FinDev

Documento de Arquitetura de Software

Versão 1.4

**Autores**

Gabriel Colling

Gian Boschetti

Joice Cazanoski Gomes

José Augusto Accorsi

Joszef Barrionuevo

Pedro Bohlmann Cascaes Silva

Histórico de Revisões

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| 09/09/19 | 1.0 | Primeira versão | Joice |
| 10/09/19 | 1.1 | Revisões e soluções | Gabriel |
| 11/09/19 | 1.2 | Adicionando mais soluções | Pedro |
| 11/09/19 | 1.3 | Revisão Geral | Gian |
| 14/09/19 | 1.4 | Seção 8 Visão de Dados. | Gian |

Sumário

1. Introdução 4

1.1 Finalidade 4

1.2 Escopo 4

1.3 Definições, Acrônimos e Abreviações 4

2. Representação da Arquitetura 4

3. Metas e Restrições de Arquitetura 4

4. Visão de Casos de Uso 5

5. Visão Lógica 5

5.1 Divisão em Pacotes 5

6. Visão de Implantação 6

7. Visão de Implementação 7

7.1 Soluções utilizadas 7

7.1.1 Typescript 7

7.1.2 React.js 7

7.1.3 Node.js 7

7.1.4 Express 7

7.1.5 PostgreSQL 7

7.1.6 Redux 7

7.1.7 Redux - Saga 7

7.1.8 I18N 7

# Introdução

## Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

## Escopo

Este documento de Arquitetura de Software se aplica ao sistema **FinDev**. Esse sistema foi elaborado pensando nas dificuldades das personas dos nossos usuários (Freelancer e Empregador), visando abranger todos os fluxos do processo principal. Assim, contempla o fluxo do empregador, que consiste em gerenciamento de projetos e pessoas interessadas, incluindo criação, atualização e fechamento de projetos, além do controle dos freelancers interessados e contratados. Ainda, contempla o fluxo do freelancer, que consiste em gerenciamento do perfil, inclusão e atualização de habilidades e de projetos em que manifestou interesse.

## Definições, Acrônimos e Abreviações

Esta subseção deve apresentar as definições de todos os termos, acrônimos e abreviações necessários para a correta interpretação do Documento de Arquitetura de Software. Essas informações podem ser fornecidas mediante referência ao Glossário do projeto.

* **API** – Application Programming Interface
* **SGBD** - Sistema Gerenciador de Banco de Dados

# Representação da Arquitetura

Esta seção descreve qual é a arquitetura de software do sistema atual e como ela é representada. Nas Visões de Casos de Uso, Lógica, do Processo, de Implantação e de Implementação, este documento enumera as visões necessárias e, para cada uma delas, explica os tipos de elementos do modelo que contém.

Este documento apresenta a arquitetura como uma série de visões: visão de casos de uso, visão lógica, visão de processos, visão de implantação e visão de implementação. Essas visões são apresentadas como Modelos do Astah e utilizam a Linguagem Unificada de Modelagem (UML).

* No momento o backend e frontend está sendo rodado localmente nas máquinas em que estamos usando para codificar. Estamos pensando na possibilidade de armazenar em um servidor para ficar disponível para uso.
* Banco de Dados PostgresSQL será armazenado em nuvem da KingHost.

# Metas e Restrições de Arquitetura

Esta seção descreve os requisitos de software e os objetivos que têm um impacto significativo na arquitetura, como proteção, segurança, privacidade, uso de um produto desenvolvido internamente e adquirido pronto para ser usado, portabilidade, distribuição e reutilização.

Existem algumas restrições de requisito e de sistema principais que têm uma relação significativa com a arquitetura. São elas:

* + A linguagem de desenvolvimento utilizada no Frontend será Typescript usando React como framework;
  + A linguagem de desenvolvimento utilizada no Backend será Typescript usando Node como interpretador e Express como framework;
  + O Sistema Gerenciador de Banco de Dados escolhido para a aplicação será o PostegreSQL;

# Visão de Casos de Uso

Esta seção lista os casos de uso ou cenários do modelo de casos de uso que representam uma funcionalidade central e significativa do sistema final ou se têm uma ampla cobertura de arquitetura, ou seja, que experimenta vários elementos de arquitetura e enfatizam ou ilustram um determinado ponto complexo da arquitetura.

Os Casos de Uso do sistema FinDev são listados a seguir. Os Casos de Uso em negrito são destacados porque são muito importante para a arquitetura:

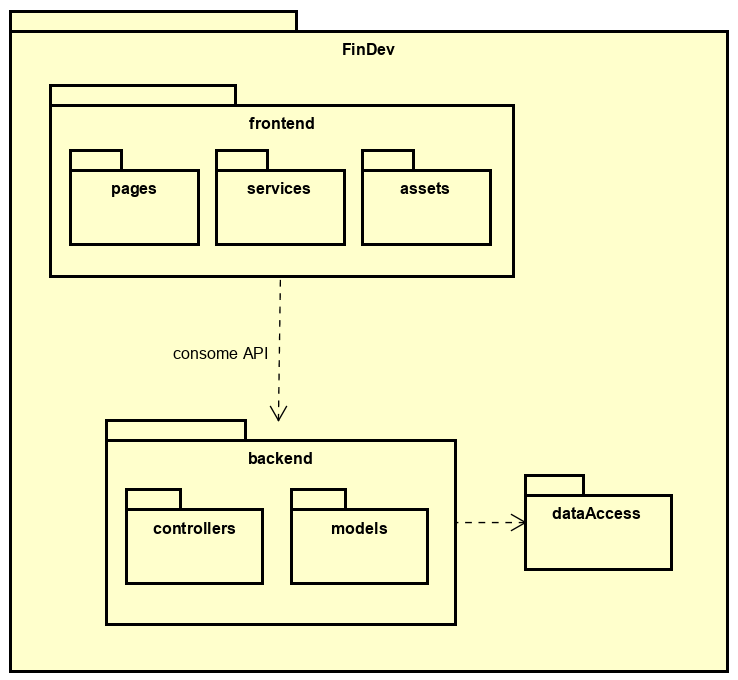
* Geral
  + Cadastrar Usuário;
  + **Login;**
  + Atualizar Perfil do Usuário;
  + Notificações;
  + Comunicação entre Freelancer e Empregador;
* Empregador
  + **Gerenciar Projetos;**
  + Gerenciar Freelancers interessados;
  + **Contratar Freelancer;**
* Freelancer
  + Buscar projetos;
  + **Manifestar interesse em um projeto;**
  + Cadastrar habilidades;

# Visão Lógica

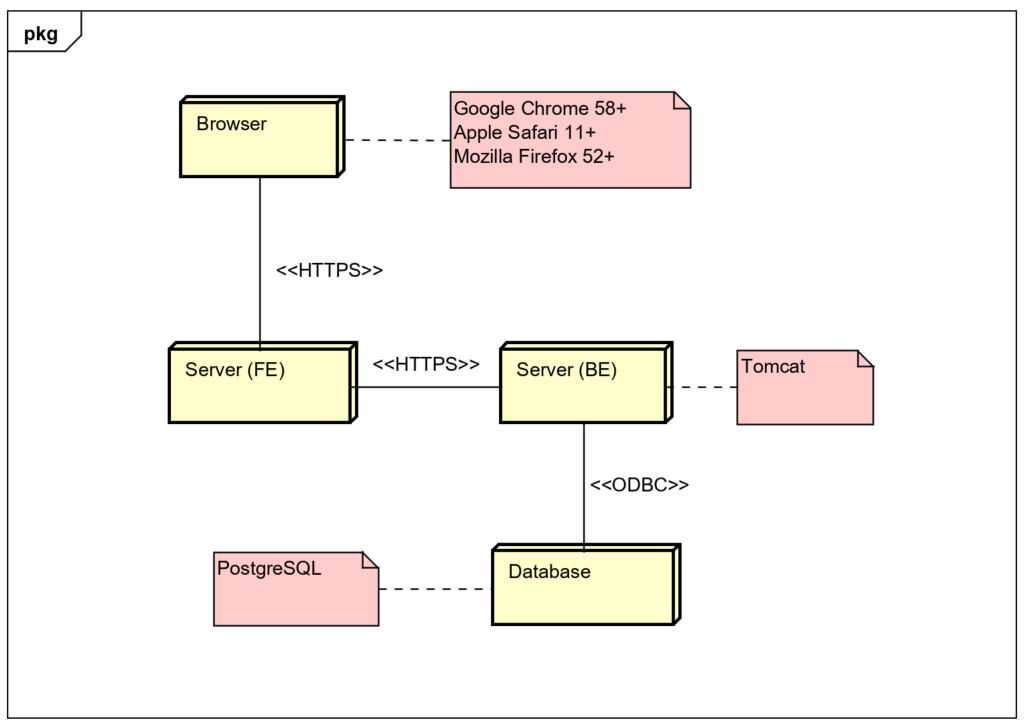
Esta seção descreve as partes significativas do ponto de vista da arquitetura do modelo de design, como sua divisão em subsistemas e pacotes. Além disso, para cada pacote significativo, ela mostra sua divisão em classes e utilitários de classe. Apresenta as classes significativas do ponto de vista da arquitetura e descreve suas responsabilidades, bem como alguns relacionamentos, operações e atributos de grande importância.

## Divisão em Pacotes

Nesta seção serão apresentadas as camadas da arquitetura proposta. Serão descritas as responsabilidades de cada camada quais tecnologias devem ser aplicadas a cada uma delas.



# Visão de Implantação



# Visão de Implementação

A visão de implementação aborda a arquitetura sobre a perspectiva do projeto estrutural dos componentes do sistema, como o sistema e cada um dos seus componentes serão organizados em termos de diretórios e como o sistema, será empacotado para publicação (deployment).

A visão de implementação retrata a composição física da implementação em termos de subsistemas e elementos (diretórios, arquivos, dados, executáveis). Geralmente essas camadas de implementação se ajustam às camadas definidas na visão lógica.

## Soluções utilizadas

### Typescript

Para desenvolvimento web de frontend e backend utilizaremos o Typescript, que é um superconjunto do Javascript, mas com fortemente tipado, com interfaces e tipos genéricos.

### React.js

Para a criação das interfaces de usuário, utilizaremos o React, que é uma biblioteca Javascript de código aberto, além de ser a mais utilizada dessa gama no mercado atualmente e ser facilmente extensível para mobile através da utilização do React Native.

### Node.js

O Node é o interpretador de Javascript que utilizaremos para possibilitar o desenvolvimento de backend usando essa linguagem de programação, tendo como grande vantagem a sua alta escalabilidade, tratando diversas requisições simultaneamente.

### Express

Express é um framework para desenvolvimento em node, que possibilita e facilita a construção de aplicações web e APIs.

### PostgreSQL

PostgreSQL é um SGBD do tipo objeto-relacional, de código aberto e com uma das maiores e mais completas gamas de funcionalidades dentre os modelos disponíveis.

### Redux

Redux é uma biblioteca Javascript que tem como objetivo funcionar como um estado global da aplicação onde qualquer componente possa se conectar e ter acesso aos dados.

### Redux - Saga

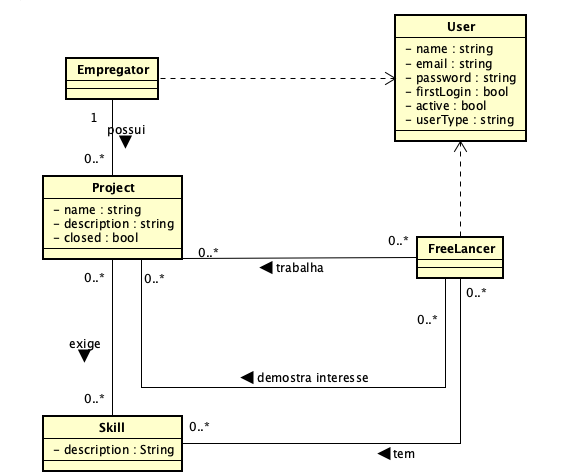
Mais conhecido por Saga, ele é uma biblioteca que escuta ações do Redux e chama funções que podem serem executadas de forma assíncrona ou síncrona.

### I18N

É um modulo que ajuda na utilização de textos em múltiplos idiomas e localizações. Com a criação da configuração do modulo é possível automaticamente trocar o idioma da página de acordo com o idioma do navegador

# Visão de Dados

## Modelo de objetos persistentes



## Estratégias

Descrição das estratégias e padrões utilizados no mapeamento:

* Nos relacionamentos N para N criamos uma terceira tabela para armazenar as PKs e FKs.
* Já nos relacionamentos 1 para N, a PK do lado 1 fica armazenado como FK no lado N da relação.

## Modelo relacional

A persistência das informações da aplicação em banco de dados será baseada no modelo a seguir. As entidades (tabelas) constam em negrito, com os nomes das colunas que as compõem listados entre parênteses. Colunas sublinhadas continuamente compõem chaves primárias, enquanto as sublinhadas de forma alternada e/ou em itálico são chaves estrangeiras.

* **User**  (id, name, email, password, active, firstLogin, userType)
* **Project** (id, name, description, closed, id*OwnerUser*)
* **Skill** (id, description)
* **SkillUser** (id, *idUser*, *idSkill*)
* **SkillProject** (id, *idProject*, *idSkill*)
* **UserWorkingProject (**id, *idProject*, *idUser*)
* **User**In**terestProject (**id, *idProject*, *idUser*)