**Documento de Arquitetura - Probex**

**1. Introdução**

O Documento de Arquitetura de Software é um artefato que permite comunicar aos stakeholders as decisões arquiteturais de um projeto de software. A presente seção introduz esse documento, descrevendo sua finalidade, as definições, acrônimos e abreviações utilizadas no projeto, suas referências e a visão geral da arquitetura do sistema Probex.

**1.1 Finalidade**

Este documento tem como objetivo apresentar a arquitetura do Sistema Probex. Por arquitetura, entende-se a estrutura ou o conjunto de estruturas do sistema, que compreendem:

(i) elementos de software;

(ii) propriedades externamente visíveis destes elementos; e

(iii) os relacionamentos entre os mesmos.

A arquitetura do sistema é formada por diversos componentes que se relacionam globalmente sob diferentes perspectivas. Sendo assim, este documento desenvolve várias visões arquiteturais, as quais representam características e interesses diferentes de diversos stakeholders sobre um mesmo objeto, ou seja, sobre o software desenvolvido.

**1.2 Definições, Acrônimos e Abreviações**

**Definições**

| **Termo** | **Significado** |
| --- | --- |
| JBoss | Servidor de aplicações baseado na plataforma JEE mantido pela Red Hat |
| REST | Padrão arquitetural usado em serviços Web |
| RESTEasy | Implementação da especificação JAX-RS usado pelo JBoss |
| SPA | Padrão de aplicações que carregam apenas uma página e atualizam dinamicamente as informações exibidas à medida que o usuário interage com a aplicação |

**Acrônimos**

| **Termo** | **Significado** |
| --- | --- |
| JEE | Java Enterprise Edition |
| REST | Representational State Transfer |
| SPA | Single Page Application |

**1.3 Referências**

Não se aplica.

**1.4 Visão Geral**

O restante do documento está organizado como segue. A Seção 2 descreve a representação global da arquitetura da aplicação. A Seção 3 apresenta as principais metas e restrições arquiteturais. As Seções de 4 a 9 apresentam as diferentes visões arquiteturais, sejam elas: visão de caso de uso, visão lógica, visão de processos, visão de implantação, visão de implementação e visão de dados, respectivamente.

**2. Representação Arquitetural**

O Probex é composto por quatro camadas:

(i) A camada de visão, construída com HTML5 e Javascript, utilizando o padrão SPA e os frameworks AngularJS e Google Material Design;

(ii) A camada de serviços, onde é disponibilizada uma API REST escrita em Java;

(iii) A camada de negócio, onde são construídos os objetos de negócio gerenciados pelo container EJB;

(iv) A camada de dados, que realiza o mapeamento objeto-relacional no padrão JPA.

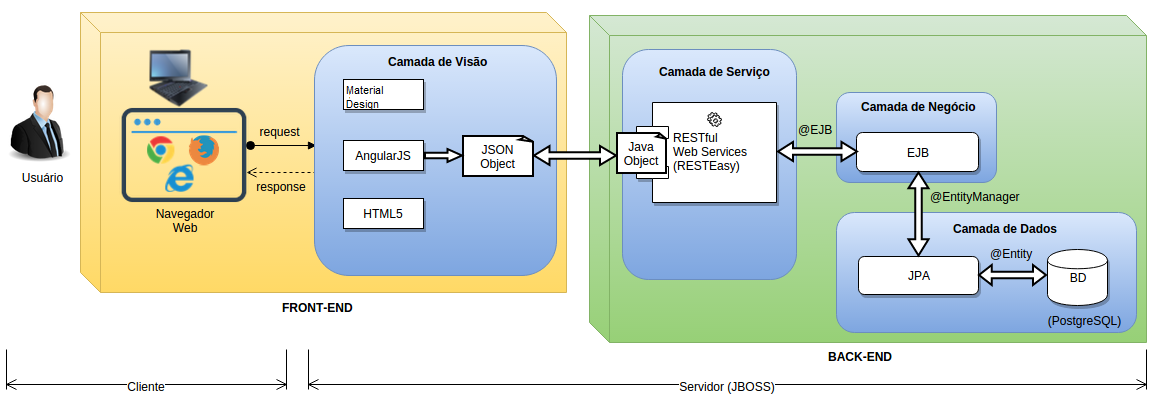
A camada de visão é executada inteiramente no navegador do cliente que, por sua vez, faz uma série de acessos assíncronos à camada de serviços com o objetivo de ler e manipular dados. O formato usado para a troca de dados é o JSON.

A API REST foi desenvolvida na plataforma JEE 6 usando RESTEasy e JBoss EAP 6.4.

Os objetos de negócio são instanciados via injeção de dependência pelo container EJB 3.1.

As entidades persistentes são construídas utilizando o padrão JPA 2 com a implementação em Hibernate 3.2.1 e mapeadas para tabelas de um banco de dados no SGBD PostgreSQL 9.3.

Abaixo, um diagrama da solução.



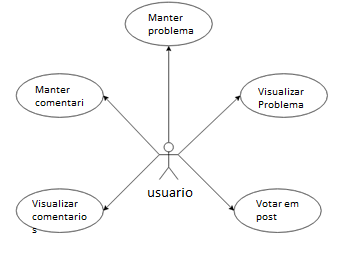
**3. Metas e Restrições da Arquitetura**

A escolha do padrão SPA permite trazer todo o processamento da camada visão para o cliente, o que minimiza a quantidade de requisições ao servidor e o tráfego de rede, tornando mais rápida e fluida a interação do usuário com a aplicação.

Embora o Probex tenha sido desenvolvido inicialmente para gerenciar “problemas , sua arquitetura foi projetada para permitir a inclusão de novas funcionalidades ao sistema, como o gerenciamento de avaliações das turmas de todos os professores em uma ou mais instituições de ensino. Dessa maneira, a criação de uma API REST tem o objetivo de facilitar a evolução do sistema, permitindo a inclusão de novas funcionalidades de forma transparente entre as camadas.

**4. Visão de Casos de Uso**

O diagrama de casos de uso a seguir ilustra o relacionamento entre as principais funcionalidades e atores com impacto na arquitetura do sistema:



O Probex possui usuários. Ao usuário são fornecidas as funcionalidades realizar um post de problema, comentar em posts, visualizar posts e comentários e votar em posts.

**5. Visão Lógica**

Esta seção descreve as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes. Além disso, para cada pacote significativo, ela mostra sua divisão em classes e utilitários de classe.

**5.1 Visão Geral**

Os módulos do sistema são descritos na tabela abaixo:

| **Módulo** | **Descrição** |
| --- | --- |
| ProbexService | Agrupa todos os pacotes e arquivos gerenciados pelo Container Web da plataforma JEE. |
| ProbexEJB | Agrupa todos os pacotes e arquivos gerenciados pelo Container EJB da plataforma JEE. |
| ProbexEAR | Representa o Enterprise Archive que agrupa os módulos ProbexService e ProbexEJB, permitindo a implantação do sistema em um servidor de aplicação. |

**5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura**

A seguir cada pacote do sistema é descrido e exemplos de artefatos que os compõem são fornecidos:

| **Pacote** | **Descrição** | **Exemplos de Artefatos** |
| --- | --- | --- |
| DTO | Agrupa todos os modelos DTOS das entidades para serem transportados pelo REST | *PostDTO.java, UserDTO.java* |
| API REST | Agrupa todos os serviços REST disponibilizados pela aplicação. | *PostResource.java*, *UserResource.java* |
| BeanService | Agrupa todas as interfaces dos Session Beans remotos da aplicação. | *PostBORemote.java*, *UserBORemote.java* |
| BeanServiceImpl | Agrupa todas as implementações das interfaces dos Session Beans remotos da aplicação. | *PostBO.java*,  *UserBO.java* |
| Entity | Agrupa todas as entidades persistentes da aplicação no padrão JPA 2. | *Post.java, User.java* |
| Dao | Agrupa todas as classes utilizadas para acesso a dados da aplicação. | *PostDAO, UserDAO* |

**6. Visão de Processos**

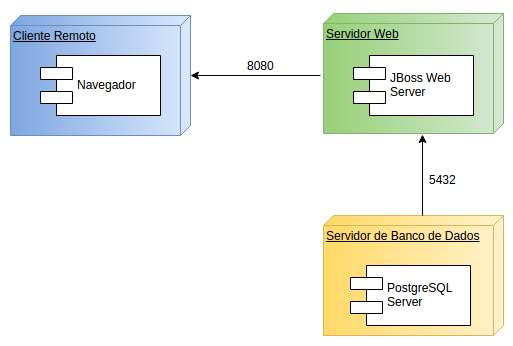
Não se aplica, pois não existe processamento paralelo simulado no nível da aplicação.

**7. Visão de Implantação**

Esta seção descreve a configuração da rede física (hardware) na qual o Probex é implantado e executado.

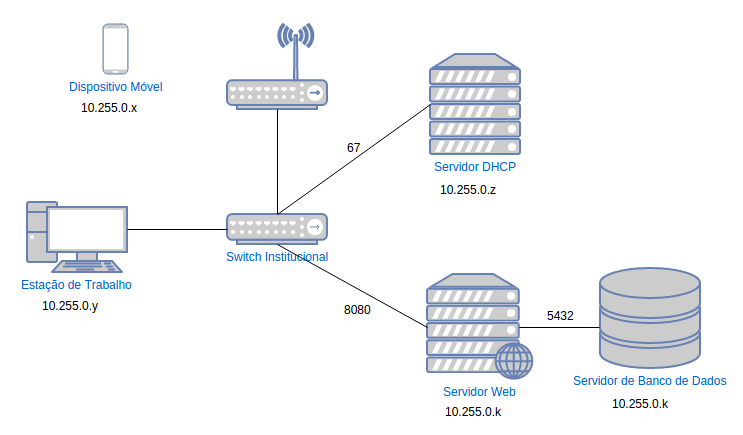
**7.1 Diagrama de Implantação**

A figura abaixo mostra o Diagrama de Implantação da solução, onde o host do cliente acessa o servidor Web na porta 8080 através de um navegador Web, e o servidor de banco de dados é acessado na porta 5432.



**7.2 Diagrama de Rede**

O Diagrama de Rede a seguir ilustra a infraestrutura de comunicação na qual o sistema é executado. Cada host da rede recebe um endereço ip dinâmico fornecido por um servidor DHCP institucional. Dessa maneira, as estações de trabalho e dispositivos móveis conectados à rede podem realizam requisições ao servidor de aplicação.

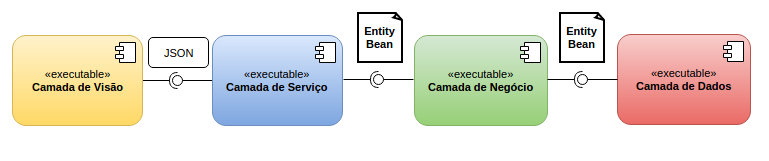


**8. Visão da Implementação**

Esta seção descreve a estrutura geral do modelo de implementação, a divisão do software em camadas e todos os componentes significativos do ponto de vista da arquitetura.

**8.1 Visão Geral**

As camadas de visão, serviço, negócio e dados, que compõem o Probex, bem como as interfaces que permitem a comunicação entre elas, são mostradas no diagrama de componentes abaixo:

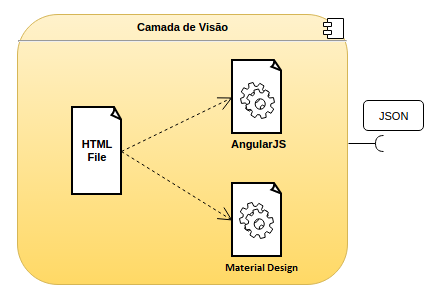


**8.2 Camadas**

A seguir são descritas em detalhes as camada do Probex, seus subsistemas e a natureza dos artefatos que os compõem. Cada camada possui responsabilidades e componentes bem definidos, comunicando-se com camadas adjacentes através de objetos padronizados.

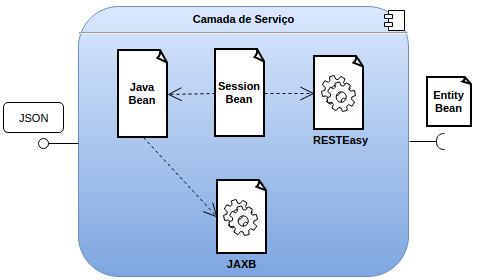
**8.2.1 Camadas de Visão**

A camada de visão é responsável por exibir as funcionalidades do sistema, capturar os eventos do usuário e realizar solicitações à camada de serviços. Os artefatos que compõe essa camada são arquivos HTML e Javascript, além da bibliotecas do AngularJS. As páginas HTML são estendidas através das diretivas do AngularJS. As solicitações a serviços são realizadas dentro dos arquivos Javascript que têm o papel de controladores em cada funcionalidade, utilizando o formato JSON como padrão de comunicação.



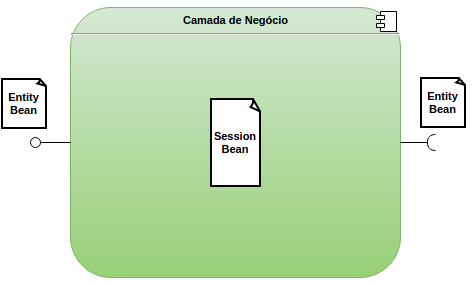
**8.2.2 Camadas de Serviço**

A camada de serviço provê uma interface de acesso à API dos serviços REST providos pela aplicação. Ela é composta por Session Beans, que implementam os serviços utilizando o RESTEasy, e por Java Beans, onde são mapeados, através do JAXB, os dados dos objetos JSON recebidos a partir de solicitações da camada de visão. O acionamento de objetos de negócio é realizado através de um Entity Bean, passado para a interface que interliga as camadas de serviço e negócio.



**8.2.3 Camadas de Negócio**

A camada de negócio encapsula todas as regras de negócio da aplicação. Essa camada é formada por Session Beans, que manipulam Entity Beans recebidos da camada de serviço e enviados à camada de dados.



**8.2.4 Camadas de Dados**

A camada de dados contém os Session Beans responsáveis por gerenciar a persistência dos Entity Beans recebidos da camada de negócio, utilizando a API do Hibernate para a realização do mapeamento objeto-relacional.

