**UNIVERSIDADE FEDERAL DE GOIÁS**

**INSTITUTO DE INFORMÁTICA**

**DOCUMENTO DE ARQUITETURA DE SOFTWARE**

**IV/UFG Elaboração: sistema web de elaboração de questões de prova**

**Goiânia - GO**

**2023**

**ÍNDICE**

[**1 Introdução**](#_heading=h.kduvfkcniham) **3**

[1.1 Finalidade](#_heading=h.8kqfx74gy7zz) 3

[1.2 Escopo](#_heading=h.72is72dasa5h) 3

[1.3 Definições, Acrônimos e Abreviações](#_heading=h.bcyx9n3j7j6y)3

[**2 Contexto da Arquitetura**](#_heading=h.9ppmulvieldr) **3**

[2.1 Restrições Arquiteturais](#_heading=h.c7di7qmsma46) 3

[**3 Representação da arquitetura**](#_heading=h.kzkgqw9di5du) **3**

[3.1 Detalhamento das camadas](#_heading=h.3pj6rsez3bz3) **4**

[**4. Tecnologias Utilizadas**](#_heading=h.am1qdkfhy5xd) **4**

[4.1 Vaadin](#_heading=h.2nm6c4vzg4n1) 4

[4.2 PostgreSQL](#_heading=h.iyaphvlgl0fj) 5

[4.3 Hibernate](#_heading=h.kk487r8ous1u) **5**

**1 Introdução**

**1.1 Finalidade**

O objetivo deste documento é registrar as definições sobre os aspectos arquiteturais do SEP. É destinado à equipe de Tecnologia da Informação do Instituto Verbena da Universidade Federal de Goiás, sobretudo à equipe de desenvolvimento, aos membros da Equipe Pedagógica e aos membros da Diretoria.

**1.2 Escopo**

Este documento se baseia na definição de requisitos do SEP, registrada no artefato "Documento de Especificacao de Requisitos.docx", constante da pasta "Documentacao - artefatos" do repositório no GitHub. Aqui serão registradas todas as visões de arquitetura, bem como as motivações para cada decisão e os requisitos que as embasaram.

**1.3 Definições, Acrônimos e Abreviações**

**API**: Interface de Programação de Aplicação, que é um conjunto de definições e protocolos para criar e integrar softwares de aplicações

**Browser**: Programa que permite a navegação pela internet

**CSS**: Sigla para a linguagem de estilo Folhas de Estilo em Cascata

Framework: estruturas compostas por um conjunto de códigos genéricos que permite o desenvolvimento de sistemas e aplicações

**HTML**: Linguagem de Marcação de Hipertexto

**HTTP**: Protocolo de Transferência de Hipertexto

**HTTPS**: Protocolo de Transferência de Hipertexto Seguro

**UI**: Interface do Usuário (User Interface).

**IV**: Instituto Verbena

**LGPD**: Lei Geral de Proteção de Dados

**RA**: Restrições Arquiteturais

**RNF**: Requisito Não Funcional

**SEP**: Sistema de Elaboração de Provas

**SGBD**: Sistema de Gerenciamento de Banco de Dados

**UFG**: Universidade Federal de Goiás

**2 Contexto da Arquitetura**

Este tópico descreve os requisitos e restrições utilizadas para a definição da arquitetura a ser implementada.

**2.1 Restrições Arquiteturais**

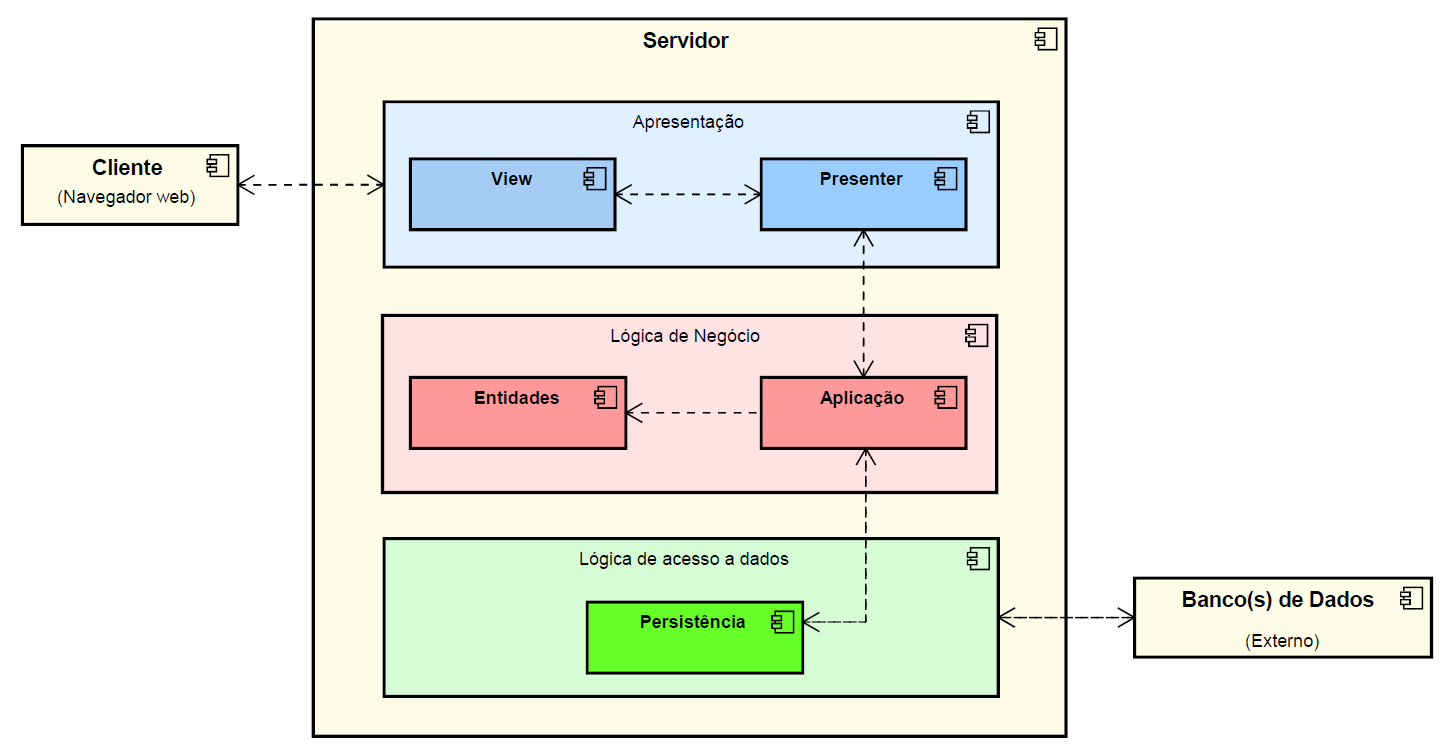
| **ID** | **Requisito fonte** | **Tipo do RA** |
| --- | --- | --- |
| RA\_1 | R.N.F. 01 | Portabilidade |
| RA\_2 | R.N.F. 02 | Disponibilidade |
| RA\_3 | R.N.F. 03 | Seg. da Informação(LGPD) |
| RA\_4 | R.N.F. 04 | Eficiência |
| RA\_5 | R.N.F. 05 | Dados |
| RA\_6 | R.N.F. 06 | Seg. da Informação |

*Tabela 1. Restrições Arquiteturais*

**3 Representação da arquitetura**

O modelo arquitetural proposto para o desenvolvimento do SEP é composto por 3 componentes, sendo cliente, servidor e banco de dados. O servidor, por sua vez, é composto por 3 camadas principais: sendo elas *Apresentação*, *Lógica de Negócio* e *Lógica de acesso a dados.* As camadas são compostas por componentes independentes, sendo cada um responsável por uma função específica dentro do software. Ainda, devido a esta arquitetura é possível que o desenvolvimento do sistema escalone, adicionando novas camadas e/ou componentes.

Na *Figura 1* pode ser visualizada a representação da arquitetura proposta, bem como seus componentes e a interação entre eles.



*Figura 1. Diagrama de Arquitetura do Software*

**3.1 Detalhamento das camadas**

Neste tópico, descreveremos as funcionalidades das camadas que irão compor o servidor do software.

**Apresentação**

A camada de apresentação é responsável por criar a UI do sistema, reagir aos eventos do usuário e comunicar com a camada de Lógica de negócio para requisitar ou atualizar dados.

* **View**: provê uma interface de acesso para a interação do cliente com o sistema. Se comunica apenas com a camada *Presenter*.
* **Presenter**: realiza a comunicação com a lógica de negócios. Responsável por tratar os eventos da camada view, realizar chamadas a camada de Aplicação e modificar a *View*.

**Lógica de Negócio**

A camada de Lógica de Negócios contém as entidades do domínio e as regras de negócio da aplicação.

* Entidade: contém os conceitos do domínio do problema (Classes).
* Aplicação: funções, validações e outras regras de negócio da aplicação. Esta camada se comunica com Entidade e Persistência.

**Lógica de Acesso a Dados**

Responsável pela persistência e interação com o banco de dados.

* Persistência: Recebe as requisições da camada de aplicação e contém os métodos responsáveis pelas requisições ao banco de dados.

**4. Tecnologias utilizadas**

**4.1 Vaadin**

Vaadin é um framework para desenvolvimento de aplicações web. Ele incorpora programação orientada a eventos e widgets, o que possibilita um modelo de programação similar à programação para GUI desktop.

A ferramenta inclui um amplo conjunto de componentes UI, como formulários, diálogos e tabelas. Estes componentes podem ser estendidos diretamente do código Java e, por meio da JVM, roda em todos os navegadores sem a necessidade de plugins.

As aplicações desenvolvidas com Vaadin são renderizadas no browser, como aplicação HTML. O motor de exibição (*Client-Side Engine*) é o responsável em realizar o trabalho necessário para apresentar visualmente os componentes e tratar a interação do usuário com a tela. O estado e comportamento da aplicação são tratados no servidor, que envia para o cliente somente os dados que são necessários para alterar a visualização dos componentes.

Utilizando Vaadin não é necessário definir os métodos de comunicação entre componentes, uma vez que eles são definidos pela arquitetura da ferramenta.

**4.2 PostgreSQL**

O PostgreSQL é um SGBD relacional *open source*, que possui a aplicação *pgAdmin*, consistente em uma interface para administração e desenvolvimento para o PostgreSQL. Logo, considerando se tratar de um SGDB *open source* e de fácil manipulação, justifica-se sua escolha.

**4.3 Hibernate**

Hibernate é uma implementação do JPA (Java Persistence API), que como o nome já diz, implementa uma API para interagir com o banco de dados e persistir os dados. Sua vantagem é o mapeamento intuitivo das classes e objetos Java para a lógica relacional do SQL, o que previne erros e diminui a verbosidade do código.

**4. Tecnologias Utilizadas**

Nesta seção apresentaremos as tecnologias escolhidas para o desenvolvimento deste projeto.

**4.1 Vaadin**

Vaadin é um framework Java para desenvolvimento de aplicações web. Ele incorpora programação orientada a eventos e widgets, o que possibilita um modelo de programação similar à programação para GUI desktop.

A ferramenta inclui um amplo conjunto de componentes UI, como formulários, diálogos e tabelas. Estes componentes podem ser estendidos diretamente do código Java e, por meio da JVM, roda em todos os navegadores sem a necessidade de plugins.

As aplicações desenvolvidas com Vaadin são renderizadas no browser, como aplicação HTML. O motor de exibição (*Client-Side Engine*) é o responsável em realizar o trabalho necessário para apresentar visualmente os componentes e tratar a interação do usuário com a tela. O estado e comportamento da aplicação são tratados no servidor, que envia para o cliente somente os dados que são necessários para alterar a visualização dos componentes.

Utilizando Vaadin não é necessário definir os métodos de comunicação entre componentes, uma vez que eles são definidos pela arquitetura da ferramenta.

**4.2 PostgreSQL**

O PostgreSQL é um SGBD relacional open source, que possui a aplicação pgAdmin, consistente em uma interface para administração e desenvolvimento para o PostgreSQL. Logo, considerando se tratar de um SGDB open source e de fácil manipulação, justifica-se sua escolha.

**4.3 Hibernate**