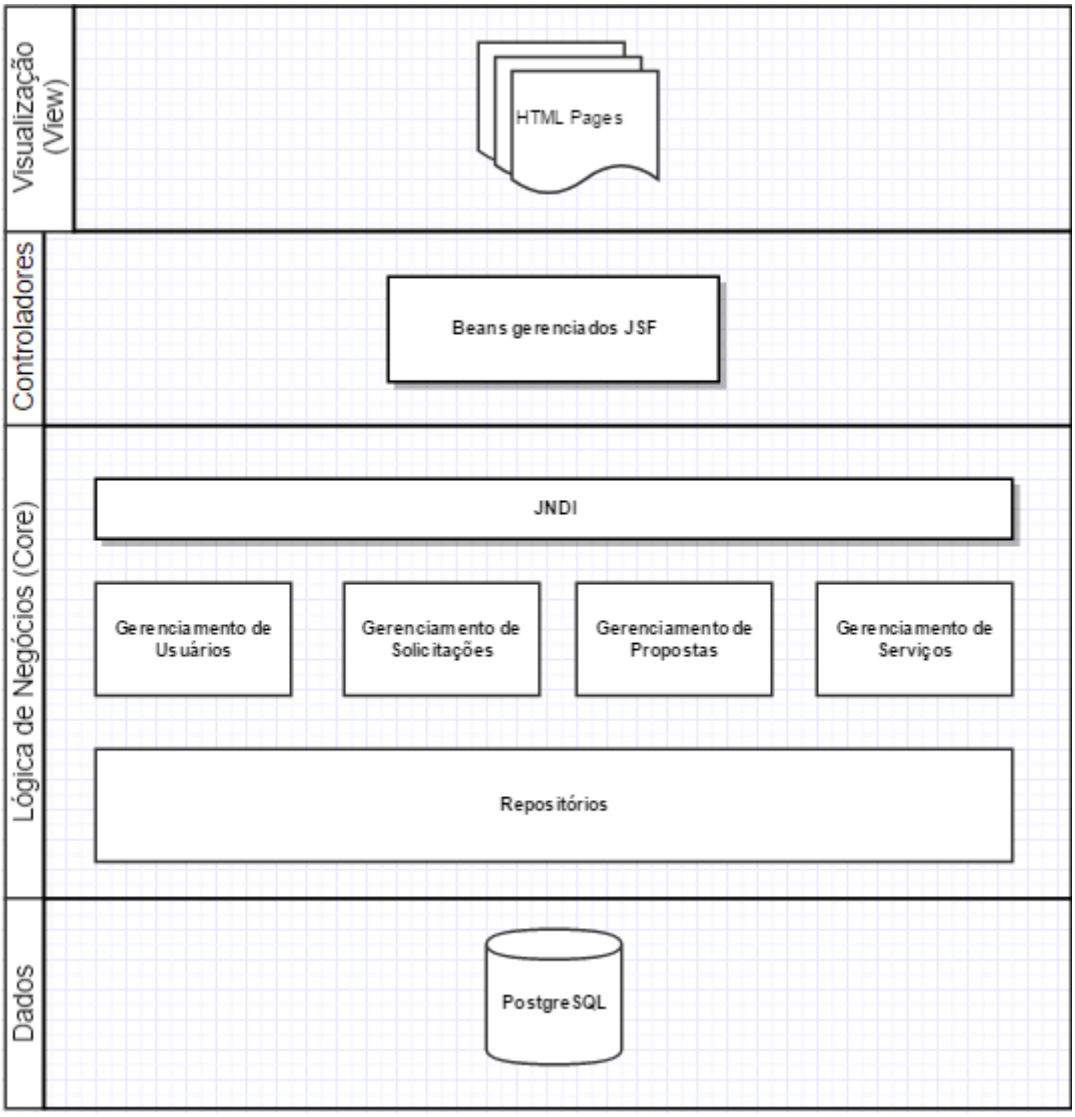
Quickserv
PROJETO ARQUITETURAL

Organização geral do sistema

O Quickserv trata-se de um sistema web que auxilia as pessoas na procura de profissionais qualificados de acordo com o serviço desejado, fornecendo uma área para clientes divulgarem suas solicitações por serviços e uma forma dos profissionais responder as solicitações oferecendo uma proposta adequada à procura.

O sistema desenvolvido seguirá o padrão de projeto *MVC* (*Model*, *View* and *Controller*) e terá suas camadas distribuídas entre 3 máquinas servidoras. As camadas possuem funções distintas e operam de maneira conjunta para atender os requisitos da aplicação.

Uma representação gráfica da arquitetura do *Quickserv* pode ser observada na Figura 1.



Camadas Físicas

Camada da Apresentação (View / Interface com Usuário):

Na camada de visão é fornecida uma interface gráfica de usuário (GUI) de fácil uso e acesso via *browser* que funciona em navegadores de internet populares como *Google Chrome*, *Mozila Firefox* e *Microsoft Edge*. Além da visualização esta camada é responsável pela realização de requisições por meio do protocolo *HTTP* à camada controladora para acesso as funcionalidades.

Para o desenvolvimento da interface com o usuário foram utilizadas tecnologias como *HTML*, *CSS* e *Javascript*.

Camada de Controle (Controller):

Esta camada faz o intermédio entre a camada de visualização (*View*) e a camada de lógica de negócio (*Core*), para tal, o *Quickserv* utiliza o *framework Java Server Faces (JSF)* para tratar todas as requisições vindas da camada de visão por meio do protocolo *HTTP*.

Camada de Lógica de Negócio (Core):

Como o próprio nome já diz, esta camada possui todas as características que fazem o sistema funcionar corretamente, todas as regras de negócio do *Quickserv* se concentra nesta camada. Implementada usando a tecnologia *EJB (Enterprise JavaBeans)*, essa camada recebe todas as requisições da camada controladora por meio de comunicação *RPC (Remote Procedure Call),* realiza o tratamento necessário de acordo com as regras de negócios definidas na elicitação de requisitos e devolve a resposta correspondente a camada controladora para então, ser devolvida a camada de visualização.

Para persistência de dados foi utilizado a especificação *JPA (Java Persistence API)* com a implementação do *Eclipselink*.

Possibilidade de Reúso

Com o uso da arquitetura proposta, a camada de regra de negócio (core) é isolada da camada de apresentação (*View*). Isto propicia um baixo acoplamento entre ambas as camadas e um alto grau de reúso uma vez que eu posso implementar diferentes camadas de visualização para a mesma aplicação sem alterar em nada a camada de regra de negócio.

Linguagens e tecnologias de desenvolvimento

Para o desenvolvimento do sistema utilizamos a linguagem Java, que apresenta vantagens como uma boa portabilidade, uma grande variedade de bibliotecas para reúso, uma grande comunidade de desenvolvedores e além disso, familiarização da equipe do projeto com a linguagem.

A linguagem *javascript* também foi utilizada para criação de *scripts* que serão executados no lado do cliente para tornar a interface de usuário ainda mais fácil de ser operada.

Ainda foram utilizados *frameworks* como:

- *Java Server Faces (JSF)* – na camada de controle para gerenciamento das requisições *HTTP* vindas da camada de visualização;
- *JUnit* – para testes de unidade;
- *Arquillian* – para testes de integração na plataforma Java EE;
- *DbUnit* – para testes que usam persistência na base de dados;
- *Mockito* – para testes que abstraiam o funcionamento de outras partes;

Além dos frameworks citados acima foram utilizados ferramentas como o *Selenium* para testes de sistema e componentes da plataforma *Java Enterprise Edition (JEE),* como *Enterpise JavaBeans (EJB)* e *Java Persistence API (JPA)*;

A persistência dos dados manipulados pelo Quickserv se dá por meio de um banco de dados relacional. O banco de dados utilizado foi o *PostgreSQL*.

Por fim, o gerenciamento de dependências é feito utilizando a ferramenta *Maven* e o controle de versões utilizado foi o *Git*.