TÉCNICAS DE PROGRAMAÇÃO 1

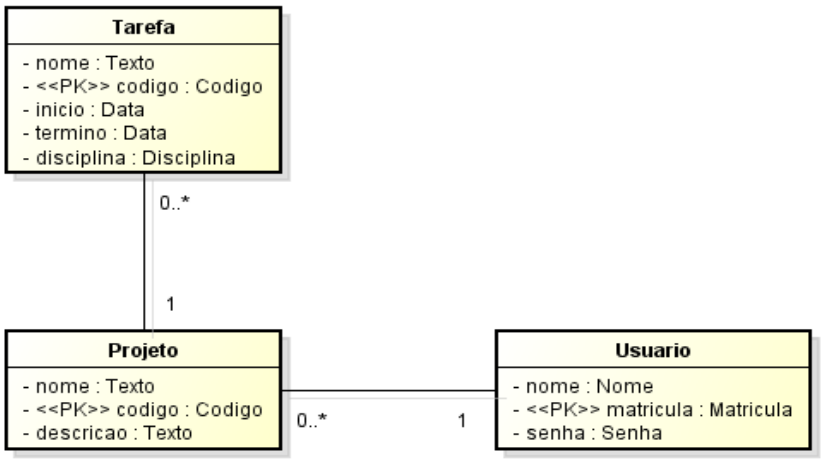
TRABALHO PRÁTICO

1. INTRODUÇÃO

O trabalho prático consiste no desenvolvimento de sistema de software com os requisitos descritos a seguir.

2. REQUISITOS FUNCIONAIS

O sistema de software a ser desenvolvido possibilitará o suporte ao gerenciamento de projetos. Cada usuário pode cadastrar uma conta informando nome, matrícula e senha. Uma vez cadastrado, para ser autenticado, o usuário deve informar matrícula e senha. Após autenticado, o usuário tem acesso aos seguintes serviços: visualizar, editar (exceto matrícula) e descadastrar a sua conta de usuário; visualizar, cadastrar, editar (exceto código) e descadastrar projeto associado à sua conta de usuário; visualizar, cadastrar, editar (exceto código) e descadastrar tarefa associada a projeto que esteja associado à sua conta de usuário. Para visualizar determinada instância de entidade, o usuário deve informar a chave que a identifica. A visualização de determinada instância de entidade resulta na apresentação dos valores dos seus atributos. O sistema deve assegurar, além das regras expressas no seguinte diagrama, que o descadastramento de conta de usuário resulta no descadastramento dos projetos associados à sua conta de usuário; que o descadastramento de projeto resulta no descadastramento das tarefas associadas ao projeto.



3. REQUISITOS NÃO FUNCIONAIS

1. Adotar o estilo de arquitetura em camadas (*layers*).
2. A arquitetura do software deve ser composta por camada de apresentação e por camada de serviço.
3. A camada de apresentação deve ser responsável pela interface com o usuário e pela validação dos dados de entrada.
4. A camada de serviço deve ser responsável pela lógica de negócio e por armazenar dados.
5. Cada camada deve ser decomposta em módulos de software.
6. Módulos de software devem interagir por meio de serviços especificados em interfaces.
7. Módulos de software devem ser decompostos em classes.
8. Devem ser implementadas classes que representem domínios, entidades e controladoras.
9. Implementar o código na linguagem de programação C++.
10. Prover projeto compatível com o ambiente de desenvolvimento Code::Blocks.

4. DOMÍNIOS

DOMÍNIO	FORMATO
CODIGO	Formato DDDDDDDDDDX D é dígito (0-9). X é dígito verificador calculado através de algorítmo módulo 11.
DATA	Formato DD-MES-ANO DD - 01 a 31 MES - 01 a 12 ANO - 00 a 99 Deve ser levado em consideração se ano é ou não é bissexto.
MATRICULA	Formato LLLLDDDD L é letra maiúscula (A-Z). D é dígito (0-9).
NOME	Nome é composto por prenome e até dois sobrenomes. Texto (prenome mais sobrenomes e espaços em branco) é composto por total de até 20 caracteres. Cada caractere é letra (A-Z a-z) ou espaço em branco. Primeira letra de prenome ou de sobrenome é maiúscula (A-Z) e as outras são minúsculas (a-z). Não há espaços em branco em sequência. Acentuação pode ser desconsiderada.
SENHA	Formato XXXXXX Cada caractere X é letra maiúscula (A-Z) ou dígito (0-9). Não pode haver caractere duplicado. Existem pelo menos duas letras maiúsculas e dois dígitos.
TEXTO	10 a 40 caracteres. Cada caractere X é letra (A-Z ou a-z), dígito (0-9) ou sinal de pontuação (. , ; ? ! : -). Não há espaços em branco em sequência. Não há sinal de pontuação (. , ; : ? ! -) em sequência. Acentuação pode ser desconsiderada.
DISCIPLINA	Arquitetura, Desenvolvimento, Gerenciamento, Implantacao, Requisitos, Teste

TÉCNICAS DE PROGRAMAÇÃO 1

TRABALHO 2

MATRÍCULAS : **NOTA:**

1. ATIVIDADES A SEREM REALIZADAS

1. Construir modelo de arquitetura do software.
2. Declarar em código as interfaces entre módulos.
3. Projetar e implementar camada de apresentação.
4. Projetar e implementar camada de serviço.
5. Criar vídeo que demonstre a execução com sucesso do código integrado.

2. REQUISITOS A SEREM CUMPRIDOS

1. Trabalho pode ser realizado individualmente ou por equipe com até três participantes.
2. Desenvolver o sistema de software seguindo os requisitos especificados (funcionais e não funcionais).
3. Preencher os documentos com clareza e atentar para ortografia.
4. Adotar uma convenção de codificação (informe a convenção de codificação adotada no seu trabalho).
5. Fornecer os códigos em formato fonte e em formato executável.
6. Em cada classe, identificar por comentários, a matrícula do aluno responsável pela implementação da classe.
7. Modelo de arquitetura deve conter diagrama composto por módulos, interfaces entre módulos e relacionamentos.
8. Modelo de arquitetura deve conter descrições textuais das responsabilidades de cada módulo.
9. Fornecer modelo de arquitetura de software em arquivo PDF.
10. Declarar em código as interfaces entre módulos por meio de classes abstratas.
11. Classes abstratas devem ser compostas por métodos virtuais puros.
12. Camada de apresentação pode ser codificada usando *cin* e *cout, PDCurses* ou *wxWidgets*.
13. Camada de apresentação deve depender dos serviços declarados nas interfaces.
14. Camada de serviço dever implementar os serviços declarados nas interfaces.
15. Camada de serviço pode armazenar os objetos em estrutura de dados em memória (fila, pilha etc.).
16. Camada de serviço pode armazenar os objetos em banco de dados relacional usando o produto SQLite.
17. Nesse trabalho, associações entre entidades são implementadas.
18. Vídeo a ser fornecido deve demonstrar que o código integrado é executado com sucesso.
19. Deve exisitr um caso de teste para cada funcionalidade relacionada nos requisitos funcionais.
20. Cada caso de teste deve contemplar um cenário de sucesso.
21. Fornecer projeto Code::Blocks que possibilite compilar e executar códigos sem erros na plataforma de correção.
22. Incluir todos os artefatos construídos em um arquivo zip com nome T2-TP1-X-Y-Z.ZIP (incluindo o vídeo criado).
23. No nome do arquivo, os valores de X, Y e Z são os números de matrícula dos autores do trabalho.
24. Testar se o arquivo pode ser descompactado com sucesso e se não há vírus no mesmo.
25. Enviar o arquivo dentro do prazo.
26. Não cumprimento de requisitos resulta em redução de nota do trabalho.

CRITÉRIOS DE CORRREÇÃO

1 – Construir modelo de arquitetura do software.		PONTOS (% ACERTO)
	Diagrama composto por módulos, interfaces entre módulos e relacionamentos.	0, 25, 50, 75, 100
	Providas descrições textuais das responsabilidades de cada módulo.	0, 25, 50, 75, 100
	Providas assinaturas de métodos de cada interface.	0, 25, 50, 75, 100
2 – Declarar em código as interfaces entre módulos.		
	Interfaces entre módulos declaradas por meio de classes abstratas.	0, 25, 50, 75, 100
	Classes abstratas são compostas por métodos virtuais puros.	0, 25, 50, 75, 100
3 – Implementar camada de apresentação.		
	Presença de classes que implementam interfaces e de classes que dependem de interfaces.	0, 25, 50, 75, 100
	Presença de classes que implementam interações com o usuário e validações de dados de entrada.	0, 25, 50, 75, 100
4 – Implementar camada de serviço.		
	Presença de classes que implementam interfaces.	0, 25, 50, 75, 100
	Presença de classes que implementam lógica de negócio e de classes que realizam armazenamento de dados.	0, 25, 50, 75, 100
5 – Criar vídeo que demonstre a execução com sucesso do código integrado.		
	Vídeo demonstra a execução com sucesso do código integrado.	0, 25, 50, 75, 100