FIAP

MBA+ 2015

DESENVOLVIMENTO DE APLICAÇÕES JAVA – SOA E INTERNET DAS COISAS

FIAP

MBA+ 2015

Desenvolvimento de Projeto Corporativo JEE

Daniel Lemeszenski, MSc

FIAP

10 anos de experiência em arquitetura e desenvolvimento de soluções corporativas em TI para grandes empresas de telecomunicações e internet.

Formação Acadêmica:

- ✓ Bacharel em Ciências da Computação na UFSCar;
- ✓ Mestre em ciências – Escola Politécnica da USP – Engenharia de computação.

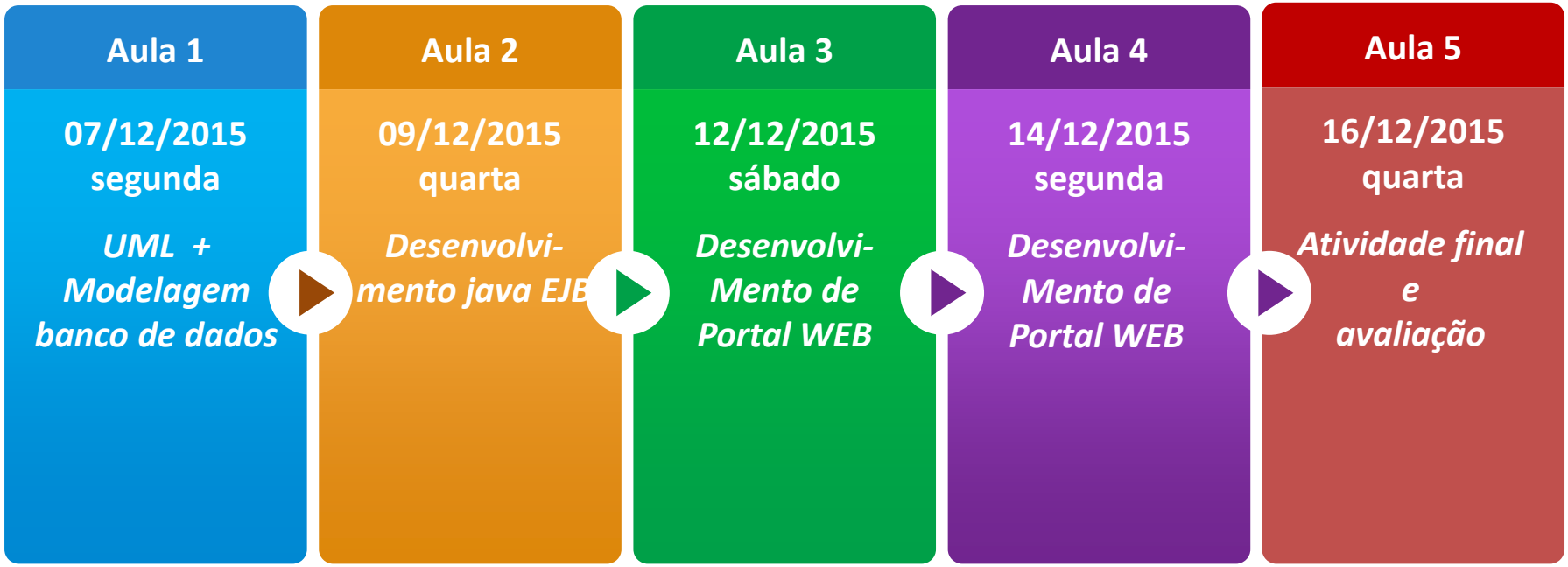
Experiência Profissional:

- ✓ Coordenador da equipe suporte SOA na Nextel;
- ✓ Líder técnico de grandes projetos de desenvolvimento de sistemas na Claro, Vivo, HP, UOL e Nextel;
- ✓ Vivência internacional na Ericsson Alemanha desenvolvendo soluções de CRM e BPM utilizadas em operadoras de celular ao redor do mundo;
- ✓ Autor e revisor de artigos na área de tecnologias interativas pela Escola Politécnica da USP.

https://www.linkedin.com/in/daniellemeszenski

LinkedIn

Nossos encontros...



Objetivo da disciplina

FIAP

- ✓ Dado o estudo de caso, os grupos deverão modelar e desenvolver um sistema JEE com os principais componentes dessa plataforma;

- ✓ O intuito dessa disciplina é consolidar os conceitos técnicos apresentados no decorrer do curso sobre a arquitetura da tecnologia Java através do desenvolvimento de um exemplo prático de uma aplicação de e-commerce;

FIAP e-commerce

Caso de Uso

Oportunidade

Hoje a empresa **FIAP** possui um processo de vendas manual e engessado, e gostaríamos de inovar seu modelo de negócio, o que o traz para uma nova necessidade:

- **Unificar o processo de vendas de produtos**, pois iremos lançar uma plataforma de E-commerce utilizando um portal WEB.

Com essa iniciativa, surgiu a preocupação e a possibilidade da empresa sofrer grandes perdas financeiras, pois atualmente não se conhece o ticket médio das vendas, ou se existe algum valor médio que deveria se aprovado por uma auditoria ou área financeira antes da liberação de crédito.

Comportamento

E-Commerce

A navegação pelo site e a interação com o “carrinho de compras” pode ser feita de forma anônima ou através da identificação do cliente. As páginas Web de listagem de produtos (vitrines) mostram os detalhes técnicos dos produtos, assim como a quantidade de itens que se deseja adicionar ao carrinho (valor padrão das páginas é de 1 item), e podem-se fazer buscas de produtos pelo nome ou marca do produto. As operações que o “carrinho de compras” oferece são:

- ✓ acrescentar os itens,
- ✓ alterar as quantidades dos itens dentro do carrinho e
- ✓ remover itens do carrinho.

Para finalizar a compra no site, o cliente precisa estar devidamente identificado e “logado”, e informar um endereço para entrega, que pode ser o mesmo endereço de seu cadastro.

Comportamento *(continuação)*

E-Commerce

Para o cadastro do cliente são necessárias e obrigatórias as seguintes informações:

- **E-mail**
- **Senha**
- **Endereço com CEP**
- **CPF**
- **Nome**
- **Nome da mãe**
- **Telefone para contato**
- **Profissão (CBO)**
- **Salário**
- **Data de nascimento**

Deve ser gerado um número de pedido para fazer a rastreabilidade da compra a partir do cliente, e também deve ser exibida uma mensagem ao cliente informando que seu pedido foi enviado para análise e que o mesmo receberá a confirmação e o boleto por e-mail. Um recurso que deve ser adicionado ao E-Commerce é a recuperação de senha através da funcionalidade disponibilizada: ***"esqueci minha senha"***. Lembrando que por motivos de segurança, não podemos fornecer a senha antiga, obrigando o reset.

Comportamento *(continuação)*

E-Commerce

Ao finalizar a venda no site é enviado o pedido composto pelo cliente e o “carrinho de compras” para as validações, a geração do boleto e o envio de confirmação do pedido do cliente. Para esse processo, deve ser feita uma consulta na base de dados de clientes para averiguar se o cliente já tem pedidos no mesmo mês em aberto e sem a realização do pagamento, o que pode caracterizar inadimplência. Também deve-se verificar se os itens no carrinho de compras não são iguais aos pedidos realizados por um período, previamente cadastrado, que será configurado pela área de Canais Digitais. Com as informações de todas as compras feitas no mês pelo mesmo cliente, deve, também, ser feita uma análise do ticket médio, que será configurado pela área de Canais Digitais, mas nesse caso, se ultrapassar esse valor no período, deve ser encaminhada para a área de Análise de Crédito para uma solicitação de análise manual.

Comportamento *(continuação)*

E-Commerce

Esses pedidos de compra podem ser aceitos e rejeitados pelos seguintes motivos (tabela de decisão):

1. **Primeira compra - cliente novo não está na base de clientes (em conjunto com outra negativa);**
2. **Cliente está na *blacklist*;**
3. **Tem dívida com a empresa;**
4. **Tem um parcelamento anterior vencido;**
5. **Tem dívida contestada no Serasa**
6. **A região de entrega é atendida pela Logística;**

Caso seja aprovada a análise, ou caso não atinja os valores previamente definidos pela área de Canais Digitais, devem ser gerado um boleto com o valor total da compra, uma nota fiscal com cada item, suas quantidades, seus valores unitários discriminados separadamente, e com todos os dados do cliente que realizou a compra, além é claro, dos impostos calculados. De tempos em tempos deve-se gerar extratos de clientes (débitos e créditos) para futuras consultas pelos funcionários de Canais Digitais.

Comportamento *(continuação)*

FIAP

Controle de Estoque

A aplicação deve controlar o estoque de produtos por Região. Caso o produto esteja indisponível, na região selecionada pelo cliente, deve ser exibido um aviso de produto indisponível (não deve ser exibido o preço neste caso) na vitrine de produtos (ou esvaziar o carrinho).

Na tela de finalização do pedido (checkout) deve ser feita a baixa do produto no estoque daquela região. Caso o cliente não pague o boleto, ou o crédito seja cancelado o produto deve voltar para o estoque em 24horas .

Validação dos Preços

Ao clicar em finalizar pedido, o sistema deve validar se o preço do produto no carrinho é igual ao preço configurado na base de dados com intuito de evitar fraudes, ou em caso de atualização do preço, o carrinho deve ser esvaziado automaticamente informando o cliente que houve uma atualização nos preços enquanto ele navegava.

Relatórios Estratégicos

Consultas

- ✓ Quantidade de pedidos que foram rejeitados por dia/semana/mês
- ✓ Valor médio de compra por pedido por dia/semana/mês
- ✓ Top 3 dos motivos de rejeição de crédito
- ✓ Top 5 dos itens mais vendidos por dia/semana/mês
- ✓ Quantidade de pedidos por região (estado);
- ✓ Ticket médio do pedido por região (estado);

Atividades em grupo

1. Regras de Negócios (proposta inicial):

Lista dos principais Processos de Negócios:

1. **Cadastro de Cliente**
2. **Cadastro de Produto**
3. **Controle de Estoque**
4. **Liberação de Crédito**
5. **Tela de checkout (finalizar pedido)**
6. **Consulta de Pedidos do cliente**
7. **Auditoria das Compras**
8. **Cancelamento de Pedido por falta de pagamento**

Atividades em grupo

2. Complemento – Regras de Negócios (proposta inicial)

A tabela de decisão proposta abaixo está incompleta, por favor desenvolva um modelo de um serviço denominado "MotorCredito" que calcule essas regras de forma dinâmica, e caso seja necessário realizar alguma alteração, a mesma seja feita em tempo de execução sem necessidade de um novo *deployment*.

Condição	R1	R2	R2	R3	R4	...
Valor do produto	*	<500	>=500	>=500	>=500	
Cliente da base	*	sim	sim	Sim	Sim	
Está na BlackList	Sim	não	Não	Não	Não	
Tem dívida com a empresa	*	*	Sim	*	Não	
Tem o nome no SPC	*	*	*	Sim	Não	
...						
Resultado	rejeitado	aprovado	rejeitado	rejeitado	aprovado	...

Atividades em grupo

4. MODELO DE DADOS

Defina um Modelo de Dados comum para os Serviços de Negócios - EJB (desenhe um **Diagrama de Classes ou Dicionário de Dados**) para o Estudo de Caso.

Principais Entidades de Dados:

- ✓ **Cliente**
- ✓ **Pedido**
- ✓ **Cobrança**
- ✓ **Entrega**
- ✓ **Políticas de crédito**
- ✓ **Boleto**
- ✓ **Nota Fiscal**
- ✓ **Produto**
- ✓ **Estoque**
- ✓ **Rastreabilidade**
- ✓ **Carrinho de compras**

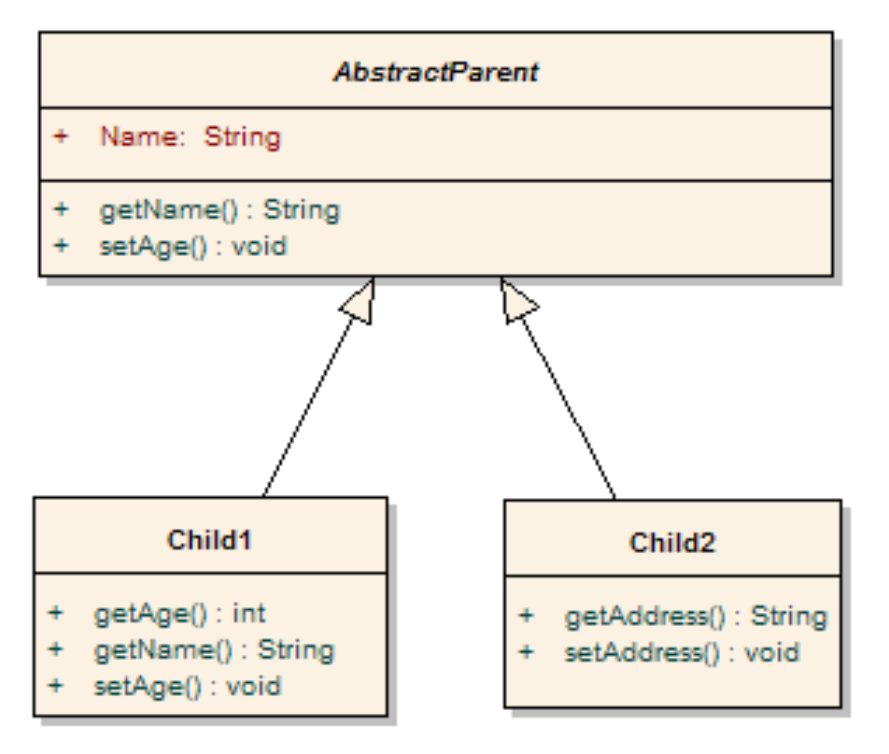
Aula 1

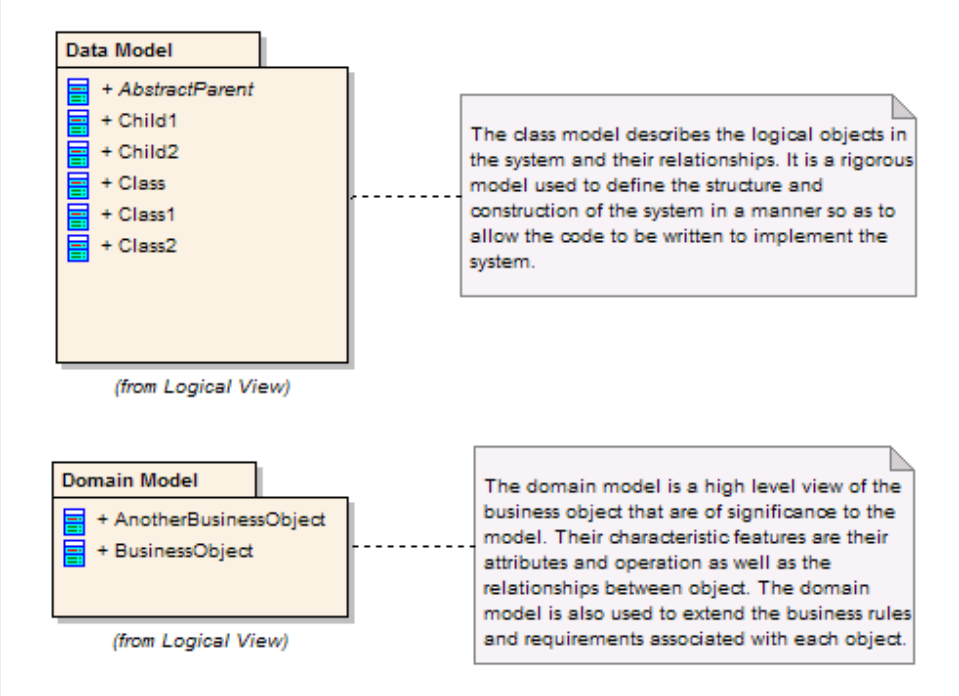
1. Elaboração de diagramas UML:
   - Diagramas de classes;
   - Diagramas de sequência;
   - Diagramas de pacotes;

2. Modelagem das entidades levantadas e desenvolvimento das tabelas em banco de dados;

UML - Modelagem

- Com as informações do estudo de caso elaborar os diagramas de classes e pacotes

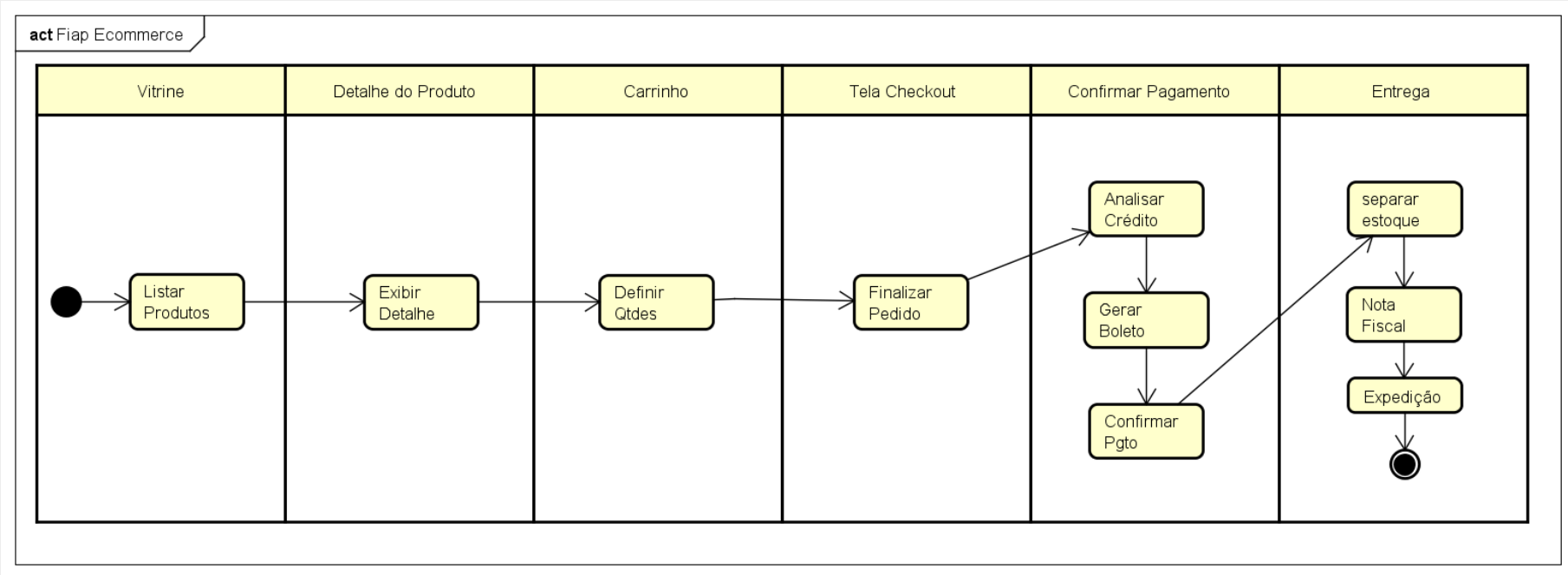




http://www.uml.org/

UML - Modelagem

- Diagrama de atividades – Passos de uma compra



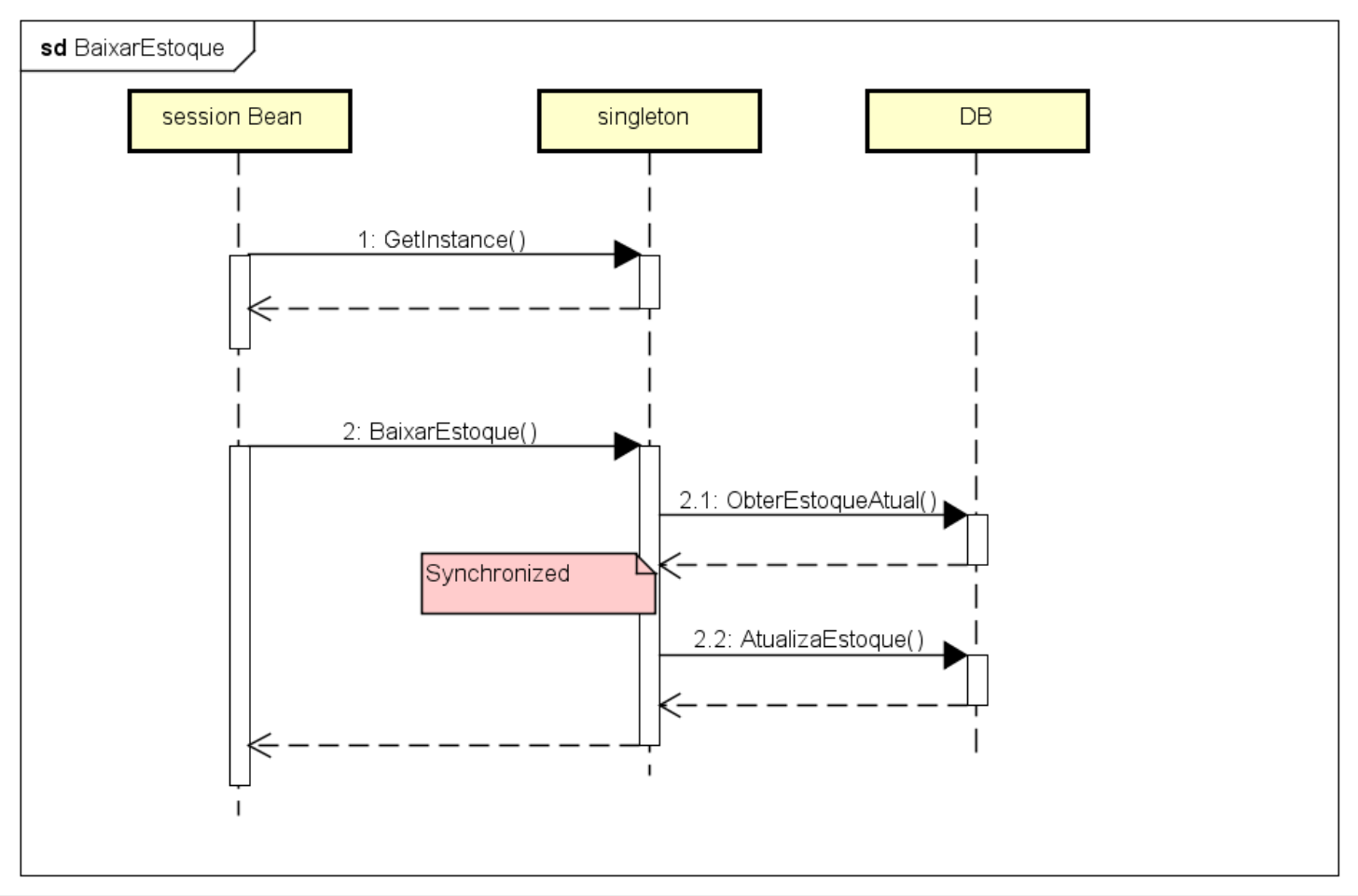
Exemplo: Controle de Estoque

Desafio: Evitar problemas de sincronismo quando uma ou mais sessões solicitar baixa de estoque.

Possível solução: validar estoque em diversos passos da compra e criar método *singleton* e *synchronized* para manipular o estoque.

UML - Modelagem

- Exemplo: Diagrama de sequência BaixarEstoque



http://www.sparxsystems.com/resources/uml2_tutorial/uml2_sequencediagram.html

FIAP

- Desenvolvimento de classes de negócio utilizando **EJB e JPA** para**:**
  1. manipular dados das entidades envolvidas (cliente, produtos, carrinho de compras etc.);
  2. operar as regras do motor de crédito;
  3. "*login*" do usuário.

DESAFIO:
Utilizar cache nos métodos utilizados pela tela de vitrine e detalhes do produto com intuito de melhorar a experiência do usuário e diminuir a carga nos servidores.

HIBERNATE 3 [Configuração]

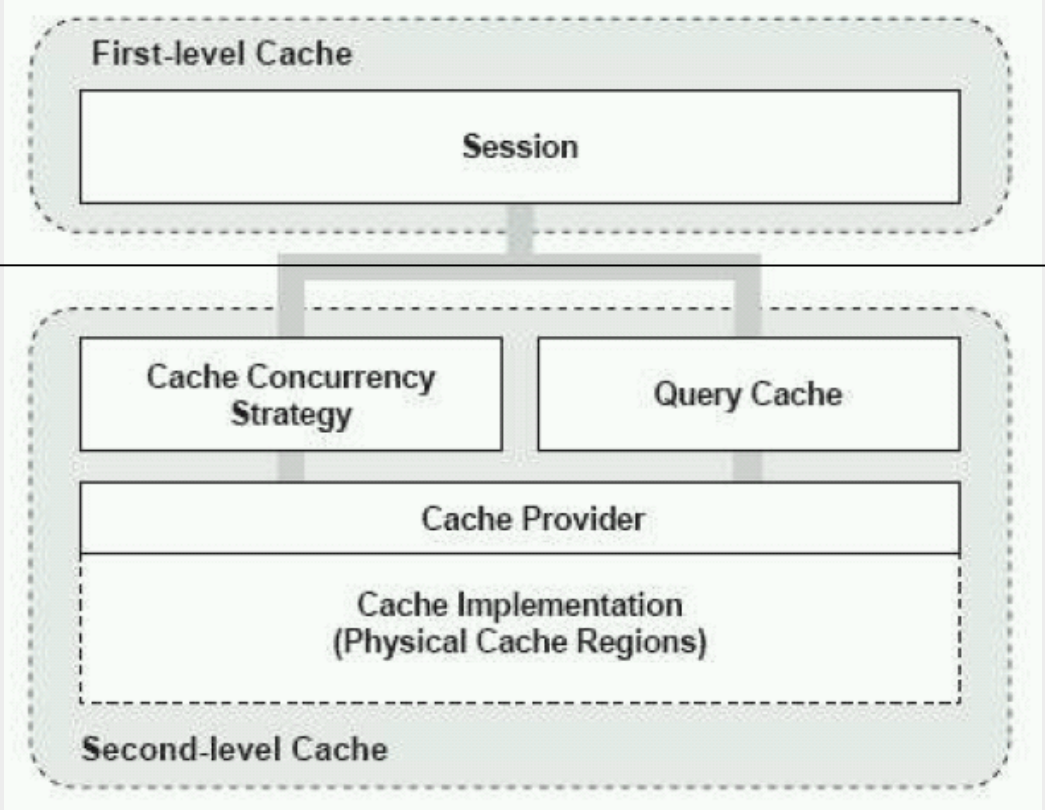
FIAP

- Caching
  - Os frameworks de mapeamento objeto relacional são ideais para uso de Caching.

- Usos do cache:
  - Busca pela chave primária
  - Associações com Lazy Loading
  - Queries

HIBERNATE 3 [Configuração]

Níveis de cache



Hibernate: Cache de 1º Nível - Session Cache

FIAP

- Sempre que é executado o método get ou load o resultado é armazenado no cache da sessão (em memória);
- Boa prática (fechar a sessão após o uso);
- session.evict(n) limpa o cache na sessão;
- O cache de uma sessão não interfere em outra sessão.

Hibernate: Cache de 2º. Nível - objeto Query

FIAP

- É utilizado diretamente através do objeto Query:

```
Vitrine = (Vitrine) session.createQuery("from Vitrine where id =
1").setCacheable(true).uniqueResult();
```

Classe: org.hibernate.cache.StandardQueryCache

- É necessário configurar o comportamento do cache no arquivo de configuração do hibernate.

Hibernate: Cache de 2º. Nível ehcache

- Ehcache

Incluir ehcache.jar no classpah

Configuração no arquivo ehcache.xml

```
<cache name="model.Vitrine"
      maxElementsInMemory="100"
      eternal="false"
      timeToIdleSeconds="300"
      timeToLiveSeconds="600"
      overflowToDisk="false"
/>
```

EJB – Enterprise Java Beans

Session Beans
– Modelam processos de negócio. São ações, verbos.
– Fazem coisas: acessam um banco de dados, fazem cálculos
– Podem manter ou não manter estado persistente
– Ex: processar informação, comprar produto, validar cartão

Entity Beans
– Modelam dados de negócios. São coisas, substantivos
– Representam informações em bancos de dados
– Mantêm estado persistente
– Ex: um produto, um empregado, um pedido

Message Driven Beans
Desenvolvimento de Aplicações Java - SOA 67 EJB 3.0 – Enterprise Java Beans
– Modelam processos assíncronos. Respondem a eventos
– Agem somente quando’

JOB de devolução de estoque

- Job de controle de pagamento/devolução de estoque;
- Após 24horas, se o pagamento não for confirmado, devolver o item para o estoque.

EJB 3.1 TimeService

```
1.  public void agenda() {
2.    // cada segunda e quarta as 8:30
3.    ScheduleExpression expression = new ScheduleExpression();
4.    expression.dayOfWeek("Mon,Wed");
5.    expression.hour("8");
6.    expression.minute("30");
7.    this.timerService.createCalendarTimer(expression);
8.    System.out.println("Agendado: " + expression);
9.  }
```

EJB 3.1 TimeService

```
1.  //dentro do session bean
2.  @Schedule(dayOfWeek="Mon,Wed", hour="8", minute="30")
3.  void agendado() {
4.    System.out.println("agendado pela anotacao @Schedule");
5.  }
```

Se não for persistido, o job é perdido no restart do servidor

```
1.  //dentro do session bean
2.  @Schedule(dayOfWeek="Mon,Wed", hour="8", minute="30", persistent=false)
3.  void agendado() {
4.    System.out.println("agendado pela anotacao @Schedule");
5.  }
```

FIAP

Exemplo de expressão:

```
@SuppressWarnings("unused")
@Schedule(second="*/30", minute="*", hour="8-23", dayOfWeek="Mon-Fri",
    dayOfMonth="*", month="*", year="*", info="MyTimer", persistent=true )
```

Especificando intervalos

Para a expressão x/y, x representa o momento início e y o intervalo para a próxima execução;

minute="*/10“ (significa que o método é executado a cada 10 minutos).

- Desenvolvimento das páginas JSF da loja virtual visão cliente:
    - Página da vitrine com categorias de produtos;
    - Página do detalhe do produto;

DESAFIO: Utilizar cache/compactação de dados

Compactação de dados - Apache

- **Ativando compressão com o DEFLATE**

1. <**IfModule** mod_deflate.c>

2. *# Compactar por tipo - html, text, css, xml*
3. AddOutputFilterByType DEFLATE text/html text/plain text/css text/xml

4. *# Compactar por tipo - javascript* AddOutputFilterByType DEFLATE application/x-javascript application/javascript text/javascript text/x-js text/x-javascript

5. *# Compactar por extensão*
6. AddOutputFilter DEFLATE js css htm html xml ttf eot

7. </**IfModule**>

- Desenvolvimento das páginas JSF da loja virtual visão cliente:
    - Página do carrinho de compras;
    - Página de cadastro do cliente/login;
    - Página meus pedidos;

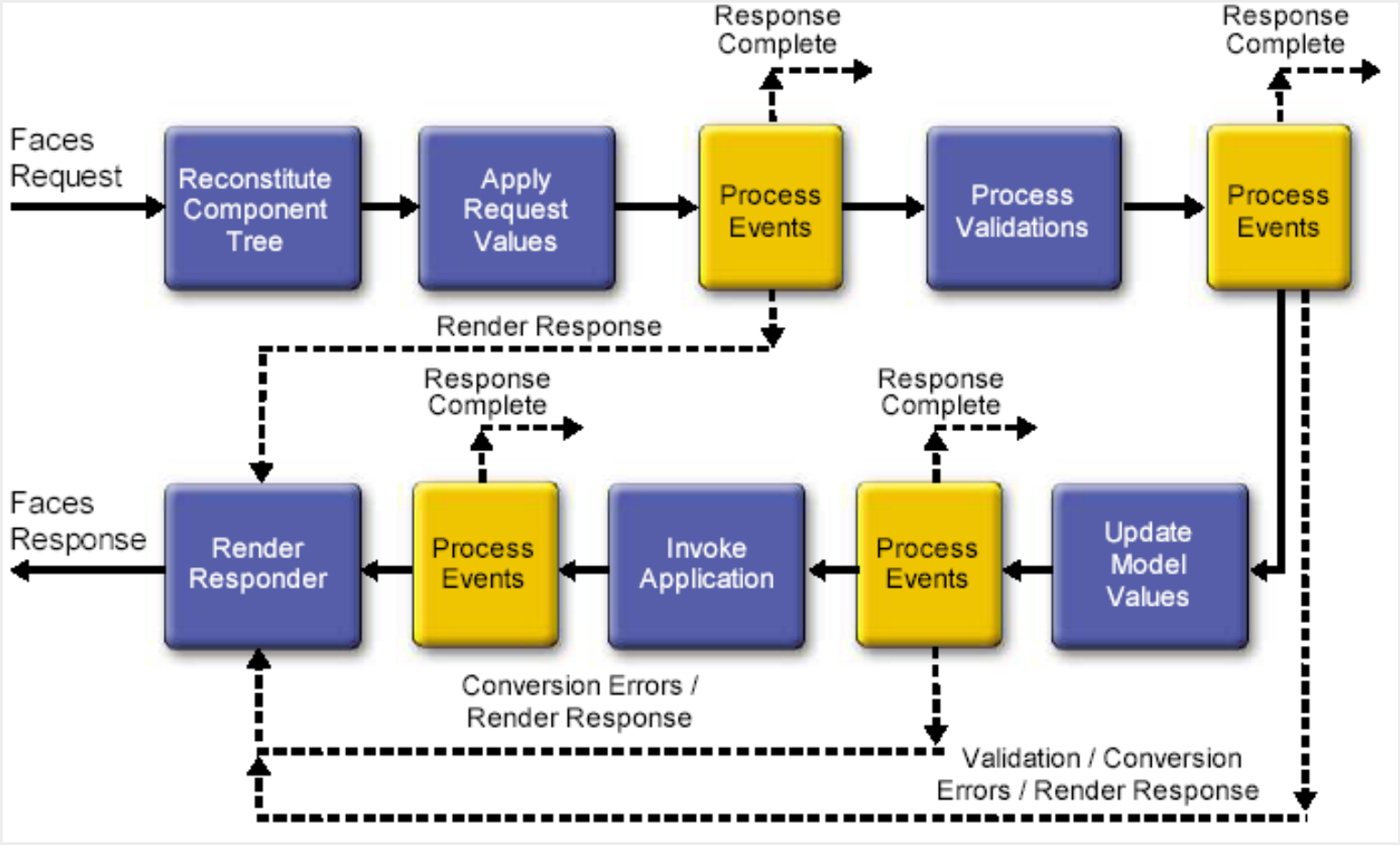
DESAFIO

Desenvolver Admin com as telas:

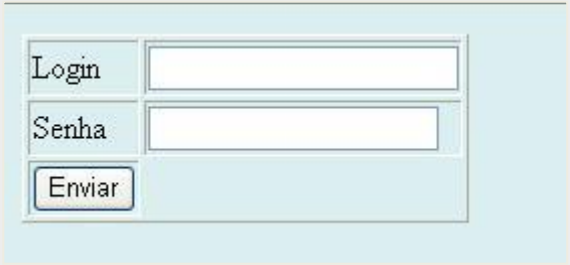
- *Cadastrar/editar/remover produtos;*
- *Cadastrar/editar/remover estoque de produtos;*
- *Cadastrar/editar/remover preços;*
- *Visualizar/editar pedidos;*

JSF - Fases do Processamento de um Request

FIAP



JSF – Exemplo usando Facelets




```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html xmlns=http://www.w3.org/1999/xhtml
      xmlns:h=http://java.sun.com/jsf/html
      xmlns:f=http://java.sun.com/jsf/core>
<h:head>
     <title>Exemplo Facelet</title>
</h:head>
<h:body bgcolor="#DBEFF0">
<f:loadBundle basename="resources.ApplicationResources" var="bundle"/>
<f:view>
<h:form>
<h:panelGrid columns="2" border="1">
   <h:outputText value="#{bundle.login}"/>
   <h:inputText id="nome" value="#{loginBean.nome}" required="true"/>
   <h:outputText value="#{bundle.password}"/>
   <h:inputSecret id="senha" value="#{loginBean.senha}" required="true"/>
   <h:commandButton action="#{loginBean.submit}" value="#{bundle.submit}"/>
</h:panelGrid>
</h:form>
</f:view>
</h:body>
</html>
```

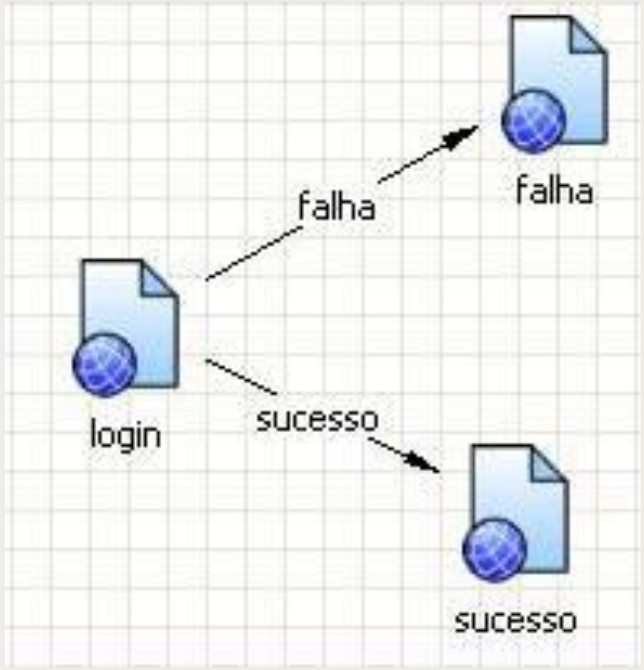
JSF – Managed Bean - Exemplo

```
package com.fiap.curso.web.jsf.bean;

public class LoginBean {

private String nome;
private String senha;

   ...
   // Métodos getters e setters
   ...
   public String submit() {
      /*
       * Simula pesquisa no banco de dados
       * e retorna resultado.
       */
       if ("jose".equalsIgnoreCase(getNome())) {
          return "sucesso";
       } else {
          return "falha";
       }
   }

}
```

faces-config.xml

```
<managed-bean>
 <managed-bean-name>loginBean</managed-bean-name>
    <managed-bean-class>
      com.fiap.curso.web.jsf.bean.LoginBean
    </managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
 </managed-bean>
```

JSF – Regras de Navegação



faces-config.xml

```
<navigation-rule>
   <!-- Navigation Rules for login.jsp -->
   <from-view-id>/login.jsp</from-view-id>
   <navigation-case>
      <from-outcome>sucesso</from-outcome>
      <to-view-id>/sucesso.jsp</to-view-id>
   </navigation-case>
   <navigation-case>
      <from-outcome>falha</from-outcome>
      <to-view-id>/falha.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

Aula 5 – Avaliação da disciplina

- Avaliação:
  1. Change Request solicitada pelo professor;
  2. Apresentação do sistema pelo grupo.

Dúvidas devem ser tiradas por e-mail:
profdaniel.andrade@fiap.com.br

Ou hangouts:
Daniel.lemes@gmail.com