Objetivo: Este documento tem como objetivo descrever as decisões arquiteturais relacionadas ao desenvolvimento do padrão de desenvolvimento para o Projeto PLUGIN-IFBA *Campus* Vitória da Conquista.

Desenvolvedor
Ivick Roberta Leite Ferreira

Avaliador
Luis Paulo da Silva Carvalho

INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA - IFBA

PROJETO PLUGIN-IFBA *Campus* Vitória da Conquista
DOCUMENTO DE ARQUITETURA

Versão: 01.00

Data: 08/08/2015

Identificador do documento:

Versão do *Template* Utilizada na Confecção: 00.01

HISTÓRICO DE REVISÕES

Versão	Data	Autor	Descrição
01.00	08/08/2015	Ivick Roberta	O Documento foi criado.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUÇÃO

1.1 Propósito

Este documento tem como principal objetivo a definição da arquitetura do padrão de interface do projeto PLUGIN-IFBA *Campus* Vitória da Conquista.

1.2 Público Alvo

Este documento se destina aos alunos e professores envolvidos no projeto PLUGIN-IFBA do *Campus* Vitória da Conquista.

1.3 Definições, Acrônimos e Abreviações

IFBA	Instituto Federal de Educação Ciência e Tecnologia
PLUGIN	Plataforma Unificada de Gestão Institucional
CRUD	*Create, Read, Update e Delete*
CSS	*Cascading Style Sheets*
HTML	*Hypertext Markup Language*
JDBC	*Java Database Connectivity*
JPA	*Java Persistence API*
JSF	*Java Server Faces*
MVC	*Model-View-Controller*
XHMTL	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*
UML	*Unified Modeling Language*

1.4 Referências

GITHUB. Disponível em: < https://github.com>. Acesso em 27 de Março 2015.

LEMOS, M. F.; OLIVEIRA, P. C.; RUELA, L. C.; SANTOS, M. da S.; SILVEIRA, T. C.; REIS, J. C. de S. **Aplicabilidade da Arquitetura MVC em uma aplicação web(webapps)**. 2013. Disponível em: < http://www.trabalhosfeitos.com/ensaios/Aplicabilidade-Da-Arquitetura-Mvc-Em-Uma/60713989.html>. Acesso em 02 de Março 2015.

PostgreSQL.1996-2015. Disponível em: <http://www.postgresql.org/about/>. Acesso em 08 Maio 2015.

Primefaces. Disponível em:< http://www.primefaces.org/ >. Acesso em 28 Jul. 2015.

FARIA, T. **Java EE 7 com JSF, PrimeFaces e CDI**. Uberlândia: AlgaWorks Softwares, 2013.

ZURB. **Zurb Foundation**. Campbell: 2011. Disponível em:<http://foundation.zurb.com/index.html>. Acesso em 02 de Março de 2015.

1.6 Visão geral do documento

O restante do documento encontra-se dividido da seguinte forma:

- **A seção 2.0** especifica as premissas e restrições dos requisitos levantados.
- **Na seção 3.0** são enumerados todos os requisitos funcionais do padrão implementado.
- **Na seção 4.0** são descritos os requisitos não funcionais do padrão de interface.

REQUISITOS IMPACTANTES

Este trabalho teve como objetivo principal o desenvolvimento de um padrão de interfaces para a implementação das aplicações componentes do projeto PLUGIN - IFBA sob os paradigmas de *Web Design* Responsivo e padrão MVC. Entre os objetivos específicos definidos, destacam-se: (a) utilizar as tecnologias HTML5, CSS3, *JavaScript* e padrão MVC e (b) definir como estes itens podem ser integrados, visando a criação de uma arquitetura de *software* reusável para os projetos PLUGIN. Para contemplar o primeiro, foi realizada uma pesquisa sobre *Web Design* Responsivo e suas tecnologias (HTML 5, CSS3 e *JavaScript*) e padrão MVC. Para o segundo objetivo, foi efetuada uma integração entre as telas implementadas em *Web Design* Responsivo à arquitetura MVC, para isso foi estabelecida a união do HTML5, JSF e *JavaScript* com o pacote Visão do padrão MVC.

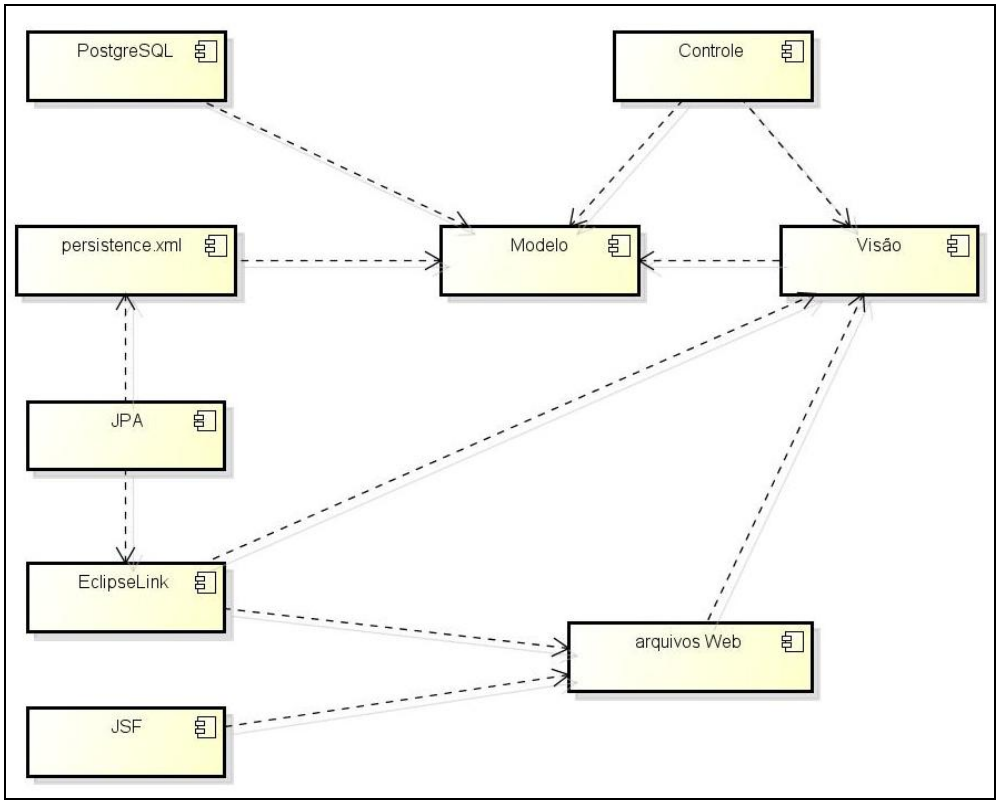
VISÃO DE COMPONENTES

O padrão de telas implementado em *Web Design* Responsivo foi integrado à arquitetura MVC. Para isso foi estabelecida a união do HTML5, JSF e *JavaScript* com o pacote Visão do padrão MVC. No projeto foram utilizadas as bibliotecas *EclipseLink*, o JSF e o *drive* do JDBC do *PostgreSQL* (banco de dados). A Figura 1 representa o diagrama de componentes do projeto. Esses componentes estão relacionados aos pacotes MVC do *template* PLUGIN da seguinte forma: (i) o Modelo é integrado ao *PostgreSQL* e ao JPA, pois é nesta camada que são realizadas as integrações com os bancos de dados (acesso a informações e persistência de dados); (ii) a Visão precisa de referências ao JSF, pois ele oferece um modelo de interfaces gráficas baseado em eventos, e é nesta camada que são realizadas as requisições dos usuários na *web*; (iii) O *EclipseLink*, é através dele que o Modelo interage com vários tipos de serviços de informação como banco de dados, *web services* e os objetos XML.

3.1. Diagrama de componentes

O diagrama de componentes da UML descreve graficamente os componentes de *software* utilizados e seus relacionamentos. Este diagrama se encontra exibido na Figura 1.

Figura 1 - Diagrama de Componentes do projeto *template* PLUGIN



Fonte: Própria Autora

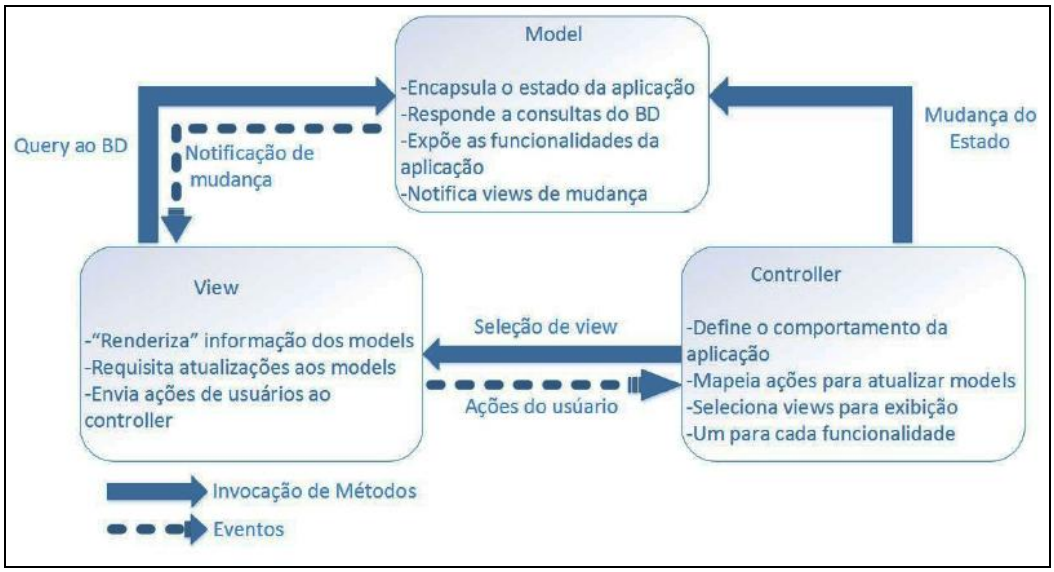
VISÃO LÓGICA

O padrão MVC fornece a vantagem de tornar os sistemas escaláveis e portáveis, além de permitir o desenvolvimento paralelo aos pacotes de responsabilidade. A integração entre as tecnologias, *Web Design* Responsivo e arquitetura MVC, tem como objetivo possibilitar a transferência de conhecimento para reuso nos projetos PLUGIN - IFBA *Campus* Vitória da Conquista.

4.1. Visão de padrões arquiteturais

A arquitetura se encontra ilustrada na Figura 2.

Figura 2 - Arquitetura MVC



Fonte: LEMOS et al. (2013)

A arquitetura MVC foi adotada. Esta arquitetura prevê o particionamento das classes em três pacotes de artefatos de código:

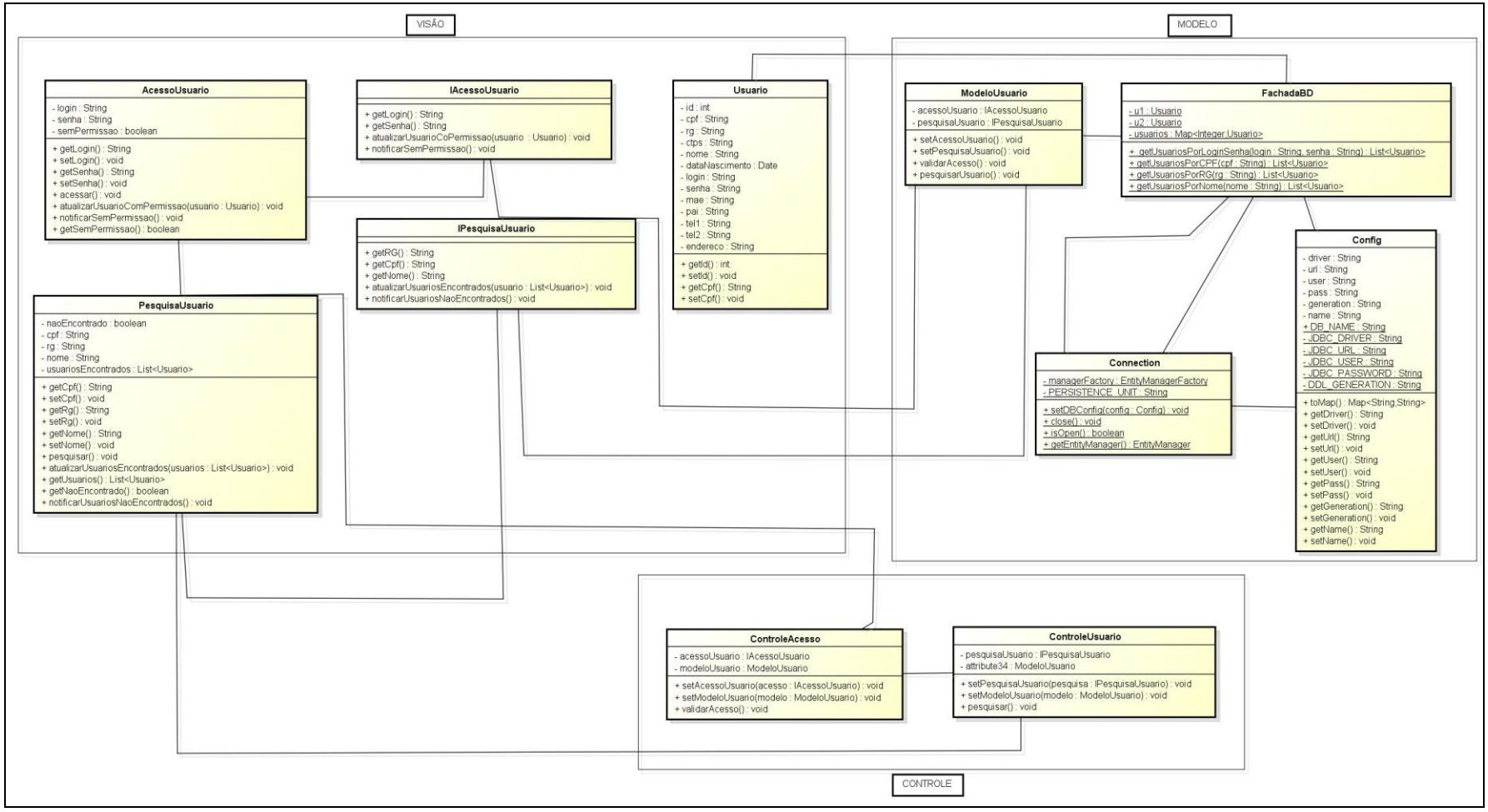
- Modelo (*Model*) → Sua função é realizar as operações sobre o banco de dados, *CREATE*, *READ*, *UPDATE* e *DELETE* (CRUD);
- Visão (*View*) → As classes contidas neste pacote representam o que será visualizado pelo usuário, ela captura as suas ações e envia ao Controle;

- Controlador (*Controler*) → O propósito é administrar todo o fluxo da aplicação, controlando a movimentação entre a Visão e o Modelo.

4.2. Visão de classes

O projeto *template* PLUGIN tem classes e interfaces distribuídas entre os pacotes Visão, Controle e Modelo da arquitetura MVC. Trata-se de um projeto exemplo formado por um conjunto mínimo de classes, que pode ser evoluído a partir da entrada de novas classes e demais itens. A Visão é composta pelas classes "AcessoUsuario" e "PesquisaUsuario", e pelas interfaces "IAcessoUsuario" e "IPesquisaUsuario". O Controle é constituído pelas classes "ControleAcesso" e ControleUsuario". E o componente Modelo é formado pelas classes de configuração e conexão "Config" e "Connection", além da "FachadaBD", "Usuario" e "ModeloUsuario". Este exemplo tem como finalidade demonstrar como o *template* PLUGIN deve ser utilizado, onde o escopo de classes e interfaces representam um exemplo mínimo para que o padrão funcione. A Figura 3 representa um diagrama de classes do projeto *template* PLUGIN.

Figura 3 - Diagrama de classes do projeto *template* PLUGIN

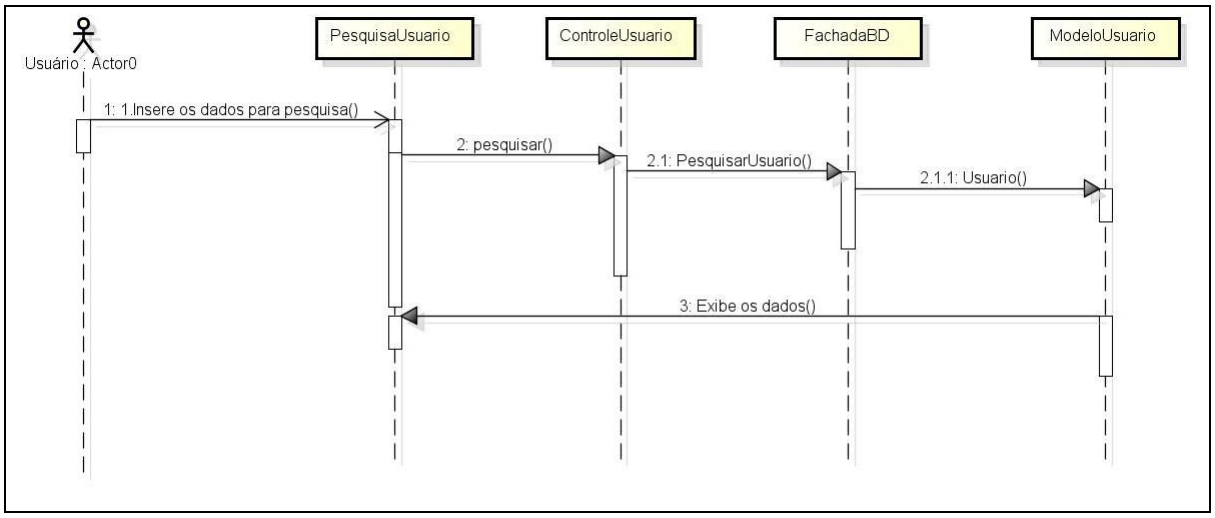


Fonte: Própria Autora

4.3. Visão de sequência

Esta seção contém informações sobre as sequências de atividades necessárias para automatizar as funcionalidades da tela pesquisa de usuário, exemplo do projeto PLUGIN. Na tela de pesquisa de usuário, por exemplo, ao pressionar o botão "Pesquisar", após digitar os dados solicitados, um evento é acionado e realiza a chamada do MVC para realizar a pesquisa a partir dos dados digitados. Na classe "PesquisaUsuario" foi criado um método *void* "pesquisar()", onde ele aciona chamadas aos componentes Modelo e Controle do MVC. Se o usuário for encontrado seus dados serão exibidos na pesquisa. Caso contrário aparecerá uma frase informando que nenhum registro foi encontrado. Para ilustrar tais sequências graficamente, é utilizado o diagrama de sequencia da UML, ela mostra como ocorre a comunicação entre os componentes do MVC, representando todos os passos da pesquisa, mostrado na Figura 4.

Figura 4 - Diagrama de sequência de pesquisa de usuário



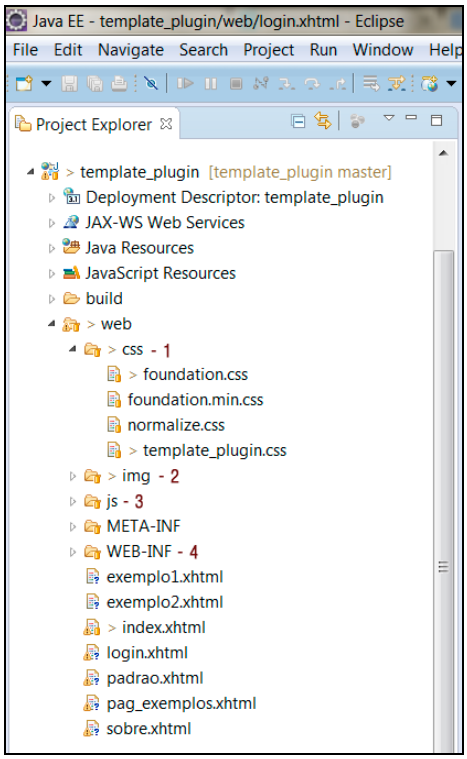
Fonte: Própria Autora

4.4. Visão de configuração

A implementação do padrão de interface para as aplicações componentes do projeto PLUGIN - IFBA *Campus* Vitória da Conquista foi construída sob os paradigmas do *Web Design* Responsivo, utilizando como base o *framework Foundation*.

Para a implementação do padrão, foi criado um projeto na IDE Eclipse *Luna*, com o nome *template*_plugin. Como a base do padrão foi o *Foundation*, o projeto é composto dos arquivos de estilo (CSS) e *JavaScript* do *framework*, eles são obtidos no momento do *download* do pacote de desenvolvimento. Na Figura 5 é mostrada como as pastas do projeto foram dispostas.

Figura 5 - Projeto do padrão *template*_plugin



Fonte: Própria Autora

Dentro da pasta "*web"* encontram-se todos os arquivos utilizados para a implementação do padrão, na primeira pasta de numeração 1 "css", podem ser identificados os arquivos de CSS básicos do *Foundation* e o criado especificamente para o padrão: *template*_plugin.css. Na pasta seguinte, de numeração 2 "img" encontram-se todas as imagens utilizadas no desenvolvimento. Na pasta de numeração 3 "js" estão todos as funções *JavaScript* do *Foundation.* Por fim, a pasta de numeração 4 "*WEB-INF*" onde estão todos os arquivos "XHTML" do projeto. É importante que todos os arquivos para *web*

estejam com extensão XHMTL, pois trata-se do formato esperado para a integração com o JSF, visto que a partir dele será possível a criação de artefatos de código em Java para *web.*

O *Foundation* é um *framework* que possibilita sua customização, proporcionando ao desenvolvedor uma liberdade maior ao criar o *design* de uma interface. Devido a essa característica, além de utilizar as folhas de estilo básicas do *framework*, caso seja necessário, é possível criar uma ou mais folhas de estilo para o projeto. Para o padrão PLUGIN foi necessário criar uma folha para ele, para que as aplicações componentes desenvolvidas tivessem características gráficas próprias e identidade visual diferenciada.

Durante o processo de implementação do padrão de interfaces, uma conta foi criada no site *GitHub* como o objetivo hospedar todo o projeto. O *GitHub* trata-se de um serviço de *Web Hosting* Compartilhado para projetos que usam o controle de versionamento Git. Para realizar os *"commit's"* foi utilizado um *plugin* no próprio Eclipse para que as alterações feitas no projeto fossem enviadas para um repositório na conta do *GitHub.*

O projeto *template*_plugin tem classes e interfaces distribuídas entre os pacotes de componentes: Visão, Controle e Modelo da arquitetura MVC, como foi dito anteriormente. Trata-se de um projeto exemplo formado por um conjunto mínimo de classes, que pode ser evoluído a partir da entrada de novas classes e demais itens. Este exemplo tem o propósito educativo, tem como finalidade demonstrar como o *template* PLUGIN deve ser utilizado, onde o escopo de classes e interfaces representam um exemplo mínimo para que o padrão funcione. Por conseguinte, cada projeto específico deve evoluí-lo adicionando novos componentes.