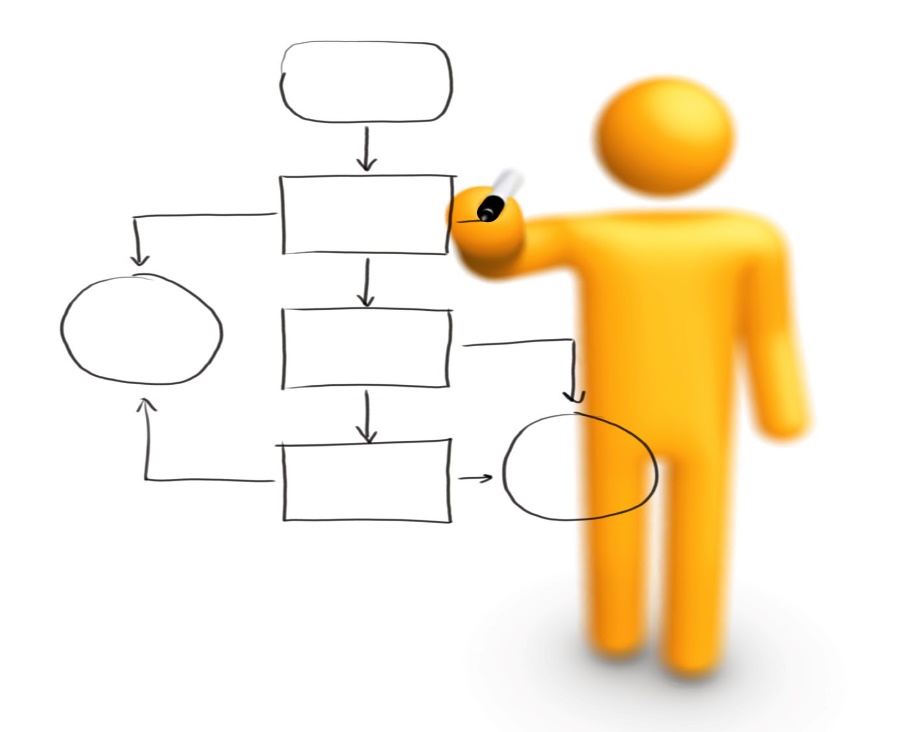


Gestão de Projetos

01 de Novembro de 2014



**Estudantes & Autores:**

Ricardo Cerqueira – [up2010304000@fe.up.pt](mailto:up2010304000@fe.up.pt) – 201304000

Inês Carneiro – [up2010303501@fe.up.pt](mailto:up2010303501@fe.up.pt) - up201303501

Marta Lopes – [ei12106@fe.up.pt](mailto:ei12106@fe.up.pt) – ei12106

**Índice:**

Resumo**3**

Solução Implementada **4**

Casos de utilização**7**

Utilização da interface**8**

Dificuldades**10**

Esforço dedicado pelos elementos do grupo**10**

**Resumo:**

Neste trabalho procuramos implementar uma aplicação para a gestão de projetos de informática, encomendados por vários clientes da empresa.

A aplicação desenvolvida é composta por um conjunto de clientes, um conjunto de projetos, um conjunto de tarefas e um conjunto de colaboradores.

Os clientes contratam a empresa para a realização de um ou mais projetos. Um projeto é composto por tarefas, que podem ou não depender umas das outras. Isto é, só é possível o progresso de uma tarefa se todas as tarefas das quais depende estiverem completas. Todos os projetos vão ter um prazo de conclusão associado, um conjunto de colaboradores encarregues de o executarem.

Um colaborador pode ser de vários tipos, e o seu custo, por hora, será função do tipo. Terá também um limite de horas de trabalho semanal.

**Solução Implementada:**

Cada cliente, representado pela classe *Client* será composto pelo seu nome, um código identificador e estará associado a um conjunto de projetos encomendados.

Cada projeto, representado pela classe *Project*, é composto pelo seu nome, um código identificador, o seu tipo, o custo do projeto até ao momento, a data do prazo para a sua conclusão e está associado ao cliente que o encomendou, às tarefas que o compõe e aos colaboradores que nele trabalham.

Cada colaborador, representado pela classe *Collaborator*, é composto pelo seu nome, um código identificador, o número máximo de horas que pode trabalhar, o número de horas que está a trabalhar atualmente, e tem associado os projetos em que trabalha atualmente, as tarefas em que trabalha atualmente, assim como o número de horas semanais que trabalha para cada tarefa e ainda as tarefas que já completou. Os vários tipos de colaboradores, programadores, arquitetos, gestores, e *testers* são representados pelas classes *Programmer*, *Architect*, *Manager*, e *Tester* respetivamente, e que derivam da classe *Collaborator*.

Cada tarefa, representada pela classe *Task*, é composta pelo seu nome, um código identificador, uma breve descrição, o número de horas de trabalho restantes para terminar a tarefa e está associado ao projeto a que pertence, aos colaboradores que nela trabalham, assim como o número de horas que cada um nela trabalha, as tarefas de que depende, e as tarefas que dependem dela.

A aplicação é representada pela classe *Application*, que guarda um vetor de apontadores para objetos de cada uma das classes apresentadas acima.

A aplicação funciona avançando de semana a semana e, para cada tarefa, caso esta não esteja dependente de outra que ainda não esteja completa, é subtraído ao esforço restante o número total de horas que a tarefa foi trabalhada por todos os colaboradores e é adicionado ao custo do projeto a que pertence, o custo total de todos os colaboradores (que é obtido multiplicando o número de horas que o colaborador trabalha pelo seu custo).

Caso este processo resulte na terminação da tarefa, todos os colaboradores que trabalhavam na tarefa são retirados da mesma, o que resulta num aumento das horas disponíveis para cada um correspondente ao número de horas que trabalhavam na tarefa.

É obtida uma estimativa da data de terminação de uma tarefa sem dependências dividindo o esforço restante para a terminação da tarefa pelo total de horas que a tarefa é trabalhada pelos seus colaboradores. Se a tarefa depender de outras, a sua data de terminação é obtida somando ao valor obtido pelo processo anterior o máximo dos tempos para terminação de todas as tarefas de que depende. Os atrasos nas tarefas são modelados por um aumento no esforço necessário para terminar. Com este aumento, o tempo de terminação da tarefa aumenta, o que resulta um aumento do tempo de terminação de todas as tarefas que dela dependam.

De modo a manter a informação do sistema de sessão para sessão os dados são guardados em ficheiros no início de sessão e lidos aquando do arranque do programa.

De modo a minimizar os recursos utilizados, na escrita dos ficheiros, sempre que um objeto contem uma referência a outro, é escrito o seu ID. Na leitura dos ficheiros é lido este ID e, quando estiverem criados todos os objetos do sistema, estes IDs são substituídos pelos apontadores para os objetos correspondentes.

Aquando da criação de uma tarefa, de um cliente ou de um projeto é verificada a sua pré-existência que caso se verifique são lançadas exceções.

/\*Cada Colaborador tem como parâmetro um vetor de pares em que primeiro elemento representa uma tarefa e o segundo o número de horas dedicadas a essa tarefa e por sua vez cada tarefa terá também um vetor de pares em que o primeiro elemento é um colaborador e o segundo é um inteiro que indica as horas que esse colaborador trabalha.\*/

Na eliminação de um objeto são sempre eliminados automaticamente todas as referências ao objeto no sistema.

/\*Na eliminação de tarefas foi tido em conta as tarefas de que depende e dependentes, com a eliminação da tarefa selecionada as tarefas de que depende ficam agora com apontadores para as tarefas dependentes da eliminada.\*/

Na eliminação de uma tarefa, cada dependência, caso exista, passa a ter como dependente todas as dependentes da tarefa a ser eliminada, e cada dependente, caso exista, passa a ter como dependências todas as dependências da tarefa a ser eliminada.

Quando um projeto é eliminado, todas as suas tarefas são também eliminados.

Quando um cliente é eliminado todos os projetos que lhe estão associados também são eliminados.

A lista de projetos de um colaborador e a lista de colaboradores de um projeto são atualizadas automaticamente; um colaborador e um projeto não podem estar associados se o colaborador não trabalhar em pelo menos uma tarefa do projeto.

Uma tarefa só pode estar associada a um projeto, e um projeto só pode estar associado a um cliente. Só podem existir dependências entre tarefas que pertençam ao mesmo projeto. Se uma tarefa estiver associada a outras por dependências não pode ser removida do projeto.

Para ordenação dos vetores de objetos, para cada classe, foi definida uma interface de comparação que define o comportamento das classes que implementam efetivamente a comparação, quer entre objetos quer entre apontadores para os objetos, com diferentes critérios para cada classe. Esta estrutura permite a utilização dos comparadores polimorficamente, usando um vetor de comparadores das várias classes, é possível trocar facilmente o critério de ordenação.

Para a

**Diagrama de classes (UML):**

**Casos de utilização:**

* Criar projetos
* Apagar projetos
* Apagar clientes
* Apagar tarefas
* Eliminar dependências
* Apagar colaboradores
* Desassociar clientes de projetos
* Desassociar tarefas de projetos
* Desassociar tarefas a colaboradores
* Desassociar projetos de clientes
* Desassociar colaboradores a tarefas
* Alterar informação de projetos
* Alterar informação de clientes
* Alterar informação de tarefas
* Alterar informação de colaboradores
* Adicionar dependências a tarefas
* Adicionar dependentes a tarefas
* Criar colaboradores
* Criar clientes
* Adicionar tarefas a colaboradores
* Adicionar colaboradores a tarefas
* Adicionar projetos a clientes
* Adicionar tarefas a projetos
* Visualizar os projetos ordenados por tempo para terminar, por prazo, ordem alfabética, por custo, por ID, por número de tarefas, número de colaboradores e por tipo
* Visualizar tarefas ordenadas por tempo para completar, ordem alfabética, por esforço, por ID e por tempo estimado.
* Visualizar clientes ordenados por custo total dos seus projetos, número de projetos, ordem alfabética e por ID
* Visualizar colaboradores ordenados por ordem alfabética, por ID, por experiência (número de tarefas terminadas), por numero de tarefas, por número de horas que trabalha, por número de projetos, e por número máximo de horas de trabalho por semana
* Visualizar tarefas e projetos associadas a um colaborador
* Visualizar projeto, colaboradores e tarefas associadas a uma tarefa, assim como um tempo estimado para a sua conclusão
* Visualizar cliente, colaboradores, tarefas associadas a um projeto, assim como o seu custo, o seu prazo e uma data estimada para a sua conclusão
* Visualizar projetos associados a um cliente assim como o total dos custos dos seus projetos

**Utilização da interface:**

De modo a modificar os objetos do sistema é necessário introduzir um dos seguintes comandos:

Comandos para projetos:

- *delete:* elimina projeto do sistema, elimina também todas as tarefas a ele associados;

- *setName <name>:* modifica nome do projeto;

- *setType <type>:* modifica tipo do projeto;

- *addTask <taskID>:* adiciona tarefa ao projeto, não é possível se a tarefa já estiver associada a um projeto;

- *removeTask <taskID>:* remover tarefa de projeto, não é possível se tarefa não estiver associada ao projeto, ou se a tarefa estiver associada a outras por dependências;

- *setDeadline <deadline>:* modifica prazo do projeto;

- *setClient<ClientID>:* associa projeto a cliente, não é possível se o projeto já tiver um cliente;

- *removeClient*: desassocia cliente do projeto, não é possível se o projeto não tiver cliente;

Comandos para clientes:

- *delete:* elimina cliente do sistema, elimina também todos os seus projetos e todas as tarefas de cada um;

*- setName <name>:* modifica nome do cliente;

- *addProject <projectID>:* adiciona projeto ao cliente, não é possível se o projeto já tiver cliente;

*- removeProject <projectID>:* remove projeto do cliente, não é possível se o projeto não estiver associado ao cliente;

Comandos para colaboradores:

*- delete:* elimina colaborador do sistema;

- *setName <name>:* modifica nome do colaborador;

- *setWeeklyHours <hours>:* modifica horas semanais máximas do colaborador;

*- addTask <taskID> <hours>:* adiciona tarefa ao colaborador, não é possível se o novo número de horas do colaborador exceder o máximo;

- *removeTask <taskID>:* remove tarefa do colaborador, não é possível se a tarefa não estiver associada ao projeto;

Comandos para tarefas:

*- delete:* elimina tarefa;

- *setName <name>:* modifica nome da tarefa;

*- setDescription <description>:* modifica descrição da tarefa;

- *addCollaborator <colID> <hours>:* adiciona colaborador à tarefa, não é possível se o número de horas total do colaborador exceder o seu máximo;

- *removeCollaborator <colID>:* remove colaborador, não é possível se o colaborador não estiver associado à tarefa;

*- addDependency <taskID>:* adiciona dependência à tarefa, não é possível se as tarefas não pertencerem ao mesmo projeto;

- *removeDependency <taskID>:* remove dependência da tarefa, não é possível se a dependência não existir;

- *addDependant <taskID>:* adiciona dependente à tarefa, não é possível se as tarefas não pertencerem ao mesmo projeto;

- *removeDependant <taskID>:* remove dependente da tarefa, não é possível se a dependente não existir;

*- setEffort <effort>:* modifica esforço restante da tarefa;

*- setProject <projectID>:* associa tarefa a um projeto, não é possível se a tarefa já estiver associada a um projeto;

- *removeProject:* remove projeto da tarefa, não é possível se a tarefa não tiver projeto;

**Dificuldades:**

Utilização adequada de exceções.

**Esforço dedicado pelos elementos do grupo:**