TÉCNICAS DE PROGRAMAÇÃO 1

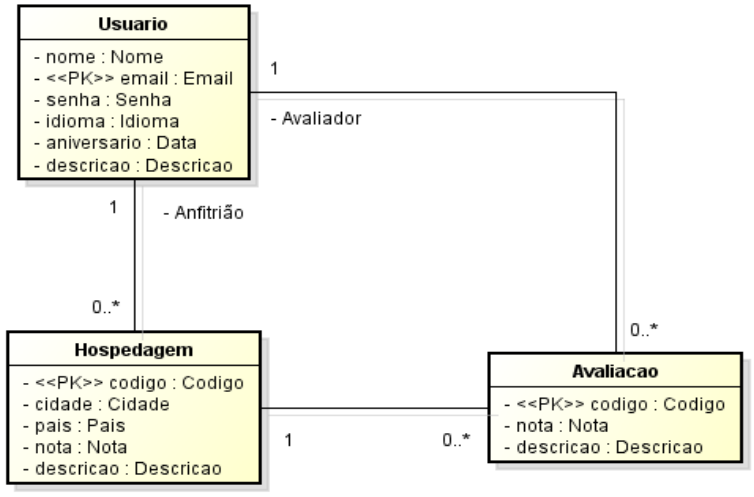
TRABALHO PRÁTICO

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 a prestação de serviço de hospedagem gratuita para turistas. Por meio desse sistema, qualquer usuário pode listar hospedagens disponíveis, acessar dados de hospedagens disponíveis e acessar dados de anfitriões de hospedagens disponíveis (exceto senha). Cada usuário pode cadastrar uma conta. Ao cadastrar uma conta, o usuário deve informar nome, endereço de correio eletrônico e senha. Uma vez cadastrada a conta, para ser autenticado, o usuário deve informar endereço de correio eletrônico e senha. Após autenticado, o usuário tem acesso aos seguintes serviços: editar (exceto correio eletrônico) e descadastrar a sua conta; cadastrar hospedagem, editar (exceto código) e descadastrar hospedagem da qual é anfitrião; cadastrar avaliação de qualquer hospedagem, editar (exceto código) e descadastrar avaliação onde é o avaliador. O sistema deve assegurar, além das regras expressas por meio do seguinte diagrama de classes, as seguintes regras: nota de cada hospedagem é a média das notas das avaliações associadas à hospedagem; descadastramento de conta de usuário descadastra hospedagens onde o usuário é o anfitrião e avaliações onde o usuário é o avaliador; descadastramento de hospedagem resulta no descadastramento de avaliações da hospedagem. O sistema deve garantir que os serviços prestados não resultem em inconsistências. A seguir, tem-se diagrama de classes composto por entidades e por relacionamentos entre as mesmas.



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

NOME	FORMATO
CIDADE	Antalya, Bangkok, Delhi, Dubai, Hong Kong, Londres, Macau, Mumbai, Paris, Rio de Janeiro, São Paulo, Seul, Istambul, Kuala Lumpur, Nova Iorque, Osaka, Phuket, Shenzhen, Tóquio Desconsiderar a acentuação.
CÓDIGO	Formato DDDDDDDDDDX D é dígito (0-9). X é dígito verificador calculado via algorítmo de Luhn (vide Luhn algorithm - Wikipedia)
DATA	Formato DD/MES DD - 01 a 31 MES - Jan, Fev, Mar, Abr, Mai, Jun, Jul, Ago, Set, Out, Nov, Dez
DESCRICAO	0 a 40 caracteres. Não há espaços em branco em sequência. Não há caracteres de pontuação (. , ; : ? ! -) em sequência.
EMAIL	Formato *parte-local@domínio* Nome de *parte-local* é composto por até 64 caracteres. Caractere pode ser letra (A-Z ou a-z). Caractere pode ser dígito (0-9). Caractere pode ser hífen (-), sublinhado (_) ou ponto (.) desde que seguido por letra ou dígito. Caractere ponto (.) não pode ser o primeiro caractere no nome. Nome de *domínio* é composto por lista de termos separados por pontos. Ponto (.) não pode ser o primeiro caractere do nome de domíno e não pode ocorrer em sequência. Cada termo é composto por até 63 carateres. Caractere no termo pode ser letra (A-Z ou a-z). Caractere no termo pode ser dígito (0-9). Caractere no termo pode ser hífen (-) desde que não seja o primeiro ou o último caractere.
IDIOMA	Inglês, Chinês Mandarim, Hindi, Espanhol, Francês, Árabe, Bengali, Russo, Português, Indonésio Desconsiderar a acentuação.
NOME	Nome é composto por prenome e sobrenome. Nome é composto por até 30 caracters. Cada caractere é letra (A-Z a-z) ou espaço em branco. Não há espaços em branco em sequência. Primeiro caractere de prenome ou de sobrenome é maiúscula (A-Z) e os outros são minúsculas (a-z).
NOTA	0 a 10
PAIS	Estados Unidos, Brasil, China, Coreia do Sul, Emirados, França, Índia, Japão, Malásia, Reino Unido, Tailândia, Turquia Desconsiderar a acentuação.
SENHA	Formato XXXXX Cada caractere X é letra (A-Z ou a-z), dígito (0-9) ou caractere especial (! # $ % &). Existe pelo menos uma letra (maiúscula ou minúscula), um dígito e um caractere especial.

TÉCNICAS DE PROGRAMAÇÃO 1

TRABALHO 1

1. ATIVIDADES A SEREM REALIZADAS

1. Codificar classe para cada domínio (*domain*).
2. Codificar classe para cada entidade (*entity*).
3. Codificar e executar teste de unidade (*unit test*) para cada classe domínio.
4. Codificar e executar teste de unidade (*unit test*) para cada classe entidade.
5. Documentar classes que representam domínios e entidades por meio de texto em formato HTML.

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 um padrão de codificação (*coding standard*).
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. Cada classe domínio deve conter atributo que seja instância de tipo suportado pela linguagem de programação.
8. Cada classe domínio deve permitir acesso ao atributo por meio de métodos públicos *set* e *get*.
9. Método *set* de cada classe domínio deve lançar exceção em caso de formato incorreto.
10. Cada classe de entidade deve conter atributos onde cada atributo é instância de classe domínio.
11. Cada classe de entidade deve permitir acesso aos atributos por meio de métodos públicos *set* e *get*.
12. Nesse trabalho, associações entre entidades não são implementadas.
13. Cada teste de unidade deve ser classe com diferentes métodos para diferentes casos de teste.
14. Cada teste de domínio deve exercitar o domínio por meio de um cenário de sucesso e de um de falha.
15. Cada teste de entidade deve invocar cada método público da entidade em teste pelo menos uma vez.
16. Classes devem funcionar corretamente segundo os testes de unidade fornecidos.
17. Fornecer projeto Code::Blocks que possibilite compilar e executar códigos sem erros na plataforma de correção.
18. Gerar documentação dos domínios e das entidades em formato HTML por meio da ferramenta Doxygen.
19. Escrever documentação das classes em formato HTML segundo perspectiva dos usuários das classes.
20. Incluir todos os artefatos construídos em um arquivo zip com nome T1-TP1-X-Y-Z.ZIP.
21. No nome do arquivo, os valores de X, Y e Z são os números de matrícula dos autores do trabalho.
22. Testar se o arquivo pode ser descompactado com sucesso e se não há vírus no mesmo.
23. Enviar o arquivo dentro do prazo.
24. Não cumprimento de requisitos resulta em redução de nota do trabalho.