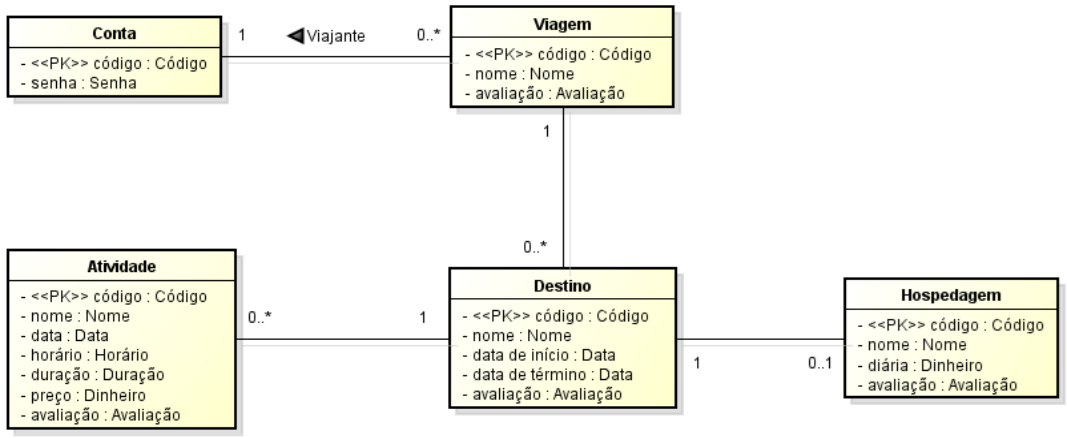
TÉCNICAS DE PROGRAMAÇÃO 1

ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA A SER DESENVOLVIDO

1. REQUISITOS FUNCIONAIS

O sistema a ser desenvolvido possibilitará operações designadas pela sigla CRUD (Create, Read, Update, Delete). O sistema visa facilitar o planejamento de viagens. Para usar o sistema, o viajante deve criar uma conta. Após ser autenticado, o viajante tem acesso aos seguintes serviços: criar, ler, atualizar, e excluir viagem; criar, ler, atualizar, e excluir destino; criar, ler, atualizar, e excluir atividade; criar, ler, atualizar, e excluir hospedagem. A atualização de um registro (viagem, destino, atividade ou hospedagem) não possibilitar atualizar dado que identifica o registro (chave primária). O sistema deve também prover os seguintes serviços: consultar custo de viagem; listar viagens associadas a viajante, listar destinos associados a viagem, listar atividades associadas a destino e listar hospedagens associadas a destino. Ao apresentar uma listagem, o sistema deve ser apresentar código e nome de cada registro integrante da listagem. O sistema deve assegurar as seguintes regras: não pode ser excluído registro se existir registro associado a ele; data de atividade deve estar no intervalo definido pelas datas de início e término do destino associado. O sistema deve também assegurar as regras de negócio representadas no seguinte diagrama:



2. 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.
11. Nas implementações dos códigos de validação não é necessário considerar acentuação e nem a letra ç.

3. DOMÍNIOS

NOME	FORMATO VÁLIDO
Avaliação	Dígito 0, 1, 2, 3, 4 ou 5
Código	Seis caracteres Cada caracter pode ser letra (A - Z ou a - z) Cada caracter pode ser dígito (0 - 9)
Data	Formato DD-MM-AA DD - 00 a 31 MM - 01 a 12 AA - 00 a 99 Levar em consideração anos bissextos
Dinheiro	Valor de 0,00 a 200.000,00
Duração	Valor de 0 a 360
Horário	Formato HH:MM HH pode ser 00 a 23 MM pode ser 00 a 59
Nome	Texto com até 30 caracteres
Senha	Cinco dígitos (0 - 9) Não há dígito duplicado Os seis dígitos não podem estar em ordem crescente (01234, 12345, 23456 etc.) Os seis dígitos não podem estar em ordem descrescente (43210, 54321, 65432 etc.)

TÉCNICAS DE PROGRAMAÇÃO 1

TRABALHO 2

1. ATIVIDADES A SEREM REALIZADAS

1. Construir um modelo de arquitetura do software, codificar e documentar as interfaces entre módulos.
2. Projetar e codificar a camada de apresentação.
3. Projetar e codificar a camada de serviço.
4. 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 cinco participantes.
2. Preencher os documentos com clareza, atentar para a ortografia e adotar um padrão de codificação (*coding standard*).
3. Fornecer os códigos em formato fonte e em formato executável.
4. Desenvolver o sistema segundo os requisitos funcionais e não funcionais especificados.
5. Em cada classe, identificar por comentários, a matrícula do aluno responsável pela implementação da classe.
6. Modelo de arquitetura deve conter diagrama composto por módulos, interfaces entre módulos e relacionamentos.
7. Modelo de arquitetura deve conter descrições textuais das responsabilidades de cada módulo.
8. Fornecer modelo de arquitetura de software em arquivo PDF.
9. Declarar em código as interfaces entre módulos por meio de classes abstratas.
10. Classes abstratas devem ser compostas por métodos virtuais puros.
11. Documentar as classes que representam interfaces usando formato HTML por meio da ferramenta Doxygen.
12. Escrever documentação das classes em formato HTML segundo perspectiva dos usuários das classes.
13. Camada de apresentação deve implementar e depender de serviços relacionados em interfaces declaradas.
14. Camada de apresentação pode ser codificada usando *cin* e *cout, PDCurses* ou interface gráfica.
15. Camada de serviço dever implementar serviços relacionados em interfaces declaradas.
16. Camada de serviço pode armazenar os objetos em estrutura de dados em memória (fila, pilha etc.).
17. Camada de serviço pode armazenar os objetos em banco de dados relacional usando o produto SQLite.
18. Neste trabalho, associações entre entidades são implementadas.
19. Vídeo a ser fornecido deve demonstrar que o código integrado é executado com sucesso.
20. Deve exisitr um caso de teste para cada funcionalidade relacionada nos requisitos funcionais.
21. Cada caso de teste deve contemplar um cenário de sucesso.
22. Fornecer projeto Code::Blocks que possibilite compilar e executar códigos sem erros na plataforma de correção.
23. Incluir todos os artefatos construídos em um arquivo zip e atribuir o nome T1-TP1-X-Y-Z.ZIP ao arquivo zip.
24. No nome do arquivo zip, X, Y e Z devem ser os números de matrícula dos autores do trabalho.
25. Testar se o arquivo pode ser descompactado com sucesso e se não há vírus no mesmo.
26. Enviar o arquivo dentro do prazo.
27. Não cumprimento de requisitos resulta em redução de nota do trabalho.

3. CRITÉRIOS DE CORRREÇÃO

ITEM	CRITÉRIO	% ACERTO
1	Provido modelo de arquitetura do software, codificadas e documentas interfaces entre módulos.	0, 25, 50, 75, 100
2	Código implementa e depende de interfaces declaradas, realiza interface com usuário e valida dados de entrada.	0, 25, 50, 75, 100
3	Código implementa interfaces declaradas, implementa lógica de negócio e armazena dados.	0, 25, 50, 75, 100
4	Vídeo demonstra a execução com sucesso do código integrado.	0, 25, 50, 75, 100