

2ª PARTE TRABALHO DA DISCIPLINA

PROGRAMAÇÃO APLICADA DE COMPUTADORES –

ENG. ELÉTRICA

1 Considerações Gerais

1.1 Data de Entrega

O trabalho deve ser entregue até as 23:59 h do dia 10/12/2019.

1.2 Grupo

O trabalho deve ser realizado pelo mesmo grupo da primeira parte.

1.3 Ferramenta de Desenvolvimento

Para o desenvolvimento dos código recomenda-se utilizar a IDE Code::Blocks.

1.4 Código

O código deve ser encapsulado, organizado e modularizado. Deve-se desenvolver as classes, as funções do programa e a main em arquivos diferentes, de um mesmo projeto no CodeBlocks. Em outras palavras, deve ser implementado um arquivo [.h] e um [.cpp] para cada classe, e um arquivo [.cpp] para a main.

Atraso

Não serão aceitos trabalhos fora da data de entrega.

1.5 Forma de entrega

1.5.1 Envie o trabalho por e-mail para gabrielmartinsmiranda@gmail.com.

1.5.2 O assunto do e-mail deverá ser o que está descrito entre aspas a seguir: "PAC:trab2:nome1;nome2;nome3", onde nome1 é o primeiro nome e o último sobrenome do aluno1, nome2 é o primeiro nome e o último sobrenome do aluno2 e nome3 é primeiro nome e último nome do aluno3.

1.5.3 Compacte seus arquivos e envie o arquivo compactado. Não coloque os códigos ou arquivos texto no corpo do e-mail.

1.5.4 Um exemplo de envio de um e-mail:

Para: gabrielmartinsmiranda@gmail.com

De: Gabriel Miranda

Assunto: PAC:trab2:GabrielMiranda;VictorAmorim;CássioReginato

Anexo: PAC.zip

1.6 Atenção:

- 1.6.1 Trabalhos considerados iguais receberão nota zero. Será considerado plagio os trabalhos que tiverem mais de 50% de similaridade.
- 1.6.2 Caso um ou mais trabalhos sejam considerados iguais, o plagiador deve enviar um e-mail para o professor informando que copiou o trabalho. Desta forma, o aluno que teve o seu trabalho plagiado receberá a nota devida do trabalho. O plagiador ficará com a nota zero no trabalho.

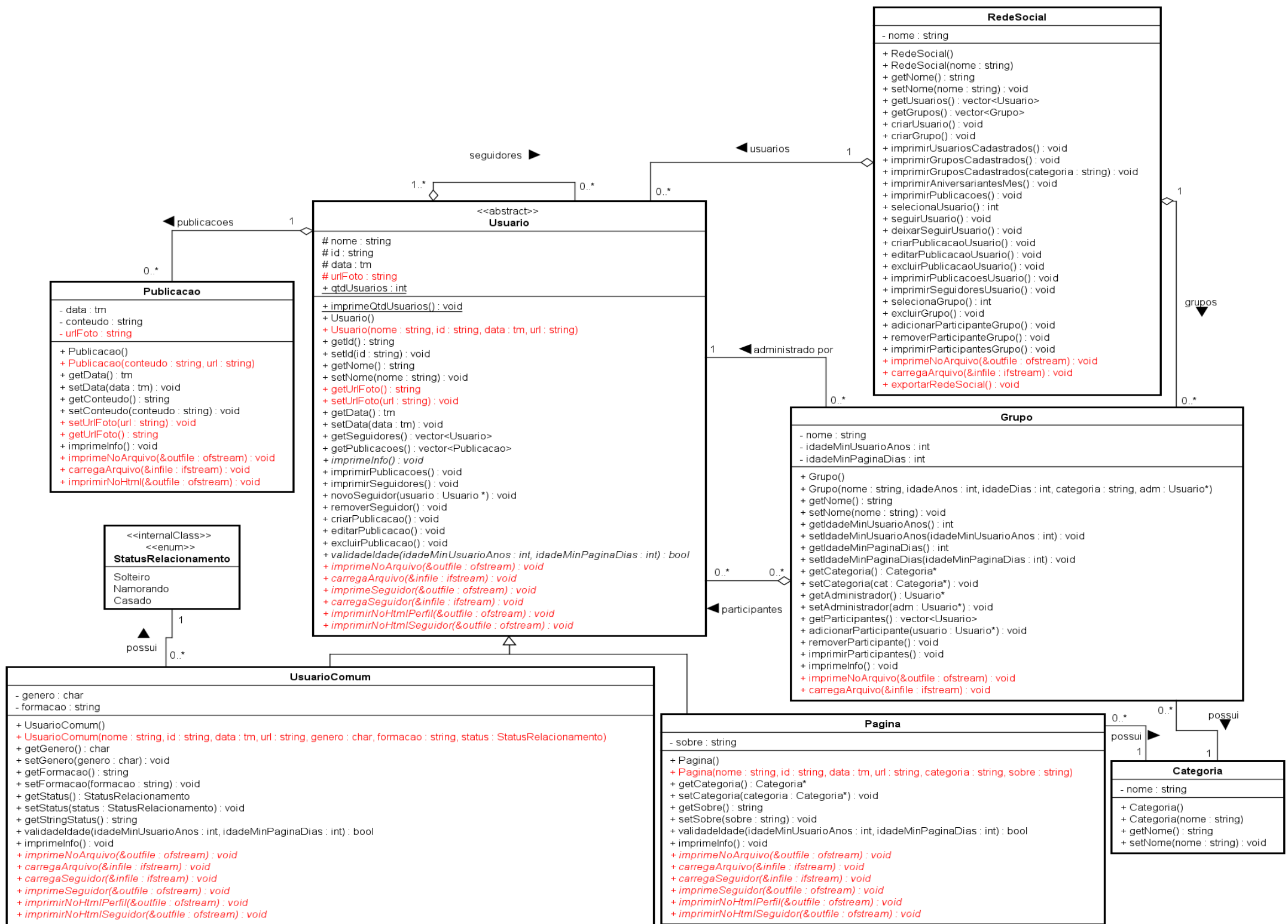
2 Especificação do trabalho

2.1 Objetivo do Trabalho

O trabalho tem como objetivo desenvolver um sistema que simule uma Rede, onde Usuários (Usuário Comum e Página) e Grupos serão gerenciados. Nesta segunda parte, será aproveitada a estrutura anterior, adicionando funcionalidades de arquivo. O principal objetivo do desenvolver é estruturar um código para salvar as informações do programa e o do mesmo modo que as informações são salvas elas também devem ser recuperadas.

2.2 Descrição do Trabalho

O grupo deve aproveitar a estrutura desenvolvida na 1ª parte, realizando algumas alterações e desenvolvendo novas funcionalidades:



Novo Menu:

O programa deve executar até que o usuário deseje finalizá-lo. As novas funcionalidades a serem desenvolvidas nessa segunda etapa estão representadas pela cor vermelha. Deve-se apresentar o seguinte menu para o usuário:

1. Criar novo Usuário (Rede Social)
2. Criar novo Grupo (Rede Social)
3. Excluir Grupo (Rede Social)
4. Imprimir todos Aniversariantes do Mês (Rede Social)
5. Imprimir todos Usuários Cadastrados (Rede Social)
6. Imprimir todos Grupos Cadastrados (Rede Social)
7. Imprimir todos Grupos Cadastrados por Categoria (Rede Social)
8. Imprimir todas Publicações Cadastradas (Rede Social)
9. Adicionar Participante (Grupo)
10. Remover Participante (Grupo)
11. Imprimir Participantes (Grupo)
12. Seguir Usuário (Usuário)
13. Deixar de Seguir Usuário (Usuário)
14. Criar Publicação (Usuário)
15. Editar Publicação (Usuário)
16. Excluir Publicação (Usuário)
17. Imprimir Publicações (Usuário)
18. Imprimir Seguidores (Usuário)
19. Imprimir Quantidade Usuário Cadastrados Geral (Sistema)
20. Salvar Dados (Sistema)
21. Carregar Dados (Sistema)
22. Exportar Biblioteca (Sistema)
23. Sair (Sistema)

Novas Funcionalidades do Menu:

17. Salvar Dados: na main deve-se chamar as funções salvarDadosRedesSociais, passando por parâmetro o vector de redes sociais criado na main.

18. CarregarDados: na main deve-se abrir os arquivos "RedesSociais.txt", valide se o arquivo existe e se foi aberto corretamente (utilize a função *is_open()*), caso não exista o arquivo informe o usuário que não existe arquivo à ser carregado. Caso o arquivo seja aberto com sucesso, apague todos os objetos das plataformas (utilize a função

clear) na main, que foram instanciados ao iniciar o programa. Chame a função carregarDadosRedesSociais passando por parâmetro o arquivo de redes sociais aberto e atribuindo o retorno da função ao vector de redes sociais (ex: *redesSociais = carregarDadosRedesSociais(arqRede)*).

19. Exportar Rede Social: na main, o usuário deve selecionar a rede social desejada e desse objeto selecionado chamar o método exportarRedeSocial.

ARQUIVO

Modificações na Main:

- Na main devem-se criar as seguintes funções:
 - *void salvarDadosRedesSociais(vector <RedeSocial*> redes);*
 - *vector <RedeSocial*> carregarDadosRedesSociais(ifstream &arqRede);*
- A função salvarDadosRedesSociais deve criar um arquivo de texto com o nome "RedesSociais.txt". Utilize um objeto do tipo **ofstream**. Após o arquivo ser criado, deve-se imprimir no arquivo a quantidade de redes sociais existentes no sistema (atualmente são 3, mas esse número deve ser parametrizado, ou seja, pegue o tamanho [size] da lista de redes sociais). Deve-se então percorrer a lista de redes sociais, chamando o método ImprimeNoArquivo de cada RedeSocial passando o arquivo criado como parâmetro. Ao final da função, feche o arquivo e imprima uma mensagem de sucesso (ex: "Arquivo com as Redes Sociais foi criado com sucesso!")
- A função carregarDadosRedesSociais deve receber como parâmetro um arquivo de entrada (do tipo **ifstream**) que foi aberto na main, e criar um vector de redes sociais. Lembre-se de zerar a quantidade de usuários armazenadas no sistema utilizando o atributo estático qtdUsuarios. Após isso, leia a quantidade de redes sociais salva no arquivo. Para a quantidade de redes sociais, crie um novo objeto RedeSocial e chame o método carregaArquivo (da classe RedeSocial) do objeto criado, passando o arquivo de entrada como parâmetro. Adicione cada rede criada no vector de redes sociais. Ao final da função, feche o arquivo, imprima uma mensagem de sucesso (ex: "Todas as Redes Sociais foram carregadas com sucesso!"), e retorne o vector de redes sociais para a main.

Modificações na classe RedeSocial:


- A função ImprimeNoArquivo da RedeSocial deve receber como parâmetro o arquivo de saída ("RedesSociais.txt") que foi enviado pela função

salvarDadosRedesSociais (da main). Deve-se salvar no arquivo o nome da rede, a quantidade de usuarios cadastrados, as informações de cada usuario (utilize o método ImprimeNoArquivo de Usuario), a quantidade de grupos cadastrados, as informações de cada grupo (utilize o método ImprimeNoArquivo de Grupo). Ao final exiba uma mensagem de sucesso (ex: "Rede Social X salva com sucesso!").

- A função CarregaArquivo deve receber como parâmetro o arquivo de entrada ("RedesSociais.txt") que foi enviado pela função carregarDadosRedesSociais (da main). Deve-se carregar do arquivo o nome da rede social e a quantidade de usuarios cadastrados (a ser incrementada no atributo estatico qtdUsuarios). Para a quantidade de usuarios cadastrados, leia o tipo do usuario (UsuarioComum ou Pagina), crie um objeto do tipo lido, chame o método carregaArquivo desse objeto e adicione esse objeto criado no vector de usuarios. Após isso, carregue a quantidade de grupos cadastrados, e para essa quantidade, instancie objetos do tipo grupo, chame o método carregaArquivo desse objeto e adicione esse objeto criado no vector de grupos. Ao final exiba uma mensagem de sucesso (ex: "Rede Social X carregada com sucesso!").
- Altere a função criarUsuario solicitando ao usuário a url da imagem do usuario.

Modificações nas classes Usuario, UsuarioComum e Pagina:

- Inclua os atributos urlFoto na classe Usuario, esse atributo armazenará links de uma imagem de perfil da web sobre o usuário. E inclua os métodos get e set para o atributo urlFoto.
- Altere os construtores parametrizados de Usuario, UsuarioComum e Pagina para incluir o atributo urlFoto.
- Defina as funções imprimeNoArquivo, imprimeSeguidor, carregaArquivo e carregaSeguidor como virtual pura na classe Usuario e sobrescreva nas classes UsuarioComum e Pagina (assim como foi feito com o método imprimeInfo).
- A função imprimeNoArquivo, deve imprimir no arquivo de saída, recebido de parâmetro, o tipo do objeto (se é UsuarioComum ou Pagina) e na sequencia TODOS os atributos da classe separadas por linha. Ex:

<pre>void Jogo::imprimeNoArquivo(ofstream &outfile) { outfile << this->nome << endl; outfile << this->valor << endl; outfile << this->urlImagem << endl; outfile << this->anoLancamento << endl; }</pre>		<pre>God of War 125 https://site.com/godofwar.jpg 2018</pre>
--	---	--

- Do usuário comum deve-se salvar no arquivo o tipo do objeto (ex: "Comum"), o id do usuário, nome, urlFoto, dia de nascimento, mês de nascimento, ano de nascimento, genero, formação, status, a quantidade de seguidores, as

informações de cada seguidor (utilize o método ImprimeSeguidor de Usuario), a quantidade de publicacoes, as informações de cada publicação (utilize o método ImprimeNoArquivo de Publicacao).

- Do página deve-se salvar no arquivo o tipo do objeto (ex: "Página"), o id da página, nome, urlFoto, dia de nascimento, mês de nascimento, ano de nascimento, categoria, sobre, a quantidade de seguidores, as informações de cada seguidor (utilize o método ImprimeSeguidor de Usuario), a quantidade de publicacoes, as informações de cada publicação (utilize o método ImprimeNoArquivo de Publicacao).
- A função imprimeSeguidor, é similar a imprimeNoArquivo, uma vez que o seguidor também é um usuário, a diferença é que para o seguidor não é necessário imprimir nem a lista de seguidores e nem a lista de publicações.
- A função carregaArquivo, deve ler do arquivo de entrada, recebido de parâmetro a sequência de dados, alterando cada um no respectivo atributo do objeto. Ex:

```
void Jogo::carregaArquivo(ifstream &infile) {  
    getline(infile, this->nome);  
    infile >> this->valor;  
    infile.ignore();  
    getline(infile, this->urlImagem);  
    infile >> this->anoLancamento;  
    infile.ignore();  
}
```

- Do usuário comum deve-se carregar no arquivo o id do usuário, nome, urlFoto, dia de nascimento, mês de nascimento, ano de nascimento, genero, formação, status, a quantidade de seguidores, as informações de cada seguidor (utilize o método CarregaSeguidor de Usuario), a quantidade de publicacoes, as informações de cada publicação (utilize o método CarregaArquivo de Publicacao).
- Do página deve-se carregar no arquivo o id da página, nome, urlFoto, dia de nascimento, mês de nascimento, ano de nascimento, categoria, sobre, a quantidade de seguidores, as informações de cada seguidor (utilize o método CarregaSeguidor de Usuario), a quantidade de publicacoes, as informações de cada publicação (utilize o método CarregaArquivo de Publicacao).
- A função carregaSeguidor, é similar a carregaArquivo, uma vez que o seguidor também é um usuário, a diferença é que para o seguidor não é necessário carregar nem a lista de seguidores e nem a lista de publicações.

Modificações na classe Grupo:

- A função ImprimeNoArquivo de Grupo deve receber como parâmetro o arquivo de saída ("RedesSociais.txt") que foi enviado pela função ImprimeNoArquivo da Rede Social. Deve-se salvar no arquivo o nome do grupo, idade mínima usuários comuns, idade mínima página dias, categoria, o administrador (utilize o método ImprimeSeguidor de Usuario), a quantidade de participante e as informações de cada participante (utilize o método ImprimeSeguidor de Usuario). Não é necessário exibir nenhuma mensagem ao final.
- A função CarregaArquivo de Grupo deve receber como parâmetro o arquivo de entrada ("RedesSociais.txt") que foi enviado pela função CarregaArquivo da Rede Social. Deve-se carregar do arquivo o nome do grupo, idade mínima usuários comuns, idade mínima página dias, categoria, o administrador (utilize o método CarregaSeguidor de Usuario), a quantidade de participante e as informações de cada participante, leia o tipo do usuario (UsuarioComum ou Pagina), crie um objeto do tipo lido, chame o método CarregaSeguidor desse objeto e adicione esse objeto criado no vector de participantes. Não é necessário exibir nenhuma mensagem ao final.

Modificações na classe Publicação:

- A função ImprimeNoArquivo de Publicação deve receber como parâmetro o arquivo de saída ("RedesSociais.txt") que foi enviado pela função ImprimeNoArquivo de Usuário. Deve-se salvar no arquivo o conteúdo da publicação, a url da foto, dia da publicação, mês da publicação e o ano da publicação. Não é necessário exibir nenhuma mensagem ao final.
- A função CarregaArquivo de Publicação deve receber como parâmetro o arquivo de entrada ("RedesSociais.txt") que foi enviado pela função CarregaArquivo de Usuário. Deve-se carregar do arquivo o conteúdo da publicação, a url da foto, dia da publicação, mês da publicação e o ano da publicação. Não é necessário exibir nenhuma mensagem ao final.

HTML


Modificações na classe RedeSocial:

- A função ExportarRedeSocial deve criar um arquivo de saída com o nome da rede social e a extensão .html (ex: Facebook.html). Após o arquivo ser aberto imprima no arquivo as tags html padrões (<html>, <head>, <meta>); a tag de <style> com as classes do css que serão utilizadas para alterar o html (css completo disponível no código da Aula 9), e a tag <body> onde o código a ser exibido na página deve ser escrito.

Em relação ao conteúdo a ser exibido na página, para cada usuário deve-se imprimir seu perfil, suas publicações e os seguidores. Exemplo:

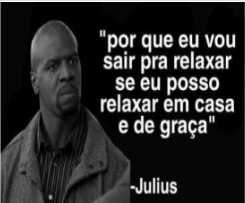
Ei Nerd (Pagina)

Perfil



einerd.com.br
9/8/2019
Entreterimento
Noticias de cinema, games, quadrinhos, animes e tudo do mundo nerd.


Publicacoes




9/8/2019
Baita Empreendedor

Eu com 8 anos depois de passar pela sala e ouvir na TV o cara do Linha Direta: "O assassino pode estar aí... na sua cidade."

9/8/2019
Bons tempos



Seguidores



gmmiranda
13/6/1992
Masculino
Casado
Bacharel em Sistemas de Informacao

Ao final do método imprima (utilizando o cout) uma mensagem de sucesso informando que o arquivo html foi criado com sucesso.

OBS: exemplos de arquivos htmls mostrando uma página de saída esperada, além de um código sobre a criação de um arquivo html pelo código C++ estão disponíveis no site.

Modificações na classe Usuario, UsuarioComum e Pagina:

- Defina as funções imprimirNoHtmlPerfil e imprimirNoHtmlSeguidor como virtual pura na classe Usuario e sobrescreva nas classes UsuarioComum e Pagina.
- A função imprimirNoHtmlPerfil, deve imprimir no arquivo html de saída, recebido de parâmetro, a foto de perfil, o id, a data de nascimento/criação e as informações específicas de cada classe, gênero, status de relacionamento e

formação para Usuários comuns; e categoria e sobre para Páginas. Exemplo de código html:

```
<div id="perfil">
  <div class="row">
    <div class="col-12">
      <h3>Perfil</h3>
    </div>
  </div>
  <div class="bg-img">
    
  </div>
  <div id="sobre">
    <h4>einerd.com.br</h4>
    <h4>9/8/2019</h4>
    <h4>Entreterimento</h4>
    <h4>Noticias de cinema, games, quadrinhos, animes e tudo do mundo nerd.</h4>
  </div>
</div>
```

- A função imprimirNoHtmlSeguidor é similar a função anterior, alterando o formato de exibição e imprimindo todos os seguidores na lista de seguidores:

Perfil	Seguidores
 <p>desimpedidos 9/8/2019 Futebol</p> <p>Este eh o canal Desimpedidos. Onde a zueira nao tem limites, onde seu time eh pequeno, onde o Jobson eh o melhor do mundo e onde o CR7 eh so mais um.</p>	 <p>gmmiranda 13/6/1992 Masculino Casado Bacharel em Sistemas de Informacao</p>
 <p>einerd.com.br 9/8/2019 Entreterimento Noticias de cinema, games, quadrinhos, animes e tudo do mundo nerd.</p>	

- Exemplo de código html:

```

<div id="seguidores">
  <div class="row">
    <div class="col-12">
      <h3>Seguidores</h3>
    </div>
  </div>

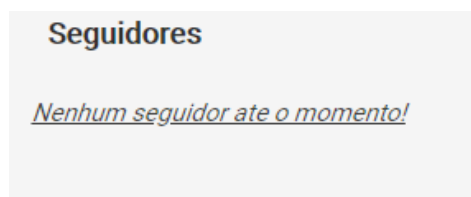
  <div id="card">
    <div class="row">
      <div id="divImage">
        <div class="bg-img">
          
        </div>
      </div>
      <div id="info">
        <h4>gmmiranda</h4>
        <h4>13/6/1992</h4>
        <h4>Masculino</h4>
        <h4>Casado</h4>
        <h4> Bacharel em Sistemas de Informacao</h4>
      </div>
    </div>
  </div>

  <div id="card">
    <div class="row">
      <div id="divImage">
        <div class="bg-img">
          
        </div>
      </div>
      <div id="info">
        <h4>einerd.com.br</h4>
        <h4>9/8/2019</h4>
        <h4>Entreterimento</h4>
        <h4>Noticias de cinema, games, quadrinhos, animes e tudo do mundo nerd.</h4>
      </div>
    </div>
  </div>
</div>

```

- Caso não existam seguidores, imprima um texto informando essa condição.

Exemplo:



Modificações na classe Publicação:

- A função imprimeNoHtml, deve imprimir no arquivo html de saída, recebido de parâmetro, a imagem da publicação, a data da publicação e o conteúdo da publicação. Exemplo:

Publicacoes	
	<div>9/8/2019</div> <div>Baita Empreendedor</div>
<p>Eu com 8 anos depois de passar pela sala e ouvir na TV o cara do Linha Direta: "O assassino pode estar aí... na sua cidade."</p> 	<div>9/8/2019</div> <div>Bons tempos</div>

- Exemplo de código html:

```
<div id="publicacoes">
  <div class="row">
    <div class="col-12">
      <h3>Publicacoes</h3>
    </div>
  </div>
  <div id="card">
    <div class="row">
      <div id="divImage">
        <div class="bg-img">
          
        </div>
      </div>
      <div id="info">
        <div>
          <h3>9/8/2019</h3>
        </div>
        <h4>Baita Empreendedor</h4>
      </div>
    </div>
  </div>
  <div id="card">
    <div class="row">
      <div id="divImage">
        <div class="bg-img">
          
        </div>
      </div>
      <div id="info">
        <div>
          <h3>9/8/2019</h3>
        </div>
        <h4>Bons tempos</h4>
      </div>
    </div>
  </div>
</div>
```

- Caso não existam publicações, imprima um texto informando essa condição.

Exemplo:

Publicacoes

Nenhuma publicacao ate o momento!