IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

| | |
|---|---|
| Bruna da Silva Pires | SP3056651 |
| Daniel Roberto Pereira | SP3046702 |
| Igor Nathan de Oliveira Rocha | SP305263X |
| Leonardo Marques da Silva | SP3052591 |
| Lucas Lima de Santana | SP3046559 |
| Marcelo Carlos Olimpio Junior | SP3046583 |


# estagiei
# *Website* de vagas de estágio

São Paulo - SP - Brasil

2022

IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

| | |
|---|---|
| Bruna da Silva Pires | SP3056651 |
| Daniel Roberto Pereira | SP3046702 |
| Igor Nathan de Oliveira Rocha | SP305263X |
| Leonardo Marques da Silva | SP3052591 |
| Lucas Lima de Santana | SP3046559 |
| Marcelo Carlos Olimpio Junior | SP3046583 |


# estagiei

# *Website* de vagas de estágio

Desenho de aplicação para desenvolvimento na disciplina de Projeto Integrado I no 1º semestre de 2022.

Professor: Carlos Henrique Veríssimo Pereira

IFSP - Instituto Federal de Educação, Ciência e Tecnologia
Câmpus São Paulo

Tecnologia em Análise e Desenvolvimento de Sistemas

PI1A5 - Projeto Integrado I

São Paulo - SP - Brasil

2022

# Lista de ilustrações

# Lista de quadros

# Lista de abreviaturas e siglas

| | |
|---|---|
| API | *Application Programming Interface* - Interface de Programação de Aplicativos - Citado em 14, 15, 24, 29, 33 |
| CLT | Consolidação das Leis do Trabalho - Citado em 12 |
| CNPJ | Cadastro Nacional da Pessoa Jurídica - Citado em 29 |
| CSS | *Cascading Style Sheets* - Folhas de Estilo em Cascata - Citado em 13 |
| HTML | *Hypertext Markup Language* - Linguagem de Marcação de Hipertexto - Citado em 13, 25 |
| HTTP | *Hypertext Transfer Protocol* - Protocolo de transferência de hypertexto - Citado em 14, 15, 16 |
| HTTPS | *Hypertext Transfer Protocol Secure* - Protocolo seguro de transferência de hypertexto - Citado em 15, 29 |
| JSON | *JavaScript Object Notation* - Notação de Objeto JavaScript - Citado em 15, 16 |
| LGPD | Lei Geral de Proteção de Dados - Citado em 20, 29 |
| LIBRAS | Língua Brasileira de Sinais - Citado em 25, 34 |
| MVP | *Minimum Viable Product* - Produto Mínimo Viável - Citado em 11, 23 |
| POC | *Prove of Concept* - Prova de Conceito - Citado em 11, 23 |
| REST | *Representational State Transfer* - Transferência de Estado Representacional - Citado em 14, 15, 33 |
| RH | Recursos Humanos - Citado em 23 |
| SPA | *Single Page Application* - Aplicação de Página Única - Citado em 2, 13, 16 |
| SSD | *Solid-State Drive* - Unidade de Estado Sólido - Citado em 30 |
| SSO | *Single Sign-On* - Login único - Citado em 23, 24 |
| URL | *Universal Resource Locator* - Localizador universal de recurso - Citado em 33 |
| USD | *United States Dollar* - Dólares Americanos - Citado em 30, 31 |

# Sumário

# 1 Introdução

Nesse capítulo serão mostrados os principais pontos do nosso projeto, os objetivos e quais os problemas que queremos solucionar com nossa aplicação.

## 1.1 Justificativa

Existe, na contemporaneidade, uma grande dificuldade em adquirir experiência profissional através da prática de estágio, muitas vezes obrigatória no projeto pedagógico de cursos das universidades. Tal problema se dá por meio das plataformas que disponibilizam tais vagas, as quais frequentemente exigem habilidades dos candidatos além do devidamente esperado para uma vaga de estágio. É também notável que existe uma certa dificuldade de conexão entre a empresa e o candidato, que muitas vezes não obtém o retorno sobre o processo de seleção da vaga.

## 1.2 Proposta de solução

Tendo em vista os problemas anteriormente descritos, *estagiei* é um sistema para aproximar novos estudantes e empresas com vagas de estágio disponíveis, de modo que os candidatos possam receber indicações de vagas condizentes com seu perfil e empresas recebam recomendações de candidatos possivelmente adequados às vagas anunciadas.

## 1.3 Objetivos

O objetivo principal da nossa solução é promover um meio de conexão mais direto entre os estudantes em busca de estágio e empresas que buscam interessados em suas vagas de estágio alinhados com o perfil buscado. Através do sistema de recomendações, tantos os estudantes quanto as empresas têm papel ativo no processo de encontrar um(a) estudante/vaga ideal, cujas as competências e perfil sejam condizentes com o que é procurado.

A partir do nosso objetivo principal, podemos listar alguns objetivos mais práticos da nossa solução:

- Ser um *website* de fácil usabilidade, onde os estudantes encontrem vagas sem passar por longos processos seletivos.

- Ser uma aplicação onde de fato os estudantes encontrem vagas que condizem com a realidade de um estagiário.
- Pensar sempre na experiência dos usuários, de modo que a aplicação seja simples e efetiva ao mesmo tempo.

## 1.4 Análise de Concorrentes

Para a elaboração da proposta, foram verificadas algumas soluções já existentes no mercado. A partir disso, as soluções que mais se assemelham com a proposta são *Companhia de Estágios*, *Cia de Talentos* e *Nube*. Com base neste levantamento, podemos observar algumas intersecções de funcionalidades oferecidas. O Quadro 1 permite uma melhor visualização deste levantamento.

Quadro 1 – Comparação dos aplicativos concorrentes

| **Funcionalidades** | **Cia de Estágios** | **Cia de Talentos** | **Nube** | **Nosso Proj.** |
|---|---|---|---|---|
| Login/Cadastro. | x | x | x | x |
| Aplicar em uma vaga. | x | x | x | x |
| Notificação a cada mudança do status no processo seletivo. | | | x | x |
| Recomendação de vagas e/ou empresas aos estudantes de acordo com as suas características. | | | | x |
| Recomendação de estudantes mais compatíveis com as vagas registradas pelas empresas, de acordo com as características da vaga e da empresa. | | | | x |
| Simplificação de contato via *WhatsApp*. | | | | x |
| Denúncias de vagas incoerentes com a realidade. | | | | x |
| *Feedback* de empresas pós-entrevista. | | | | x |

Fonte: Os Autores

# 2 Planejamento e Gerenciamento do Projeto

Neste capítulo abordaremos a metodologia e ferramenta da gestão da equipe e do projeto, os papéis dos integrantes da equipe e informações a cerca do cronograma sendo seguido no desenvolvimento do projeto e sua documentação.

## 2.1 Gestão e Desenvolvimento do Projeto

A equipe decidiu por utilizar a metodologia ágil Scrum, juntamente com a ferramenta de gerenciamento Jira Software. O Scrum possui três fases, uma inicial de planejamento geral, uma intermediária de produção e uma final de encerramento. A fase intermediária se trata de uma série de ciclos, onde em cada ciclo é desenvolvido atividades/funcionalidades a serem entregues/incrementadas. Estes ciclos são chamados de Sprints, cuja duração é fixa e a equipe decidiu por durar uma semana (7 dias). Todas as atividades, elementos e artefatos que precisarão ser produzidos serão organizados, monitorados e atribuídos aos membros da equipe via Jira Software, onde pode-se verificar o status da atividade (Não Iniciado, Em Progresso e Concluído), assim como marcar prazos.

## 2.2 Organização da equipe

Após avaliarmos as principais competências de cada integrante da equipe, resolvemos separar as tarefas de cada um como indicado no Quadro 2.

Quadro 2 – Divisão de responsabilidades da equipe.

| **Responsabilidade** | **Bruna** | **Daniel** | **Igor** | **Leonardo** | **Lucas** | **Marcelo** |
|---|---|---|---|---|---|---|
| Back-End. | | | x | x | | x |
| Front-End. | x | x | | x | x | |
| Banco de Dados. | | x | x | | | |
| Blog. | x | x | x | x | x | x |
| Documentação. | x | x | x | x | x | x |
| Design. | | | | | x | |
| Gestão. | x | | | | | |

Fonte: Os Autores

Considerando os papéis inerentes ao Scrum e as responsabilidades expostas no Quadro 2, o papel do Scrum Master será desempenhado pela integrante Bruna da Silva Pires, já a equipe de desenvolvimento será composta por todos os integrantes da equipe, sem exceção.

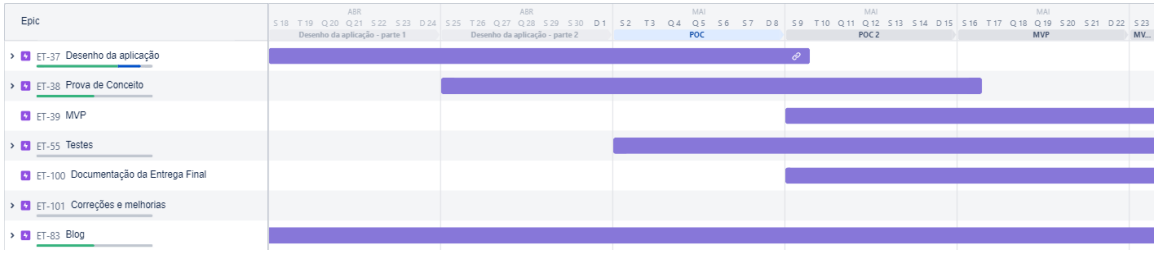
## 2.3 Cronograma

A princípio temos uma organização dos macro-itens (Epics) que precisam ser desenvolvidos durante o projeto, dentro dos quais estipulamos as tarefas a serem feitas. Por meio do Jira Software podemos ter uma visão geral do andamento das Epics e os prazos, além das Sprints planejadas e a atual.

Figura 1 – Roteiro Geral

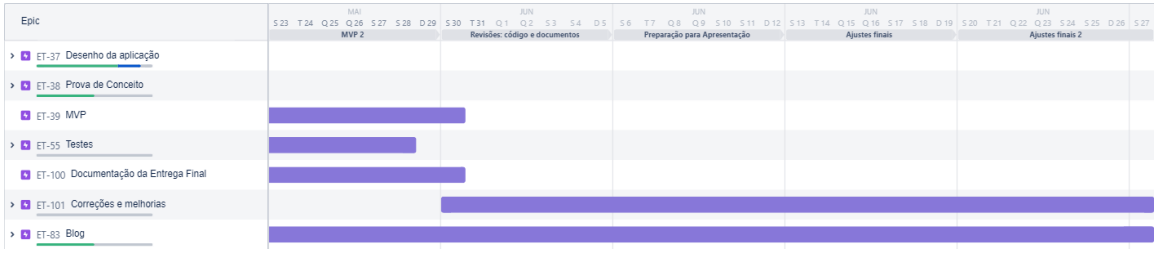
Fonte: Roteiro via ferramenta Jira Software

Figura 2 – Roteiro Geral - Detalhe Inicial




Fonte: Roteiro via ferramenta Jira Software

Figura 3 – Roteiro Geral - Detalhe Final




Fonte: Roteiro via ferramenta Jira Software

Apresentamos em Quadro 3 as Sprints e algumas informações expostas em Figura 1, Figura 2 e Figura 3.

Quadro 3 – Cronograma de Sprints

| **Sprint** | **Data Inicial** | **Data Final** | **Descrição** | **Status** |
|---|---|---|---|---|
| Desenho da aplicação 1 | 18/04/22 | 25/04/22 | Elaboração da documentação do Desenho da Aplicação. | Concluída |
| Desenho da aplicação 2 | 25/04/22 | 02/05/22 | Continuação da elaboração do Desenho da Aplicação. Planejamento para a *Prove of Concept* (POC). | Concluída |
| POC | 02/05/22 | 09/05/22 | Finalização do Desenho da Aplicação. Início do desenvolvimento dos itens da POC | Em progresso |
| POC 2 | 09/05/22 | 16/05/22 | Continuação do desenvolvimento dos itens da POC. | Não iniciada |
| MVP | 16/05/22 | 23/05/22 | Aproveitamento do que foi desenvolvido para a POC com melhorias e ampliação conforme possível para o *Minimum Viable Product* (MVP). | Não iniciada |
| MVP 2 | 23/05/22 | 30/05/22 | Continuação do trabalho no desenvolvimento do MVP. | Não iniciada |
| Revisões: código e documentos | 30/05/22 | 06/06/22 | Finalização e revisão tanto do desenvolvimento quanto da documentação. | Não iniciada |
| Preparação para a Apresentação | 06/06/22 | 13/06/22 | Organização e planejamento da apresentação do projeto e sua documentação. | Não iniciada |
| Ajustes finais | 13/06/22 | 20/06/22 | Ajustes a serem feitos para correção e/ou melhoria do projeto apresentado. | Não iniciada |
| Ajustes finais 2 | 20/06/22 | 27/06/22 | Finalização dos ajustes finais para a entrega definitiva do projeto no semestre. | Não iniciada |

Fonte: Os Autores

# 3 Revisão da Literatura

Nesta capítulo buscamos explicitar conceitos e informações relevantes para o desenvolvimento da nossa proposta de solução *estagiei*, um *website* de vagas de estágio.

## 3.1 Estágio

### 3.1.1 Definição

De acordo com a lei nº 11.788, de 25 de setembro de 2008, define-se estágio da seguinte forma:

> Art. 1º Estágio é ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho, que visa à preparação para o trabalho produtivo de educandos que estejam freqüentando o ensino regular em instituições de educação superior, de educação profissional, de ensino médio, da educação especial e dos anos finais do ensino fundamental, na modalidade profissional da educação de jovens e adultos. (BRASIL, 2008)

### 3.1.2 Tipos de estágio

Os estágios podem ser obrigatórios ou não-obrigatórios, dependendo do que foi previsto no projeto pedagógico do curso no qual o estudante está matriculado. O estágio do tipo obrigatório se caracteriza pelo requisito de cumprimento de uma determinada quantidade de horas estágio, juntamente com a aprovação nas disciplinas do curso, para a obtenção de diploma. O estágio não-obrigatório é opcional e as horas cumpridas são acrescidas às carga obrigatória do curso. (BRASIL, 2008)

### 3.1.3 Carga horária

O estágio não é regido pela Consolidação das Leis do Trabalho (CLT), assim possui sua própria especificação de jornada e carga horária. De acordo com o Art. 10 (BRASIL, 2008), a jornada do estágio é definida em um acordo entre a escola e a empresa, ressaltando que não pode ultrapassar:

> I – 4 (quatro) horas diárias e 20 (vinte) horas semanais, no caso de estudantes de educação especial e dos anos finais do ensino fundamental, na modalidade profissional de educação de jovens e adultos;
> II – 6 (seis) horas diárias e 30 (trinta) horas semanais, no caso de estudantes do ensino superior, da educação profissional de nível médio e do ensino médio

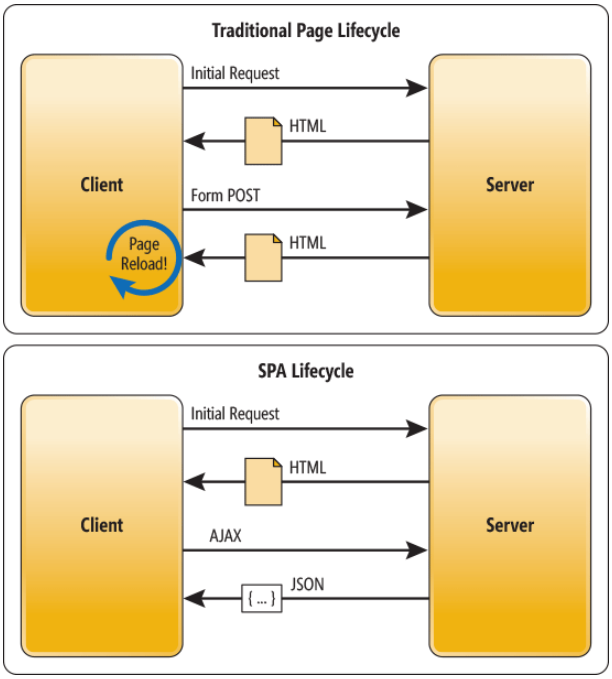
> regular.
> § 1° O estágio relativo a cursos que alternam teoria e prática, nos períodos em que não estão programadas aulas presenciais, poderá ter jornada de até 40 (quarenta) horas semanais, desde que isso esteja previsto no projeto pedagógico do curso e da instituição de ensino.
> § 2° Se a instituição de ensino adotar verificações de aprendizagem periódicas ou finais, nos períodos de avaliação, a carga horária do estágio será reduzida pelo menos à metade, segundo estipulado no termo de compromisso, para garantir o bom desempenho do estudante.(BRASIL, 2008)

## 3.2 Single-page Application (SPA)

*Single Page Application* (SPA) é uma implementação de aplicação web que carrega uma única página, um único arquivo do tipo *Hypertext Markup Language* (HTML), então de modo dinâmico modifica e atualiza o conteúdo da página de acordo com as ações do usuário, resultando em ganho de performance e melhor experiência de usuário. (MDN, 2021)

Em uma SPA toda a codificação HTML, JavaScript e CSS é carregada de uma vez logo no primeiro acesso ou os recursos são recuperados (carregados) e incorporados à página conforme a necessidade, apenas o que for necessário, geralmente em resposta à interação do usuário (WIKIPEDIA, 2022), como ilustrado na Figura 4.

Figura 4 – Ciclo de vida: página web tradicional X SPA




Fonte: (WASSON, 2015)

## 3.3 Application Programming Interface (API)

*Application Programming Interface* (API), Interface de Programação de Aplicativos, é um sistema intermediário de mediação que permite a comunicação entre outros sistemas/softwares/aplicações a partir de um conjunto de protocolos e definições, ou seja,

> APIs funcionam como se fossem contratos, com documentações que representam um acordo entre as partes interessadas. Se uma dessas partes enviar uma solicitação remota estruturada de uma forma específica, isso determinará como a aplicação da outra parte responderá.(REDHAT, 2017)

Essa comunicação possibilita uma integração entre produtos e serviços sem que seus desenvolvedores conheçam como o software alheio foi feito, basta saberem as regras para requisitar uma informação e como tratar a resposta. É certo que há formas de incluir segurança no tráfego de informações, essencialmente através do gerenciamento da API com gateways (REDHAT, 2017). Desde modo, podemos entender API como

> [...] um mediador entre os usuários ou clientes e os recursos ou serviços web que eles querem obter. As APIs também servem para que organizações compartilhem recursos e informações e, ao mesmo tempo, mantenham a segurança, o controle e a obrigatoriedade de autenticação, pois permitem determinar quem tem acesso e o que pode ser acessado. (REDHAT, 2017)

### 3.3.1 API REST

*Representational State Transfer* (REST) é um estilo de arquitetura com um conjunto de restrições (REDHAT, 2020). Uma API que segue todas as seis restrições é chamada de API RESTful (FIELDING, 2000). Como não se trata de um protocolo específico, não há um padrão de implementação das restrições REST, que seguem:

- Arquitetura cliente-servidor: a arquitetura REST é composta por clientes, servidores e recursos. Ela lida com as solicitações via HTTP.
- Sem monitoração de estado: nenhum conteúdo do cliente é armazenado no servidor entre as solicitações. Em vez disso, as informações sobre o estado da sessão são mantidas com o cliente.
- Capacidade de cache: o armazenamento em cache pode eliminar a necessidade de algumas interações entre o cliente e o servidor.
- Sistema em camadas: as interações entre cliente e servidor podem ser mediadas por camadas adicionais. Essas camadas podem oferecer recursos extras, como balanceamento de carga, caches compartilhados ou segurança.
- Código sob demanda (opcional): os servidores podem ampliar a funcionalidade de um cliente por meio da transferência de códigos executáveis.
- Interface uniforme: essa restrição é essencial para o design de APIs RESTful e inclui quatro vertentes:

> -Identificação de recursos nas solicitações: os recursos são identificados nas solicitações e separados das representações retornadas para o cliente.
>
> -Manipulação de recursos por meio de representações: os clientes recebem arquivos que representam recursos. Essas representações precisam ter informações suficientes para permitir a modificação ou exclusão.
>
> -Mensagens autodescritivas: cada mensagem retornada para um cliente contém informações suficientes para descrever como ele deve processá-las.
>
> -Hipermídia como plataforma do estado das aplicações: depois de acessar um recurso, o cliente REST pode descobrir todas as outras ações disponíveis no momento por meio de hiperlinks. (REDHAT, 2017)

Caso não seja o objetivo criar uma API RESTful, as restrições da arquitetura REST podem ser implementadas conforme a necessidade, tornando o desenvolvimento da API mais fácil por não haver exigências rígidas, como um formato específico para a informação de resposta às requisições via *Hypertext Transfer Protocol* (HTTP) ou *Hypertext Transfer Protocol Secure* (HTTPS), por exemplo (REDHAT, 2020). Porém, ainda que não haja uma obrigatoriedade, o formato *JavaScript Object Notation* (JSON) é o mais utilizado, pois é de fácil manipulação, organização e inteligível para pessoas e máquinas.

# 4 Desenvolvimento da Aplicação

Neste capítulo apresentaremos a arquitetura do *estagiei*, seu escopo, integrações, questões de segurança, privacidade e legislação, assim como itens de manutenibilidade e viabilidade financeira.

## 4.1 Arquitetura

Para o desenvolvimento do projeto, e tendo em vista que será construída uma aplicação web de página única, utilizaremos de ferramentas que cerceiam o ecossistema de SPA. Para isso, teremos a divisão do projeto em frontend e backend de modo que eles se comuniquem via protocolo HTTP com requisições e respostas no formato JSON. Para o desenvolvimento do frontend utilizaremos TypeScript por meio da biblioteca React; o backend será desenvolvido utilizando Java com o micro framework Spring Boot.

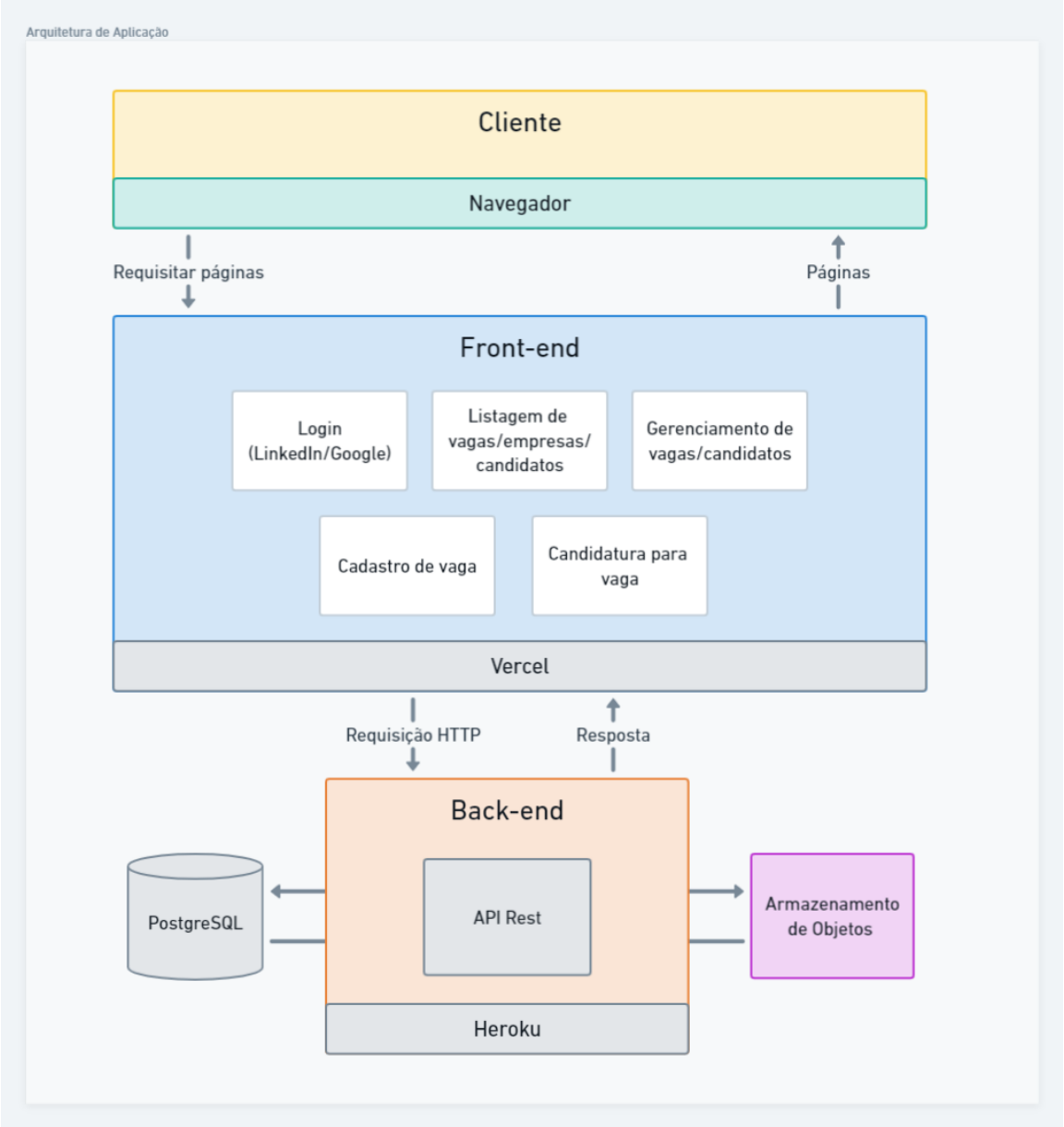
Em relação ao deploy das aplicações, o frontend será hospedado na plataforma Vercel, que é primariamente voltada para JavaScript, proporcionando uma melhor agilidade de desenvolvimento, enquanto o backend será hospedado no Heroku, que é uma plataforma como serviço de fácil manuseio e que nos permitirá ter um maior foco no desenvolvimento do projeto. Através do Heroku podemos também fazer a utilização do PostgreSQL por meio do serviço de apoio Heroku Postgres.

Ademais, se for necessário o armazenamento de objetos como arquivos ou imagens, utilizaremos a plataforma Cloudinary, principalmente por sua fácil integração com a linguagem de programação Java através de bibliotecas.

### 4.1.1 Diagramas de arquitetura

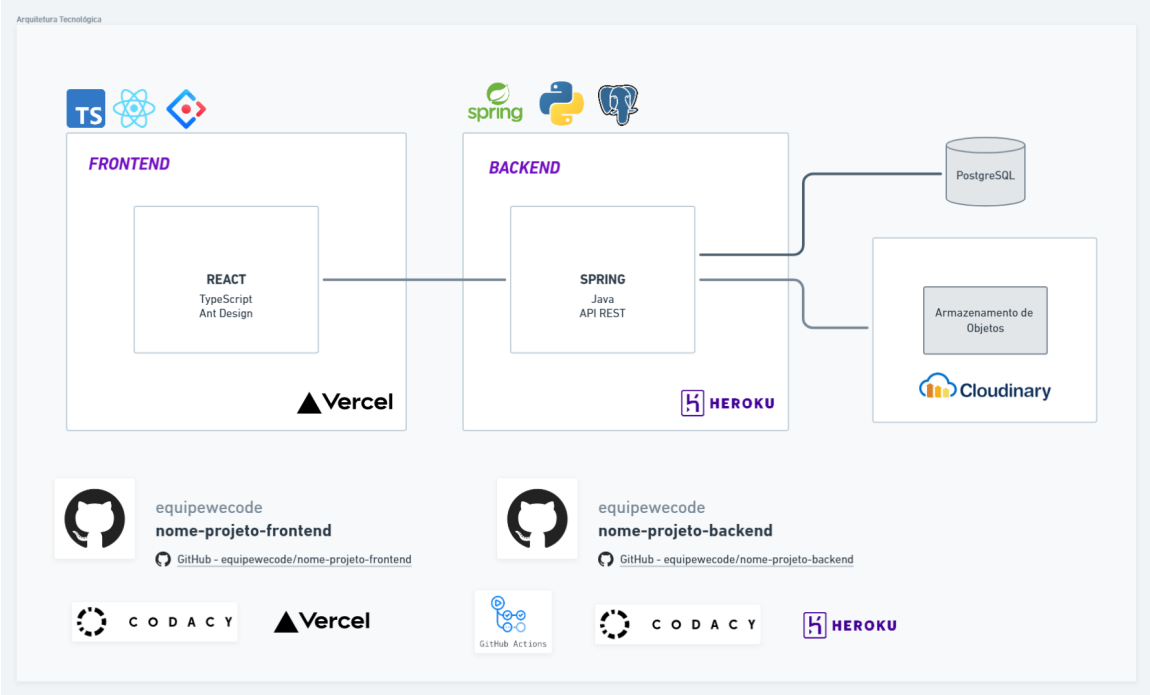
Os diagramas Figura 5, Figura 6 e Figura 7 ilustram de modo geral a arquitetura pensada para a solução proposta, utilizando das tecnologias já citadas.

Figura 5 – Arquitetura de Aplicação



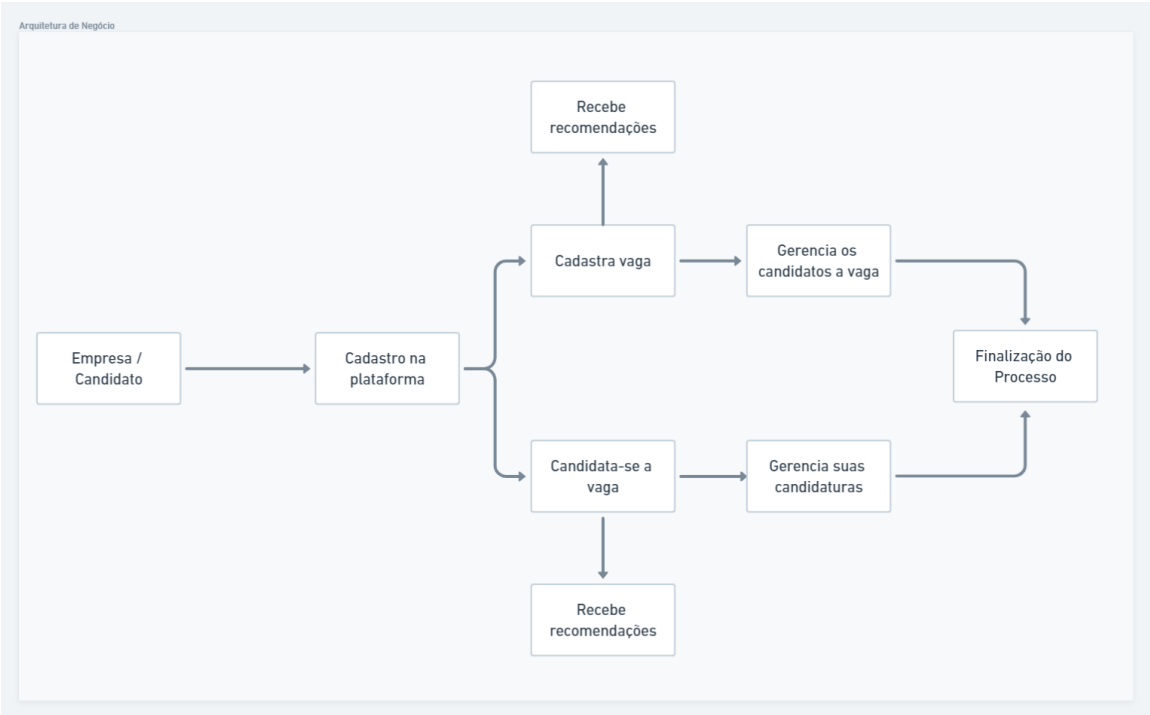

Fonte: Produzido pelos autores utilizando a ferramenta *Whimsical*

Figura 6 – Arquitetura Tecnológica




Fonte: Produzido pelos autores utilizando a ferramenta *Whimsical*

Figura 7 – Arquitetura de Negócios




Fonte: Produzido pelos autores utilizando a ferramenta *Whimsical*

## 4.2 Escopo

Neste tópico abordaremos os casos de uso da aplicação (forma de descrever uma funcionalidade do sistema); diagrama de requisitos (identificação das funcionalidades a serem implementadas); histórias de usuário (descrição das necessidades do usuário); e definição de entregas (quais funcionalidades estarão disponíveis nas principais entregas).

### 4.2.1 Requisitos

Para o desenvolvimento da aplicação *estagiei*, serão expostos os requisitos funcionais, não-funcionais e regras de negócio que nossa aplicação terá, tais requisitos foram formados a partir de estudos de como irão funcionar os processos de nosso *website*.

#### 4.2.1.1 Requisitos Funcionais

Os requisitos funcionais dizem respeito às principais funcionalidades que o sistema deve empenhar (SOMMERVILLE, 2011). Durante nossa análise, foram decididos os principais requisitos funcionais da aplicação como descrito no Quadro 4:

Quadro 4 – Requisitos funcionais

| **Código** | **Descrição** |
|---|---|
| RF-001 | Permitir a busca de vagas por filtros |
| RF-002 | Recomendar vagas para estudantes, empresas para estudantes, estudantes para vagas/empresas |
| RF-003 | Manter um histórico de vagas tanto para o candidato, quanto para a empresa |
| RF-004 | Exibir uma linha do tempo do andamento da vaga |
| RF-005 | Alertar os estudantes aplicados à vaga sobre cada mudança em seu processo |
| RF-006 | Possibilitar que a empresa possa entrar em contato com os estudantes recomendados/aplicados à vaga |
| RF-007 | Possibilitar que a empresa realize mudanças no status de andamento da vaga |
| RF-008 | Possibilitar que o estudante realize um *feedback* da empresa pós-entrevista, que será visto por outros estudantes |
| RF-009 | Não permitir o registro de vagas cujas horas de atividades ultrapassem a carga horária prevista por lei de acordo com a situação escolar de cada estudante |
| RF-010 | Permitir o cadastro de vagas por parte da empresa, seguindo as regras estabelecidas |

Fonte: Os Autores

#### 4.2.1.2 Requisitos Não-funcionais

Ao contrário dos requisitos funcionais, os requisitos não-funcionais não estão ligados às principais funcionalidades de um sistema, mas sim com seus fatores de restrições e especificações. É a partir deles que observamos aspectos como desempenho, usabilidade, segurança e outros aspectos não-funcionais que tangem o sistema (SOMMERVILLE, 2011). Tendo isto em mente, no Quadro 5 são elencados os principais requisitos não-funcionais.

Quadro 5 – Requisitos não-funcionais

| Código | Descrição |
|---|---|
| RNF-001 | O sistema deve oferecer boa usabilidade (Ser fácil de aprender a usar) |
| RNF-002 | O sistema deve estar disponível 24 horas por dia, 7 dias por semana |
| RNF-003 | O sistema deve possuir possibilidade de escalabilidade |
| RNF-004 | Tempo para o carregamento que satisfaça as expectativas do cliente |
| RNF-005 | O sistema deve possuir uma taxa de ocorrência de falhas menor que 0.3% |
| RNF-006 | O sistema deve estar de acordo com a Lei Geral de Proteção de Dados (LGPD) |
| RNF-007 | O sistema deve estar de acordo com a lei Nº 11.788, de 25 de setembro de 2008, regulando a carga horária do estágio |
| RNF-008 | O sistema deve ser responsivo aos diferentes dispositivos que os usuários podem utilizar para acessá-lo |

Fonte: Os Autores

#### 4.2.1.3 Regras de Negócio

As regras de negócio, que estão ligadas aos requisitos funcionais previamente descritos, do nosso projeto estão listados no Quadro 6.

Quadro 6 – Regras de negócio

| Código | Descrição | Requisito Relacionado |
|---|---|---|
| RN-001 | As vagas a serem cadastradas devem estar coerentes com o perfil buscado | RF-010 |
| RN-002 | Os históricos das vagas devem ser mantidos por todo o período | RF-003 |
| RN-003 | A empresa é responsável pelo encaminhamento do status da vaga | RF-007 |
| RN-004 | Para o candidato enviar um *feedback*, ele deve ter pelo menos iniciado o processo seletivo | RF-008 |
| RN-005 | O *feedback* pode ser feito de forma anônima, mas o usuário deve estar logado e ter passado pelo processo seletivo | RF-008 |

Fonte: Os Autores

### 4.2.2 Histórias de usuário

No Quadro 7 estão demonstradas as histórias de usuário de nossa aplicação.

Quadro 7 – Histórias de usuário

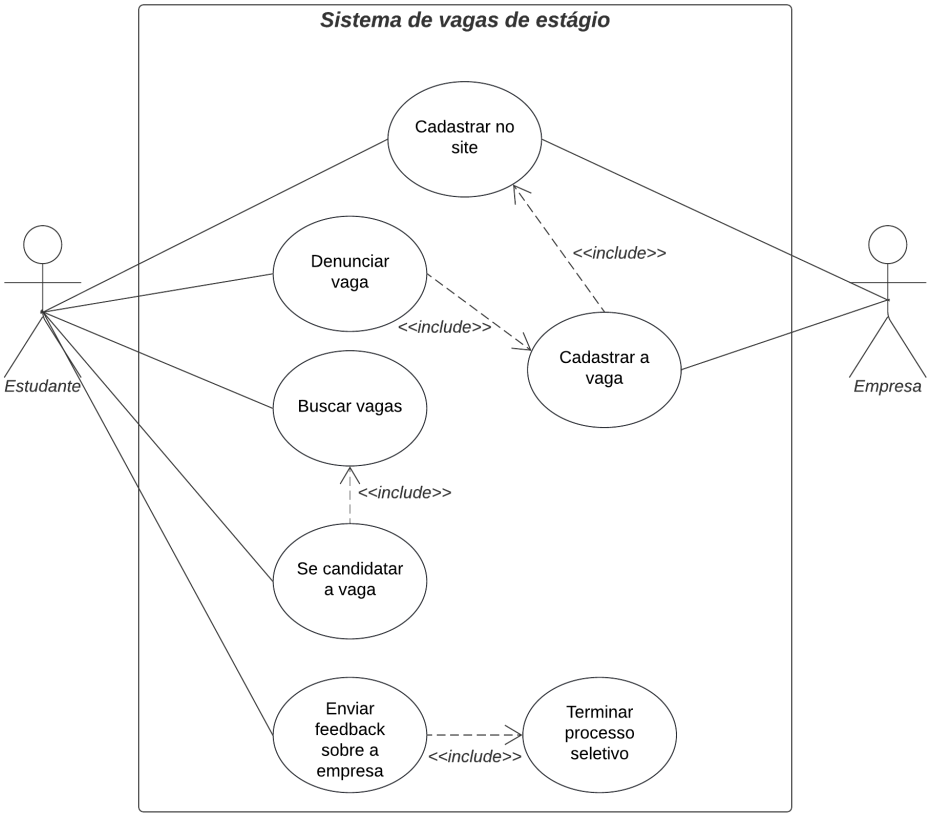
| **História** |
|---|
| Como estudante, eu quero buscar as vagas de acordo com o filtro que eu escolher. |
| Como empresa, eu quero gerenciar a minha vaga para que possa visualizar a quantidade de candidatos dentre outras informações pertinentes. |
| Como estudante, eu quero receber recomendações de vaga para que a minha pesquisa seja facilitada. |
| Como empresa, eu quero receber recomendações de estudantes para que possa enviar solicitações de candidaturas a vaga. |
| Como estudante, eu quero um histórico de todas as minhas vagas já aplicadas. |
| Como empresa, eu quero um histórico dos estudantes candidatos aplicados as vagas para que eu possa realizar levantamentos sobre as informações ali contidas. |
| Como estudante, eu quero uma linha do tempo com os principais passos do processo para que eu possa acompanhá-lo de forma fácil e rápida. |
| Como estudante, eu quero ser alertado sobre as mudanças no status da vaga para que possa saber de forma rápida as movimentações. |
| Como empresa, eu quero me comunicar de forma fácil com os estudantes candidatos para que o processo seja mais ágil. |
| Como empresa, eu quero ter a possibilidade de alterar o status da vaga para que o gerenciamento seja mais fácil. |

Fonte: Os autores

### 4.2.3 Casos de uso

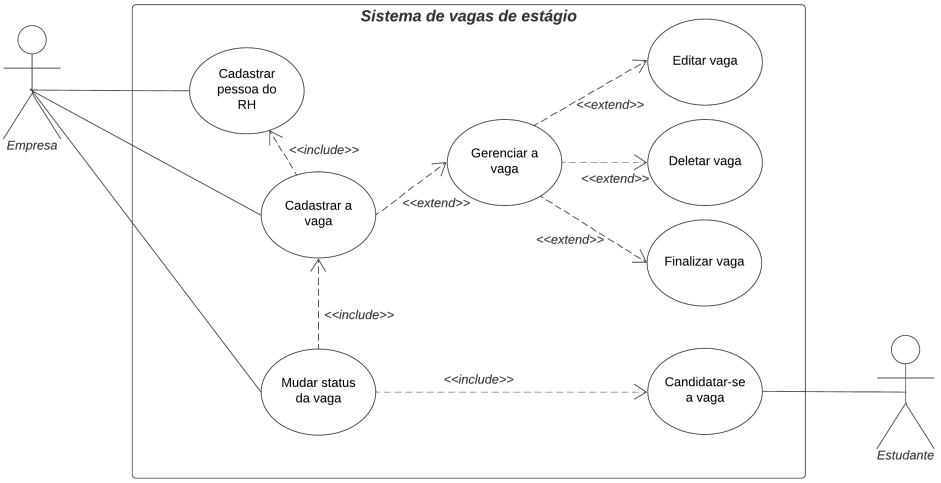
Em Figura 8, Figura 9 e Figura 10 estão demonstrados os diagramas de casos de usos que são pertinentes à nossa aplicação.

Figura 8 – Caso de Uso 1 - Funcionalidades do estudante



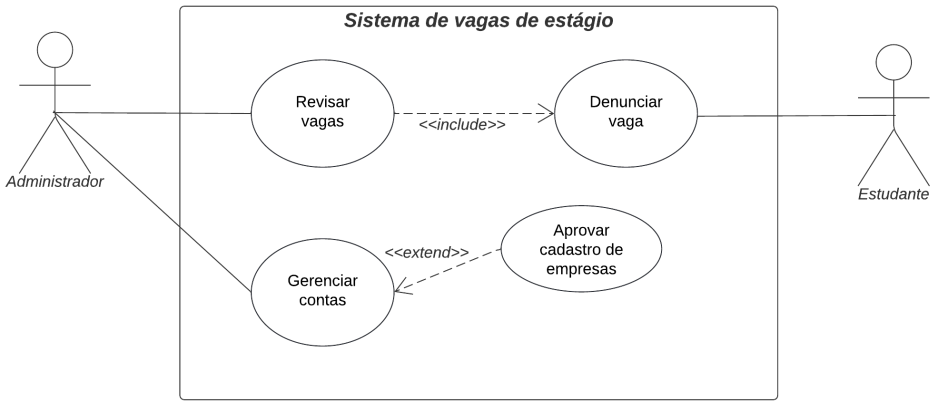

Fonte: Os autores

Figura 9 – Caso de Uso 2 - Funcionalidades da empresa




Fonte: Os autores

Figura 10 – Caso de Uso 3 - Funcionalidades do administrador




Fonte: Os autores

### 4.2.4 Fases de entrega

Nessa seção, iremos expor quais funcionalidades do sistema pretendemos desenvolver tendo em vista as principais fases de entrega da disciplina, sendo elas a POC, o MVP e a Entrega Final.

#### 4.2.4.1 Prova de Conceito (POC)

Na fase de POC, pretendemos entregar as funcionalidades mais básicas do nosso software. Dentre elas, o cadastro de estudantes via *Single Sign-On* (SSO) da Google, onde é explicado o processo no site possibilitando a criação de uma conta com informações básicas, necessárias apenas para o funcionamento padrão do sistema, e o cadastro de empresas, que será feito no próprio *website estagiei*, onde a empresa preenche as informações e passa por uma aprovação nossa. Além disso, o software permitirá o login desses usuários já cadastrados, onde poderão consultar suas informações básicas.

Ao se cadastrar no sistema, a empresa também poderá registrar uma pessoa do Recursos Humanos (RH), que será responsável por gerenciar as vagas daquela organização, e essa pessoa poderá criar novas vagas com informações básicas, apenas para serem visíveis na tela de consulta de vagas. Na parte do estudante, será possível para ele(a), consultar as vagas que existem no sistema através de filtros básicos e internacionalização de linguagem.

#### 4.2.4.2 Produto Mínimo Viável (MVP)

Na entrega do MVP, pretendemos incrementar o que já foi desenvolvido durante a POC com funcionalidades importantes ao nosso sistema, como a candidatura do estudante à uma vaga; a possibilidade do estudante denunciar uma vaga por não ser coerente com a proposta da nossa aplicação, que é ser um *website* que possua vagas de estágio coerentes com a realidade de um estagiário; funcionalidade de login via LinkedIn para os estudantes;

recomendações de vagas para os candidatos; recomendação de candidatos para empresas e opção de contato com o candidato via *Whatsapp*. Além disso, serão feitos os teste unitários e testes de qualidade de software, a fim de garantir que a aplicação esteja em conforme com os requisitos solicitados.

#### 4.2.4.3 Entrega Final

Na entrega final, iremos acrescer nosso projeto com o restante das funcionalidades, tais como o *dashboard* de vagas para a empresa; histórico de vagas para os estudantes; mudança de status das vagas por parte da empresa; *feedback* de empresas após o processo seletivo; acessibilidade com o VLibras e utilização de mais campos do banco de dados para termos mais detalhes de vagas, usuários, etc.

Além disso, requisitos não-funcionais, como a refatoração do código, a fim de deixá-lo mais limpo e performático, e mais testes de qualidade de software, utilizando um banco de dados com mais registros, para manter o mesmo nível de performance e usabilidade que tinha antes, com poucos registros.

## 4.3 Integrações

Nessa seção serão citadas as possíveis integrações que nossa aplicação terá, que foram decididas baseadas em outras aplicações do mercado.

### 4.3.1 Login com o Google e LinkedIn

Pensando na experiência de usuário, nossa aplicação terá a opção do estudante se logar através do SSO dessas empresas. Dessa forma, não será necessário digitar a senha toda vez que o usuário for usar nosso *website*, precisando apenas clicar um botão e fazer o login em uma dessas alternativas.

### 4.3.2 Entrar em contato via *Whatsapp*

Nossa aplicação terá, também, uma forma da empresa contatar o estudante via *Whatsapp*. Essa integração será feita via API disponibilizada pela própria empresa que mantém o aplicativo. Dessa forma, com apenas um clique, será possível enviar uma mensagem diretamente ao estudante.

### 4.3.3 Acessibilidade com VLibras

A Lei Brasileira de Inclusão, Art. 63, estipula que os sites devem ser acessíveis de modo a garantir o acesso às informações disponíveis (BRASIL, 2015), assim, realizaremos a integração com a aplicação VLibras, que é um tradutor de texto escrito em Português

para Língua Brasileira de Sinais (LIBRAS). De acordo com o manual do VLibras, esta integração pode ser realizada com a inclusão de um trecho de código na página HTML da aplicação (SGD, 2021).

## 4.4 Manutenibilidade

Para que a aplicação atinja um nível adequado de qualidade é fundamental que se estabeleça certos requisitos e parâmetros de manutenibilidade, tal como ferramentas que facilitam esse processo. Através dos critérios estabelecidos, podemos medir o quanto o processo de desenvolvimento concorda com as boas práticas e incentivar o uso das mesmas.

### 4.4.1 Logs

Para o monitoramento da aplicação em tempo de execução, essencialmente na camada de servidor, os *logs* serão usados para monitorar o estado dos objetos. A ferramenta a ser utilizada será a implementação de *logs* do Spring Boot que utiliza a implementação Logback. A ferramenta permite diversos registros, como:

- *debug*
- *info*
- *warn*
- *error*

Assim, a cada bloco de falha da aplicação um *log* será colocado para que os problemas sejam identificados, analisados e resolvidos.

### 4.4.2 Code Convention

Visando facilitar o entendimento mútuo entre a equipe, são feitas as convenções de código com o propósito de padronizar como os integrantes da equipe produzem seus respectivos códigos, de modo que o estilo de programação seja independente de seus autores. As convenções de código estabelecem estilos para a organização do código textualmente, ou seja, como os comentários são posicionados, nome de variáveis escolhidas.

As convenções adotadas são baseadas na especificação da SUN MICROSYSTEMS, de 1996. É comumente usada no desenvolvimento na linguagem java, e relativamente próxima do padrão adotado no JavaScript, podendo destacar os seguintes pontos:

- Minimização do uso de variáveis, funções e objetos globais.

- Declarações globais estarão de forma preferencial no início do arquivo.
- Declaração de variáveis próximo do ponto onde são inicializadas.
- Indentação de 4 espaços.
- Classes e interfaces em **CamelCase** e substantivos.
- Métodos em **camelCase** e verbos.
- Constantes em **UPPER_CASE**.

No backend os pacotes serão bem divididos, tendo o pacote *model* para os *models*, *controllers* para os *controllers* e endpoints.

### 4.4.3 Design Patterns

Para padrões de projetos, serão essencialmente utilizados 3 padrões muito utilizados pela comunidade de desenvolvimento: Clean Code, SOLID e 12 Factor App.

#### 4.4.3.1 Clean Code

O Clean Code é um conjunto de boas práticas de programação que visam melhorar o entendimento do código, para que facilite a leitura do mesmo. Algumas boas práticas principais listadas abaixo:

- Nomes significativos para as variáveis, classes, métodos, atributos e objetos.
- Utilização de constantes e enums para evitar números mágicos.
- Evitar comentários que são redundantes e podem ser convertidos em códigos
- Utilização de funções pequenas, com uma única responsabilidade abstrata
- Evitar booleanos de forma explícita.
- Diminuir a redundância e a repetição de código (Don't Repeat Yourself).
- Aumentar a ortogonalidade do código, diminuindo as dependências e o aumentando o desacoplamento e a independência entre os módulos, de modo a de deixa-lo mais fácil de mudar (Easy To Change).

#### 4.4.3.2 SOLID

O SOLID é um acrônimo para 5 princípios da programação orientada a objetos, fundamental para o desenvolvimento e manutenção de software, visto que traz uma facilidade e flexibilidade no código em se adequar a mudanças, frequente no desenvolvimento.

- Single Responsiblity Principle: Uma classe deve ter apenas um motivo para mudar.
- Open-Closed Principle: Uma classe deve estar aberta para extensão, e fechada para modificação, recomendando sempre utilizar a herança e não modificar o código-fonte original.
- Liskov Substitution Principle: Uma classe derivada deve ser substituível por sua classe base.
- Interface Segregation Principle: Utilizar muitas interfaces específicas é melhor que uma interface genérica.
- Dependency Inversion Principle: Dependa de abstrações e não de implementações.

#### 4.4.3.3 12 Factor App

A aplicação doze-fatores é uma metodologia para construir softwares como serviço que seguem os seguintes parâmetros:

- Base de Código: Uma base de código com rastreamento utilizando controle de revisão, muitos deploys.
- Dependências: Declare e isole as dependências.
- Configurações: Armazene as configurações no ambiente.
- Serviços de Apoio: Trate os serviços de apoio, como recursos ligados.
- Construa, lance, execute: Separe estritamente os builds e execute em estágios.
- Processos: Execute a aplicação como um ou mais processos que não armazenam estado.
- Vínculo de porta: Exporte serviços por ligação de porta.
- Concorrência: Dimensione por um modelo de processo.
- Descartabilidade: Maximizar a robustez com inicialização e desligamento rápido.
- Dev/prod semelhantes: Mantenha o desenvolvimento, teste, produção o mais semelhante possível.

- Logs: Trate logs como fluxo de eventos.
- Processos de Admin: Executar tarefas de administração/gerenciamento como processos pontuais.

### 4.4.4 Integração continua

Para manter o serviço sempre atualizado para o usuário, a ferramenta de integração contínua do Heroku CI foi selecionada para a implantação da aplicação no backend em produção.

1. Após uma mudança do código no GitHub, uma instância da Heroku CI que tem acesso ao código do GitHub, identifica automaticamente a linguagem do código;
2. No momento do deploy a Heroku CI constroi o código e da deploy em uma aplicação temporária.
3. Essa aplicação passa por testes paralelos, cujos resultados são mostrados ao usuário através de uma interface.
4. Após a build passar nos testes com sucesso é feito o deploy da aplicação

### 4.4.5 Testes

Testes são ferramentas indubitáveis para o desenvolvimento da aplicação, pois garante, no processo de compilação, o comportamento esperado do programa. Além disso, testes exercem um papel na documentação, visto que abstraem de forma breve o comportamento esperado de classes e métodos, podendo ser consultados em caso de dúvida em relação a algum método. Esta categoria de teste é chamado teste unitário, que diferente dos testes de integração, que verificam o funcionamento do programa de uma chamada a um endpoint, verificando apenas os serviços externos.

Logo, a construção dos testes, de qualquer natureza é de suma importância para a confecção do projeto no quesito manutenibilidade, seguiremos os princípios do Test Driven Development (TDD). Como as ferramentas de teste são específicas para cada linguagem, cada camada fará uso do seu respectivo framework.

O backend deverá ser testado com o framework JUnit, já no frontend serão feitos com a biblioteca Jest, para a confecção de testes unitários.

## 4.5 Segurança, Privacidade e Legislação

Para o desenvolvimento de nossa aplicação, temos que levar em consideração alguns aspectos de segurança, privacidade e legislação. A lei brasileira que diz respeito a como

lidar com dados de pessoas em plataformas digitais (sobretudo em aplicações disponíveis na internet) é a Nº 13.709 (BRASIL, 2018), que está em vigor desde 2020, a LGPD.

De acordo com o estabelecido na LGPD, nossa aplicação irá, se necessário, recuperar o mínimo de dados possíveis do usuário para prosseguir com a sua utilização, como e-mail, nome e informações sobre a instituição de ensino do usuário por parte do candidato e o Cadastro Nacional da Pessoa Jurídica (CNPJ) da empresa por parte da empresa que irá cadastrar as vagas. Sempre que necessário a obtenção de tais informações por parte do sistema, o usuário será alertado de tal ocorrência.

Também podemos levar em consideração algumas outras questões fundamentais de segurança enquanto se dá o desenvolvimento da aplicação, visto que utilizaremos no backend uma API para a transferência de dados e comunicação com o nosso frontend:

- Autenticação e Autorização: As requisições apenas serão aceitas se o usuário estiver autenticado no sistema e os endpoints funcionarão de acordo com a autorização baseada em papeis;
- Criptografia: Seguiremos o protocolo e padrão HTTPS para a transferência de mensagens entre o backend e o frontend, de modo a ficarem encriptadas e garantir maior segurança na aplicação;
- Não exposição de dados sensíveis à aplicação: Durante o desenvolvimento da aplicação, senhas para comunicação com serviços externos e outras ferramentas não ficarão expostas em código, e sim passados através de variáveis de ambiente de modo a não expor chaves e/ou senhas importantes.
- Política de senhas: nunca iremos armazenar as senhas dos usuários diretamente no banco de dados, teremos um algoritmo gerando um *hash* e fazendo a sua comparação no momento da autenticação. Também será crucial impor uma política de segurança que obriga os usuários a informarem uma senha com mais de 8 dígitos, contendo letras e números, pelo menos uma letra maiúscula e um caractere especial. Dessa forma, o fator humano da segurança de nossa aplicação é levemente reforçado.

## 4.6 Viabilidade Financeira

A análise de viabilidade financeira consiste em averiguar a viabilidade da manutenibilidade do projeto e da possibilidade de lucro do mesmo, a fim de fazer essa verificação será descrito cada processo.

### 4.6.1 Gerenciamento de custos

Aqui serão abordados os custos de desenvolvimento e o porte inicial do projeto.

#### 4.6.1.1 Desenvolvimento

O projeto não possuirá nenhum custo de implementação, devido ao fato de ser um projeto educacional, todo o tempo de desenvolvimento da aplicação e documentação serão totalmente voluntários, sem custo adicional ao projeto.

### 4.6.2 Ambiente de produção

São apresentados os custos de manutenibilidade do projeto para os usuários. Onde será feita uma previsão anual de cada plataforma utilizada.

#### 4.6.2.1 Frontend

A camada cliente da aplicação será hospedada na plataforma Vercel, sendo o custo de processamento e requisições da aplicação baixo inicialmente, a hospedagem da camada cliente não apresentará custo adicional.

#### 4.6.2.2 Backend

Inicialmente gratuito na plataforma Heroku.

A partir do momento que for necessário grande porte, será indicado a migração para a AWS ou Azure, visto que garante viabilidade econômica e estratégica (pois o preço é calculado a partir do uso).

Utilizando a calculadora da AWS (AWS, 2022) e optando por um servidor Linux da instância t4g.micro com 1 vCPU e 1GiB, com armazenamento SSD de uso geral, será custeado o valor de 5,76 USD mensalmente para operar o mês inteiro.

Utilizando a calculadora da Microsoft Azure (AZURE, 2022) e optando por um servidor Linux da instância A1 v2 com 1 núcleo e 2GB de RAM, com 10GB de armazenamento temporário, será custeado o valor de 57,10 USD mensalmente para operar o mês inteiro.

#### 4.6.2.3 Banco de dados

Inicialmente gratuito na plataforma Heroku através do serviço de apoio Heroku Postgres.

Caso a aplicação fique com um porte maior, será indicado a migração para a RDS, que suporta o serviço de banco de dados, cujo o custo é calculado em relação ao uso.

Utilizando a calculadora da AWS (AWS, 2022) e optando por um servidor da instância t3.micro de modelo Single-AZ OnDemand, com armazenamento SSD para cada instância, será custeado o valor de 27,36 USD mensalmente para operar o mês inteiro.

### 4.6.3 Monetização

Afim de gerar receita para a plataforma, são consideradas duas possibilidades de monetização.

- Propagandas: Será utilizado mediador de anúncio *Google Adsense*, onde o valor varia por visualizações de anúncios e cliques nos anúncios, quanto maior a quantidade de conversão de cliques por visualização, maior será a sua renda.
- Contratos: Empresas interessadas em impulsionar as suas vagas para atingir um número maior de visualizações ou oferecer ferramentas de análises mais precisas e um melhor suporte, feito por intermédio da realização de contratos com a plataforma e que consequentemente gerará renda.

Com a estimativa de 100 a 250 visitantes por dia, considerando que pelo menos 2 páginas são visualizadas por visitantes, sendo a taxa de cliques em anúncios 1% e o custo do clique 0.20 USD, o valor mensal será de aproximadamente 10.5 USD. A monetização por propaganda seria a forma de renda mais rápida para o projeto e os contratos seriam feitos a médio/longo prazo.

### 4.6.4 Conclusão

Utilizando inicialmente os servidores de baixo porte detalhados acima, não haverá custo adicional a priori. Contudo, o valor calculado para 250 visitantes diários com os parâmetros detalhados arrecadará 10.5 USD mensalmente.

Caso o engajamento da aplicação aumente, a medida que o número de usuários aumenta, incrementando proporcionalmente o rendimento com o *Google Adsense*, poderá ser revisto os planos dos servidores para atender maiores níveis de requisições e buscar contratos com empresas para aumentar a rentabilidade da plataforma.

# Referências


AMAZON WEB SERVICES. *Definição de Preço do Amazon S3*. 2022. Disponível em: <https://aws.amazon.com/pt/s3/pricing/>. Citado na página 30.

BRASIL. *Lei de Estágios*. 2008. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11788.htm>. Acesso em: 24 abr. 2022. Citado 4 vezes nas páginas 12 e 13.

BRASIL. *Lei Brasileira de Inclusão da Pessoa com Deficiência*. 2015. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2015/lei/l13146.htm>. Acesso em: 25 abr. 2022. Citado na página 24.

BRASIL. *Lei Geral de Proteção de Dados*. 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 24 abr. 2022. Citado na página 29.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, CA, 2000. Citado na página 14.

MICROSOFT AZURE. *Calculadora de Preço*. 2022. Disponível em: <https://azure.microsoft.com/pt-br/pricing/calculator/>. Citado na página 30.

MOZILLA FOUNDATION. *SPA (Single-page application)*. 2021. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>. Acesso em: 24 abr. 2022. Citado na página 13.

REDHAT. *What is an API?* 2017. Disponível em: <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>. Acesso em: 25 abr. 2022. Citado 4 vezes nas páginas 14 e 15.

REDHAT. *What is a REST API?* 2020. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 25 abr. 2022. Citado 2 vezes nas páginas 14 e 15.

SECRETARIA DE GOVERNO DIGITAL. *Manual de Instruções da Ferramenta VLibras Widget 6.0.0*: Integrando a uma página web. [S.l.], 2021. Disponível em: <https://vlibras.gov.br/doc/widget/installation/webpageintegration.html>. Acesso em: 25 abr. 2022. Citado na página 25.

SOMMERVILLE, I. *Engenharia de Software*. São Paulo, SP: Pearson, 2011. Citado na página 19.

WASSON, M. *Aplicativos de página única:*: Crie aplicativos web dinâmicos e modernos com o asp.net. 2015. Disponível em: <https://docs.microsoft.com/pt-br/archive/msdn-magazine/2013/november/asp-net-single-page-applications-build-modern-responsive-web-apps-with-asp-net>. Citado na página 13.

WIKIPEDIA. *Single-page application*. 2022. Disponível em: <https://en.wikipedia.org/wiki/Single-page_application>. Acesso em: 01 maio 2022. Citado na página 13.

# Glossário

| | |
|---|---|
| API RESTful | API que segue todas as restrições da arquitetura REST. - Citado em 14, 15 |
| AWS | Amazon Web Services - Plataforma em nuvem *on-demand* que disponibiliza diversos serviços web. - Citado em 30, 34 |
| backend | Camada do sistema da aplicação que não é acessado diretamente pelo usuário, responsável pelo processamento de dados e a implementação de funcionalidades que satisfazem uma ou mais regras de negócios da aplicação. - Citado em 6, 16, 26, 28, 29, 30 |
| deploy | Refere-se ao processo de configuração de um computador ou sistema até o ponto em que esteja pronto para o processamento em ambiente de produção. - Citado em 16, 27, 28 |
| endpoint | Localização digital onde uma API recebe requisições sobre um recurso específico em seu servidor. Os endpoints comumente são uma *Universal Resource Locator* (URL), indicando uma ponta da conexão para a recuperação do recurso através da API. - Citado em 26, 28, 29 |
| framework | Estrutura base para desenvolvimento de um sistema e/ou projeto com um conjunto de elementos e conexões pré-estabelecidas e/ou indicadas. - Citado em 16, 28, 34 |
| frontend | Camada do sistema da aplicação que é responsável pela integração do usuário com o sistema, oferecendo uma interface que se comunica com o usuário e com o sistema. - Citado em 6, 16, 28, 29, 30 |
| Git | Sistema de controle de versão de arquivos. - Citado em 33 |
| GitHub | Provedor de hospedagem na internet para desenvolvimento de software e controle de versionamento usando Git. - Citado em 28 |
| Heroku | Plataforma em nuvem como um serviço que suporta diversas linguagens de programação. - Citado em 16, 30, 33 |
| Heroku CI | Instância da Heroku responsável pela integração contínua. - Citado em 28 |
| Jira Software | Ferramenta de gerenciamento que permite o monitoramento de tarefas e acompanhamento de projetos. - Citado em 9, 10 |
| LinkedIn | Rede social focada em vagas de emprego. - Citado em 23 |
| Linux | Kernel open-source usado em diversos sistemas operacionais. - Citado em 30 |

| | |
|---|---|
| Logback | Logback é uma estrutura de log para aplicações java, criada como sucessora do popular projeto log4j. - Citado em 25 |
| PostgreSQL | Sistema de Gerenciamento de Banco de Dados Relacional, gratuito e open-source. - Citado em 16 |
| RDS | Amazon Relational Databases - Serviço de banco de dados da AWS. - Citado em 30 |
| React | Biblioteca JavaScript gratuita e open-source para a construção de interfaces baseadas em componentes. - Citado em 16 |
| Scrum | Metodologia ágil de software concebida por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 90. - Citado em 9, 34 |
| Scrum Master | Papel de gerência e coordenação na metodologia Scrum. O Scrum Master é o intermediário entre a equipe de desenvolvimento e os clientes. - Citado em 9 |
| Spring Boot | Spring Boot é um framework baseado em Java de código aberto usado para criação de micro serviços e aplicações web no geral. - Citado em 16, 25 |
| Sprint | Unidade de planejamento do Scrum na qual se verifica o trabalho (funcionalidade) a ser entregue, os recursos necessários e ocorre o desenvolvimento do software de fato. - Citado em 3, 9, 10, 11 |
| SUN MICROSYSTEMS | I. Java coding conventions, 1996. - Citado em 25 |
| TDD | Test Driven Development - Uma prática de desenvolvimento de software que se concentra na criação de casos de teste de unidade antes de desenvolver o código real. - Citado em 28 |
| TypeScript | Linguagem de programação fortemente tipada sobre JavaScript. - Citado em 16 |
| Vercel | Plataforma em nuvem que faz o *host* de páginas web. - Citado em 16, 30 |
| VLibras | Conjunto de ferramentas para a tradução de texto em Português para LIBRAS gratuítas e de código aberto, mais informações disponíveis no endereço <https://www.gov.br/governodigital/pt-br/vlibras>. - Citado em 6, 24, 25 |