BookMinder

Projeto Arquitetural

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Objetivo

A arquitetura utilizada no sistema baseia-se na arquitetura MVC.

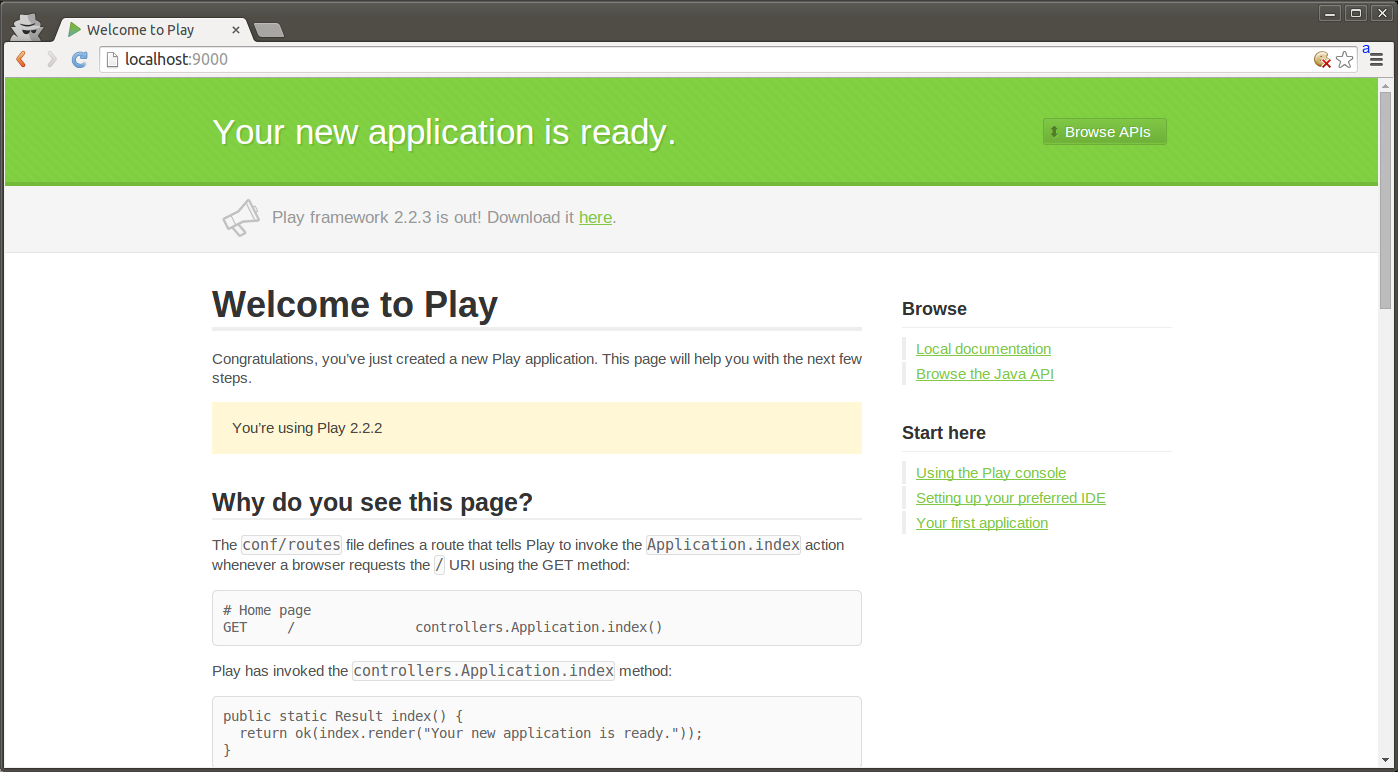
Objetivos:

* Separação de componentes por responsabilidades em comum.
* Utilizado para estruturar e definir melhor as funcionalidades de um sistema.
* Separar o código de acesso aos dados do código de negócio e interface com o usuário (Modularidade)

# Requisitos Críticos da Arquitetura

Para este projeto faremos uso do [Play! Framework](http://playframework.com/), framework que tem ficado cada vez mais famoso devido à sua facilidade de uso, rapidez no desenvolvimento, otimizações de performance e várias outras vantagens. Segue link da documentação do Play! Framework e seus demais módulos utilizados.

* <https://www.playframework.com/>
* <https://www.playframework.com/download>
* <https://www.playframework.com/documentation/2.3.x/Installing>



Vantagens da utilização do Play! Framework:

* **Instantâneo**: Todas as alterações em telas ou no código java são recompiladas e executadas sem reiniciar o servidor, só é necessário um “refresh”;
* **Depuração**: é possível conectar o debugger do Eclipse com a aplicação executando em modo de desenvolvimento, mas nem é tão necessário, pois quando há algum erro no Java ou na tela, a mensagem do framework é tão amigável que mostra, além da pilha, o próprio código fonte e qual o comando ou erro de sintaxe que causou a falha;
* **Sessão:** não existe uma *session*, os únicos valores que são armazenados entre as requisições ficam em cookies criptografados. Isso faz com que a aplicação seja altamente escalável com múltiplas instâncias, embora seja confuso para quem está acostumado com as tecnologias JEE padrão;
* **Threads**: não há criação de threads por requisição como nos Servlets, os métodos dos controladores são static e processamentos “demorados” devem ser assíncronos;
* **Totalmente REST**: todas as entradas de url são configuradas no arquivo “routes”, como no *rails*;

GET /sobre controllers.App.sobre()

* **CoC (Convensão sobre Configuração / *Convention over Configuration*)**: a ideia é não reinventar a roda, basta executar um comando “play new [nome sistema]” e sair programando, obviamente se você seguir o padrão adotado pelo framework. Já se quiser fazer diferente aí vai ter tanto ou mais trabalho do que num framework tradicional;
* **Builders para as views**: as telas, por exemplo, possuem parâmetros e você “invoca” uma tela a partir do código chamando um método de uma classe gerada dinamicamente que recebe os devidos parâmetros, então se você mudar uma tela para receber algo novo, não tem como esquecer de atualizar o código, já que isso vai gerar erros de compilação;
* **Servidor de aplicação**: esqueça Tomcat, Glassfish, Weblogic ou Websphere, ele já vem com um servidor leve embutido, o comando “run” inicia a aplicação em modo desenvolvimento e o comando “start” em modo de produção;
* **Ciclo de vida**: o framework gerencia o ciclo de vida através de uma ferramenta de build chamada SBT (uma espécie de ANT), gerencia as dependências através do IVY e também é compatível com Maven;
* **Integração com Eclipse**: o comando “eclipsify” gera os arquivos necessários para importar o projeto no Eclipse, quando incluir uma nova dependências, basta executar o comando novamente para atualizar o classpath e demais configurações;
* **Banco de dados**: integração nativa com JPA, inclusive gerenciando as evoluções da estrutura do banco e gerando a DDL necessária a partir das classes
* **Telas**: templates feitos numa mistura de HTML com a liguagem Scala. É simples e rápido para implementar, porém lento demais para compilar.

Notas:

* Para executar o Play! Framework, é necessário o JDK 6 ou superior. Além disso, configurar o diretório bin do JDK na variável de ambiente PATH do seu Sistema Operacional (O Play! Framework irá utilizar o javac, além do java);
* Incluir o diretório que contém os programas play e play.bat também na variável PATH (No meu caso, Figura 1, o caminho será C:\dev\frameworks\play-2.0.4\play-2.0.4);
* No caso do Linux, não esqueça do chmod no arquivo play.

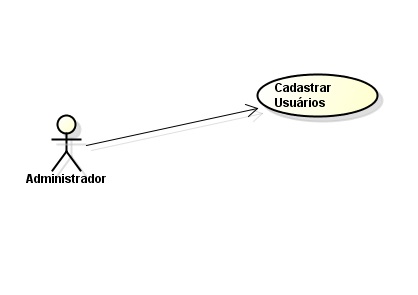
# Mecanismos Arquiteturais

Os diretórios mais importantes de um projeto Play! são os seguintes:

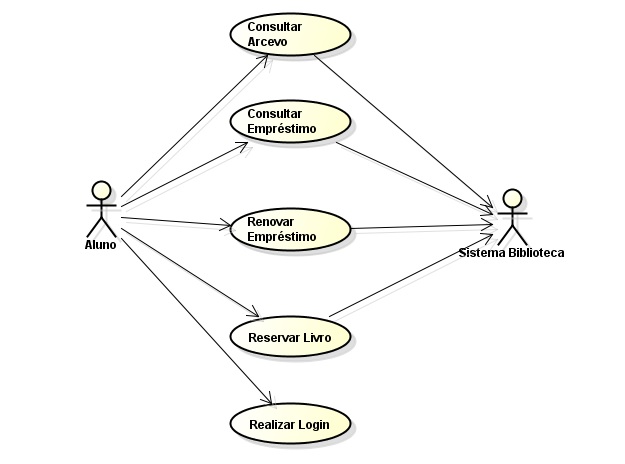
* **app** - contêm os models, controllers e views do projeto;
* **conf**- contêm os arquivos de configuração como o arquivo de rotas (routes), configurações gerais da aplicação (application.conf), internacionalização, etc.;
* **project** – contêm os build scripts do projeto (os scripts são criados para a ferramenta sbt, que é uma ferramenta de build para Scala e Java). Raramente você terá que alterá-lo;
* **public** – contêm recursos como arquivos javascript, css e imagens.

# Principais Abstrações

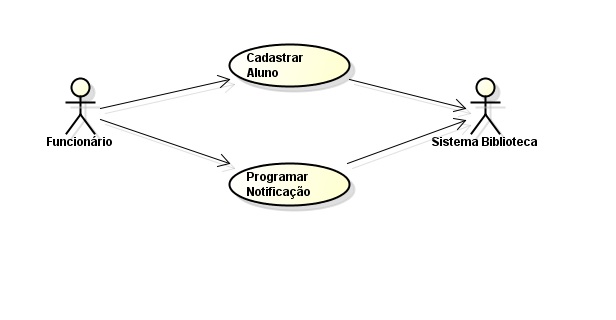
1. Administrador



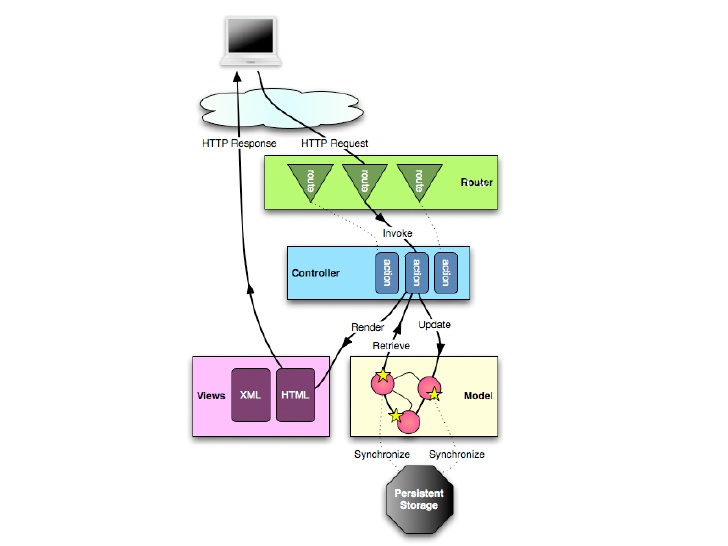
2. Aluno



3. Funcionário



# Camadas do Framework da Arquitetura



# Visões Arquiteturais

• Lógica:

• Operacional:

• Caso de uso: