Advanced Institute for Artificial Intelligence

Programação em Python

https://advancedinstitute.ai

Programação Python

Projeto de Software

Referências

Referências e Fontes das Imagens

- UML2: Uma Abordagem Prática (Book)
- UML: Guia do usuário (Book)

Classificação, Abstração e Instanciação

> *No início da infância, o ser humano aprende e pensa de maneira bastante semelhante à filosofia da orientação a objetos, representando seu conhecimento por meio de abstrações e classificações.*

- Intuição de classes como **grupos de objetos com as mesmas características e comportamentos**
  - E.g.: qualquer peça de metal com quatro rodas que se locomova de um lugar para outro transportando pessoas recebe a *denominação de carro*.
    - Esforço de abstração: carros apresentarem diferentes formatos, cores e estilos;

UML2: Uma Abordagem Prática

Classes, Atributos e Objetos

- Quando objeto é instanciando a partir de uma classe, estamos **criando um novo item do conjunto representado por essa classe**;
- Possuem as **mesmas características e comportamentos** de todos os outros objetos já instanciados;
  - Possuem mesmos atributos porém **não são exatamente iguais**: cada objeto armazena valores diferentes em seus atributos:
  - Pessoas possuem valores diferentes para `CPF`, `nome`, `telefone`, etc.
- Uma classe representa uma **categoria** e os objetos são os **membros ou exemplos** dessa categoria

Projeto de Software

- **Análise Orientada a Objetos**
  - Identificar os **objetos e as interações** entre esses objetos
  - Sobretudo, sobre o **que precisa ser feito**.
    - Entrevistar clientes, estudar processos, eliminar possibilidades;
    - **Requisitos!**
- **Projeto Orientado a Objetos**
  - Processo de conversão de **requisitos em uma especificação de implementação**
  - Descrever uma coleção de objetos que interagem por meio de seus **dados e comportamento**.
  - Output: **especificação de implementação**
    - **Classes e Interfaces** independente de linguagem de programação.

Projeto de Software

- **Programação Orientada a Objetos**
  - Converter Projeto Orientado a Objetos em um **software funcional**.
  - Nem sempre é possível um **mapeamento 1-para-1 entre design e implementação**;
- No mundo real as **coisas são mais complicadas**:
  - Sempre encontraremos **coisas que precisam de mais análise durante o projeto**;
  - Descobrimos p**artes do projeto que precisam ser mais claras, durante a implementação**;
- **Desenvolvimento iterativo**:
  - **Pequena parte da tarefa** é modelada, projetada e programada
  - Incluir novos recursos em uma **série de curtos ciclos de desenvolvimento**;

Linguagem de Modelagem Unificada - UML

Linguagem de Modelagem Unificada - UML

- União de três métodos orientados a objeto: o método de Booch, o método OMT (Object Modeling Technique) o método OOSE (Object-Oriented Software Engineering);
- Lançamento, em 1996, da primeira versão da UML;
- Versão 2.0 lançada em julho de 2005;
- Atualmente a UML encontra-se na versão 2.5 (lançada em 2015)
- 23 Diagramas agrupados em **Estruturais** e **Comportamentais**
  - **Nem todos diagramas são necessários no projeto de um dado software**;
  - **Múltiplas visões** do sistema a ser modelado, analisando-o e modelando-o sob **diversos aspectos**;

Caso de Uso

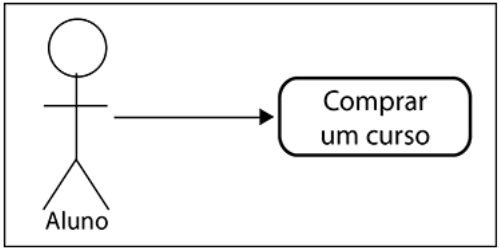
- Um dos mais importantes e utilizados da UML;
- Aplicado quando precisamos conversar com um cliente para entender melhor os requisitos e etapas do desenvolvimento do sistema;
- Tenta esclarecer quais são as funcionalidades do sistema e quem irá fazer uso delas;
- O diagrama de caso de uso nos ajudará a modelar e esclarecer alguns pontos para a equipe;

Caso de Uso

- Vamos imaginar um cenário para uma plataforma de cursos *E-learning*, onde teremos:
  - Alunos e Professores são considerados **Atores**;
  - Os atores representam um papel que um ser humano, um dispositivo de hardware ou até outro sistema desempenham com o sistema.
  - O **Caso de Uso** representa as funcionalidades e,
  - A **Comunicação**, são as setas que ligam os **Atores** com as **funcionalidades**.

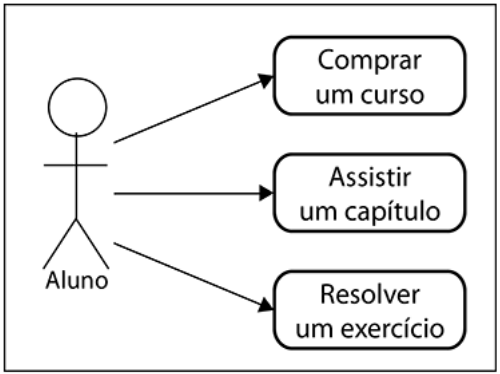
Caso de Uso

- No diagrama, as funcionalidades são representadas por círculos com texto dentro e então, são conectados por setas.



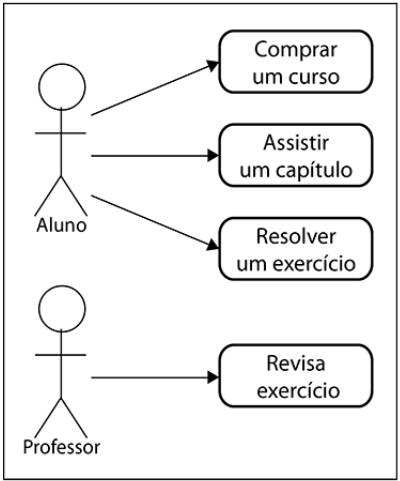
Caso de Uso

- Podemos colocar mais funcionalidades, que também são feitas por alunos:



Caso de Uso

- Podemos ter mais de um ator no diagrama:

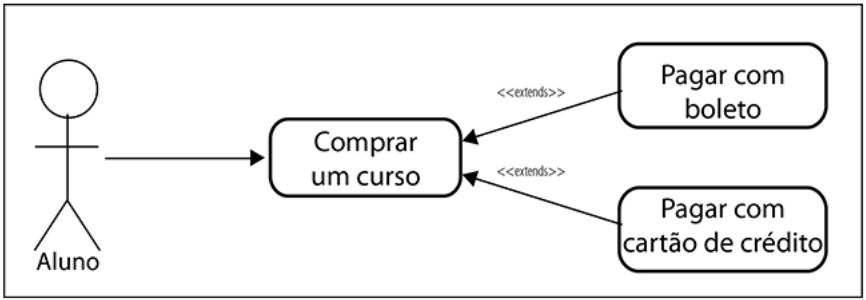


Caso de Uso

- Um dos grandes problemas enfrentados pelas equipes que usam UML é que elas resolvem seguir a linguagem de maneira rígida, sendo que a orientação é usar UML somente quando fizer sentido.
- Uma das grandes vantagens nesse tipo de notação é a liberdade de estender a UML e criar uma nova notação para simplesmente facilitar o entendimento pelos membros da equipe.

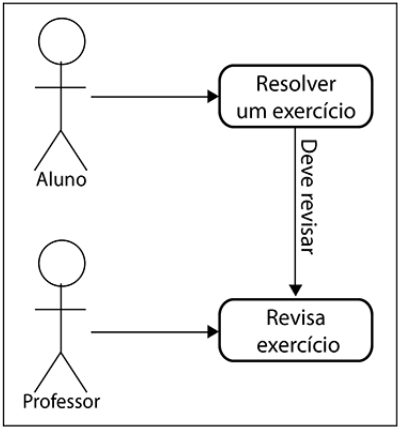
Caso de Uso

- Utilizando a notação ***extends***.



Caso de Uso

- Nada nos impede de evoluir a UML para facilitar nossa comunicação. Podemos, por exemplo, fazer uma ligação entre **"aluno resolver um exercício"** e **"professor revisar um exercício"**, afinal um vai acontecer somente depois do outro:



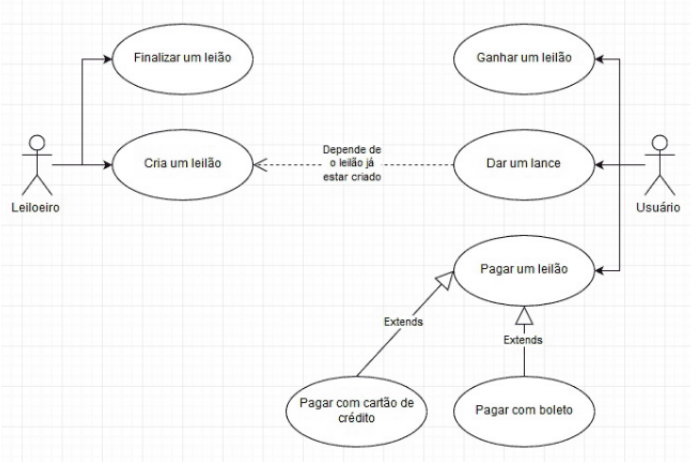
Caso de Uso

- Todo diagrama de caso de uso é geralmente acompanhado de um documento de texto que explica melhor aquela funcionalidade.
- O diagrama que vimos até então apenas nos dá uma ideia das funcionalidades existentes no sistema.

Caso de Uso - Exercício I

- Faça um diagrama de caso de uso para o seguinte sistema:
  - Um leiloeiro pode criar um leilão;
  - Um usuário pode dar lances em um leilão já criado;
  - Um leiloeiro pode finalizar um leilão;
  - Um usuário pode ganhar um leilão;
  - Um usuário pode pagar pelo produto com cartão de crédito;
  - Um usuário pode pagar pelo produto com boleto.

Caso de Uso - Resposta Exercício I



Caso de Uso - Exercício II

- Identifique os atores e as principais funcionalidades:
  - A empresa **BRASOIL** precisa de um sistema para organizar as propostas de fornecedores de material de obra. Nesse sistema o **BRASOIL** vai cadastrar o projeto com data de vencimento e os fornecedores podem colocar propostas detalhadas. Cada proposta será revisada pelo **BRASOIL** para atender os critérios de qualidade e a mais barata será escolhida. Após vencimento a entrega de material será agendado com o fornecedor.

Caso de Uso - Resposta Exercício II

- Identifique os atores e as principais funcionalidades:
  - **Atores:** BRASOIL e os fornecedores
  - **Funcionalidades:** Cadastrar Projeto, Colocar proposta, Revisar proposta e Agendar entrega
    - Muito provável que na fase de análise de requisitos, mais atores surjam e as funcionalidades serão melhor especificadas.

A análise de requisitos é a fase no desenvolvimento de software onde os analistas confirmam e priorizam as funcionalidades, e definem o escopo do projeto com as principais pessoas envolvidas (**stakeholders**). Uma vez que os requisitos foram levantados, eles serão analisados pelos analistas de requisitos.

Caso de Uso - Resposta Exercício II

- Como resultado final temos:
  - Um documento de requisitos que descreve as funcionalidades de maneira consistente, clara e entendível para o desenvolvedor.
  - Dentro desse documento encontramos então os cenários dos nossos casos de uso! Aliás, o caso de uso é muitas vezes utilizado em um documento de requisitos.

Diagrama de Classes

- Toda equipe quando vai começar o software e já tem os requisitos levantados
- A equipe para e pensa um pouco sobre como vai modelar aquele domínio:
  - Quais classes serão criadas?
  - Quais os comportamentos existentes?
  - Como uma se relaciona com a outra?

É por isso que a UML nos provê o chamado **diagrama de classes**.

Diagrama de Classes

- Um dos mais importantes e utilizados da UML;
- **Visualização das classes** que fazem parte do sistema com seus **atributos e métodos**;
- Mostrar **relacionamento entre classes**;
- **Visão estática**;
- **Construído durante a análise** e refinado durante o projeto;
  - **Modelo Conceitual** - informações necessárias ao software;
  - **Modelo de Domínio** - focado na solução do problema;
- **Vários modelos de classe** em um mesmo projeto de software com **enfoques diferentes**;

Diagrama de Classes

- Toda equipe quando vai começar o software e já tem os requisitos levantados
- A equipe para e pensa um pouco sobre como vai modelar aquele domínio:
  - Quais classes serão criadas?
  - Quais os comportamentos existentes?
  - Como uma se relaciona com a outra?

Diagrama de Classes

- O grande objetivo da UML (*Unified Modelling Language*) é facilitar a comunicação entre a equipe de desenvolvimento, entre os *stakeholders*, e quaisquer outros interessados naquele software.

Diagrama de Classes

- Notação:

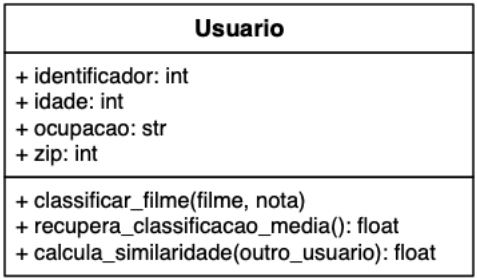


Diagrama de Classes

- Notação:

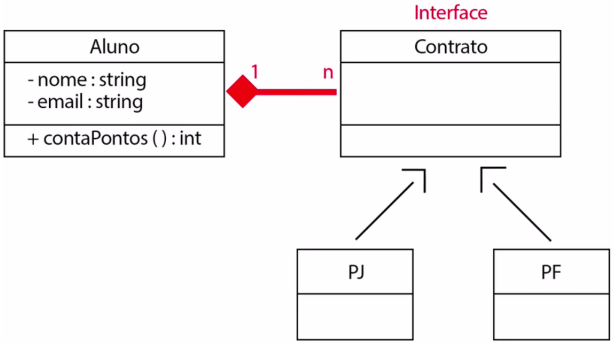
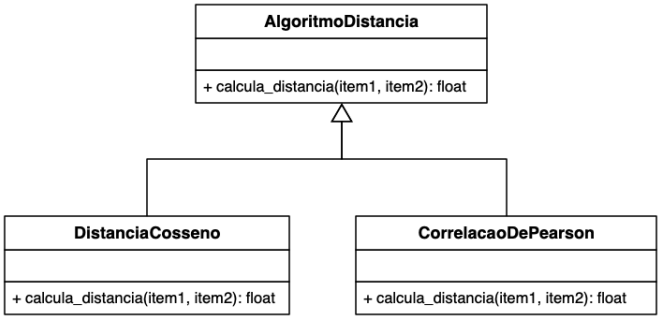


Diagrama de Classes

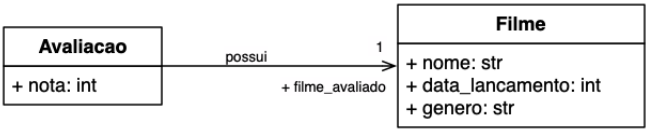
- **Atributos**:
  - Peculiaridades que costumam **variar de um objeto para outro**
  - Segunda divisão da classe e contêm **Nome e Tipo de dado**
  - Visibilidade **+** (**pública**), **#** (**protegida**) e **–** (**privada**)
- **Métodos**:
  - Atividade que um **objeto de uma classe pode executar**
  - Terceira divisão da classe e contêm assinatura do método (**visibilidade; nome; parâmetros** - contendo tipo ou não; e **tipo de retorno**
- Podem ser simplificada a **uma única divisão** contendo somente o nome da classe;

Diagrama de Classes - Herança



- Relação de **Generalização/Especialização**

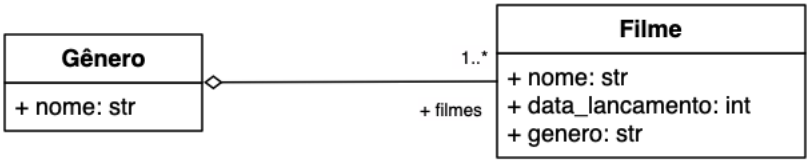
Diagrama de Classes - Associações



- **Navegabilidade**: determina que os objetos da classe para onde a seta aponta não têm conhecimento dos objetos aos quais estão associados na outra extremidade da associação;
- **Multiplicidade**: quantidade de instâncias presentes na classe oposta a seta;
  - 1, *, 1..*, 0..1, m..n
- **Papéis**: nome e visibilidade do atributo na classe oposta a seta;

Diagrama de Classes - Relacionamento *Todo-Parte*

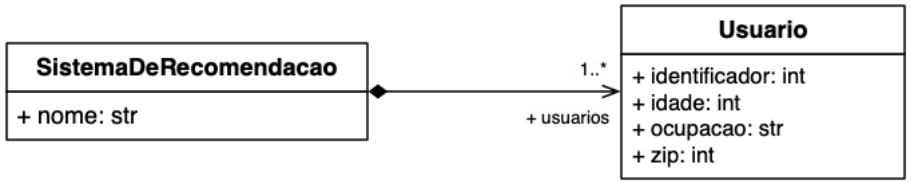
- **Agregação**:



- *Objetos-todo* que precisam ter suas informações complementadas pelos *objetos-parte*
- *Objetos-parte* continuam fazendo sentido mesmo após a destruição do *objeto-todo*

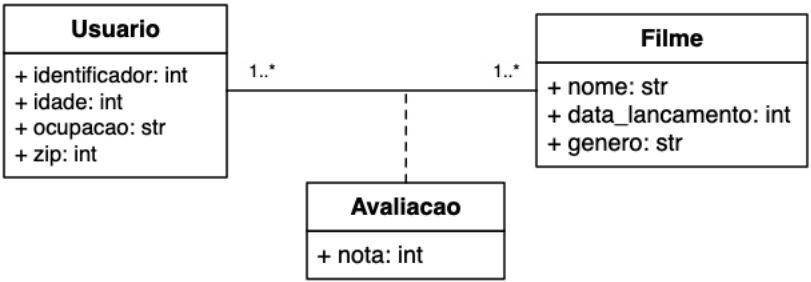
Diagrama de Classes - Relacionamento *Todo-Parte*

- **Composição**:



- *Objetos-parte* têm de estar **associados a um único *objeto-todo***
- **Não podem ser destruídos por um objeto diferente do *objeto-todo***;
- *Objetos-parte* **não fazem sentido após a destruição** do *objeto-todo*

Diagrama de Classes - Classes Associativas



- Atributos que **pertencem a associação**;
  - Não podem ser armazenados em **nenhuma das classes envolvidas**;

Diagrama de Classes - Dependência

- Identifica certo grau de dependência de um elemento em relação à outro
- Relacionamento fornecedor/cliente entre elementos do modelo

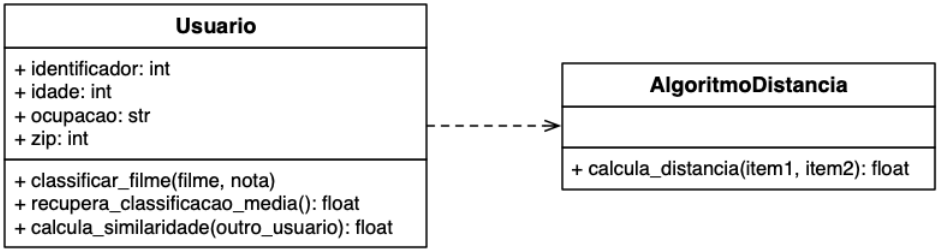


Diagrama de Classes - Exercício III

- Escreva o diagrama de classes para o seguinte sistema:
  - Um leilão tem produto, preço, data limite, dono, e uma lista de lances.
  - Um usuário tem nome e e-mail.
  - Um lance tem um usuário, um leilão e um valor.
  - Um job tem o comportamento de fechar leilões vencidos.

Diagrama de Classes - Resposta Exercício III

- O usuário é dono de um leilão, mas também pode fazer lances. A classe Lance representa o relacionamento entre leilão e usuário.

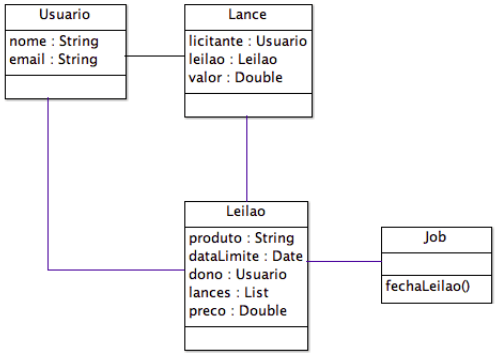


Diagrama de Classes - Exercício IV

- E se fossemos modelar nosso estudo de caso?
  - Uma trilha tem cursos
  - Um curso possui capítulos
  - Um capítulo possui exercícios
  - Um aluno se matricula no curso e resolve exercícios

Diagrama de Classes - Resposta Exercício IV

- Teremos o seguinte cenário:

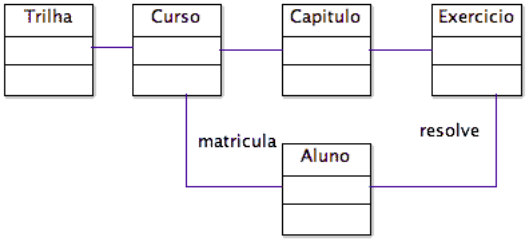


Diagrama de Sequência

- Esse tipo de diagrama irá representar o passo a passo do nosso sistema, quem e quando é chamado para a execução.
- Existe um vai-e-vem grande entre os diferentes sistemas então, vale a pena representar isso graficamente. Vamos mais uma vez representar nosso estudo de caso, imaginando o sistema de compra de cursos sendo direcionado ao site *PayPal*, preenchendo os dados e tendo como resposta o "ok" do pagamento e então, o sistema libera o curso.

Diagrama de Sequência

- Ações são sempre disparadas por um ator então, a primeira coisa que o aluno faz é escolher uma trilha para comprar no site.

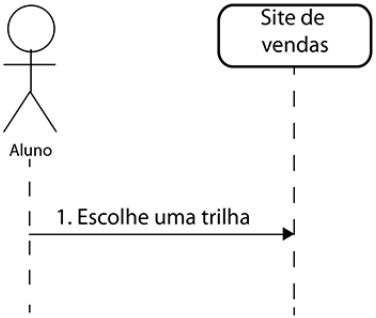


Diagrama de Sequência

- Ao receber essa informação, o site de vendas dispara uma ação no site do *Paypal*.

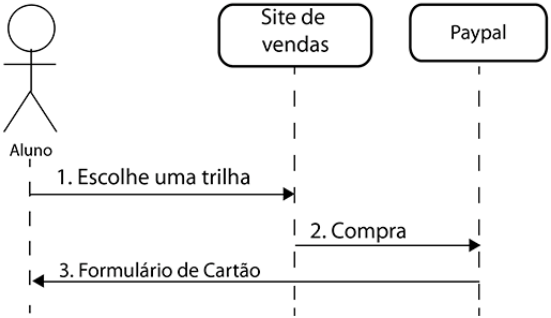


Diagrama de Sequência

- O aluno, por sua vez preenche o formulário. O *Paypal* então nos devolve a informação de "ok, aluno pagou"

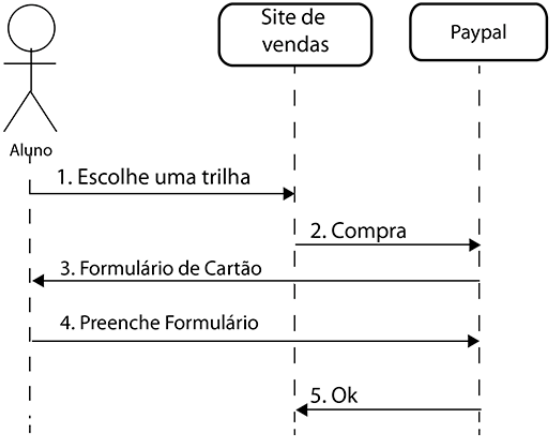


Diagrama de Sequência

- O site de vendas, por sua vez, dispara o sistema de matrículas, que libera a matrícula do usuário.

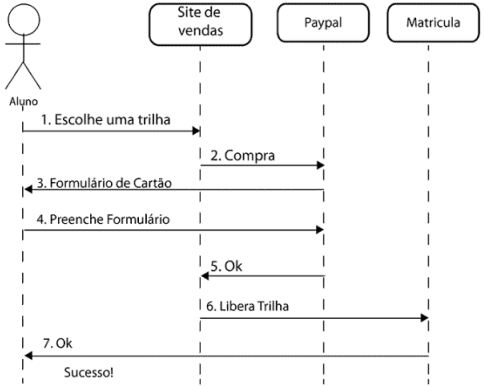


Diagrama de Sequência

- Podemos reparar então que não há segredo, usamos setas de um lado para o outro, explicando o que cada parte faz e em que sequência.

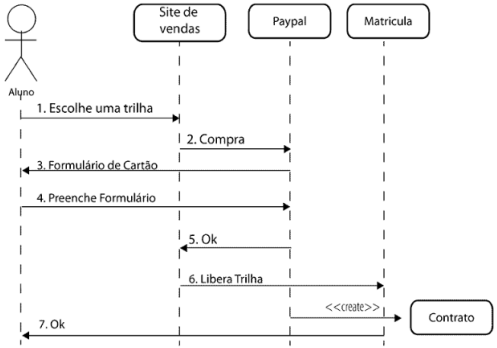


Diagrama de Sequência - Exercício V

- Tente entender o diagrama de sequência a seguir e descreva com suas palavras o que está acontecendo!

Diagrama de Sequência - Exercício V

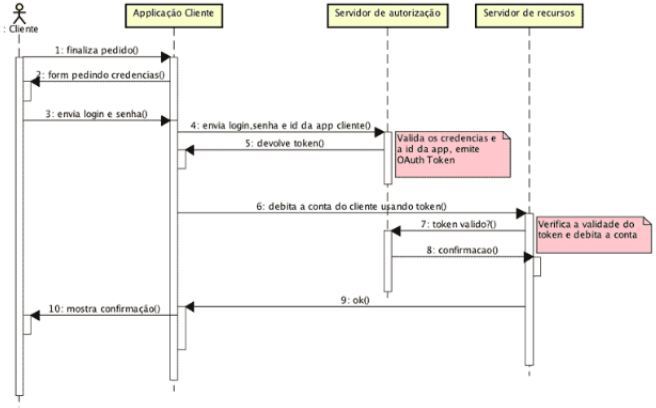


Diagrama de Sequência - Resposta
Exercício V

> *Temos um ator e 3 sistemas envolvidos nesse diagrama. O ator cliente deseja finalizar um pedido e pagar com a sua conta Paypal. Suas credenciais são solicitadas pela aplicação cliente que pode ser uma loja virtual. De posse das credenciais, essa aplicação se comunica com o servidor de autorização para validar os dados e receber um token. Esse token nada mais é do que um identificador único então, com o token a aplicação cliente pode acessar a conta do cliente (no servidor de recursos) para debitar a conta e no final recebe a confirmação do pedido.*

Diagrama de Sequência - Resposta Exercício V

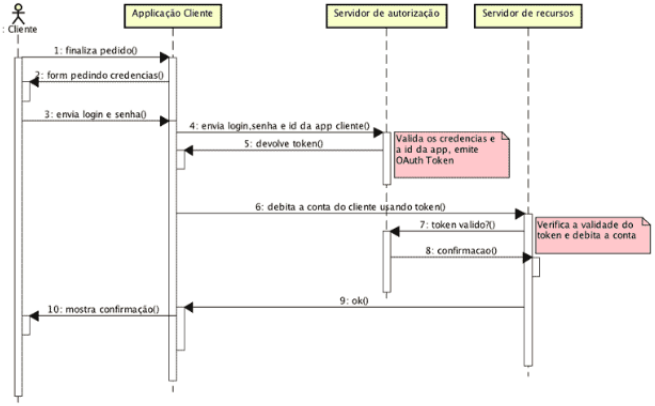


Diagrama de Atividades

- Muitas vezes precisamos representar as atividades de um ator/funcionalidade dentro do sistema, caminhos que ele pode tomar, etc.
- Um bom exemplo é o próprio aluno quando entra em um curso: ele pode assistir um capítulo ou ir para o fórum tirar dúvidas.
- Se ele assistiu um capítulo, ele verá um vídeo e responderá um exercício.
- Se ele teve dúvida, ele vai abrir a dúvida.
- Quando acabar, ele vai então dar *feedback* para um colega, ao mesmo tempo que ele o avalia.
- Então o processo termina.

Diagrama de Atividades

- A primeira notação que apresentamos é o círculo, que indica o início do processo então, por meio de setas, vamos indicando os caminhos que o aluno pode tomar.
- Depois de entrar em um curso, ele tem duas opções: ir para o fórum ou assistir um capítulo.

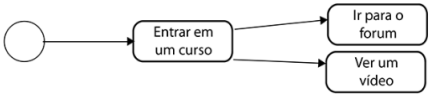


Diagrama de Atividades

- O aluno resolve um exercício porém, ele pode ter uma dúvida.
- Se ele tem uma dúvida, o caminho a ser seguido é um, caso contrário, o caminho é outro.

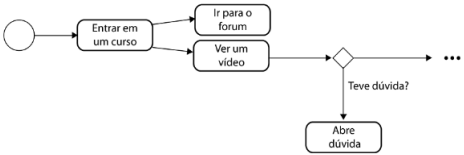


Diagrama de Atividades

- O aluno resolve um exercício porém, ele pode ter uma dúvida.
- Se ele tem uma dúvida, o caminho a ser seguido é um, caso contrário, o caminho é outro.

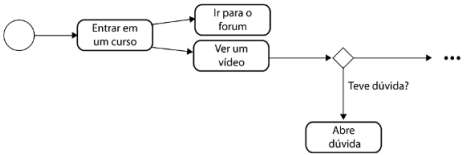


Diagrama de Atividades

- Em seguida, com ele abrindo ou não uma dúvida, o próximo passo é dar *feedback* e uma nota para o colega.
- As duas coisas devem acontecer ao mesmo tempo e para indicar simultaneidade, usamos uma "barra":

Diagrama de Atividades

- Depois disso o processo acaba. Representamos o fim, com uma bola pintada dentro.

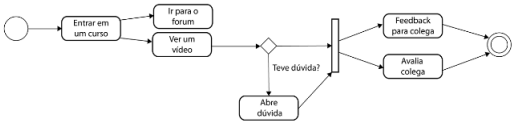
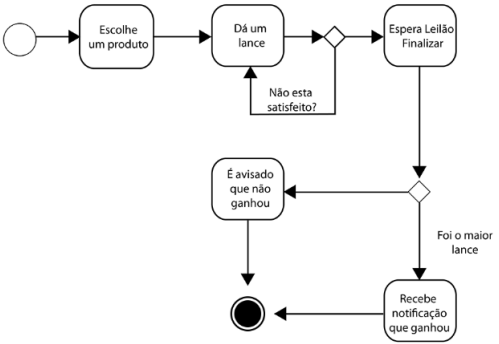


Diagrama de Atividades - Exercício V

- Faça um diagrama de atividades para o seguinte problema:
  - Usuário escolhe um produto;
  - Dá um lance;
  - Se não está satisfeito, dá um lance novamente;
  - Espera leilão encerrar
  - Se o maior lance foi o dele, ele recebe notificação que ganhou;
  - Caso contrário, é avisado que ele não foi o ganhador.

Diagrama de Atividades - Resposta Exercício VI



Dúvidas?