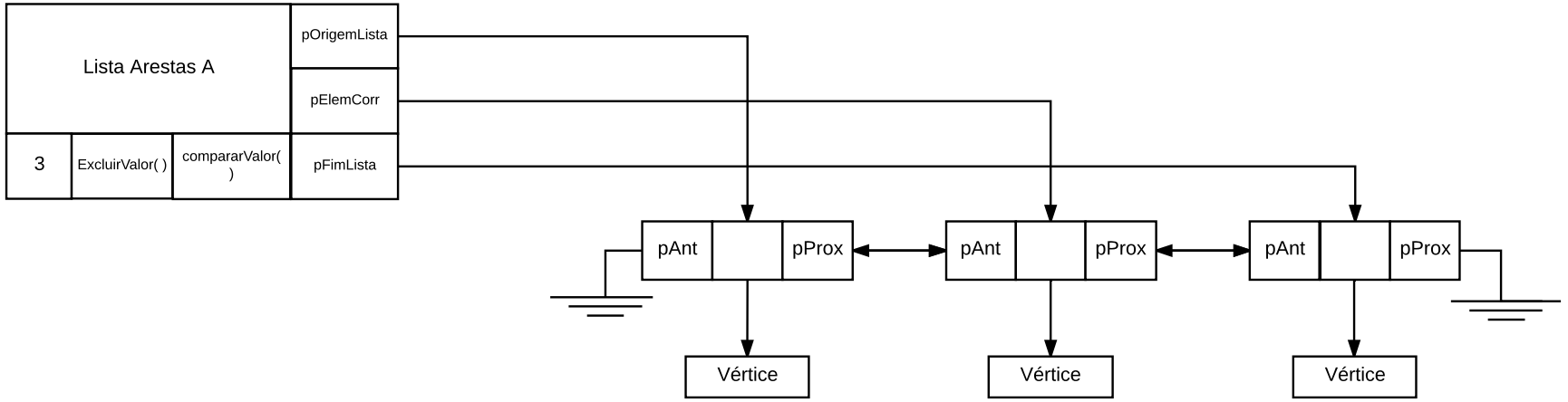
### Exemplo Físico



# GRAFO (continuação)

## Lista Arestas

Exemplo Físico




---

## Assertivas de Grafo

- Se a Lista Vértices é não vazia, então tpGrafo -> pVerticeCorrente != NULL .

-Se pVertices -> numElem ==1 , então tpGrafo -> pVerticeCorrente == tpGrafo -> pOrigem -> pElemCorr .

- Se ElemLista == tpGrafo -> pVerticeCorrente , então ElemLista pertence à tpGrafo - > pVertices .

- Se a Lista Vértices é não vazia, então sempre existe um elemento (ElemLista) pertencente à Lista Vértices tal que tpGrafo -> pOrigem == ElemLista .

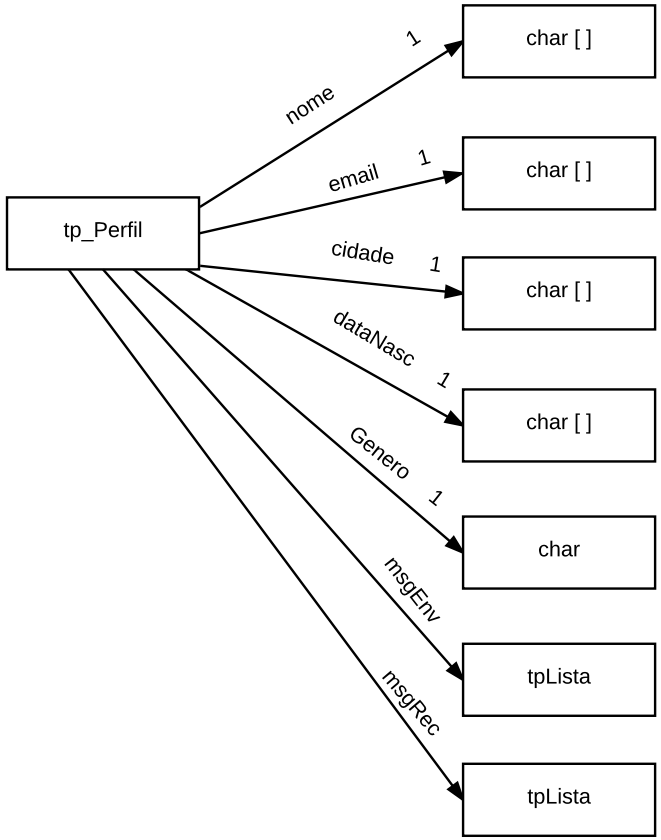
- Se a lista de arestas do vértice X possui um elemento cujo valor é o vértice Y, então a lista de arestas do vértice Y deve possuir um elemento cujo valor é o vértice X. (Grafo não-dirigido)

- A lista de arestas do Vértice X NÃO PODE conter um elemento cujo valor é o próprio elemento X

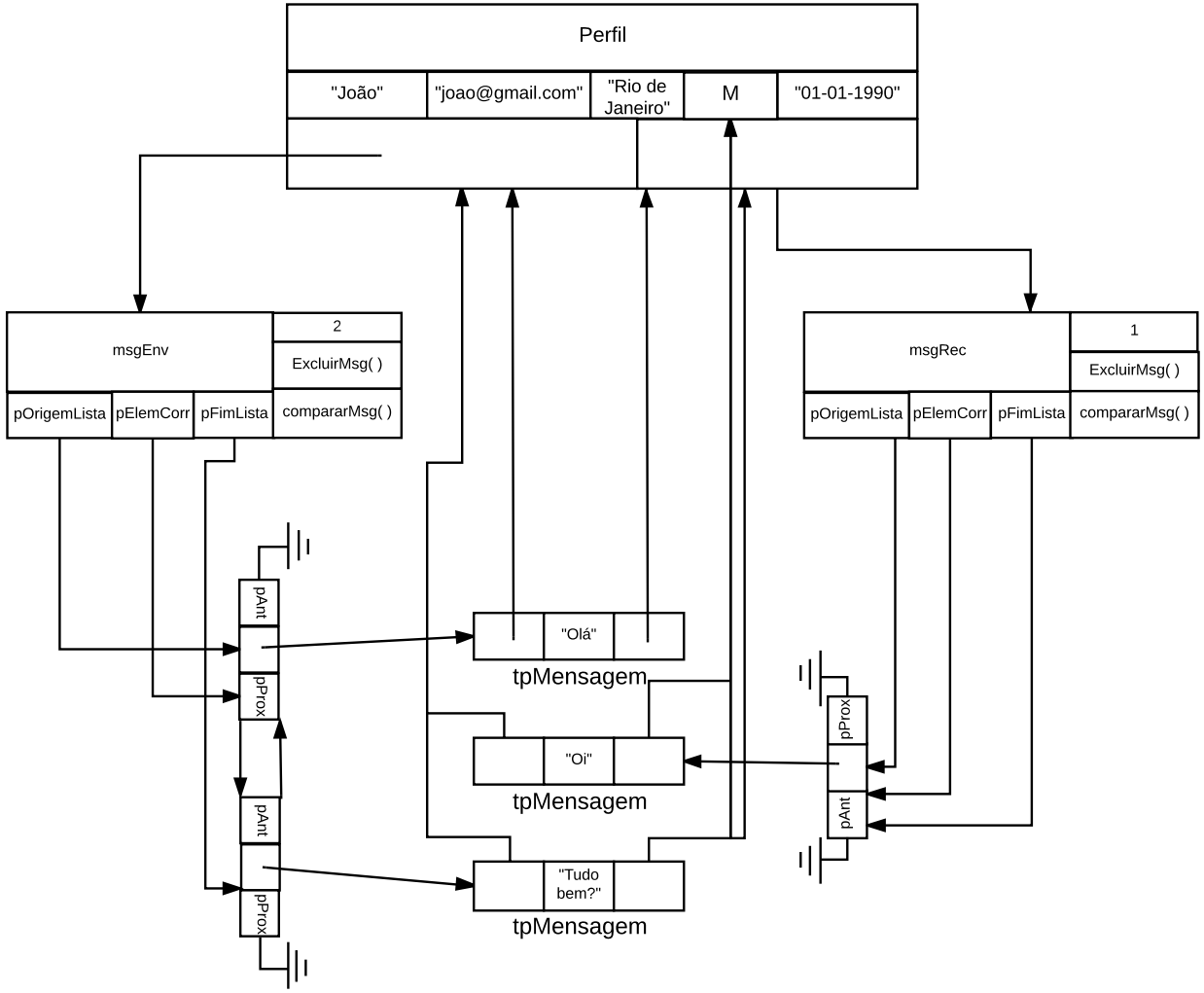
-O valor armazenado no vértice deve ser do mesmo tipo para todos os vértices

# PERFIL

## Modelo Físico




## Exemplo Físico




## Assertivas de PERFIL

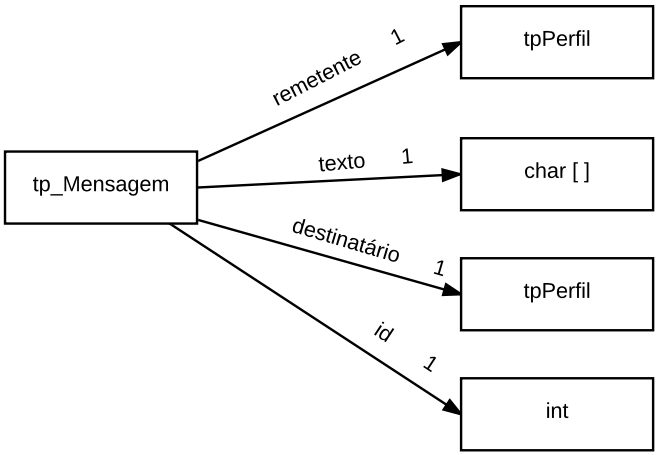
- Para todos os perfis cadastrados no sistema, não pode haver dois perfis com a mesma string no campo "email".

- Nenhum dos campos da estrutura Perfil pode ter valor nulo.

- Nenhuma das strings pode ser vazia.

# MENSAGEM

## Modelo Físico




## Assertivas de MENSAGEM

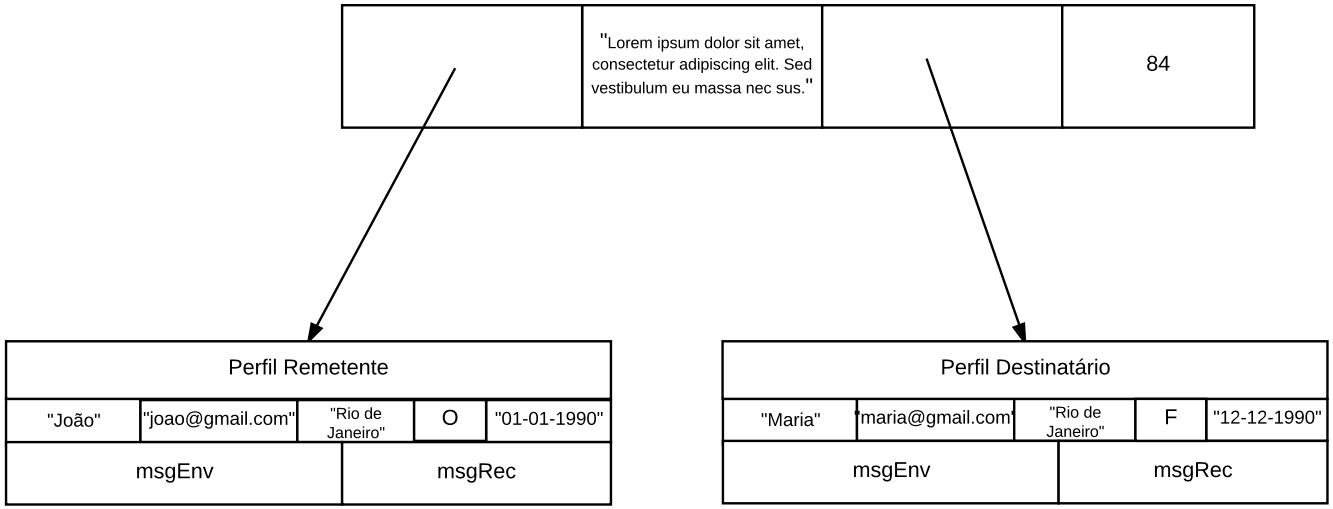
- Duas mensagens não podem ter o mesmo identificador (id)

- O id de uma mensagem de ser maior que 1.

-Os valores Remetente e Destinatário devem ser diferentes de NULL

- O campo texto de ser diferente de NULL e a string não pode ser vazia.

## Exemplo Físico



# Rede de relacionamentos - Especificação de Requisitos

PUC-Rio
INF1301 3WA
Data: 7 de Outubro de 2017
Alunos: Gabriel Brito, Gabrielle Brandenburg, Wellington Bezerra

## SOBRE O PROGRAMA

O programa a ser desenvolvido (em linguagem C) se trata de uma simulação de uma rede de relacionamentos, no qual apenas um usuário faz uso da rede, adicionando perfis fictícios e conectando perfis entre si para criar amizades. Quando dois perfis estão conectados, o usuário pode executar uma ação no qual ele digita uma mensagem e informa qual é o remetente e qual é o destinatário da mensagem (simulando o envio de uma mensagem de um perfil para o outro).

O usuário também poderá executar uma ação que irá mostrar o histórico de mensagens entre dois perfis ao informar o email dos dois. Para cada mensagem, será mostrado junto o nome do perfil que enviou a mensagem, similar ao histórico de um chat. Também será possível buscar e mostrar todas as informações sobre um perfil cadastrado, e para isso é necessário informar o e-mail do perfil para executar esta ação. Outra funcionalidade do programa será listar todas as amizades de um determinado perfil, e ao executar este comando, é necessário informar o e-mail do perfil em questão.

Os perfis criados podem ser editados ou excluídos, enquanto que as amizades criadas podem ser excluídas. Não será possível excluir uma mensagem já enviada, porém ela será excluída caso os dois perfis envolvidos na troca de mensagem sejam excluídos. O programa não fará persistência dos dados inseridos, ou seja, após finalizar a sua execução, todas as informações cadastradas serão perdidas.

As ações que podem ser executadas pelo usuários estarão disponíveis através de um menu (especificado neste documento), que irá guiá-lo durante o uso do programa.

## PARTE 1: REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

**LEGENDA**
**[RF]:** requisito funcional
**[RNF]:** requisito não funcional

### I. PERFIL

**1. [RF] Deve ser possível criar um perfil contendo as seguintes informações:**
- nome [caracteres]: obrigatório

- data de nascimento [caracteres]: obrigatório, no padrão DD/MM/AAAA
- sexo [caractere 'F', M' ou 'O']: obrigatório
- cidade [caracteres]: obrigatório
- e-mail [caracteres]: obrigatório

**1.1 [RF]** A informação "e-mail" deve ser única para cada perfil criado.
**1.2 [RF]** Se houver algum campo cadastrado de forma incorreta, o programa deve avisar ao usuário que ocorreu um erro ao cadastrar o perfil como resultado. O cadastro nesse caso não é concluído e o usuário deve solicitá-lo novamente.
**1.3. [RNF]** O limite para os campos de cadeia de caracteres é de 100 caracteres. Para data de nascimento são exatamente 10 caracteres.
**1.4 [RF]** Deve ser avisado que o perfil foi cadastrado com sucesso.
**1.5 [RNF]** O nome pode ser apenas o primeiro nome ou o nome completo.
**1.6 [RNF]** O programa não irá verificar se o e-mail existe ou se o formato é válido.

**2. [RNF] Deve ser possível excluir um perfil.**

**2.1 [RF]** A mesma informação solicitada em [1.1] será necessária para remover um perfil da rede.
**2.2 [RF]** Uma mensagem de sucesso ou insucesso na remoção de um perfil será mostrada na tela como resultado.

**3. [RF] Deve ser possível editar um perfil.**

**3.1 [RF]** A mesma informação solicitada em [1.1] será necessária para acessar o perfil que será modificado.
**3.2 [RF]** Deve aparecer na tela um menu de opções, no qual o usuário deve escolher qual informação do perfil será alterada. Ao escolher a informação, o programa irá esperar a entrada do novo valor para a informação escolhida.
**3.3 [RNF]** Não deve ser possível alterar o e-mail de cadastro do perfil.
**3.4 [RF]** Uma mensagem de sucesso ou insucesso na remoção de um perfil será mostrada na tela como resultado.

**4. [RF] Deve ser possível criar uma amizade entre dois perfis.**

**4.1 [RF]** As seguintes informações são necessárias para conectar os dois perfis: e-mail do perfil 1 e e-mail do perfil 2
**4.2 [RNF]** A amizade é criada automaticamente, ou seja, não é necessário nenhum tipo de aprovação.
**4.3 [RF]** Uma mensagem de sucesso ou insucesso da criação de amizade será mostrada na tela como resultado.

**5. [RF] Deve ser possível desfazer uma amizade.**

**5.1** As mesmas informações declaradas no item [4.2] serão necessárias para desfazer uma amizade.
**5.2** Uma mensagem de sucesso ou falha ao desfazer uma amizade será mostrada na tela como resultado.

**6. [RF] Deve ser possível buscar e visualizar as informações de um perfil.**

**6.1 [RF]** É necessário digitar apenas o número de e-mail do perfil para buscar suas informações.
**6.2 [RF]** Todas as informações cadastradas no perfil serão exibidas na tela.

**7. Deve ser possível buscar e visualizar a lista de amizades de um perfil.**

**7.1 [RF]** É necessário digitar apenas o e-mail do perfil para buscar sua lista de amigos.
**7.2 [RF]** Para cada amigo do perfil, deve ser exibido: nome, e-mail.

## II. MENSAGENS

**1. [RF] Deve ser possível um perfil enviar uma mensagem para outro perfil, desde que exista um relacionamento entre ambos.**

**1.1 [RF]** As seguintes informações são necessárias para enviar a mensagem: e-mail do perfil 1 (remetente), texto da mensagem [caracteres], e-mail do perfil 2 (destinatário).
**1.2 [RF]** Deve haver um identificador único [número inteiro] para cada mensagem enviada.
**1.3 [RF]** Caso os perfis não sejam amigos, uma mensagem de erro aparece como resultado na tela.

**2. [RF] Deve ser possível carregar o histórico de mensagens recebidas e enviadas entre dois perfis, ordenadas da mais recente para a mais antiga.**

**2.1 [RF]** As seguintes informações são necessárias para carregar o histórico de mensagens: e-mail do perfil 1 e e-mail do perfil 2
**2.2 [RF]** Cada mensagem visualizada na tela deve seguir o seguinte padrão: "[nome do perfil remetente]:", "[texto da mensagem]", linha em branco. No final da conversa será mostrado uma mensagem de fim de conversa.
**2.3 [RF]** Caso não haja mensagens entre os perfis, apenas a mensagem de fim de conversa será exibida.

**3. [RNF] Não deve ser possível excluir mensagens enviadas/recebidas.**

**3.1 [RF]** As mensagens serão excluídas automaticamente caso ambos perfis atrelados à mensagem sejam excluídos da rede.

## III. INTERFACE

**1. [RF] Deve ser mostrado um menu único com todas as opções de ação do programa, identificados por um número, da seguinte forma:**

(1) Criar novo perfil
(2) Criar uma nova amizade
(3) Enviar uma mensagem
(4) Carregar histórico de mensagens
(5) Buscar perfil
(6) Listar amizades de perfil
(7) Excluir Perfil
(8) Excluir Amizade
(9) Alterar Perfil
(10) Sair

**1.1 [RF]** O usuário deverá entrar com o número da opção acessada para acessá-la
**1.2 [RF]** Após selecionar uma opção, o usuário entra com os dados necessários (solicitados na tela) para a opção escolhida e após a resposta do programa, o menu inicial volta a aparecer na tela e aguarda a escolha de uma nova solicitação.
**1.3 [RF]** Após uma ação ser concluída, o resultado (especificado nos requisitos anteriores) daquela ação é mostrado na tela antes de voltar ao menu.

## IV. MODERADOR

**1. [RNF] A rede de relacionamentos possui um único usuário, que é o moderador que insere as informações no programa**

**1.1 [RNF]** O moderador não assume o "papel" de nenhum perfil, ou seja, o usuário que está interagindo com o programa não entra em um perfil específico para adicionar um amigo, enviar uma mensagem ou qualquer outra ação que o programa possa fazer.

## V. CORRETUDE

**1. [RNF]** Todos os módulos devem ser testados individualmente, utilizando o arcabouço de testes fornecido junto com uma boa massa de testes, onde cada função dos módulos é testada especificando-se as condições de entrada e saída, bem como as condições de estado do módulo.
**2. [RNF]** O cliente deve assegurar a validade sintática e semântica dos dados transmitidos ao servidor, assim como o servidor deve garantir o mesmo dos dados retornados ao cliente.

## VI. REUSO

**[RNF]** De forma a acelerar o processo de reutilização de projeto e implementação (e teste), deve-se maximizar a reutilização de módulos. Para isso, os módulos desenvolvidos de Lista, Grafo, Perfil e Mensagem são genéricos, podendo ser aplicados em outras soluções. O módulo lista pode utilizar qualquer tipo de valor para os seus elementos, assim como o grafo.

### VII. MANUTENIBILIDADE

**[RNF]** Todas funções e módulos deverão ser desenvolvidos utilizando padrões de documentação,, garantindo assim que o programa seja de fácil manutenção. Todas as funções de cada módulo deverão ser documentadas, explicitando entradas e possíveis saídas. A nomeação das funções e condições de retorno deverão seguir os padrões de sintaxe definidos.

### VIII. ROBUSTEZ

**[RNF]** O programa deve ser resistente à entradas incorretas do usuário (regras definidas anteriormente controlam essas entrada).

## PARTE 2: REQUISITOS TÉCNICOS

### GRAFO

Um grafo é formado por dois conjuntos. Um conjunto de elementos chamados vértices e um conjunto de elementos chamados arestas. Cada aresta está associada a um par de vértices.

**1. Cada perfil será representado por um vértice do grafo.**

**1.1** Cada vértice será armazenado em uma lista com somente um nó, a Lista Vértice
**1.2** Este nó guardará todas as informações referentes ao perfil.
**1.3** O nó também guardará uma referência para a Lista Arestas deste vértice, especificada no item 3.1 .

**2. Todos os vértices do grafo (armazenados como descrito em 1.1) serão referenciados por nós de uma nova lista, a Lista Vértices.**

**2.1** A Lista Vértices ancora todos os vértices existentes, representando todas as possíveis origens do grafo.

**3. Cada relacionamento entre perfis será representado por uma aresta entre vértices do grafo.**

**3.1** Estas arestas serão armazenadas em uma lista, a Lista Arestas.
**3.2** Nesta lista (3.1), cada nó possuirá uma referência para os outros vértices com quem este possui uma ligação (aresta).

**4. Todas as listas do grafo, incluindo Lista Vértice, Lista Vértices e Lista Arestas, possuem uma estrutura Cabeça de Lista.**

**4.1** Esta estrutura guarda uma referência para o início da lista além de uma referência para o nó corrente.

5. **Cada vértice (perfil) pode ter arestas conectadas a nenhum ou todos os vértices.**

**5.1** No caso de um vértice com nenhuma aresta, a sua Lista Arestas será uma lista vazia.

6. **Um vértice (perfil) não pode estar conectado (possuir uma aresta) com ele mesmo.**

7. **O Grafo possuirá uma estrutura principal, a Cabeça do Grafo.**

**7.1** Esta estrutura guardará a referência para a Lista Vértices, onde é possível acessar todos os vértices do grafo.
**7.2** Também guardará uma referência para o vértice corrente do grafo.

# Relatório de alterações na documentação feitas no T4

1. **Requisitos**
   **1.1** A documentação dos requisitos foi alterada para incluir as exigências funcionais do enunciado do T4 em relação à remoção de perfis, relacionamentos e alteração de dados do perfil.

2. **Modelagem da arquitetura**
   **2.1** A arquitetura foi modificada entre o relacionamento do módulo Perfil e Mensagem. O módulo Perfil passa a conhecer a interface do módulo mensagem, e o módulo mensagem não conhece a interface do perfil. Durante o desenvolvimento, percebemos que não havia necessidade do módulo mensagem acessar a interface do perfil, então para aumentar a independência do módulo mensagem, retiramos esse relacionamento. Por outro lado, para exclusão da estrutura do perfil, é necessário que este módulo tenha acesso à funções da interface da mensagem, logo foi necessário adicionar esse relacionamento.
   **2.2** As funções de interface também foram modificadas. As funções responsáveis por retornar o valor dos campos das estruturas encapsuladas foram adicionadas. Foi necessário incluir as funções DesativarRemetente/Destinatario em MENSAGEM.H e EnviarMensagem em PERFIL.H, além de incluir funções relativas aos novos requisitos funcionais mencionados no item 1.
   **2.3** Foi necessário adicionar uma nova funcao na interface do grafo (AvancarVizinho) para que fosse possível cumprir com o [RF 7] dos requisitos.
   **2.4** Foi necessário adicionar funções na interface do Controlador relacionados às novas exigências do T4 em relacao à remoção e modificação de perfis e exclusão de relacionamentos.
   **2.5** O requisito 2.3 foi alterado pois se dois amigos desfazem a amizade, não faz sentido apagar sua conversa, visto que isso não acontece em uma rede social normalmente.

3. **Modelagem física**
   **3.1** O modelo exemplo físico da estrutura Perfil foi alterado para incluir o campo "gênero" que estava faltando
   **3.2** Foram adicionados assertivas de entrada que faltavam no modelo Perfil e Mensagem.