Planejamento

(Processo de Desenvolvimento)

Alunos: Lucas Gomes Almeida – 12/0152860

Rafael da Silva Rocha – 12/0133229

**PADRÃO DE CODIFICAÇÃO:**

O padrão de codificação adotado pela equipe encontra-se no site: <http://www.facp.pro.br/cursos/docs/ps/padrao_ps.pdf>. O arquivo PDF que é alcançado através do link disponibilizado contém todas as informações a respeito de aspectos como declarações de variáveis, constantes, macros e funções; uso de espaços em branco, formatos de expressões e enunciados, diretivas de compilação, entre outros pontos importantes à atividade de programação. O padrão foi disponibilizado pelo professor Francisco A. C. Pinheiro no site do departamento de Ciência da Computação da Universidade de Brasília.

**FORMULÁRIO PARA A COLETA DE DADOS E PLANEJAMENTO:**

*- Identificador do MÓDULO 1:* Persistência dos Desenvolvedores;

*- Tempo estimado para desenvolvimento do módulo:* 2 semanas (14 dias)

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo:* 500 linhas de código

*- Tamanho total efetivo do módulo:*

*- Identificador do MÓDULO 2:* Persistência de Produtos;

*- Tempo estimado para desenvolvimento do módulo:* 2 semanas (14 dias)

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo:* 500 linhas de código

*- Tamanho total efetivo do módulo:*

*- Identificador do MÓDULO 3:* Persistência de Defeitos;

*- Tempo estimado para desenvolvimento do módulo:* 2 semanas (14 dias)

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo:* 500 linhas de código

*- Tamanho total efetivo do módulo:*

*- Identificador do MÓDULO 4:* Lógica do Negócio;

*- Tempo estimado para desenvolvimento do módulo: 10 dias*

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo: 400 linhas de código*

*- Tamanho total efetivo do módulo:*

*- Identificador do MÓDULO 5:* Interface com os Usuários;

*- Tempo estimado para desenvolvimento do módulo: 2 semanas(14 dias)*

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo: 800 linhas de código*

*- Tamanho total efetivo do módulo:*

- Plano de Desenvolvimento:



**DOCUMENTAÇÃO DA ARQUITETURA:**

Persistência de Defeitos

Interface com os Usuários

Lógica de Negócio

Persistência de Produtos

**Interface Lógica-Négocio**

**Interface P-Defeitos**

**Interface P-Produtos**

**Interface P-Desenvolvedores**

Persistência de Desenvolvedores

→ **Descrição dos Módulos:**

**MÓDULO 1: Persistência dos Desenvolvedores**

O módulo tem como objetivo realizar ações de cadastramento, pesquisa, alteração e remoção no que diz respeito aos dados relativos aos desenvolvedores. Isto se dará por meio de acesso à memória secundária e consequente manipulação dos arquivos gerados.

**MÓDULO 2: Persistência de Produtos**

O módulo tem como objetivo realizar ações de cadastramento, pesquisa, alteração e remoção no que diz respeito aos dados relativos aos produtos de software. Isto se dará por meio de acesso à memória secundária e consequente manipulação dos arquivos gerados.

**MÓDULO 3: Persistência de Defeitos**

O módulo tem como objetivo realizar ações de cadastramento, pesquisa, alteração e remoção no que diz respeito aos dados relativos aos defeitos acerca dos produtos de software. Isto se dará por meio de acesso à memória secundária e consequente manipulação dos arquivos gerados.

**MÓDULO 4: Interface do negócio**

O módulo tem como objetivo impor as regras de negócio definidas e garantir a integridade das chaves e a integridade referencial. Possuirá interface com os três módulos de persistência e também com o módulo de interface com o usuário.

**MÓDULO 5: Interface com os usuários.**

O módulo é responsável pela interface com o usuário, e, através de funções contidas nele, pela validação dos dados de entrada digitados pelo usuário. Em casos de erros o módulo tem como mais uma de suas funções a notificação do usuário.

→ **Descrição das Interfaces:**

**INTERFACE 1-4: P-Desenvolvedores**

Interface responsável por conectar os módulos 1 e 4 (Persistência dos Desenvolvedores e Lógica de Negócio, respectivamente). Essa conexão se dará por meio da disponibilização de funções pelo módulo 1.

- Os protótipos:

int cadastrar\_desenvolvedor(Desenvolvedor\*)

- Função utilizada para o cadastramento de um novo desenvolvedor;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao desenvolvedor a ser cadastrado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int remover\_desenvolvedor(char\*)

- Função utilizada para a remoção dos dados de um desenvolvedor;

- Recebe como parâmetro o e-mail do desenvolvedor a ser removido (o e-mail é um dado único);

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int editar\_desenvolvedor(Desenvolvedor\*)

- Função utilizada para a edição dos dados de um desenvolvedor;

- Recebe como parâmetro uma estrutura de dados contendo as novas informações referentes ao desenvolvedor a ser editado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int pesquisar\_desenvolvedor(Desenvolvedor\*)

- Função utilizada para a pesquisa de um desenvolvedor já cadastrado;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao desenvolvedor a ser pesquisado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

**INTERFACE 2-4: P-Produtos**

Interface responsável por conectar os módulos 2 e 4 (Persistência de Produtos e Lógica de Negócio, respectivamente). Essa conexão se dará por meio da disponibilização de funções pelo módulo 2.

- Os protótipos:

int cadastrar\_produto(Produto\*)

- Função utilizada para o cadastramento de um novo produto;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao produto a ser cadastrado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int remover\_produto(int)

- Função utilizada para a remoção dos dados de um produto;

- Recebe como parâmetro o código do produto a ser removido (o código é um dado único);

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int editar\_produto(Produto\*)

- Função utilizada para a edição dos dados de um produto;

- Recebe como parâmetro uma estrutura de dados contendo as novas informações referentes ao produto a ser editado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int pesquisar\_produto(Produto\*)

- Função utilizada para a pesquisa de um produto já cadastrado;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao produto a ser pesquisado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

**INTERFACE 3-4: P-Defeitos**

int cadastrar\_defeito(Defeito\*)

- Função utilizada para o cadastramento de um novo defeito;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao defeito a ser cadastrado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int remover\_defeito(int)

- Função utilizada para a remoção dos dados de um defeito;

- Recebe como parâmetro o código do defeito a ser removido (o código é um dado único);

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int editar\_defeito(Defeito\*)

- Função utilizada para a edição dos dados de um defeito;

- Recebe como parâmetro uma estrutura de dados contendo as novas informações referentes ao defeito a ser editado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

int pesquisar\_defeito(Defeito\*)

- Função utilizada para a pesquisa de um defeito já cadastrado;

- Recebe como parâmetro uma estrutura de dados contendo as informações referentes ao defeito a ser pesquisado;

- Retorna um número inteiro que indica o estado da operação (Os números e seus respectivos significados serão descritos em um arquivo de cabeçalho próprio).

**INTERFACE 5-5: INTERFACE COM O USUÁRIO**

void menu\_principal(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra para o usuário o menu principal;

- Função retorna nada;

void fazer\_login(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra para o usuário a página de realizar login;

- Função coleta dados para autenticar o login;

- Função retorna nada;

void menu\_principal\_login(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra o menu principal para um usuário autenticado;

- Função retorna nada;

void menu\_projeto(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra o menu de projeto para um usuário autenticado;

- Função retorna nada;

void menu\_produto(Desenvolvedor\* , Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Função mostra o menu de produto para um usuário autenticado;

- Função retorna nada;

void menu\_defeito(Desenvolvedor\* , Produto\* ,Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Função mostra o menu de produto para um usuário autenticado;

- Função retorna nada;

void cadastrar\_desenvolvedor\_usuario(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra a pagina de cadastro de um novo desenvolvedor;

- Função coleta os dados de cadastro;

- Função retorna nada;

void edita\_desenvolvedor\_usuario(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra a página de editar informações de um desenvolvedor;

- Função coleta os dados de edição do cadastro;

- Função retorna nada;

void cadastrar\_produto\_usuario(Desenvolvedor\* , Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto, respectivamente;

- Função mostra a página de cadastro de um novo produto;

- Função coleta os dados de cadastro;

- Função retorna nada;

void edita\_produto\_usuario(Desenvolvedor\* , Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto, respectivamente;

- Função mostra a página para edição de um produto;

- Função coleta os dados de edição;

- Função retorna nada;

void encerra\_conta\_desenvolvedor(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função mostra a opção correspondente ao encerramento da conta do desenvolvedor que está logado;

- Função retorna nada;

void reportar\_defeito(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito, respectivamente;

- Função mostra a pagina para reportar um defeito;

- Função coleta os dados de cadastro do defeito;

- Função retorna nada;

void descontinuar\_produto(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito, respectivamente;

- Função mostra a página para descontinuar um produto;

- Função retorna nada;

void votos\_defeito\_valida\_entrada(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito, respectivamente;

- Função coleta os dados de voto;

- Função retorna nada;

void candidatar\_para\_solucao\_defeito(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito, respectivamente;

- Função mostra a página para se candidatar à solução de um defeito;

- Função coleta os dados dos candidatos à solução do defeito;

- Função retorna nada;

void modificar\_estado\_defeito(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito, respectivamente;

- Função mostra a página para mudar estado de um defeito;

- Função coleta os dados de mudança de estado do defeito;

- Função retorna nada;

int valida\_entrada\_nome\_padrao(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_email\_padrao(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_senha\_padrao(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_nome\_produto\_padrao(Produto\*)

- Função recebe uma estrutura de dados do tipo Produto;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_versao\_padrao\_produto(Produto\*)

- Função recebe uma estrutura de dados do tipo Produto;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_codigo\_padrao\_produto(Produto\*)

- Função recebe uma estrutura de dados do tipo Produto;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_codigo\_padrao\_defeito(Defeito\*)

- Função recebe uma estrutura de dados do tipo Defeito;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_descricao\_defeito(Defeito\*)

- Função recebe uma estrutura de dados do tipo Defeito;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

int valida\_entrada\_data\_defeito\_padrao(Defeito\*)

- Função recebe uma estrutura de dados do tipo Defeito;

- Função valida a entradas do usuário;

- Função retorna se as entradas são válidas ou não;

**INTERFACE 4-5: Negócio**

Interface responsável por conectar o módulo 4 ao 5 (Negocio e Interface com o Usuário, respectivamente). Essa conexão se dará por meio da disponibilização de funções do módulo 4 (Negócio).

int cadastrar\_usuario\_negocio(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;  
 - Verifica se é possível cadastrar o desenvolvedor;

- Função retorna SUCESSO ou ERRO;

int autenticar\_usuario\_negocio(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;  
 - Verifica se é possível autenticar o desenvolvedor;

- Função retorna SUCESSO ou ERRO;

int editar\_usuario\_negocio(Desenvolvedor\*)

- Função recebe uma estrutura de dados do tipo Desenvolvedor;  
 - Verifica se é possível editar o desenvolvedor;

- Função retorna SUCESSO ou ERRO;

int cadastra\_produto\_negocio(Desenvolvedor\*, Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto;

- Verifica se é possível cadastrar o produto;

- Função retorna SUCESSO ou ERRO;

int alterar\_produto\_negocio(Desnevolvedor\*, Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto;

- Verifica se é possível alterar o produto;

- Função retorna SUCESSO ou ERRO;

int reportar\_defeito\_negocio(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Verifica se é possível reportar o defeito;

- Função retorna SUCESSO ou ERRO;

int associa\_lider\_produto\_negocio (Desenvolvedor\* , Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto;

- Verifica se é possível associar um líder ao produto;

- Função retorna SUCESSO ou ERRO;

int voto\_defeito\_negocio(Desenvolvedor\* , Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor e Produto e Defeito;

- Verifica se é possível votar o defeito;

- Função retorna SUCESSO ou ERRO;

int candidatar\_lider\_defeito\_negocio (Desenvolvedor\* , Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Verifica se é possível se candidatar para o defeito;

- Função retorna SUCESSO ou ERRO;

int escolhe\_lider\_defeito\_negocio (Desenvolvedor\* , Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Verifica se é possível escolher o defeito;

- Função retorna SUCESSO ou ERRO;

int mudanca\_estado\_defeito\_negocio(Desenvolvedor\*, Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

- Verifica se é possível mudar o estado do defeito;

- Função retorna SUCESSO ou ERRO;

int descontinuar\_produto\_negocio(Desnevolvedor\*, Produto\*)

- Função recebe duas estruturas de dados do tipo Desenvolvedor e Produto;

- Verifica se é possível descontinuar o produto;

- Função retorna SUCESSO ou ERRO;

int encerrar\_desenvolvedor\_negocio(Desenvolvedor\* , Produto\*, Defeito\*)

- Função recebe três estruturas de dados do tipo Desenvolvedor, Produto e Defeito;

-Verifica se é possível encerrar a conta do desenvolvedor;

-Função retorna SUCESSO ou ERRO;

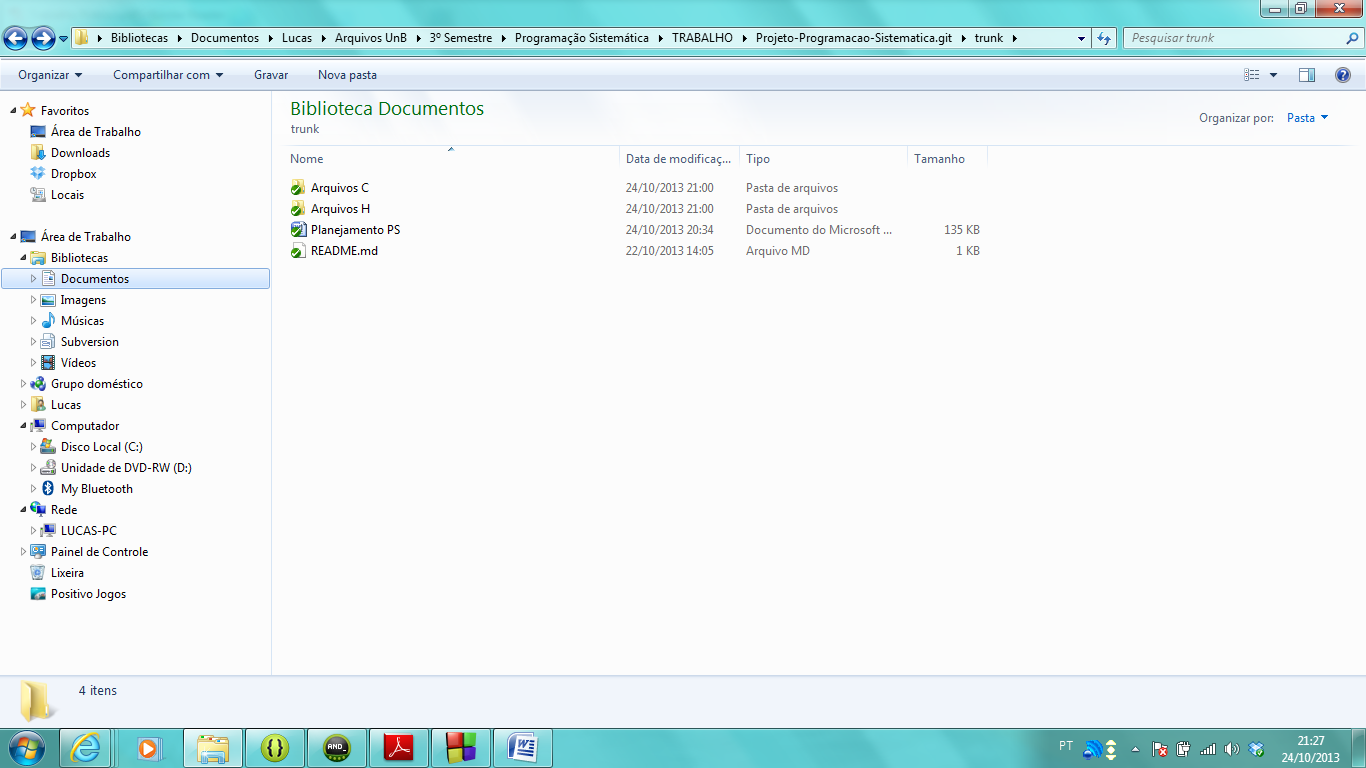
→ **Organização dos Arquivos:**

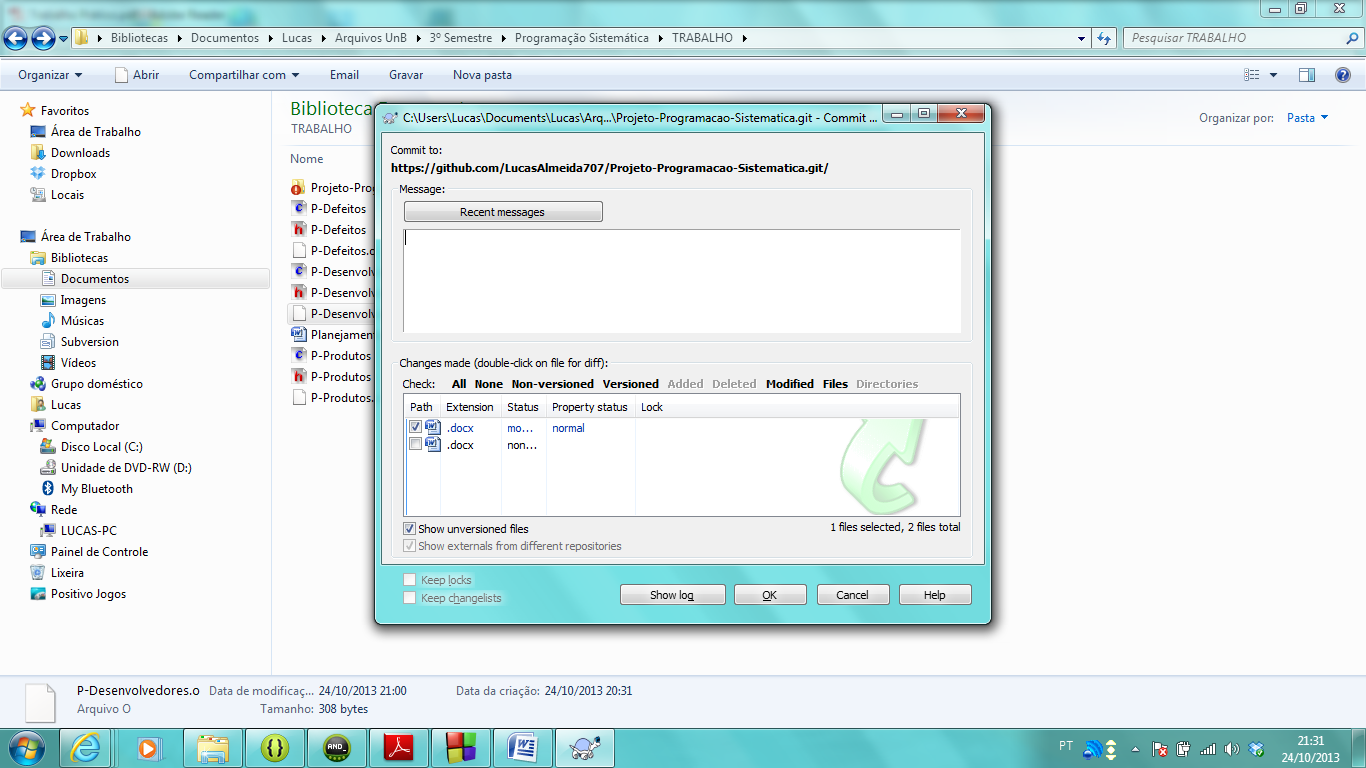
→ **Uso de Espaço em Disco:**

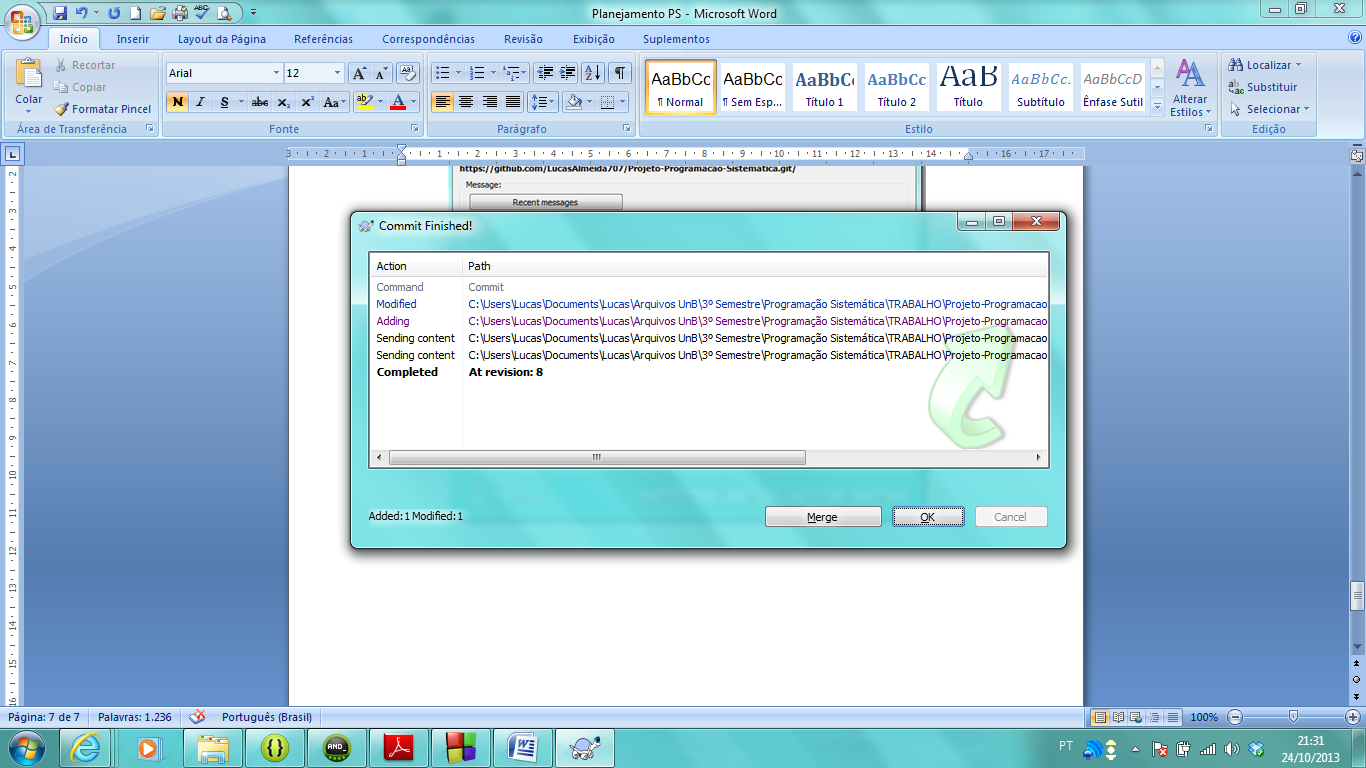
O banco de dados de informações relativas a Desenvolvedores, Produtos e Defeitos se basearão em arquivos .txt. As informações serão armazenadas de forma sequencial de forma que, com o auxilio de funções de percorrimento de arquivos, seja possível alcançar dados previamente gravados. O início de cada estrutura contará com um campo que indicará se o espaço está ou não disponível para armazenamento. Se disponível (o bloco de dados já foi descartado) será, então, possível sobreescrever o antigo conteúdo com novas informações; se ainda ocupado, será verificado o próximo bloco de dados até que seja encontrado um espaço livre. Se, por fim, não houverem espaços disponíveis no arquivo, a nova estrutura cedida pelo usuário será armazenada no final do arquivo, expandindo-o.

**- CONTROLE DE VERSÕES:**

Para o controle de versões foi utilizado o software Tortoise (uma das ferramentas do controlador de versões Subversion) em conjunto com o repositório online Github. Desta forma, foi possível manter o material relativo ao trabalho constantemente atualizado e acessível aos integrantes do grupo. O uso do Subversion propriamente dito não foi feito, pois, para tanto, seria necessária a criação de um repositório em um computador que fosse acessível ao grupo sempre que necessário; como isso não seria possível, optou-se pelo uso da ferramenta online, a qual pode ser acessada pelo link: <https://github.com/>.

****

****

****

