

Projeto Final		
INF331 – Componentização e Reúso de Software Instituto de Computação Universidade Estadual de Campinas		2022 Professor: André Santanchè

Elabore um detalhamento da modelagem baseada em componentes para o tema “Brechó Online”. Para os diagramas deste projeto, está sendo disponibilizado um arquivo de modelos no endereço: https://docs.google.com/presentation/d/1SyJ6rxg-RcKim7yOVkoQNsDbPGeuE_n2w9jh7vyvHR0/edit?usp=sharing

As equipes podem utilizar esses modelos como base e/ou usar qualquer software de elaboração de diagramas, contanto que ele apresente informações equivalentes àquelas aqui solicitadas.

O projeto deve ser elaborado de forma top-down, partindo dos componentes de maior granularidade e indo em direção aos componentes de menor granularidade, conforme detalhado a seguir.

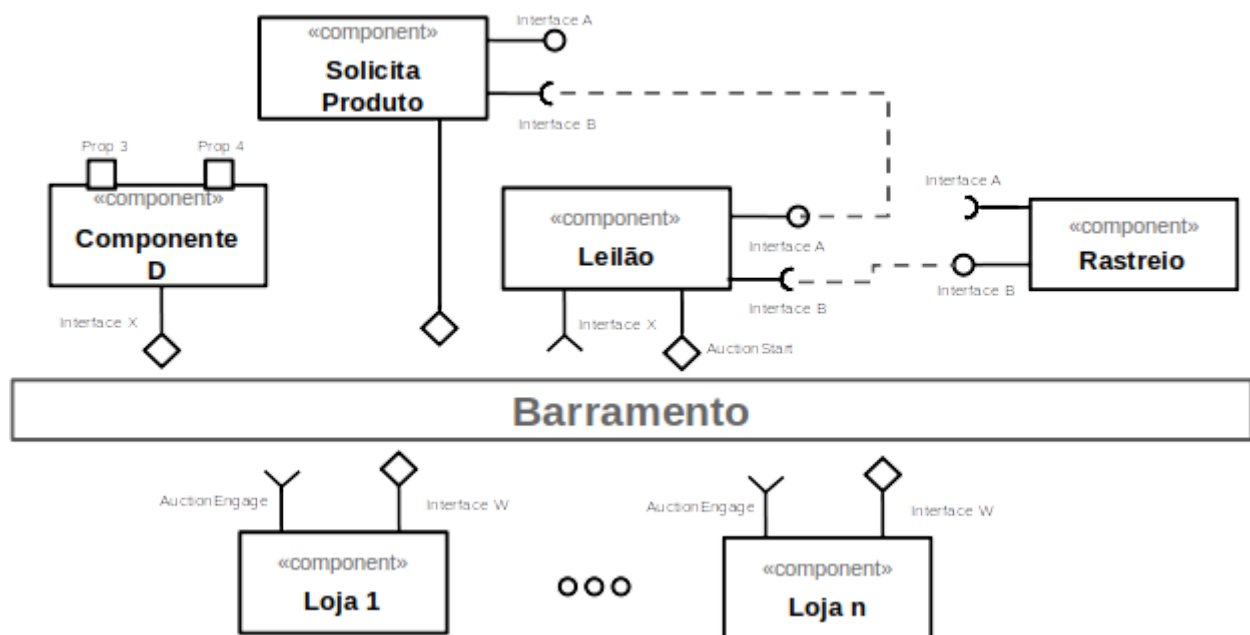
Nível 1

Neste nível de granularidade maior, devem ser considerados grandes componentes que representam módulos do sistema que gerencia o sistema de Brechó. Esses componentes são serviços que se comunicam por um barramento.

Devem ser consideradas todas as funcionalidades do sistema descritas no documento de “Brechó Online”. Deve ser dado destaque na funcionalidade de **logística de entrega**.

- Cada consumidor, vendedor e parceiro deve ser tratado como um componente se comunicando pelo barramento.
- O sistema deve funcionar para um número variável de consumidores, vendedores e parceiros, que podem aderir dinamicamente ao marketplace, sem que sejam necessárias modificações na codificação do sistema. Por essa razão, na figura exemplo a seguir elas são representadas como um conjunto genérico de vendedores (lojas);
- O sistema de **logística de entrega** deve funcionar na forma de uma coreografia entre vendedores e parceiros que façam entregas. Cada entrega deve envolver uma nova negociação associada entre o vendedor e possíveis entregadores (que pode ser variável). Será valorizada a criatividade nesse processo de negociação. Esta arquitetura deve constar com um serviço de aprendizagem de máquina para tomar as melhores decisões referentes à logística.

Sugere-se a representação a seguir para esse nível de granularidade:





Os componentes nesse nível de granularidade deverão se comunicar através de publish/subscribe. Descreva a forma como os componentes interagem em alto nível conforme o exemplo:

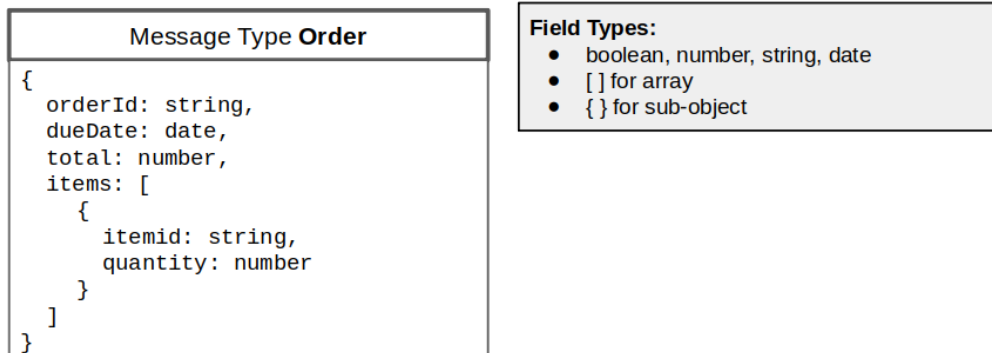
- O componente **Leilão** inicia o leilão publicando no barramento a mensagem de tópico **"auction/{auctionId}/start"** através da interface **AuctionStart**, iniciando um leilão.
- Os componentes **Loja** assinam no barramento mensagens de tópico **"auction/+start"** através da interface **AuctionEngage**. Quando recebe uma mensagem...

Há um exemplo mais detalhado na próxima seção que também pode ser adotado como base. Note que é usada uma estrutura hierárquica de tópicos inspirada no MQTT, conforme está descrito no documento: <https://harena-lab.github.io/harena-docs/dccs/tutorial/>. Recomenda-se o uso deste padrão, inclusive com coringas (wildcards).

Deve ser descrita pelo menos duas interações: (i) um processo de compra completo envolvendo todos os componentes do marketplace; (ii) o **Processo de Lançamento e Distribuição de Ofertas**.

Para cada interface (e os respectivos tópico/mensagem) deve ser detalhada conforme o modelo disponível no template:

<div>Interface SolicitaEntrega</div> <div>Type: sink Topic: pedido/+entrega Message type: Order</div>	<div>Interface Types</div> <div><div> source</div><div> sink</div></div>
<div>Interface AuctionStart</div> <div>Type: source Topic: auction/{auctionId}/start Message type: Auction</div>	
<div>Interface AuctionEngage</div> <div>Type: sink Topic: auction/+start Message type: Auction</div>	



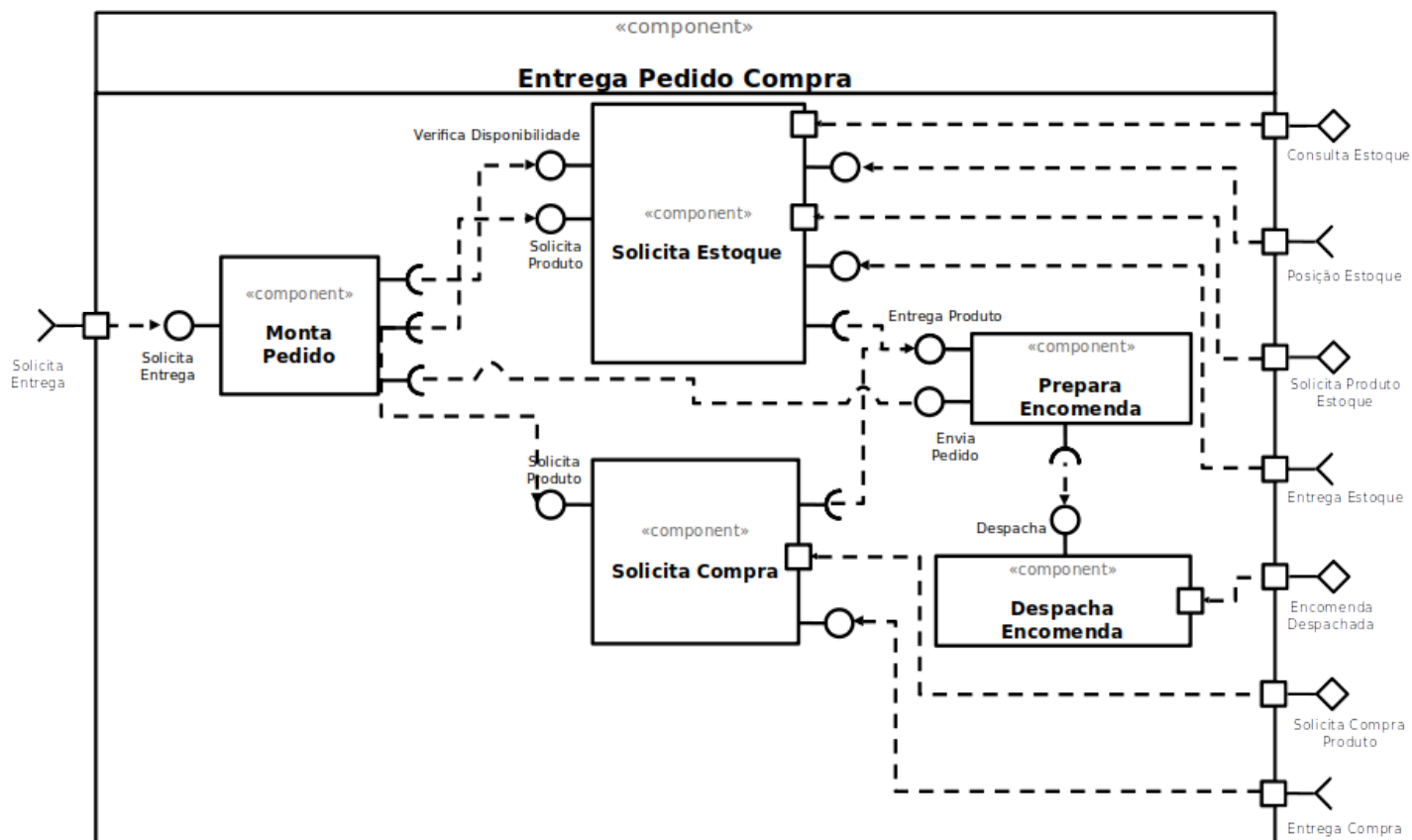
Nesse nível de detalhe a equipe decide se fará a separação MVC ou se prefere deixar para o próximo nível. Se houver separação MVC neste nível, pode haver comunicação síncrona com interface provida/requerida entre componentes para efeito do MVC.

Nível 2

Considerando que os componentes da etapa anterior são de maior granularidade e que se comunicam através de um barramento de mensagens, cada equipe deve detalhar dois desses componentes: um componente a escolha da equipe e o componente que usa aprendizagem de máquina para a tomada de decisões referente à logística.

Para o componente de escolha da equipe, se a equipe optou pela separação do MVC no diagrama anterior, deve ser escolhido pelo menos um componente Controller e os Model/View associados para detalhamento. Se a equipe optou por realizar a separação nesse estágio, o componente escolhido deve ser separado em componentes Model/View/Controller interligados.

A estrutura interna do componente conterá subcomponentes que se comunicam através de conexões, como ilustra o exemplo na figura a seguir:



Deve ser elaborado um diagrama UML seguindo o modelo acima, com eventos especificados seguindo o modelo CORBA Component Model (CCM).

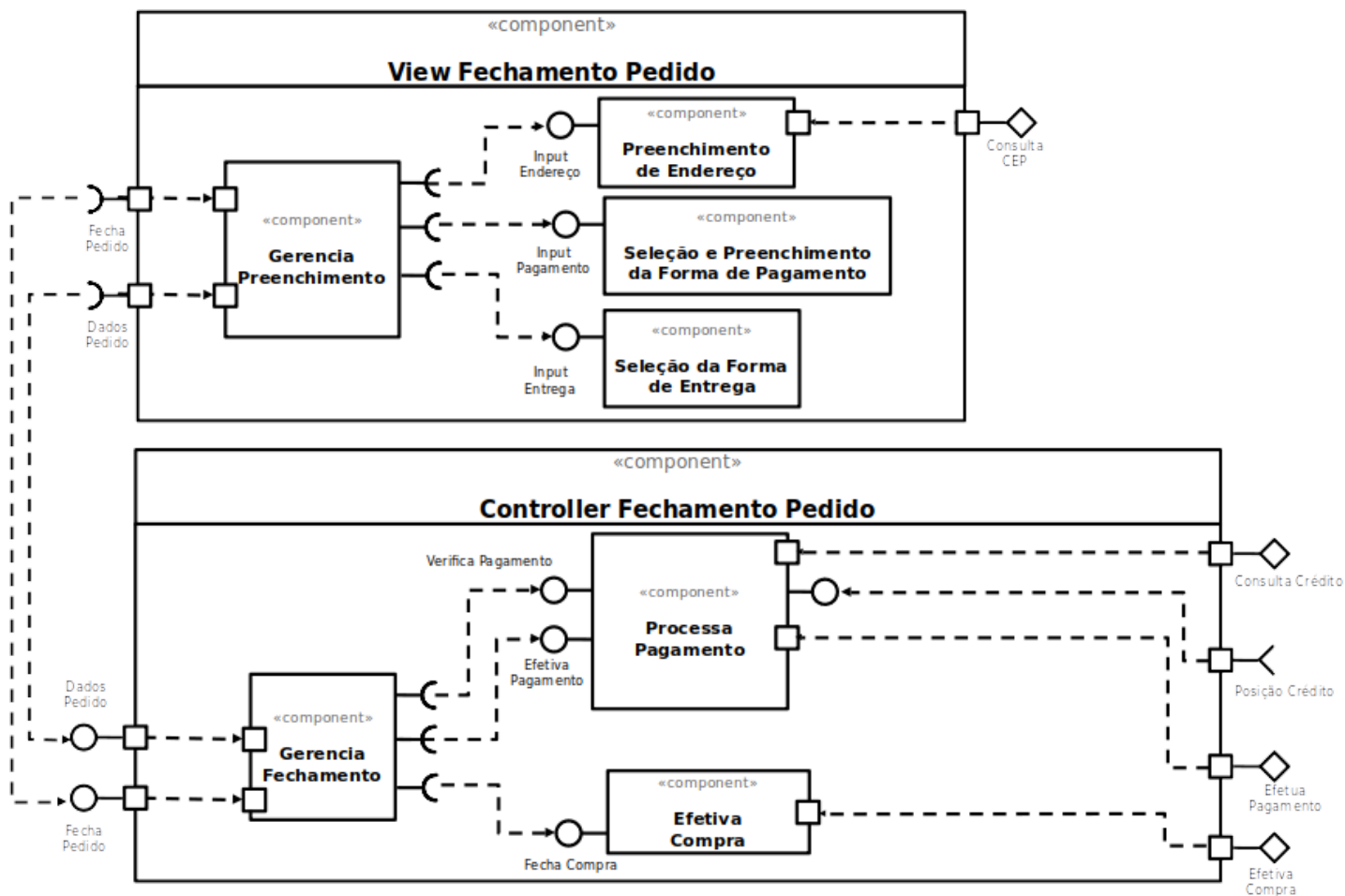
O diagrama deve ser acompanhado de uma breve explicação da interação dos subcomponentes internos para fazer operar o componente externo, conforme o exemplo de explicação do componente de **Entrega de Pedido de Compra**:

- O componente **Entrega Pedido Compra** assina no barramento mensagens de tópico “**pedido/+/entrega**” através da interface **Solicita Entrega**.
 - Ao receber uma mensagem de tópico “**pedido/+/entrega**”, dispara o início da entrega de um conjunto de produtos.
- Internamente este evento é atendido por uma interface provida do componente **Monta Pedido**, que é responsável por montar o pedido para entrega.
- Esse componente verifica a disponibilidade dos produtos em estoque, acionando o componente **Solicita Estoque** através da interface **Verifica Disponibilidade**.
- Os produtos disponíveis são solicitados para o estoque (componente **Solicita Estoque**) e para os demais é solicitada a compra (componente **Solicita Compra**). Ambas as operações são acionadas através da interface **Solicita Produto**.
- O componente **Monta Pedido** notifica o componente **Prepara Encomenda** que uma encomenda está a caminho através da interface **Envia Pedido**.
- O componente **Prepara Encomenda** aguarda a entrega dos produtos de estoque e comprados. Ele é notificado através da interface **Entrega Produto**.
- Uma vez que todos os produtos estejam disponíveis, ele ativa o despacho acionando o componente **Despacha Encomenda** através da interface **Despacha**.
- Uma vez que a encomenda seja despachada, o componente **Despacha Encomenda** publica no barramento uma mensagem de tópico “**pedido/<número>/despacha**” através da interface **Encomenda Despachada**.
- Os componentes **Solicita Estoque** e **Solicita Compra** se comunicam com componentes externos pelo barramento:
 - Para consultar o estoque, o componente **Solicita Estoque** publica no barramento uma mensagem de tópico “**produto/<id>/estoque/consulta**” através da interface **Consulta Estoque** e assina mensagens de tópico “**produto/<id>/estoque/status**” através da interface **Posição Estoque** que retorna a disponibilidade do produto.
 - Para solicitar o produto em estoque, o componente **Solicita Estoque** publica no barramento uma mensagem de tópico “**produto/<id>/estoque/solicita**” através da interface **Solicita Produto Estoque** e assina mensagens de tópico “**produto/<id>/estoque/entregue**” através da interface **Entrega Estoque** que confirma a entrega da solicitação.
 - Para solicitar a compra de produtos, o componente **Solicita Compra** publica no barramento uma mensagem de tópico “**produto/<id>/compra/solicita**” através da interface **Solicita Compra Produto** e assina mensagens de tópico “**produto/<id>/compra/entregue**” através da interface **Entrega Compra** que confirma a entrega da solicitação.

Note que eventos recebidos do componente maior podem ser convertidos em acionamento de interfaces providas em componentes internos; a produção de eventos externos não é mapeada a uma interface provida/requerida, mas é associada a um componente que a produz, conforme mostra o diagrama.

Cada equipe deve elaborar o projeto em UML envolvendo os componentes e a conexão entre componentes. Tais componentes serão detalhados no diagrama de componentes alinhado com o diagrama de interfaces.

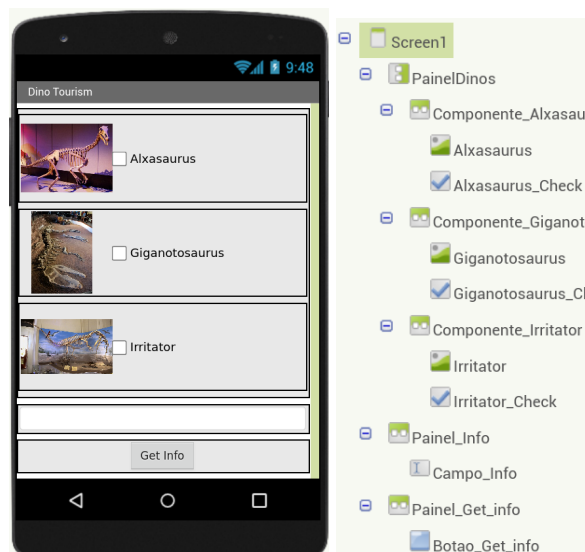
Considere que os componentes Model e View são independentes do Controller, ou seja, não são subcomponentes do Controller, mas se conectam com esse. A seguir é detalhado um diagrama sugerido para o detalhamento. Nesse caso, é detalhado o Controller e o View, mas o Model também deveria ser detalhado como um terceiro grande componente.



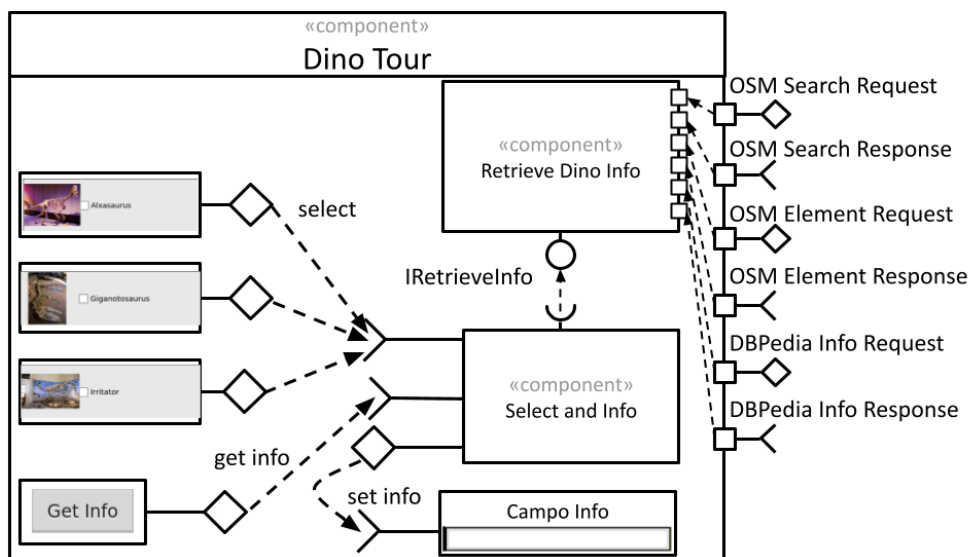
Nível 3

Considere que há um componente no Nível 2 cumpra o papel de interagir com o usuário através da GUI. Um exemplo seria: uma interface que apresenta os dados de um produto selecionado e permita ao usuário disparar uma compra direta, sem passar pelo carrinho. A interface permite que o usuário, em uma mesma tela, selecione a quantidade que deseja comprar; selecione a forma de pagamento; selecione o número de parcelas (se for no cartão de crédito) e dispare a compra.

Considerando que está sendo usado o estilo arquitetural MVC, apresente o protótipo da interface da tela descrita. Faça um protótipo de tela no MIT App Inventor, represente apenas os elementos de interface na tela. Apresente um screenshot da tela e a hierarquia de componentes como está ilustrado a seguir:



A partir desse protótipo, construa um diagrama representando componentes gráficos e outros componentes que devem participar da interação através da interface, conforme exemplo a seguir. Note que o diagrama apresenta os eventos que componentes de interação podem disparar. Para cada evento disparado, indica qual o componente que recebe o evento. Esse componente pode gerar outros eventos internamente, ou pode despachá-los para uma interface externa do componente.



É importante ressaltar que o protótipo é apenas visual, não precisa funcionar. Tal como nesse exemplo, podem ser acrescentados componentes que não foram inseridos no MIT App Inventor.

O diagrama deve ser acompanhado de uma breve explicação da interação dos componentes internos, seguindo uma abordagem equivalente à do Nível 2.

Entrega

A apresentação deve seguir a template: <https://github.com/santanche/component2learn/tree/master/templates/2021/project>

Até o dia 09/09 as equipes deverão finalizar o projeto. A submissão será em modalidade equivalente à submissão de labs, ou seja, cada equipe um fork do projeto lab2learn e criará uma pasta no diretório <https://github.com/santanche/component2learn/tree/master/project/2021/solucoes>. O nome da pasta tem o prefixo "equipe" seguido do número da equipe, exemplo: **equipe01**. A submissão será na forma de pull request para o lab2learn. As equipes poderão levar uma prévia dos modelos no dia 03/09 para debate em sala de aula.