Academic Advisor

Architecture Notebook

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**.

# Purpose

Este documento visa demonstrar principais aspectos do projeto Academic Advisor. Sua arquitetura, forma de implementação e aspectos.

# Architectural goals and philosophy

O sistema Academic Advisor prove integração de um serviço web e mobile para atender necessidades de alunos e futuro-alunos em universidades, provendo um contato inicial entre essas duas entidades, para que troquem informações sobre o curso que se tem em mente.

O sistema funciona somente com conexão à internet, podendo usar uma rede privada no caso da universidade, embora esses dados devam ser disponibilizados publicamente para acesso de usuários fora desta rede.

A primeira versão do sistema mobile vai ser desenvolvida no framawork Bootstrap, amplamente divulgada e acessível na maioria dos smartphones pois usa a tecnologia de Design Responsivo.

As demandas iniciais poderão ser atendidas sem grandes problemas. As primeiras versões usarão o mínimo de funcionalidades para atender o público. O feedback de usuários será muito importante para aperfeiçoamento do sistema.

A versão mobile pretende fornecer funcionalidades básicas como, por exemplo, consulta de agenda, cursos, etc. A versão web vai fornecer formulários para cadastro, agenda e relatórios em forma de lista. O sistema web vai ser projetado para funcionar em navegadores web, sendo que as principais funcionalidades de cadastro estarão nele.

Futuramente podem ocorrer problemas com integrações de serviços, como por exemplo, consultar serviços públicos de universidade para gerar cadastros dinâmicos.

# Assumptions and dependencies

Assume-se que a universidade vai fornecer dados para carga no sistema, e que ela possua infraestrutura de rede que permita comunicação entre outras máquinas. Qualquer problema com múltiplos acessos é de responsabilidade do cliente.

Como o Sistema vai possuir uma versão web browser e outra web mobile, será necessário manter coerência no design de interface.

Ponto crítico será a sincronia de dados, pois a base será de responsabilidade da universidade, assim como cadastros. Caso a universidade não atualize ou disponibilize a base, vai gerar conflitos. Como exemplo, temos a mudança de currículo de um curso ou extinção do mesmo.

Atualmente o projeto possui apenas um desenvolvedor previsto. Arquiteto também participará no desenvolvimento de integrações e análise de domínio.

# Architecturally significant requirements

**Requisitos Funcionais**

**RF01**​ – O futuro-aluno deve ser capaz de se cadastrar para usar o sistema.

**RF07**​ – O futuro-aluno é capaz de procurar veteranos disponíveis em um certo curso.

**RF08**​ – O admin deve ser capaz de cadastrar novos cursos.

**RF09**​ – O admin deve ser capaz de cadastrar o roteiro de cada curso.

**RF10**​ – O admin deve ser capaz de cadastrar novos veteranos no Sistema.

**Requisitos Não-Funcionais**

**RNF01** – O sistema será intuitivo mesmo para um usuário novo.

**RNF02** – O sistema manterá logs de todo o fluxo de dados para melhor monitoramento

**RNF04** – As atualizações de abas não deverão levar mais de 3 segundos.

**Algumas outras funcionalidades:**

* **Agenda atualizada (Web e Mobile)**

Os dois ambientes, tanto o mobile quanto a web, devem sempre estar com a agenda atualizada. Dada uma modificação o usuário deve ser informado se possível por e-mail.

* **Reutilização de classes e dependências**

Como são projetos separados, deve ocorrer a reutilização de objetos (dependências, script, etc) para melhor organização da arquitetura e reutilização de tais elementos. A tecnologia mobile com Web Responsive vai reutilizar elementos da arquitetura web, pois ambas são em tecnologia Java e Html.

* **Navegação simples e objetiva**

Interface não deve ser complexa, e deve ser adequada ao perfil de usuário (Administrador, Aluno, etc). Poderá utilizar o framework Spring MVC.

* **Fácil adaptação e customização para outros clientes**

Devemos ter elementos parametrizáveis para customização para clientes específicos. Por exemplo, logo de universidade no sistema.

# Decisions, constraints, and justifications

* **Perfil de administrador não deve efetuar cadastro via Sistema mobile**

Sistema mobile não visa administração. Isso será função do site.

* **Sistema mobile deve permitir apenas ações simples, como efetuar autenticação no sistema; Cancelar Agendamento; consultar listas de cursos;**

O objetivo do sistema mobile não é ser robusto.

* **O back-end do sistema deve ser reutilizável tanto na versão mobile e web**

As arquiteturas devem permitir fácil integração e reutilização de funções e serviços.

* **Usar tecnológica front-end que use design responsivo para utilização em qualquer dispositivo**

Dada a necessidade de atender muitos dispositivos diferentes, browser e mobile.

* **Não é necessário alta performance de sistema e múltiplos servidores ou espelhamento de bases**

Como o escopo é restrito a universidades, não se faz necessário grande infraestrutura.

* **Usar framework opensource onde suas comunidades a atualizem constantemente, com amplo suporte destas**

Para fácil manutenção e economia de custos de desenvolvimento.

* **Usar o paradigma orientado a objetos para o desenvolvimento**

Não se trata de uma aplicação cientifica ou financeira. Logo, o paradigma POO é o mais adequado.

* **Não é uma rede-social, não terá chat**

É um sistema de gerência de reuniões de alunos e de avaliações de cursos.

# Architectural Mechanisms

**Tecnologia de banco de dados – Java DB Apache Derby**

Apache Derby é um sistema de gerenciamento de banco de dados relacional Java que pode ser embutido em programas Java e usado para processamento de transações online.

**Servidor de aplicação - GlassFish**

GlassFish é um servidor de aplicação open source liderado pela Sun Microsystems para a plataforma Java EE.

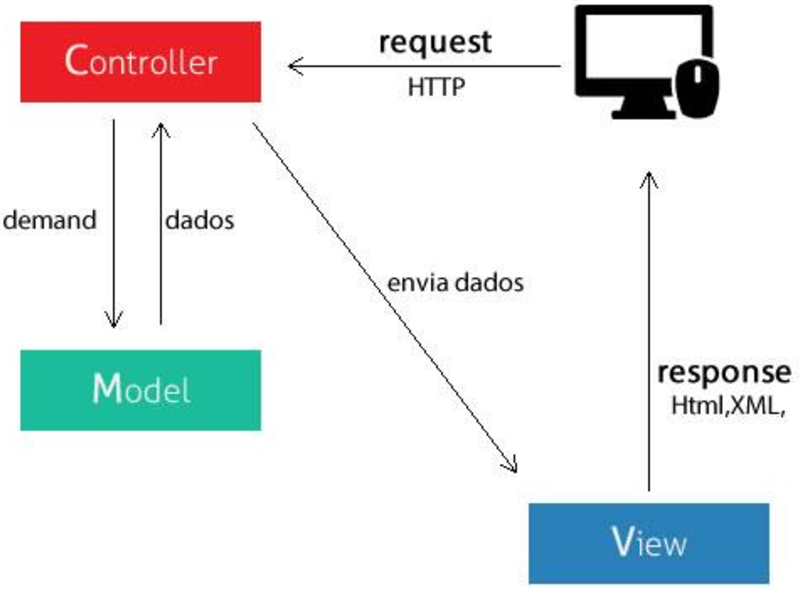
GlassFish suporta todas as especificações da API Java EE, tais como JDBC, RMI, JavaMail, JMS, JMX etc. e define como coordená-las. GlassFish também suporta algumas especificações para componentes Java EE, como Enterprise JavaBeans, conectores, servlets, portlets, JSF e diversas tecnologias de web services.

**DAO - Objeto de acesso a dados**

Objeto de acesso a dados (ou simplesmente DAO, acrônimo de Data Access Object), é um padrão para persistência de dados que permite separar regras de negócio das regras de acesso a banco de dados. Numa aplicação que utilize a arquitetura MVC, todas as funcionalidades de bancos de dados, tais como obter as conexões, mapear objetos Java para tipos de dados SQL ou executar comandos SQL, devem ser feitas por classes DAO.

**Arquitetura de camada MVC**

O MVC (Model-View-Controller) é um padrão de projeto arquitetural baseado em outros padrões como o Observer, Composite e Strategy. O MVC, como o nome sugere, possibilita a separação de um projeto em múltiplas camadas, das quais fazem parte: Modelo (Model), Visão (View) e Controlador (Controller).



**Model -** É responsável pela leitura e escrita de dados, e também de suas validações.

**View -** Simples: a camada de interação com o usuário. Apenas faz a exibição dos dados (html ou xml).

**Controller -** Responsável por receber todas as requisições do usuário. Seus métodos chamados actions são responsáveis por uma página, controlando qual model usar e qual view será mostrado ao usuário.

**Arquitetura Web Mobile**

**Bootstrap**

Bootstrap é um framework front-end que facilita o desenvolvimento web para criar sites com tecnologia mobile (responsivo). O Bootstrap possui uma diversidade de componentes (plugins) em JavaScript (jQuery) no desenvolvimento também.

Como o framework é responsivo e gera conteúdo html. Pode ser expandido para uso em outras plataformas mobile.

**Algumas características:**

* Atualmente possui uma grande diversidade de themas;
* Grande quantidade de plugins adaptados ou desenvolvidos para o framework;
* Integração com qualquer linguagem de programação;
* Sistema responsivo;

# Key abstractions

* Cliente/Servidor
* Multiplas camadas
* Sistema web
* Sistema mobile
* Linguagem java
* Tecnologia Mobile Responsive Web

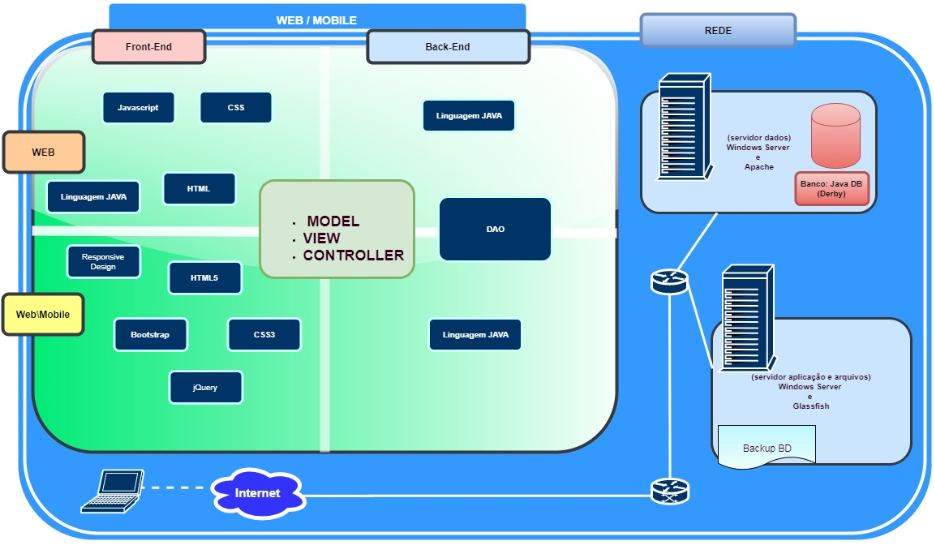
**Frameworks e pacotes**

* Bootstrap
* jQuery

**Entidades de negócio**

* Usuário Futuro-Aluno
* Usuário Veterano
* Usuário Administrador

*Figura 01 - Diagrama de arquitetura do sistema*



# Layers or architectural framework

* MVC – Model View Controller – com reutilização das camadas de Controller e Model
* Desenvolvimento divido em front-end e back-end
* Utilizar bibliotecas java para prover serviços, como envio de e-mail

# Architectural views

Na visão arquitetural teremos duas formas de acesso ao sistema.

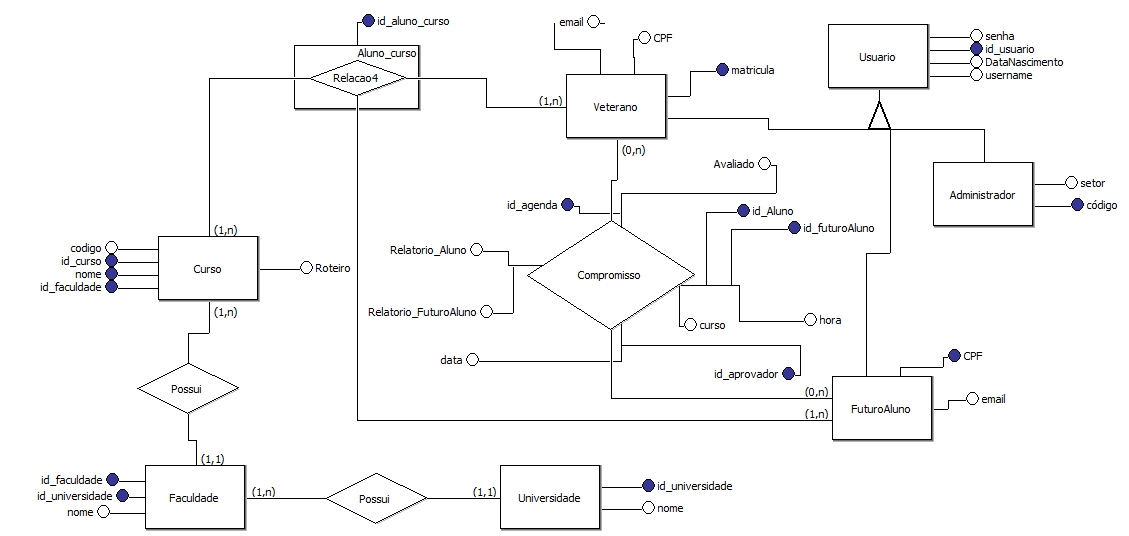
**Web** – Onde o administrador terá como dar manutenção no sistema e administrar seus cadastros. Assim como alunos poderão preencher seus relatórios e agendas.

**Mobile** – visão mais simples do sistema onde um administrador não terá acesso. Apenas perfil de usuário do tipo aluno poderá ver conteúdo.

**Alguns casos de uso relevantes para o sistema:**

* UC02 Verificar Agenda
* UC09 Cadastrar Roteiro do Curso
* UC11 Preencher Relatório
* UC07 Procurar Veteranos disponíveis em certo curso

*Figura 02 - Esquema conceitual do banco de dados*



*Figura 03 - Esquema lógico do banco de dados*

