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:*

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo:*

*- Tamanho total efetivo do módulo:*

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

*- Tempo estimado para desenvolvimento do módulo:*

*- Tempo efetivamente gasto no desenvolvimento do módulo:*

*- Tamanho total estimado para o módulo:*

*- 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.

...

→ **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).

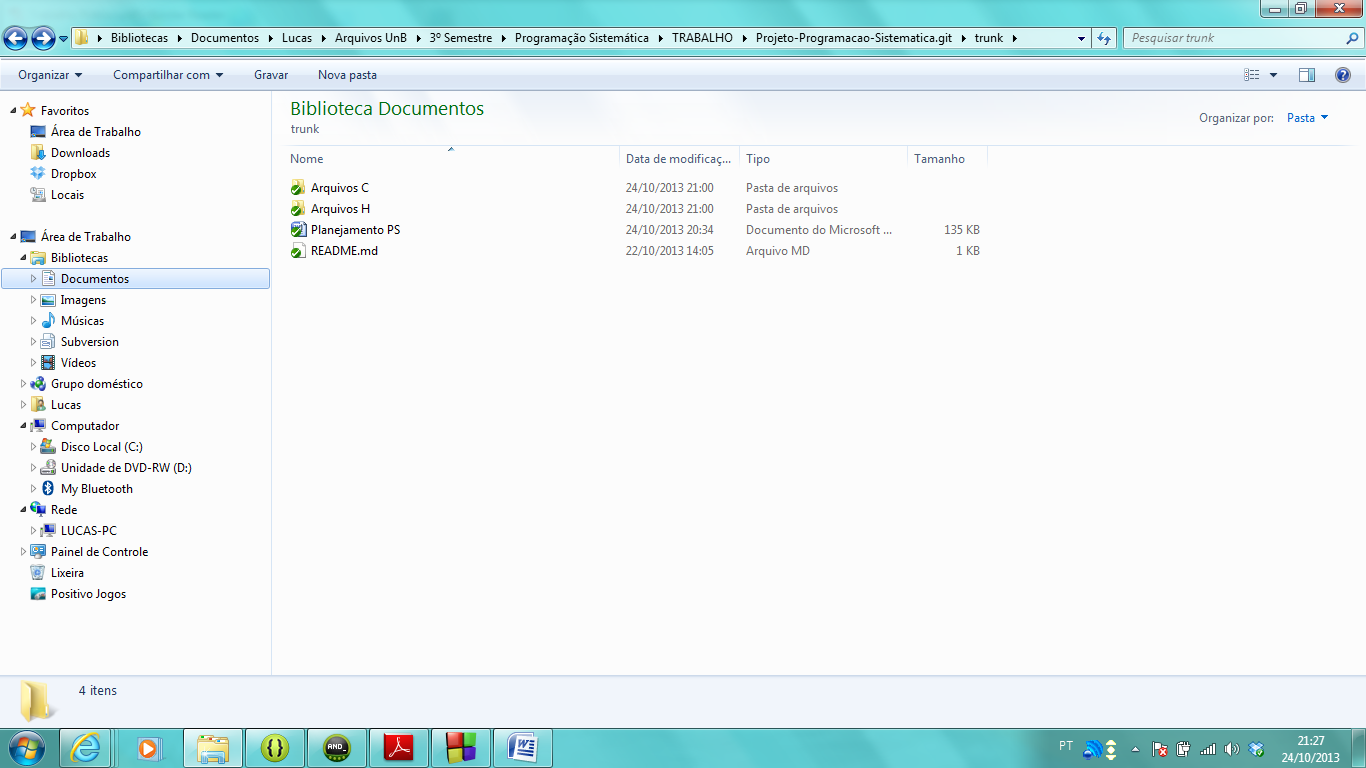
...

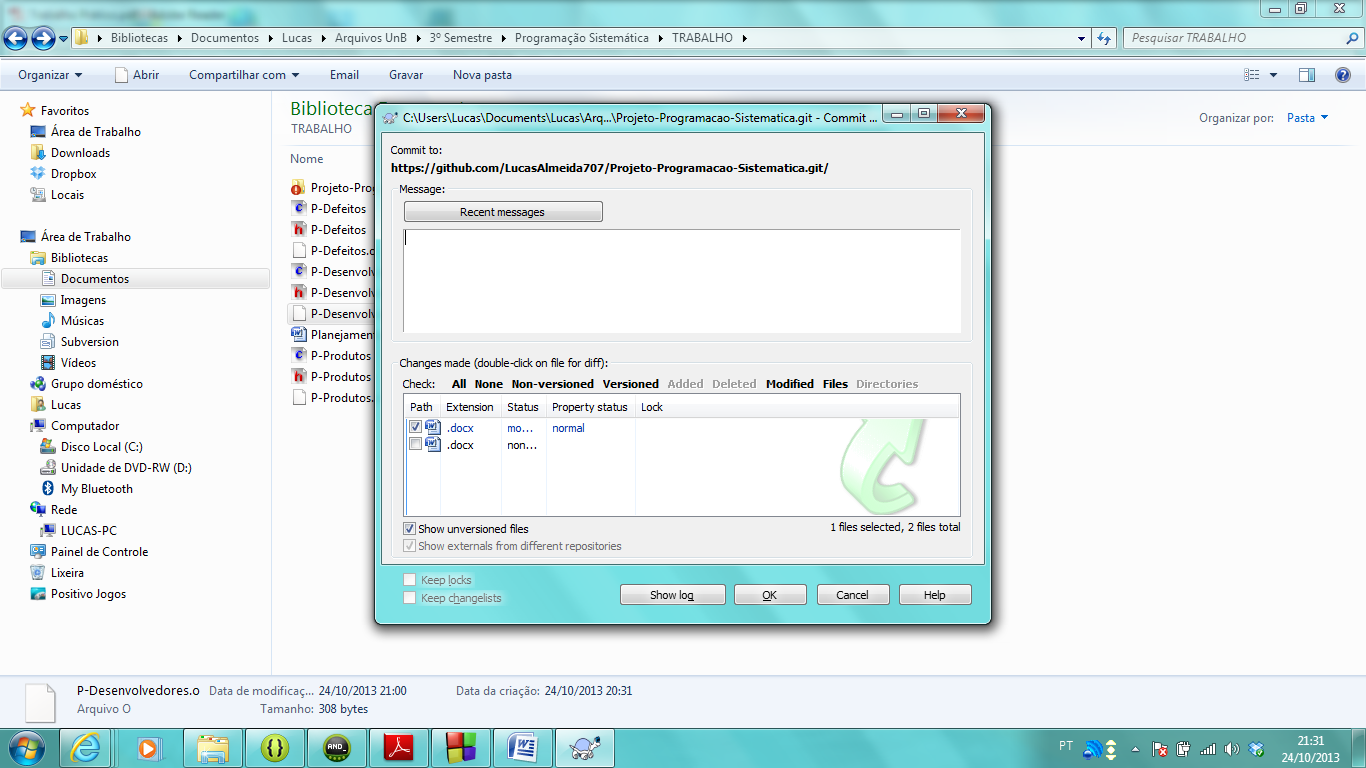
→ **Organização dos Arquivos:**

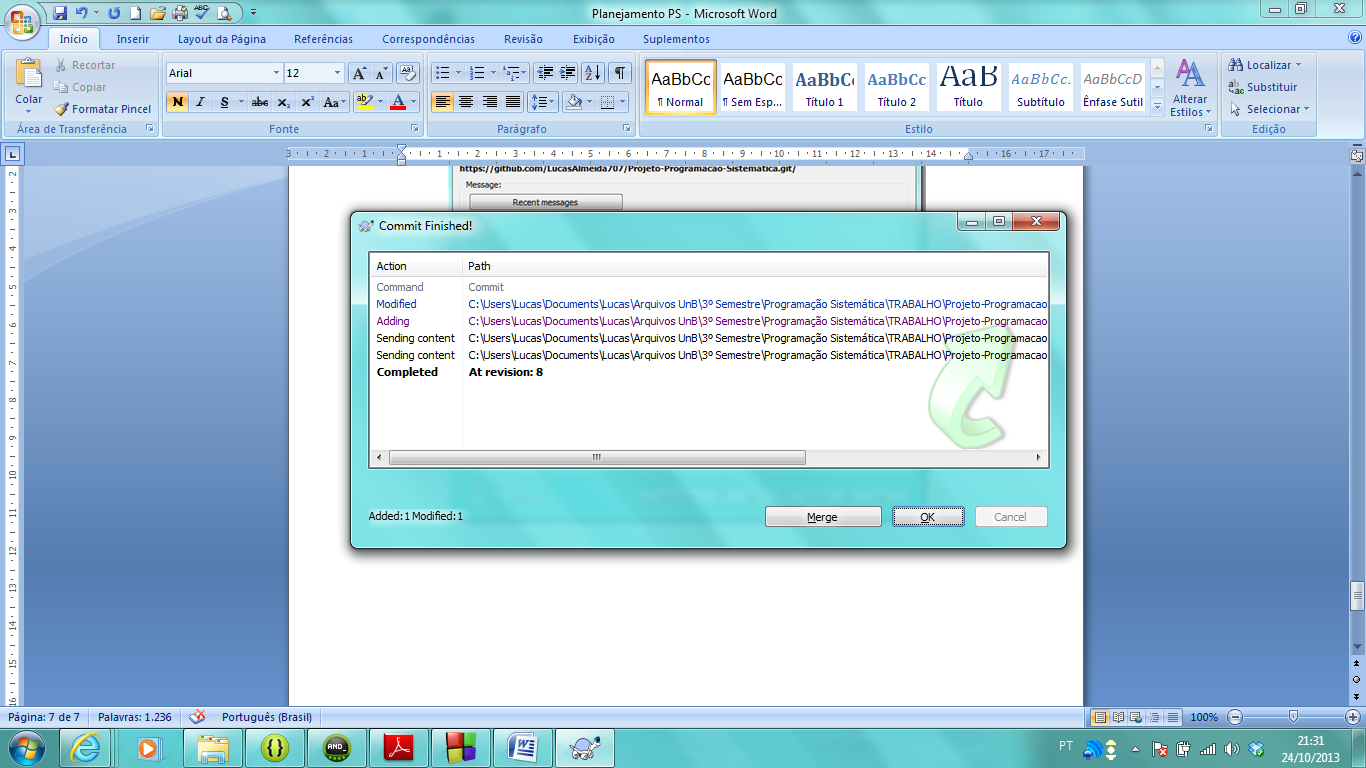
→ **Uso de Espaço em Disco:**

**- 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/>.

****

****

****

