

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

Documentação de Mínimo Produto Viável
para aprovação na disciplina de Projeto Inte-
grado 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

Listas de ilustrações

Figura 1 – Roteiro Geral	12
Figura 2 – Roteiro Geral - Detalhe Inicial	12
Figura 3 – Roteiro Geral - Detalhe Final	12
Figura 4 – Linhas de código - SVN	14
Figura 5 – Atividade por hora do dia - SVN	15
Figura 6 – Atividade por dia da semana - SVN	15
Figura 7 – Visão geral - Projeto front-end	16
Figura 8 – URL do repositório front-end	16
Figura 9 – Visão geral - Projeto back-end	17
Figura 10 – URL do repositório back-end	17
Figura 11 – Visão geral - Projeto Documentos	18
Figura 12 – URL do repositório de documentos L ^A T _E X	18
Figura 13 – Linhas de código - Projeto front-end	19
Figura 14 – Linhas de código - Projeto Backend	19
Figura 15 – Extensão de arquivos - Projeto front-end	20
Figura 16 – Dias da semana - Projeto front-end	20
Figura 17 – Lista de autores - Projeto Backend	20
Figura 18 – Lista de autores - Projeto Documentos	21
Figura 19 – Dias da semana - Projeto front-end	21
Figura 20 – Dias da semana - Projeto Backend	22
Figura 21 – Validação dos <i>headers</i>	23
Figura 22 – URL da documentação dos nossos endpoints (Swagger UI)	23
Figura 23 – Teste de <i>Transport Layer Security (TLS)</i>	24
Figura 24 – URL do front-end da nossa aplicação	24
Figura 25 – Teste de desempenho do front-end	25
Figura 26 – Análise de código do front-end	26
Figura 27 – Análise de código do back-end	27
Figura 28 – Validação do <i>Hypertext Markup Language (HTML)</i>	28
Figura 29 – Ciclo de vida: página web tradicional X SPA	30
Figura 30 – Arquitetura de Aplicação	34
Figura 31 – Arquitetura Tecnológica	34
Figura 32 – Arquitetura de Negócios	35
Figura 33 – Caso de Uso 1 - Funcionalidades do estudante	38
Figura 34 – Caso de Uso 2 - Funcionalidades da empresa	38
Figura 35 – Caso de Uso 3 - Funcionalidades do administrador	39
Figura 36 – Modelagem Entidade Relacionamento	50

Figura 37 – Diagrama Entidade Relacionamento	51
Figura 38 – Legenda	52
Figura 39 – Campos Usuário	52
Figura 40 – Campos Pessoa	53
Figura 41 – Campos Estudante	53
Figura 42 – Campos Empresa	54
Figura 43 – Campos Representante RH	54
Figura 44 – URL do blog da equipe L ^A T _E X	61
Figura 45 – Blog: Apresentação	61
Figura 46 – Blog: Semana 1	62
Figura 47 – Blog: Semana 2	63
Figura 48 – Blog: Semana 3	64
Figura 49 – Blog: Semana 4	64
Figura 50 – Blog: Semana 5	65
Figura 51 – Blog: Semana 6	65
Figura 52 – Blog: Semana 7 - 1	66
Figura 53 – Blog: Semana 7 - 2	67
Figura 54 – Blog: Semana 7 - 3	68
Figura 55 – Blog: Semana 8 - 1	69
Figura 56 – Blog: Semana 8 - 2	70
Figura 57 – Blog: Semana 9 - 1	71
Figura 58 – Blog: Semana 9 - 2	72
Figura 59 – Blog: Semana 10	73
Figura 60 – Blog: Semana 11	74
Figura 61 – Blog: Semana 12	75

Lista de quadros

Quadro 1 – Comparação dos aplicativos concorrentes	10
Quadro 2 – Divisão de responsabilidades da equipe.	11
Quadro 3 – Cronograma de Sprints	13
Quadro 4 – Requisitos funcionais	36
Quadro 5 – Requisitos não funcionais	36
Quadro 6 – Regras de negócio	37
Quadro 7 – Histórias de usuário	37
Quadro 8 – Endpoints da <i>Application Programming Interface</i> (API)	54

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i> - Interface de Programação de Aplicativos - Citado em 4 , 22 , 31 , 32 , 40 , 47 , 54 , 57
CLT	Consolidação das Leis do Trabalho - Citado em 29
CNPJ	Cadastro Nacional da Pessoa Jurídica - Citado em 46
CSS	<i>Cascading Style Sheets</i> - Folhas de Estilo em Cascata - Citado em 30
HTML	<i>Hypertext Markup Language</i> - Linguagem de Marcação de Hiper-texto - Citado em 2 , 6 , 27 , 28 , 30 , 39 , 40
HTTP	<i>Hypertext Transfer Protocol</i> - Protocolo de transferência de hiper-texto - Citado em 31 , 32 , 33 , 54
HTTPS	<i>Hypertext Transfer Protocol Secure</i> - Protocolo seguro de transfe-rência de hipertexto - Citado em 23 , 32 , 47
JSON	<i>JavaScript Object Notation</i> - Notação de Objeto JavaScript - Citado em 32 , 33
LGPD	Lei Geral de Proteção de Dados - Citado em 36 , 46
LIBRAS	Língua Brasileira de Sinais - Citado em 40 , 59
MVP	<i>Minimum Viable Product</i> - Produto Mínimo Viável - Citado em 13 , 39
POC	<i>Prove of Concept</i> - Prova de Conceito - Citado em 13 , 39
REST	<i>Representational State Transfer</i> - Transferência de Estado Repré-sentacional - Citado em 31 , 32 , 57
RH	Recursos Humanos - Citado em 39
SPA	<i>Single Page Application</i> - Aplicação de Página Única - Citado em 2 , 30 , 33
SSD	<i>Solid-State Drive</i> - Unidade de Estado Sólido - Citado em 48
SSO	<i>Single Sign-On</i> - Login único - Citado em 39 , 40
TLS	<i>Transport Layer Security</i> - Segurança da Camada de Transporte - Citado em 2 , 6 , 23 , 24 , 58
URL	<i>Universal Resource Locator</i> - Localizador universal de recurso - Citado em 57
USD	<i>United States Dollar</i> - Dólares Americanos - Citado em 48 , 49

Sumário

1	INTRODUÇÃO	9
1.1	Justificativa	9
1.2	Proposta de solução	9
1.3	Objetivos	9
1.4	Análise de Concorrentes	10
2	PLANEJAMENTO E GERENCIAMENTO DO PROJETO	11
2.1	Gestão e Desenvolvimento do Projeto	11
2.2	Organização da equipe	11
2.3	Cronograma	12
2.4	Estatísticas dos repositórios	14
2.4.1	SVN	14
2.4.2	GitHub	16
2.5	Validações de segurança, interface e código	22
2.5.1	Teste dos <i>headers</i> da API	22
2.5.2	Teste de TLS do front-end	23
2.5.3	Teste de desempenho do front-end	24
2.5.4	Análise de código	25
2.5.5	Validador HTML	27
3	REVISÃO DA LITERATURA	29
3.1	Estágio	29
3.1.1	Definição	29
3.1.2	Tipos de estágio	29
3.1.3	Carga horária	29
3.2	Single-page Application (SPA)	30
3.3	Application Programming Interface (API)	31
3.3.1	API REST	31
3.4	Competências	32
4	DESENVOLVIMENTO DA APLICAÇÃO	33
4.1	Arquitetura	33
4.1.1	Diagramas de arquitetura	33
4.2	Escopo	35
4.2.1	Requisitos	35
4.2.1.1	Requisitos Funcionais	35

4.2.1.2	Requisitos Não-funcionais	36
4.2.1.3	Regras de Negócio	36
4.2.2	Histórias de usuário	37
4.2.3	Casos de uso	37
4.2.4	Fases de entrega	39
4.2.4.1	Prova de Conceito (POC)	39
4.2.4.2	Produto Mínimo Viável (MVP)	39
4.2.4.3	Entrega Final	40
4.3	Integrações	40
4.3.1	Login com o Google e LinkedIn	40
4.3.2	Entrar em contato via Whatsapp	40
4.3.3	Acessibilidade com VLibras	40
4.3.4	API dos Correios	40
4.4	Manutenibilidade	41
4.4.1	Logs	41
4.4.2	Code Convention	41
4.4.2.1	Codificação geral	41
4.4.2.2	Commits	42
4.4.3	Design Patterns e boas práticas	42
4.4.3.1	Clean Code	42
4.4.3.2	SOLID	43
4.4.3.3	12 Factor App	43
4.4.4	Integração continua	44
4.4.4.1	Script de integração com Heroku	44
4.4.4.2	Script de integração com o Netlify	45
4.4.5	Testes	46
4.5	Segurança, Privacidade e Legislação	46
4.6	Viabilidade Financeira	47
4.6.1	Gerenciamento de custos	47
4.6.1.1	Desenvolvimento	47
4.6.2	Ambiente de produção	48
4.6.2.1	Frontend	48
4.6.2.2	Backend	48
4.6.2.3	Banco de dados	48
4.6.3	Monetização	48
4.6.4	Conclusão	49
4.7	Modelagem e definições técnicas	49
4.7.1	Modelo Entidade Relacionamento	50
4.7.2	Diagrama Entidade-Relacionamento	51

4.7.3	Dicionário de Dados	51
4.7.4	Endpoints da API	54
4.7.5	Listagem das Competências	55
GLOSSÁRIO	56
APÊNDICES		60
APÊNDICE A – PUBLICAÇÕES DO BLOG		61
APÊNDICE B – DESENHO DA APLICAÇÃO		76
APÊNDICE C – POC OVERVIEW		111
ANEXOS		113
ANEXO A – NOTA DOS HEADERS		114

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

Com nossa solução buscamos 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, tanto 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.

Podemos definir nosso objetivo principal como:

- Ser uma aplicação onde de fato os estudantes encontrem vagas que condizem com a realidade de um estagiário.

A partir do nosso objetivo principal, podemos explicitar alguns objetivos adjacentes:

- Ser um *website* de fácil usabilidade, onde os estudantes encontrem vagas sem passar por longos processos seletivos.
- 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	CIEE	EstagiEI.
Login/Cadastro.	x	x	x	x	x
Aplicar em uma vaga.	x	x	x	x	x
Notificação a cada mudança do status no processo seletivo.			x	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 <i>WhatsApp</i> .				x	x
Denúncias de vagas incoerentes com a realidade.					x
<i>Feedback</i> 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				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

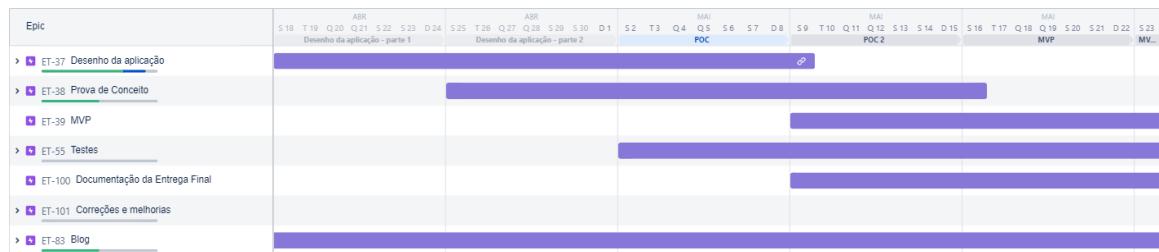
A princípio temos uma organização dos macro-itens (Epics) que precisam ser desenvolvidos durante o projeto, dentro dos quais estipulamos as tarefas a serem feitas. Por meio do [Jira Software](#) podemos ter uma visão geral do andamento das Epics e os prazos, além das [Sprints](#) planejadas e a atual.

Figura 1 – Roteiro Geral



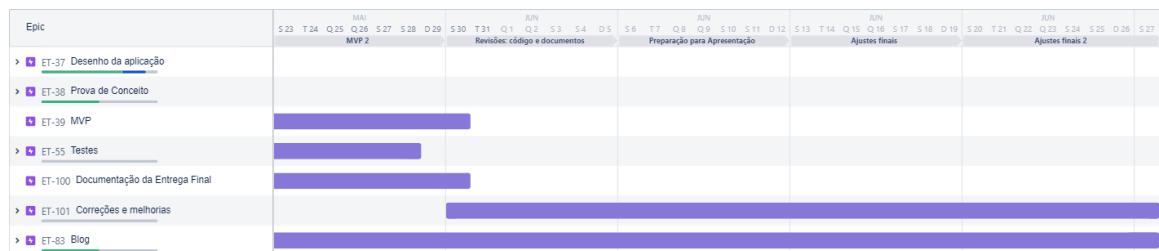
Fonte: Roteiro via ferramenta [Jira Software](#)

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 <i>Prove of Concept</i> (POC).	Concluída
POC	02/05/22	09/05/22	Finalização do Desenho da Aplicação. Início do desenvolvimento dos itens da POC	Concluída
POC 2	09/05/22	16/05/22	Continuação do desenvolvimento dos itens da POC.	Concluída
MVP	16/05/22	23/05/22	Aproveitamento do que foi desenvolvido para a POC com melhorias e ampliação conforme possível para o <i>Minimum Viable Product</i> (MVP).	Concluída
MVP 2	23/05/22	30/05/22	Continuação do trabalho no desenvolvimento do MVP.	Concluída
Revisões: código e documentos	30/05/22	06/06/22	Finalização e revisão tanto do desenvolvimento quanto da documentação.	Concluída
Preparação para a Apresentação	06/06/22	13/06/22	Organização e planejamento da apresentação do projeto e sua documentação.	Concluída
Ajustes finais	13/06/22	20/06/22	Ajustes a serem feitos para correção e/ou melhoria do projeto apresentado.	Em progresso
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

2.4 Estatísticas dos repositórios

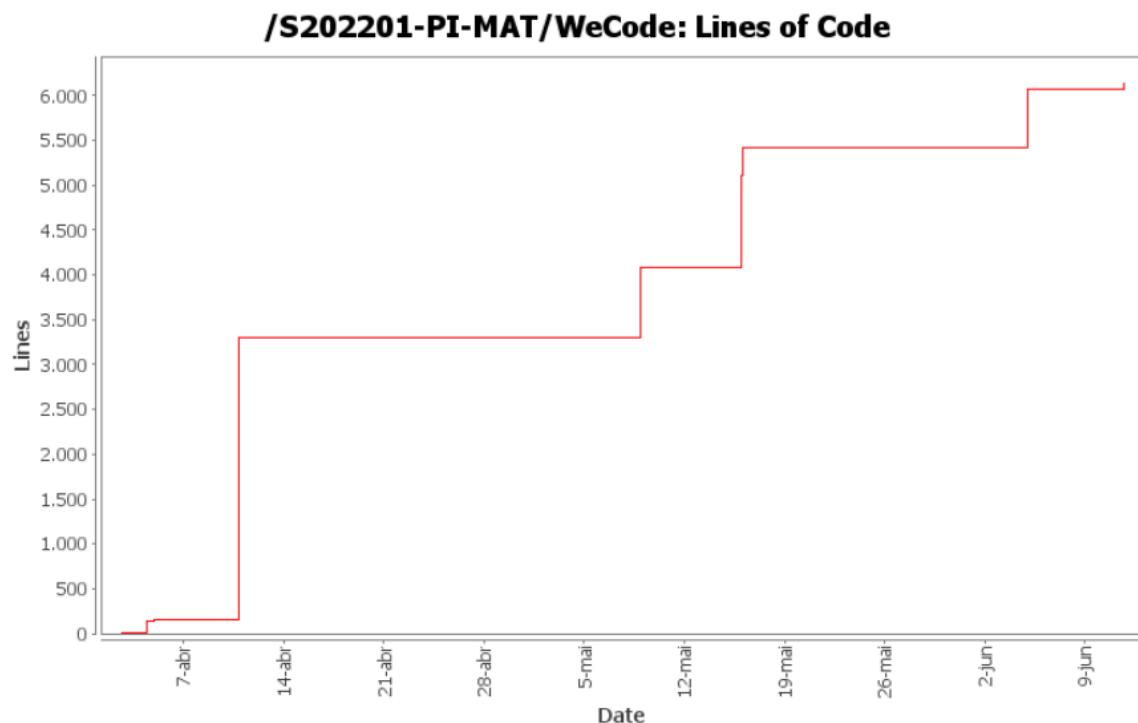
Nesta seção serão apresentadas as estatísticas de cada versionador de código, com detalhes da atuação de cada integrante da equipe e dos *commits* feitos durante o desenvolvimento.

2.4.1 SVN

Estatísticas sobre o repositório no [SVN](#) foram geradas através do [StatSVN](#) apesar de ter sido usado apenas para atualização recorrente do repositório, então não foram considerados dados estatísticos sobre atividade de cada membro da equipe.

A [Figura 4](#) mostra a evolução de linhas de código conforme o tempo.

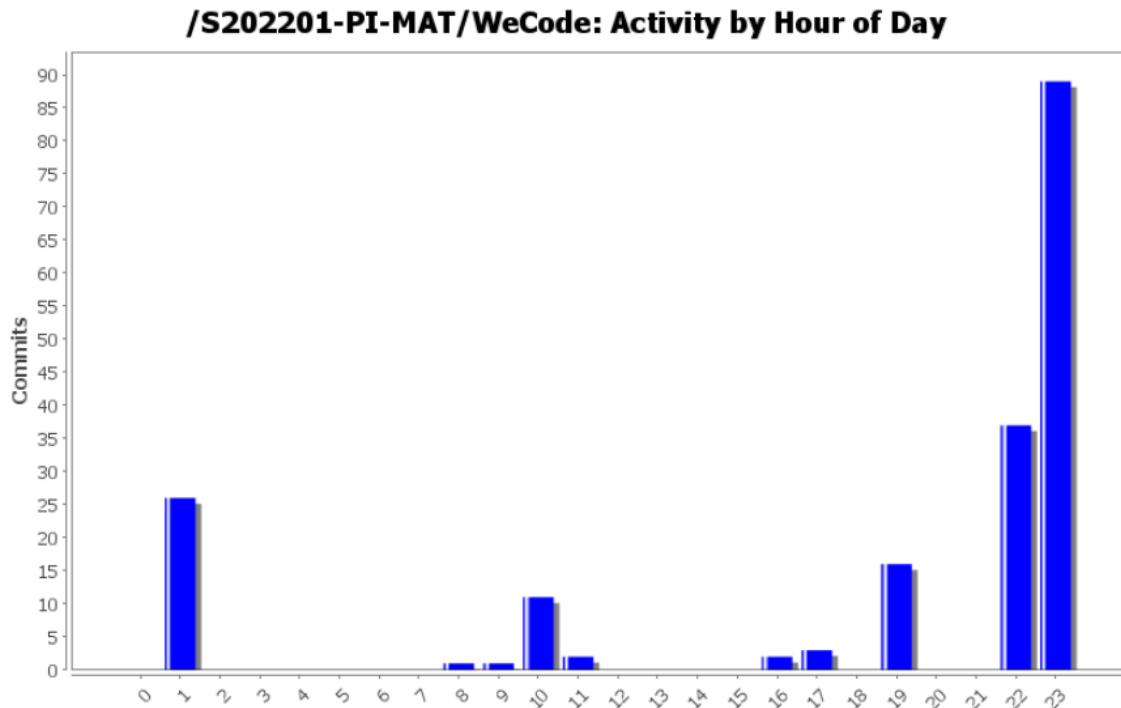
Figura 4 – Linhas de código - SVN



Fonte: Os autores.

A [Figura 5](#) mostra os horários onde mais foram feitos *commits* em todos os dias.

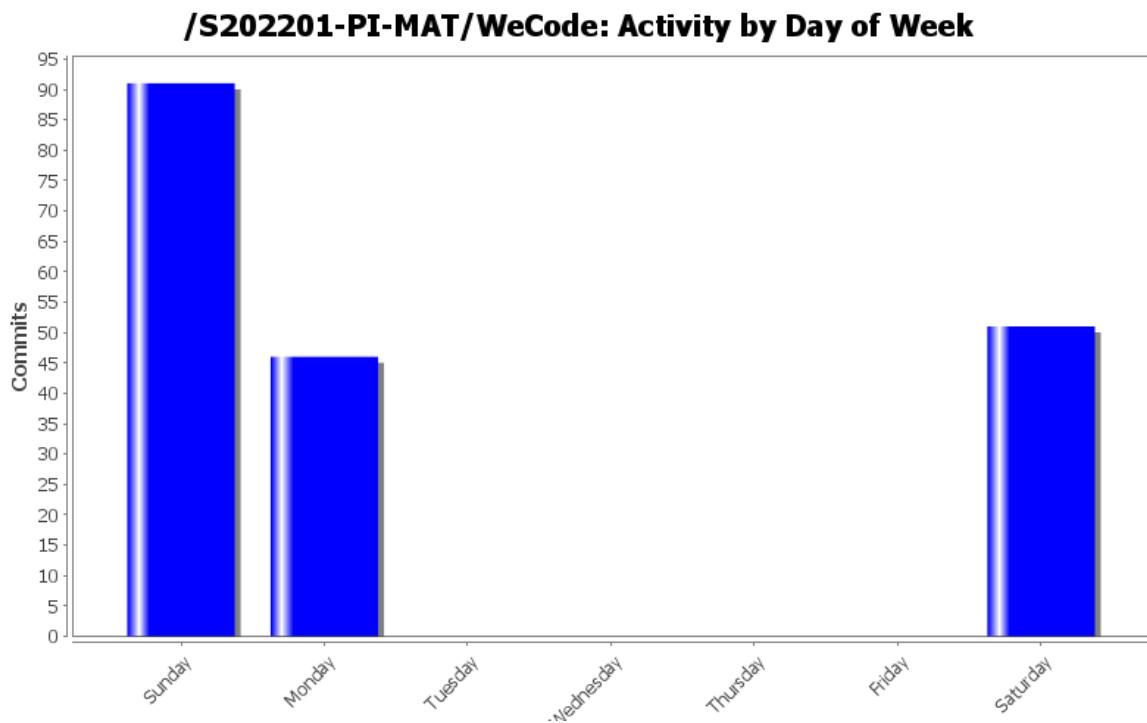
Figura 5 – Atividade por hora do dia - SVN



Fonte: Os autores.

A Figura 6 mostra a quantidade de *commits* por dia da semana, é notório a alta quantidade de atividade aos domingos.

Figura 6 – Atividade por dia da semana - SVN



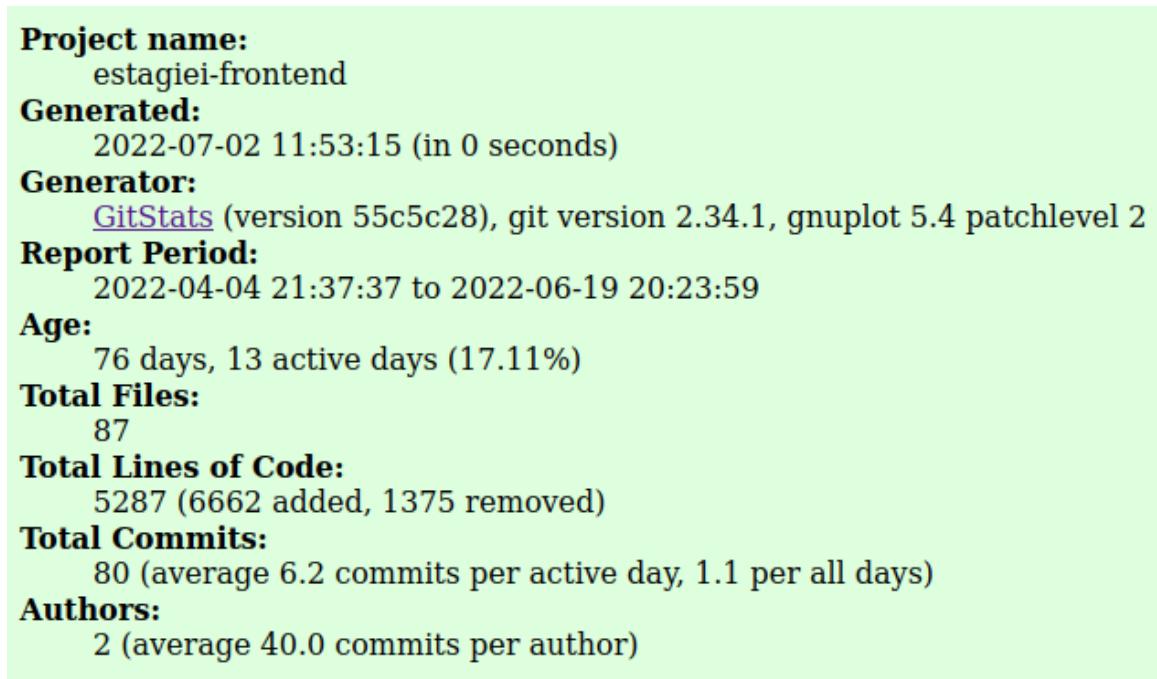
Fonte: Os autores.

2.4.2 GitHub

A partir dos commits feitos nos repositórios, foram levantadas estatísticas sobre o projeto através do [GitStats](#), que servem de base para se ter uma ideia de como foi o andamento do projeto e o que cada integrante da equipe fez durante o período. O [Git](#) foi usado como nosso versionador principal, então os dados abaixo estão de acordo com a atividade de cada integrante.

A [Figura 7](#) demonstra uma visão geral sobre alguns dados do repositório onde hospedamos o nosso [front-end](#).

Figura 7 – Visão geral - Projeto front-end



Fonte: Os autores

A [Figura 9](#) demonstra uma visão geral sobre alguns dados do repositório onde foi hospedado o backend. É possível notar algumas diferenças entre ele e o [front-end](#).

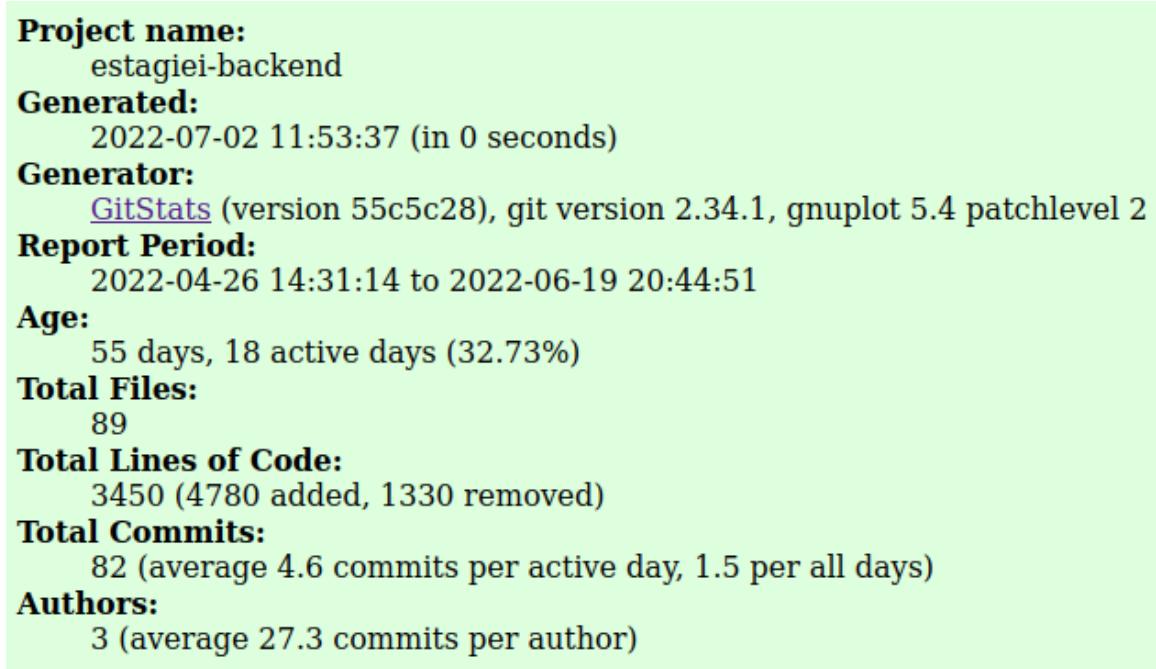
Figura 8 – URL do repositório front-end



<<https://github.com/EquipeWeCode/estagiei-frontend/>>

Fonte: Os Autores.

Figura 9 – Visão geral - Projeto back-end



Fonte: Os autores.

Figura 10 – URL do repositório back-end

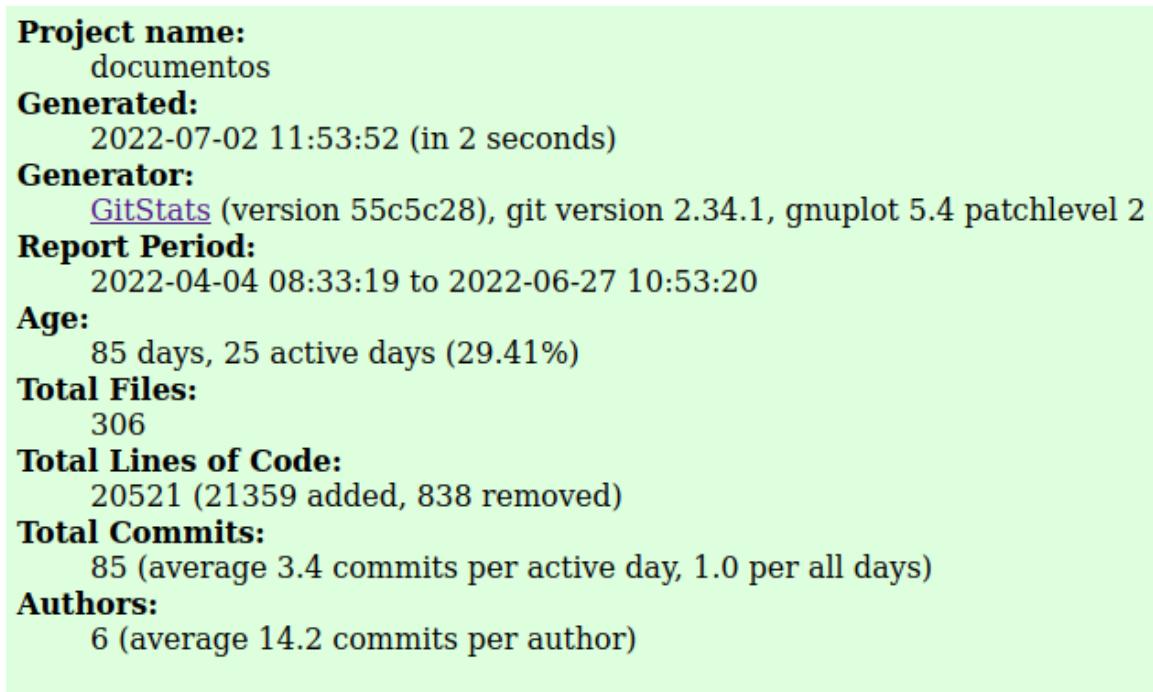


<<https://github.com/EquipeWeCode/estagiei-backend>>

Fonte: Os Autores.

A Figura 11 demonstra uma visão geral sobre alguns dados principais do repositório onde foi versionado os documentos L^AT_EX do projeto.

Figura 11 – Visão geral - Projeto Documentos



Fonte: Os autores.

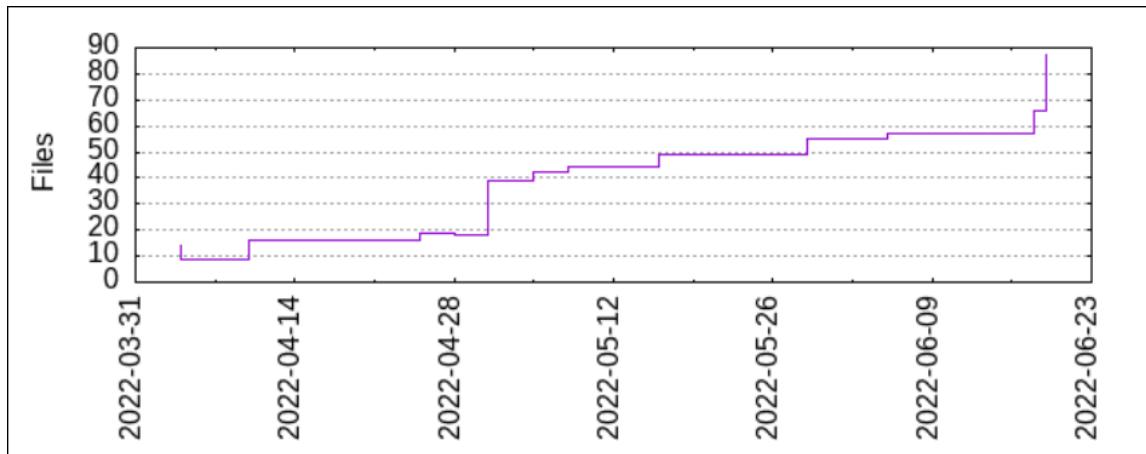
Figura 12 – URL do repositório de documentos L^AT_EX

<<https://github.com/EquipeWeCode/documentos>>

Fonte: Os Autores.

A Figura 13 demonstra a evolução em número de linhas do repositório onde foi hospedado o front-end.

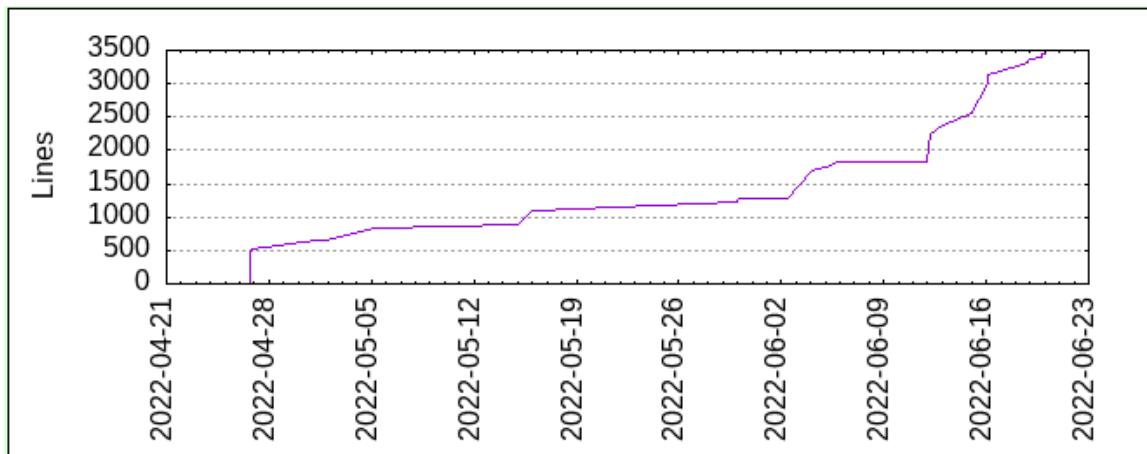
Figura 13 – Linhas de código - Projeto front-end



Fonte: Os autores.

A Figura 14 demonstra a evolução em número de linhas do repositório onde foi hospedado o back-end.

Figura 14 – Linhas de código - Projeto Backend



Fonte: Os autores.

Cabe ressaltar que o número de linhas não necessariamente diz que foi desenvolvido muito código, como demonstra a Figura 15, grande parte das linhas do nosso front-end são de arquivos .json, principalmente do package-lock.json, que é gerado automaticamente e documenta e versiona as dependências do projeto.

Figura 15 – Extensão de arquivos - Projeto front-end

Extension	Files (%)	Lines (%)	Lines/file
	3 (3.45%)	58 (1.10%)	19
css	12 (13.79%)	267 (5.05%)	22
example	1 (1.15%)	1 (0.02%)	1
html	1 (1.15%)	20 (0.38%)	20
json	6 (6.90%)	3421 (64.71%)	570
md	1 (1.15%)	14 (0.26%)	14
png	1 (1.15%)	41 (0.78%)	41
svg	5 (5.75%)	137 (2.59%)	27
toml	1 (1.15%)	9 (0.17%)	9
ts	23 (26.44%)	415 (7.85%)	18
tsx	32 (36.78%)	881 (16.66%)	27
yml	1 (1.15%)	21 (0.40%)	21

Fonte: Os autores.

A Figura 16 lista os autores que contribuíram no repositório front-end.

Figura 16 – Dias da semana - Projeto front-end

Author	Commits (%)	+ lines	- lines	First commit	Last commit	Age	Active days	# by commits
Leonardo Marques	75 (93.75%)	6557	1375	2022-04-04	2022-06-19	75 days, 22:46:22	13	1
Lucas Lima	5 (6.25%)	211	106	2022-05-29	2022-06-19	20 days, 21:49:54	2	2

Fonte: Os autores.

A Figura 17 lista os autores que contribuíram no repositório back-end.

Figura 17 – Lista de autores - Projeto Backend

Author	Commits (%)	+ lines	- lines	First commit	Last commit	Age	Active days	# by commits
Leonardo Marques	66 (80.49%)	3300	1279	2022-05-01	2022-06-19	49 days, 19:46:30	16	1
Marcelo Junior	9 (10.98%)	541	11	2022-04-26	2022-05-05	8 days, 17:26:36	2	2
Igor Nathan	7 (8.54%)	936	56	2022-05-05	2022-06-14	40 days, 19:02:13	5	3

Fonte: Os autores.

A Figura 18 lista os autores que contribuíram no repositório de documentos, no qual todos os membros da equipe participaram.

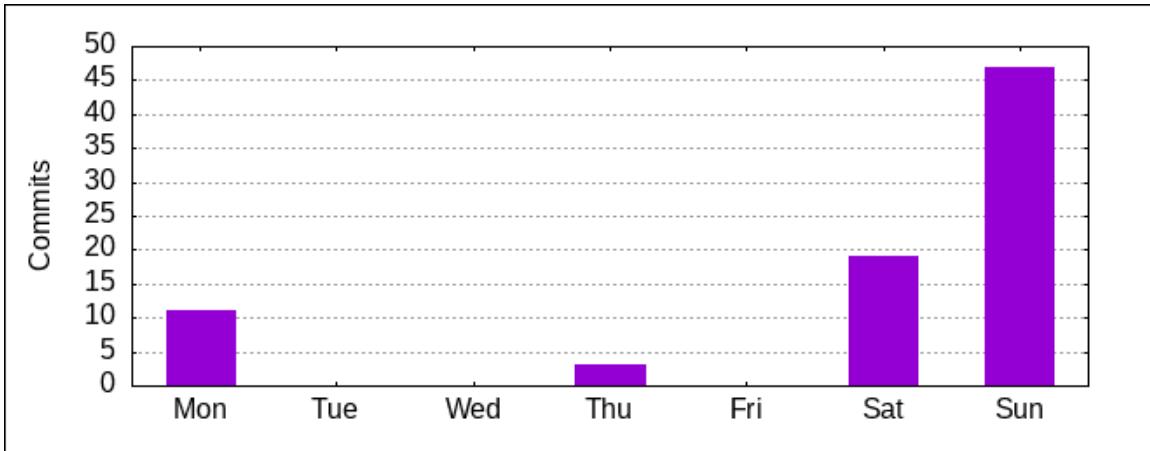
Figura 18 – Lista de autores - Projeto Documentos

Author	Commits (%)	+ lines	- lines	First commit	Last commit	Age	Active days	# by commits
Bruna S. Pires	40 (47.06%)	17697	427	2022-04-04	2022-06-13	70 days, 2:35:18	16	1
Leonardo Marques	29 (34.12%)	733	286	2022-04-09	2022-06-13	64 days, 16:46:57	9	2
Igor Nathan	6 (7.06%)	128	89	2022-04-13	2022-06-27	74 days, 13:14:58	3	3
Marcelo Junior	5 (5.88%)	2525	34	2022-04-10	2022-04-25	15 days, 4:43:36	4	4
Daniel Roberto	3 (3.53%)	17	4	2022-06-12	2022-06-27	14 days, 11:25:22	3	5
Lucas Lima	2 (2.35%)	253	10	2022-04-22	2022-04-27	5 days, 2:55:01	2	6

Fonte: Os autores.

A Figura 19 mostra os dias da semana em que foram feitos mais *commits* no repositório **front-end**, é notável que aos domingos a quantidade de *commits* nos fins de semana é maior do que nos dias da semana.

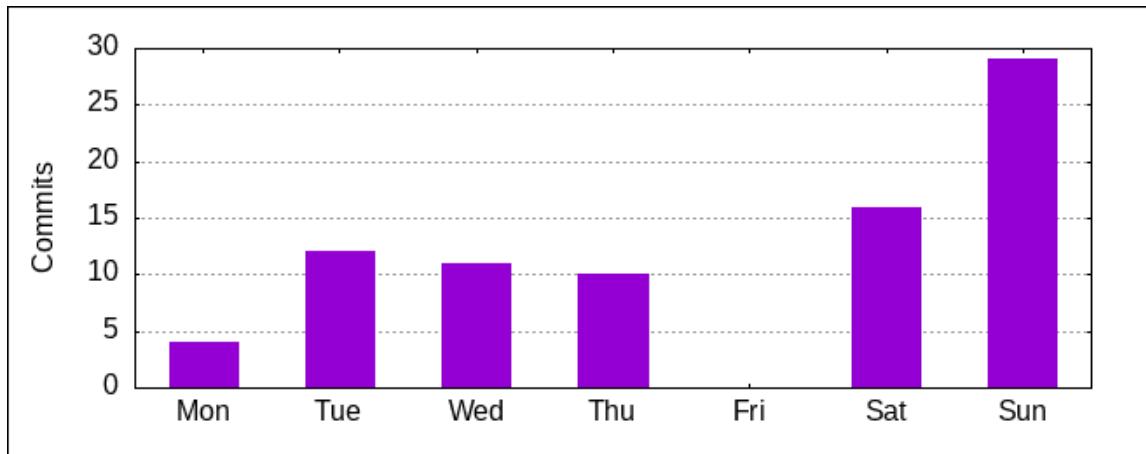
Figura 19 – Dias da semana - Projeto front-end



Fonte: Os autores.

A Figura 20 mostra os dias da semana em que foram feitos mais *commits* no repositório **back-end**, pode-se notar uma alta taxa de *commits* durante o domingo, enquanto o resto da semana permanece de forma quase igual, com exceção da sexta-feira e segunda-feira que possuem um percentual de atividade menor que os demais dias da semana.

Figura 20 – Dias da semana - Projeto Backend



Fonte: Os autores.

2.5 Validações de segurança, interface e código

Nessa seção serão abordados algumas validações que fizemos na nossa aplicação e a justificativa deles.

2.5.1 Teste dos *headers* da API

Para os testes de *headers* de segurança, foi usado o site [Security Headers](#), onde é possível verificar em qual nota se enquadrava a [API](#) do projeto. Foram feitos testes de um dos nossos *endpoints* (O resultado é o mesmo para todos), e durante o primeiro teste, o site indicou que nota F, pela resposta da aplicação não conter nenhum *header* de segurança, então com base nesse resultado, foi adicionado as dependências que faltavam e assim, foi possível subir a nota, como demonstra a Figura 21.

Figura 21 – Validação dos *headers*

The screenshot shows the homepage of the **Security Headers** tool, sponsored by Probely. It features a large green button with the text "Scan your site now". Below it is a text input field containing the URL <https://estagiei.herokuapp.com/api/vaga>. To the right of the input field is a large green "Scan" button. Underneath the input field are two checkboxes: "Hide results" (unchecked) and "Follow redirects" (checked). The main content area is titled "Security Report Summary" and displays a large green "A+" grade. To the left of the grade is a green square icon with a white "A+" symbol. On the right side of the summary area, there is a table with the following data:

Site:	https://estagiei.herokuapp.com/api/vaga
IP Address:	23.22.130.173
Report Time:	06 Jun 2022 16:15:13 UTC
Headers:	<input checked="" type="checkbox"/> Strict-Transport-Security <input checked="" type="checkbox"/> Content-Security-Policy <input checked="" type="checkbox"/> X-Frame-Options <input checked="" type="checkbox"/> X-Content-Type-Options <input checked="" type="checkbox"/> Referrer-Policy <input checked="" type="checkbox"/> Permissions-Policy

Fonte: Os autores.

Figura 22 – URL da documentação dos nossos endpoints (Swagger UI)



Fonte: Os Autores.

2.5.2 Teste de TLS do front-end

Também foi testado o *website* com relação ao certificado **TLS**, que indica se um site utiliza o protocolo *Hypertext Transfer Protocol Secure* (HTTPS) ou não. E pela aplicação estar hospedada no [Netlify](#), ele automaticamente já providencia um certificado com o [Let's Encrypt](#) quando é criado um domínio personalizado. Assim como mostra a [Figura 23](#), a aplicação possui um certificado **TLS** ativo.

Figura 23 – Teste de TLS

SSL Report: estagiei.herokuapp.com

Assessed on: Mon, 06 Jun 2022 16:28:35 UTC | [Hide](#) | [Clear cache](#)

[Scan Another >>](#)

	Server	Test time	Grade
1	54.243.238.66 ec2-54-243-238-66.compute-1.amazonaws.com Ready	Mon, 06 Jun 2022 16:22:36 UTC Duration: 89.306 sec	A+
2	107.22.57.98 ec2-107-22-57-98.compute-1.amazonaws.com Ready	Mon, 06 Jun 2022 16:24:05 UTC Duration: 89.85 sec	A+
3	3.209.172.72 ec2-3-209-172-72.compute-1.amazonaws.com Ready	Mon, 06 Jun 2022 16:25:34 UTC Duration: 91.364 sec	A+
4	23.22.130.173 ec2-23-22-130-173.compute-1.amazonaws.com Ready	Mon, 06 Jun 2022 16:27:06 UTC Duration: 88.833 sec	A+

SSL Report v2.1.10

Copyright © 2009-2022 [Qualys, Inc.](#). All Rights Reserved. [Terms and Conditions](#)
[Try Qualys for free!](#) Experience the award-winning [Qualys Cloud Platform](#) and the entire collection of [Qualys Cloud Apps](#), including [certificate security](#) solutions.

Fonte: Os autores

Figura 24 – URL do front-end da nossa aplicação

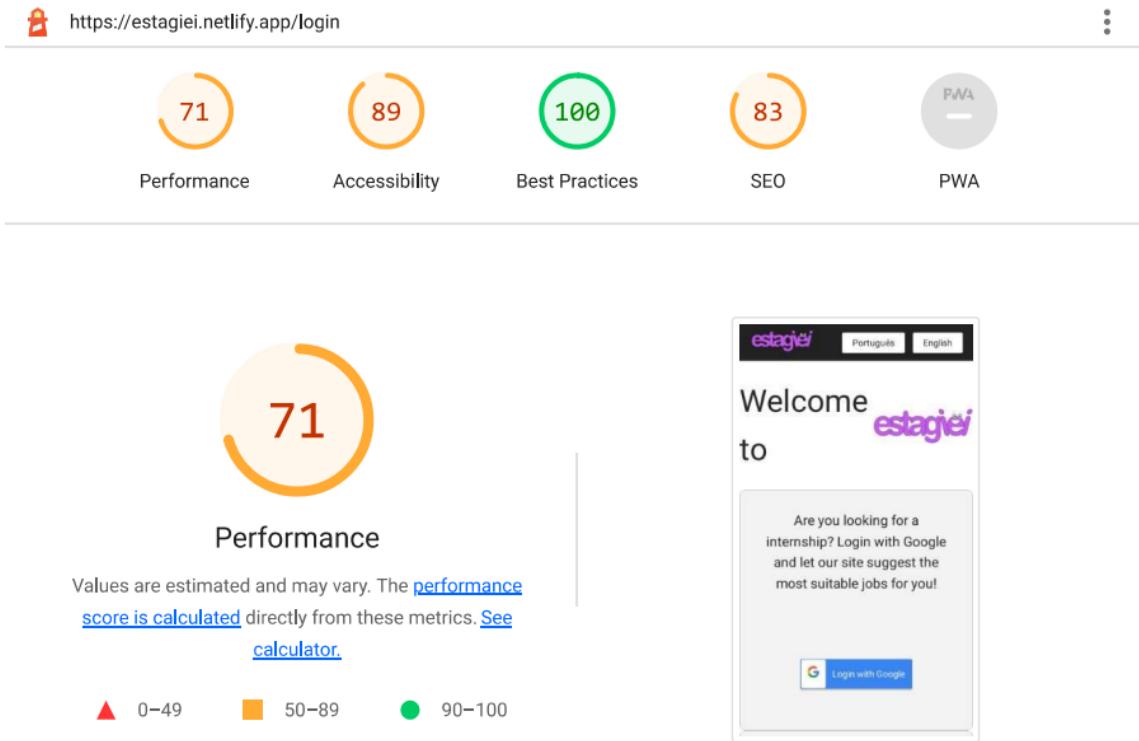
<<https://estagiei.netlify.app/>>

Fonte: Os Autores.

2.5.3 Teste de desempenho do front-end

Através da extensão [Lighthouse](#), foi verificado como estava o desempenho, acessibilidade, etc. Da tela de login, e foi percebido pontos a serem melhorados, principalmente na questão do desempenho. Pontos esses que serão ajustados no próximo semestre. A Figura 25 ilustra o resultado.

Figura 25 – Teste de desempenho do front-end

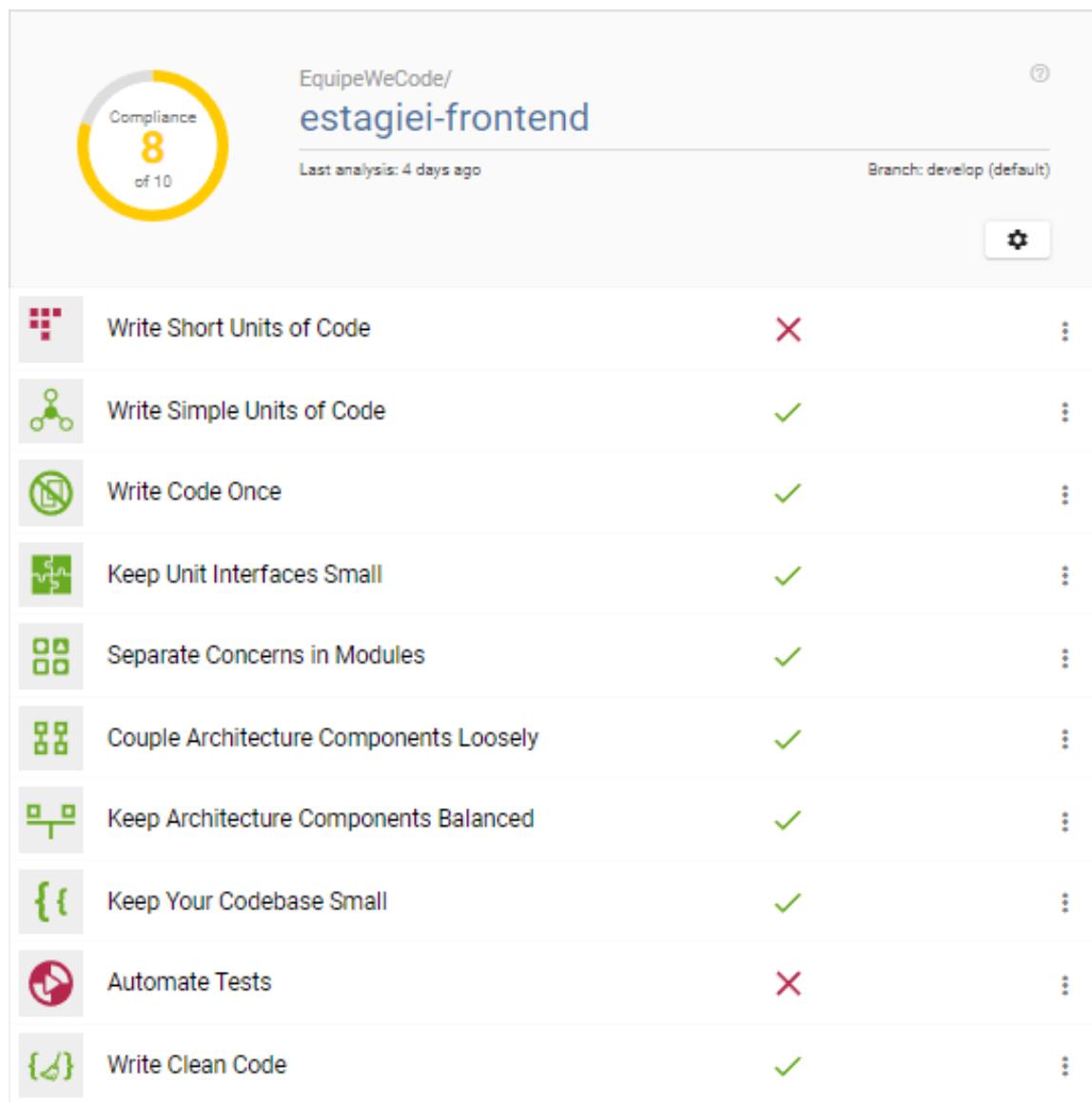


Fonte: Os autores.

2.5.4 Análise de código

Fizemos também uma verificação no nosso código, tanto no front-end como no back-end, e o resultado foi relativamente satisfatório, faltando apenas alguns pontos que serão melhorados ao longo do tempo. As figuras Figura 26 e Figura 27 demonstram os resultados do front-end e back-end respectivamente.

Figura 26 – Análise de código do front-end

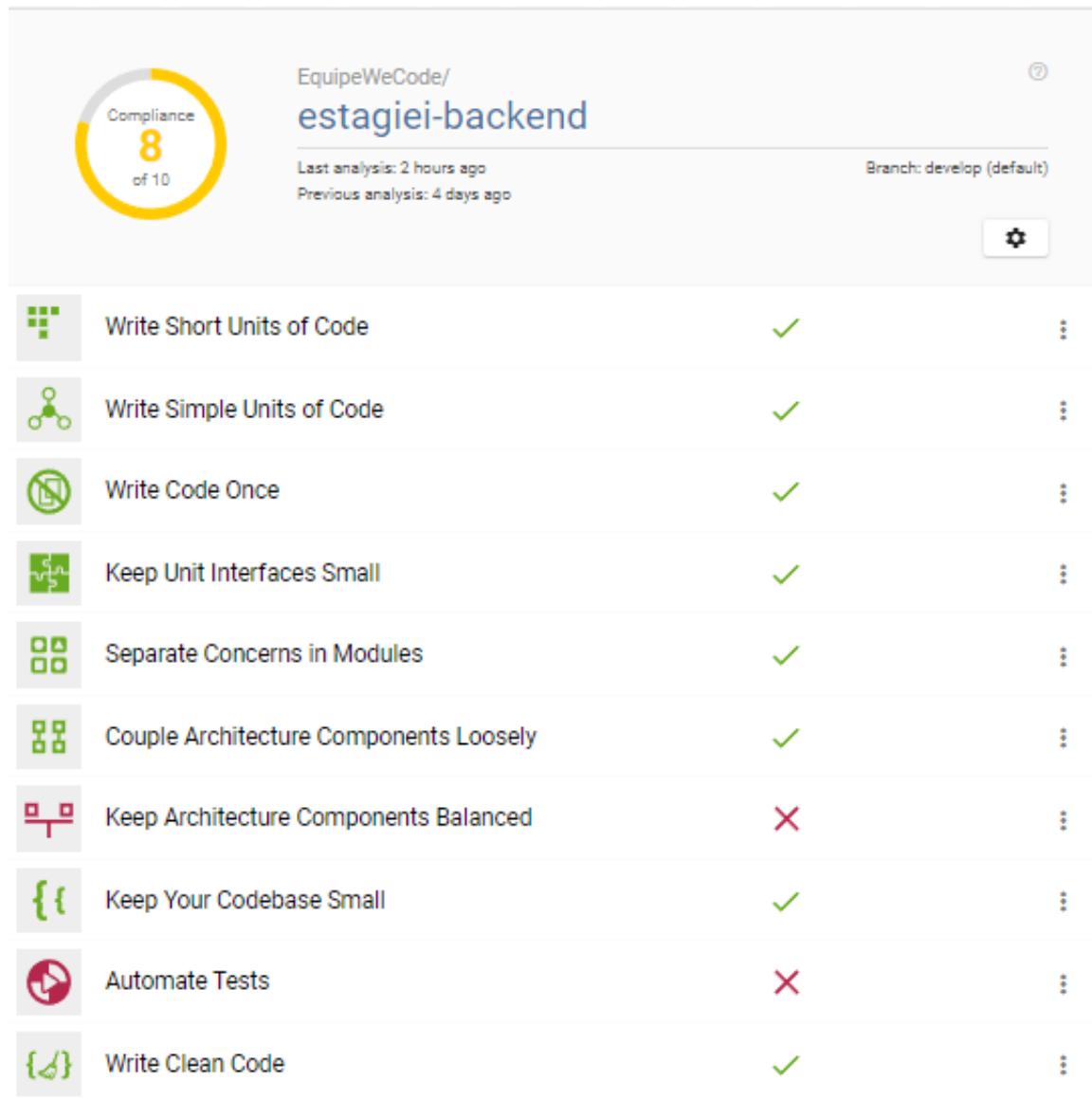


The screenshot shows a static code analysis report for the repository 'EquipeWeCode/estagiei-frontend'. The top left features a circular 'Compliance' badge with a yellow border, showing '8 of 10'. The top right displays the repository name 'EquipeWeCode/estagiei-frontend', the last analysis time '4 days ago', and the branch 'Branch: develop (default)'. A gear icon for settings is also present. The main area is a table listing ten coding best practices, each with an icon, a description, and a status indicator (green checkmark or red X). The rows are:

	Write Short Units of Code	X	
	Write Simple Units of Code	✓	
	Write Code Once	✓	
	Keep Unit Interfaces Small	✓	
	Separate Concerns in Modules	✓	
	Couple Architecture Components Loosely	✓	
	Keep Architecture Components Balanced	✓	
	Keep Your Codebase Small	✓	
	Automate Tests	X	
	Write Clean Code	✓	

Fonte: Os autores.

Figura 27 – Análise de código do back-end



Fonte: Os autores.

2.5.5 Validador HTML

Foi feito também uma verificação no **HTML** do front-end. A Figura 28 demonstra o resultado.

Figura 28 – Validação do HTML



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. (??)

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. (??)

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 (??), 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.

§ 1º 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.

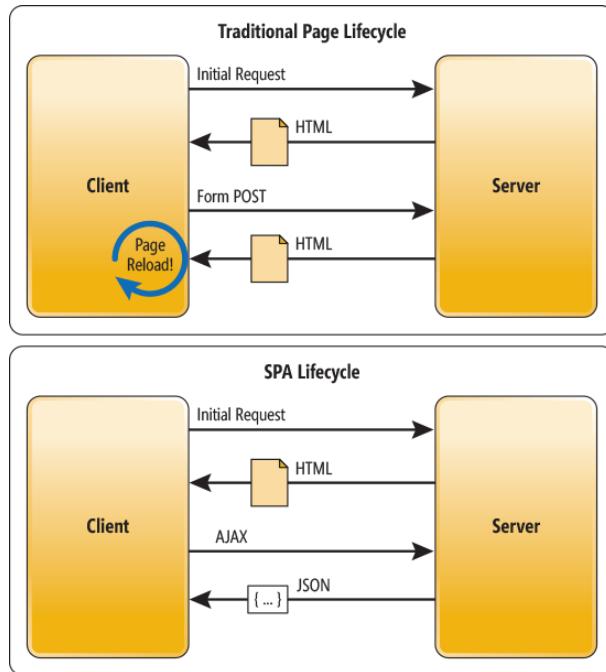
§ 2º Se a instituição de ensino adotar verificações de aprendizagem periódicas ou finais, nos períodos de avaliação, a carga horária do estágio será reduzida pelo menos à metade, segundo estipulado no termo de compromisso, para garantir o bom desempenho do estudante.(??)

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 **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. (??)

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 (??), como ilustrado na [Figura 29](#).

Figura 29 – Ciclo de vida: página web tradicional X **SPA**



Fonte: (??)

3.3 Application Programming Interface (API)

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.(??)

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 (??). 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. (??)

3.3.1 API REST

Representational State Transfer (REST) é um estilo de arquitetura com um conjunto de restrições (??). Uma **API** que segue todas as seis restrições é chamada de **API RESTful** (??). 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平衡amento 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. (??)

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 *HTTPS*, por exemplo (??). 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.

3.4 Competências

As habilidades que possuímos podem ser classificadas como técnicas e comportamentais, nos termos em inglês, *hard skills* e *soft skills*, respectivamente.

O que chamamos de competências se referem às *soft skills*, ou seja, o conjunto de características comportamentais da pessoa. Essas características podem ser determinantes na busca por estágio e emprego, assim se faz necessário darmos destaque ao entendimento do que são tais habilidades.

"São capacidades subjetivas, que atuam no espectro comportamental e social do ser humano e não dependem de diplomas ou certificados."(??) As habilidades técnicas são mais fáceis de serem registradas, detectadas, mapeadas e relacionadas, porém o caráter subjetivo das competências faz com que a mesma habilidade possa ser descrita de modos distintos, ainda que similares, como "Trabalho em equipe" pode ser dito como "Saber trabalhar em equipe" ou "Trabalhar bem em equipe", além de comumente serem importantes na obtenção e manutenção de uma vaga de estágio/emprego.

As vagas existentes de diversas áreas buscam por determinados perfis, os quais comumente se referem ao tipo de atitude procurada para preencher a vaga além dos conhecimentos técnicos. Conhecer o seu próprio perfil e aprimorar certas características dará maiores chances de encontrar vagas compatíveis consigo.

No próximo capítulo há uma lista das competências usadas como parâmetros de associação das vagas com os estudantes.

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 **front-end** e **back-end** de modo que eles se comuniquem via protocolo **HTTP** com requisições e respostas no formato **JSON**. Para o desenvolvimento do **front-end** utilizaremos **TypeScript** por meio da biblioteca **React**; o **back-end** será desenvolvido utilizando Java com o micro **framework Spring Boot**.

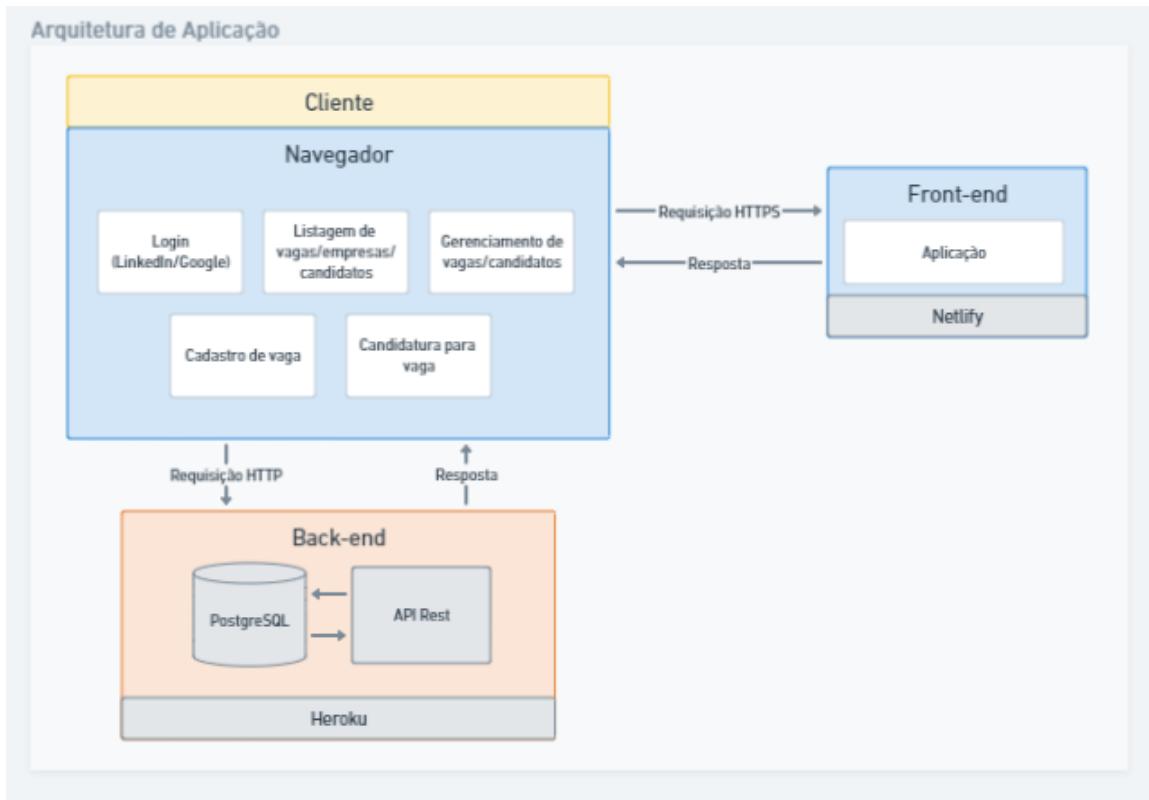
Em relação ao **deploy** das aplicações, o **front-end** será hospedado na plataforma **Netlify**, que hospeda e mantém um site com implantação contínua e HTTPS, proporcionando uma melhor agilidade de desenvolvimento, enquanto o **back-end** será hospedado no **Heroku**, que é uma plataforma como serviço 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

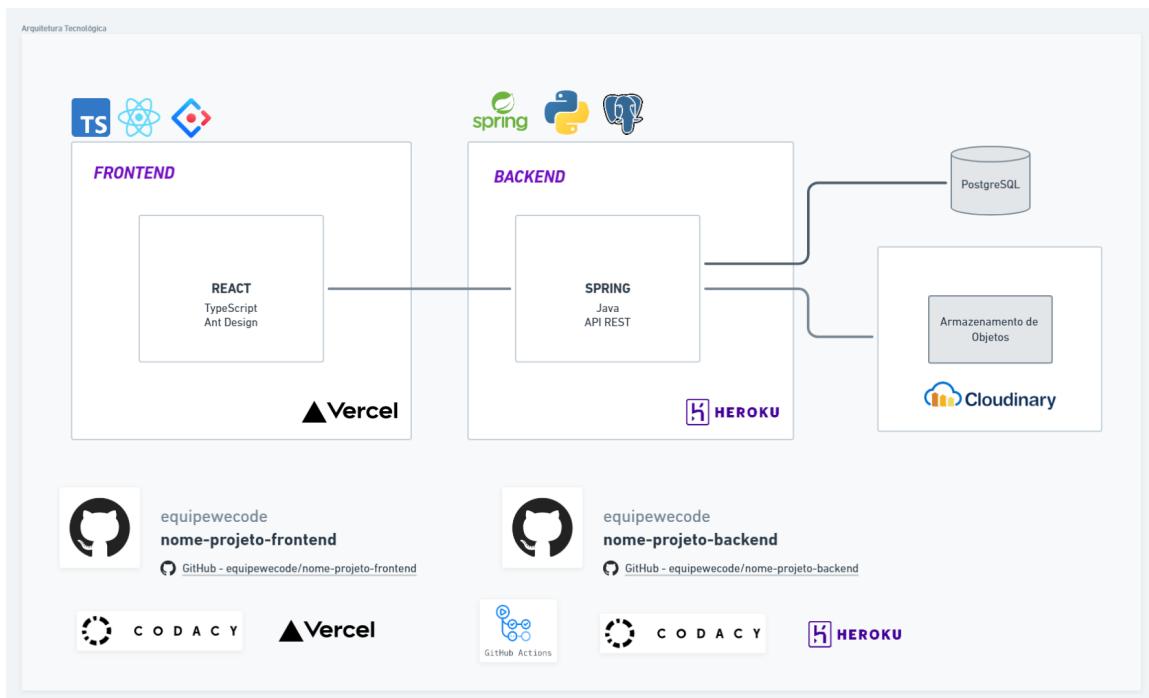
Os diagramas **Figura 30**, **Figura 31** e **Figura 32** ilustram de modo geral a arquitetura pensada para a solução proposta, utilizando das tecnologias já citadas.

Figura 30 – Arquitetura de Aplicação



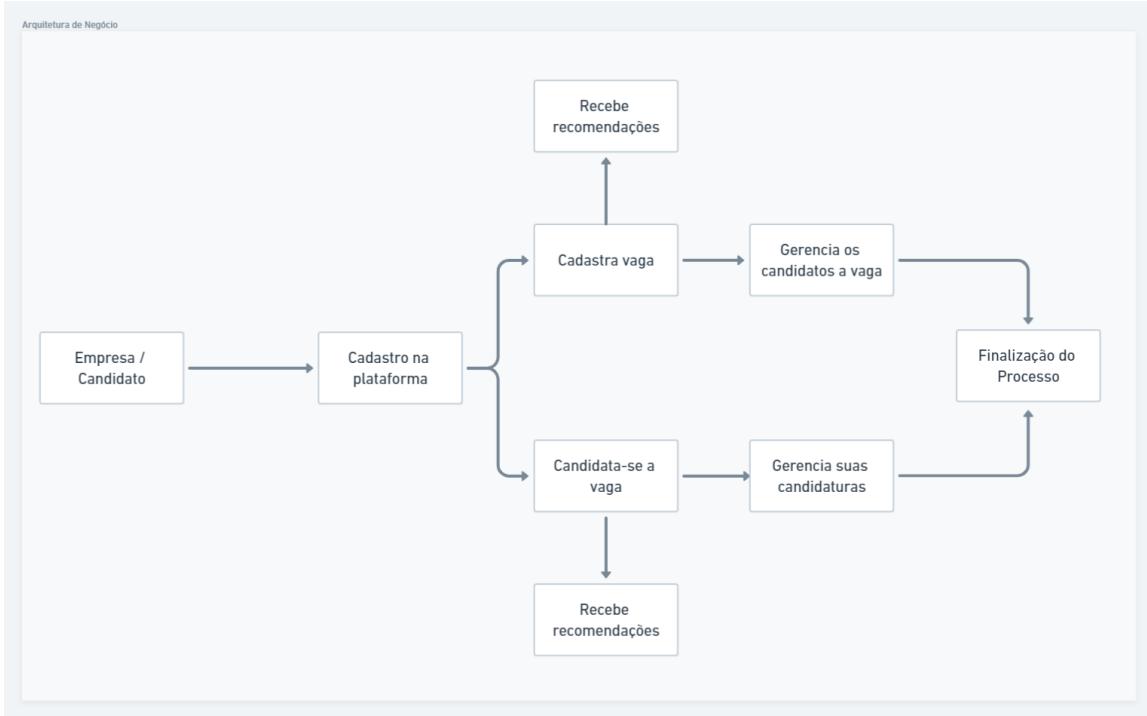
Fonte: Produzido pelos autores

Figura 31 – Arquitetura Tecnológica



Fonte: Produzido pelos autores

Figura 32 – Arquitetura de Negócios



Fonte: Produzido pelos autores

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 (??). 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 <i>feedback</i> 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 (??). 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 responsável 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 <i>feedback</i> , ele deve ter pelo menos iniciado o processo seletivo	RF-008
RN-005	O <i>feedback</i> 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

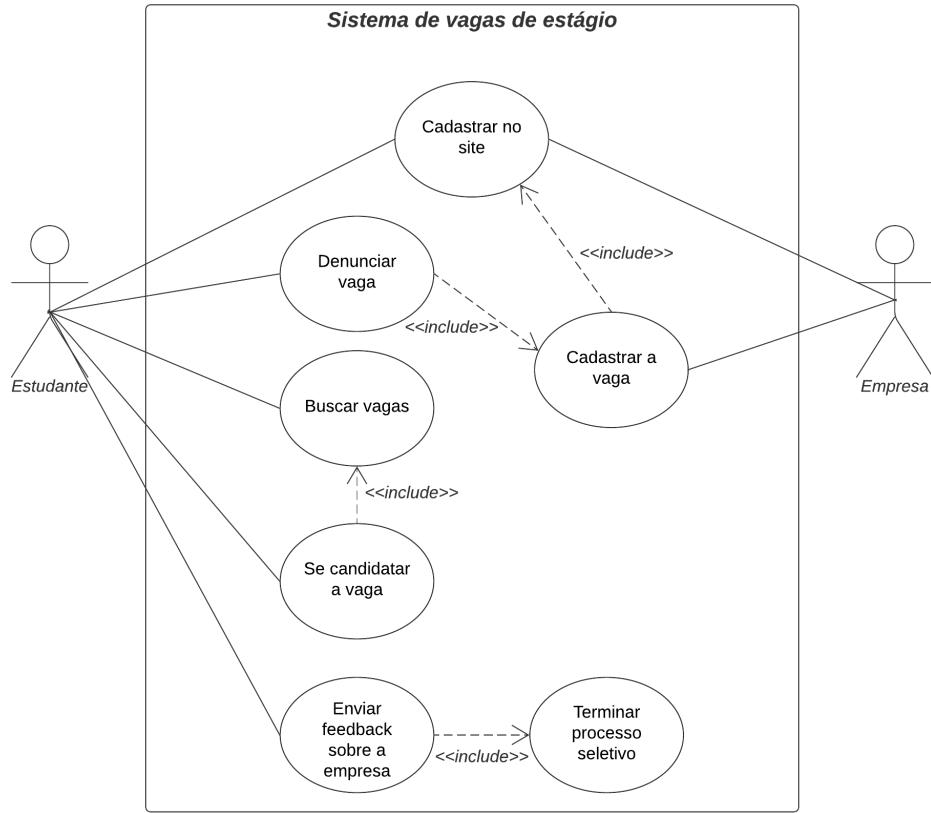
História
Como estudante, eu quero buscar as vagas de acordo com o filtro que eu escolher.
Como empresa, eu quero gerenciar a minha vaga para que possa visualizar a quantidade de candidatos dentre outras informações pertinentes.
Como estudante, eu quero receber recomendações de vaga para que a minha pesquisa 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

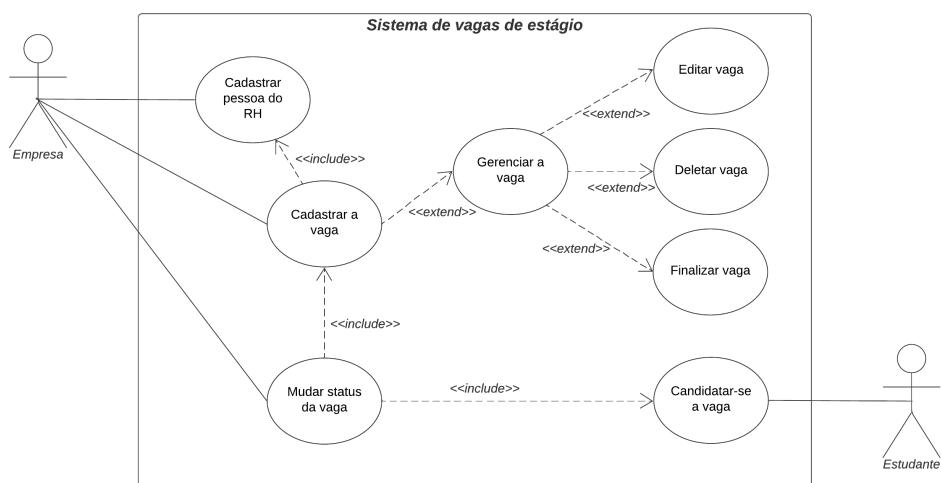
Em [Figura 33](#), [Figura 34](#) e [Figura 35](#) estão demonstrados os diagramas de casos de usos que são pertinentes à nossa aplicação.

Figura 33 – Caso de Uso 1 - Funcionalidades do estudante



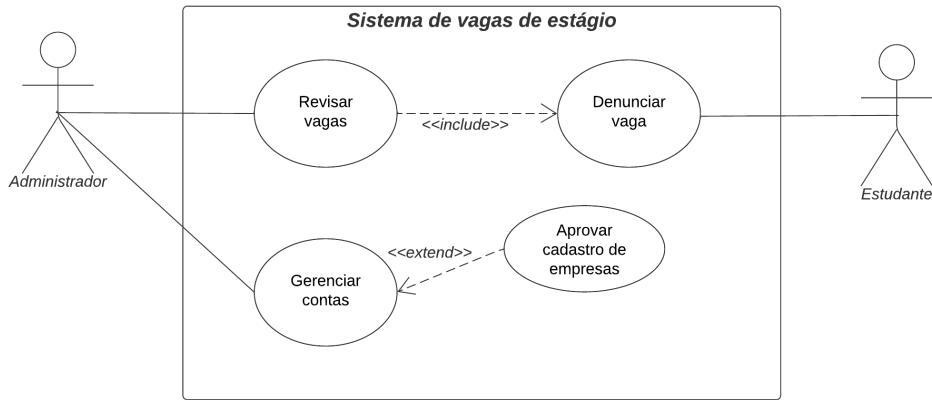
Fonte: Os autores

Figura 34 – Caso de Uso 2 - Funcionalidades da empresa



Fonte: Os autores

Figura 35 – 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**, entregamos 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**, incrementamos o que já foi desenvolvido durante a **POC** com funcionalidades importantes ao nosso sistema, como a recomendação de vagas ao estudante e edição de perfil. Além disso, serão feitos os teste unitários e validações de segurança, **HTML**, testes de interface, etc. 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 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; recomendação de candidatos para empresas; opção de contato com o candidato via *Whatsapp*; *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; e acessibilidade com o [VLibras](#).

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 (??), 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 (??).

4.3.4 API dos Correios

Realizaremos uma integração do a [API](#) dos Correios a fim de resgatar as informações de endereço dos usuários cadastrados a partir do CEP informado.

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.

4.4.2.1 Codificação geral

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 no **back-end** e 2 espaços no **front-end**.
- Classes e interfaces em **CamelCase** e substantivos.
- Métodos em **camelCase** e verbos.
- Constantes em **UPPER_CASE**.

No **back-end** os pacotes serão bem divididos, tendo o pacote *entity* para as entidades mapeadas do banco de dados, *controller* para os *controllers* e *endpoints* e *service*, além de outros pacotes para fins de separação de código.

4.4.2.2 Commits

Para os commits dos repositórios de front e back-end estamos utilizando a convenção de usar prefixos que melhor identificam do que se trata aquele commit.

São eles:

- **fix:** correção de erros no código;
- **feat:** introdução de uma nova funcionalidade;

Além disso também realizamos a prática de realizar *Pull Requests* ao invés de mandar as alterações diretamente no ramo principal do repositório.

4.4.3 Design Patterns e boas práticas

Para padrões de projetos, serão essencialmente utilizados 3 padrões muito utilizados pela comunidade de desenvolvimento: *Factory Method*, *Builder* e *Facade*, além da possibilidade de usarmos outros conforme a necessidade.

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 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 Responsibility 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 [back-end](#) 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](#) constrói 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

A implantação dos itens acima se deu por meio das *Actions* do [GitHub](#), cujos *scripts* mostramos a seguir.

4.4.4.1 Script de integração com Heroku

```
name: Deploy to development environment
```

```
on:
```

```
  push:
```

```
    branches: [ develop ]
```

```
jobs:
```

```
  check:
```

```
runs-on: ubuntu-latest
steps:
  - uses: actions/checkout@v2

  - name: Make gradlew executable
    run: chmod +x ./gradlew

  - name: Run check
    run: ./gradlew check

deploy:
  needs: check
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v2

    - name: Make gradlew executable
      run: chmod +x ./gradlew

    - name: Run build
      run: ./gradlew build

    - name: Send to heroku
      uses: akhileshns/heroku-deploy@v3.12.12
      with:
        heroku_api_key: ${{ secrets.HEROKU_API_KEY }}
        heroku_app_name: ${{ secrets.HEROKU_APP_NAME }}
        heroku_email: ${{ secrets.HEROKU_EMAIL }}
```

4.4.4.2 Script de integração com o Netlify

```
name: 'Netlify Deploy'

on:
  release:
    types: ['published']

jobs:
  deploy:
    name: 'Deploy'
    runs-on: ubuntu-latest
```

```
steps:  
  - uses: actions/checkout@v1  
  - uses: jsmrcaga/action-netlify-deploy@master  
    with:  
      NETLIFY_AUTH_TOKEN: ${{ secrets.NETLIFY_AUTH_TOKEN }}  
      NETLIFY_SITE_ID: ${{ secrets.NETLIFY_SITE_ID }}  
      NETLIFY_DEPLOY_MESSAGE: "Prod deploy v${{ github.ref }}"  
      NETLIFY_DEPLOY_TO_PROD: true  
      install_command: 'tsc && vite build'  
      build_directory: 'dist'
```

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 [back-end](#) deverá ser testado com o [framework](#) JUnit, já no [front-end](#) 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 (??), 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 **back-end** uma **API** para a transferência de dados e comunicação com o nosso **front-end**:

- 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 papéis;
- Criptografia: Seguiremos o protocolo e padrão **HTTPS** para a transferência de mensagens entre o **back-end** e o **front-end**, 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 [Netlify](#), 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](#) (??) 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 (??) 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](#) (??) 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

A fim 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.

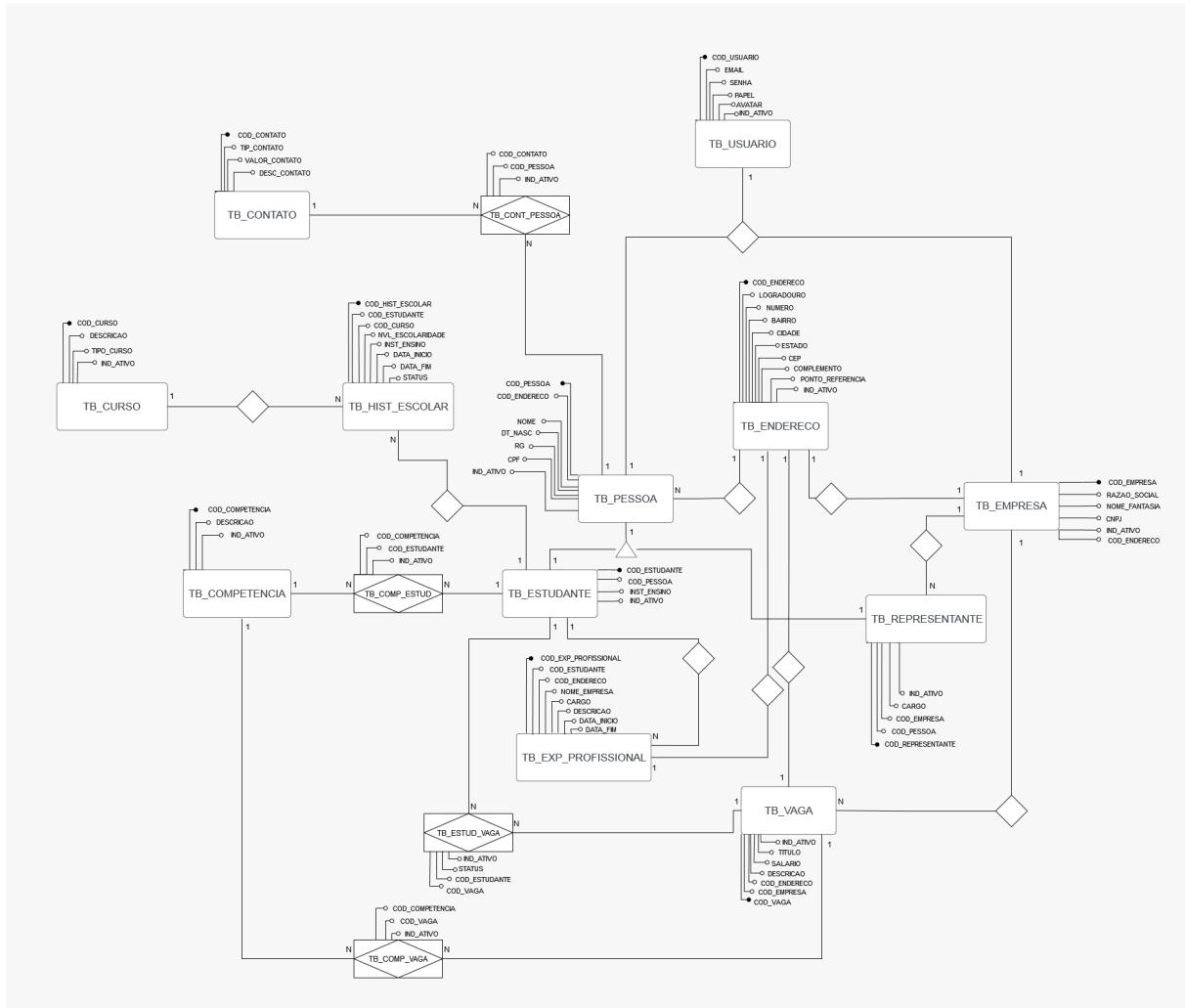
Caso o engajamento da aplicação aumente, a medida que o número de usuários aumenta, incrementando proporcionalmente o rendimento com o *Google Adsense*, poderá ser revisto os planos dos servidores para atender maiores níveis de requisições e buscar contratos com empresas para aumentar a rentabilidade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

4.7 Modelagem e definições técnicas

Esta seção tem por objetivo demonstrar as modelagens e padronizações utilizadas no desenvolvimento da aplicação.

4.7.1 Modelo Entidade Relacionamento

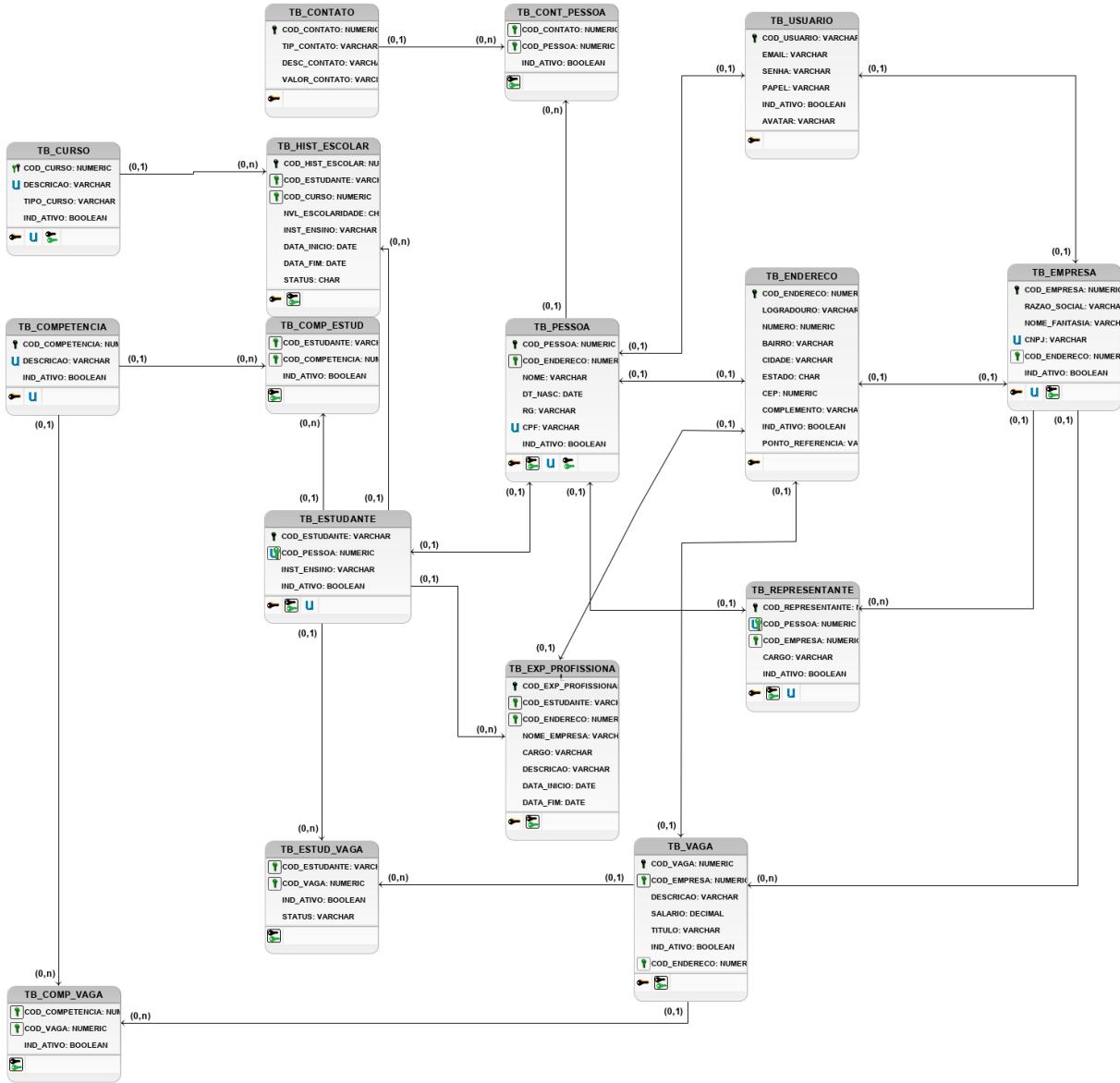
Figura 36 – Modelagem Entidade Relacionamento



Fonte: Os autores

4.7.2 Diagrama Entidade-Relacionamento

Figura 37 – Diagrama Entidade Relacionamento



Fonte: Os autores

4.7.3 Dicionário de Dados

A seguir mostramos as tabelas do nosso dicionário de dados.

Figura 38 – Legenda

LEGENDA	
SIGLA	DESCRÍÇÃO
PK	Primary Key
FK	Foreign Key
NN	Not Null
UQ	Unique
CK	Check
DEFAULT	Default

Fonte: Os autores

Figura 39 – Campos Usuário

tb_usuario		
Campo	Tipo	Restrição
Cod_usuario	SERIAL	PK
Senha	VARCHAR(50)	
Papel	VARCHAR(25)	DEFAULT
Email	VARCHAR(50)	
Ind_Ativo	BOOLEAN	DEFAULT

Fonte: Os autores

Figura 40 – Campos Pessoa

Campo	Tipo	Restrição
Cod_Pessoa	SERIAL	PK
Cod_Endereco	SERIAL	FK, NN
Tip_Contato	VARCHAR(10)	
Valor_Contato	VARCHAR(25)	
Nome	VARCHAR(50)	NN
Dt_Nasc	DATE	NN
RG	VARCHAR(11)	NN
CPF	VARCHAR(13)	NN, UQ
Ind_Ativo	BOOLEAN	DEFAULT

Fonte: Os autores

Figura 41 – Campos Estudante

Campo	Tipo	Restrição
Cod_Estudante	SERIAL	PK
Cod_Pessoa	SERIAL	FK, NN, UQ
Inst_Esino	VARCHAR(128)	NN
Nvl_Escolaridade	CHAR(2)	NN
Exp_Profissional	TEXT	
Ind_Ativo	BOOLEAN	DEFAULT

Fonte: Os autores

Figura 42 – Campos Empresa

tb_empresa		
Campo	Tipo	Restrição
Cod_Empresa	SERIAL	PK
Razao_Social	VARCHAR(50)	NN
Nome_Fantasia	VARCHAR(50)	
CNPJ	VARCHAR(20)	NN, UQ
Cod_Endereco	SERIAL	FK, NN
Ind_Ativo	BOOLEAN	DEFAULT

Fonte: Os autores

Figura 43 – Campos Representante RH

tbRepresentanteRh		
Campo	Tipo	Restrição
Cod_Representante	SERIAL	PK
Cod_Pessoa	SERIAL	FK, NN
Cod_Empresa	SERIAL	FK
Cargo	VARCHAR(50)	NN
Ind_Ativo	BOOLEAN	DEFAULT

Fonte: Os autores

4.7.4 Endpoints da API

A seguir listamos os [endpoints](#) mapeados até o momento, com seus respectivos métodos de requisição [HTTP](#).

Quadro 8 – Endpoints da API

Classe Java	Método	Endpoint
LoginController	POST	/api/loginEstudante
EstudanteController	GET	/api/estudante/{id}
VagaController	GET	/api/vaga

Fonte: Os Autores

4.7.5 Listagem das Competências

A seguir apresentamos as competências parametrizadas a fim de realizar a recomendação de vagas para os estudantes de acordo com seu perfil.

- Comunicação
- Liderança
- Flexibilidade
- Trabalho em Equipe
- Criatividade
- Proatividade
- Empatia
- Ética no trabalho
- Pensamento crítico
- Atitude positiva
- Desenvolvimento da esquipe
- Motivação
- Influência
- Capacidade de tomar decisões
- Conhecimento político e cultural
- Poder de negociação
- Estabelecimento de confiança
- Gerenciamento de conflitos
- Coaching
- Boa escrita
- Colaboração
- Organização
- Resiliência

- Trabalho sob pressão
- Capacidade de resolver problemas
- Relacionamento interpessoal
- Adaptação
- Honestidade
- Autogestão
- Autoconfiança
- Inteligência emocional
- Desenvolvimento pessoal
- Interesse em aprender

Glossário

API RESTful	API que segue todas as restrições da arquitetura REST . - Citado em 31 , 32
AWS	Amazon Web Services - Plataforma em nuvem <i>on-demand</i> que disponibiliza diversos serviços web. - Citado em 48 , 58
back-end	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 2 , 17 , 19 , 20 , 21 , 25 , 27 , 33 , 42 , 44 , 46 , 47
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 33 , 43 , 44
endpoint	Localização digital onde uma API recebe requisições sobre um recurso específico em seu servidor. Os endpoints comumente são uma <i>Universal Resource Locator (URL)</i> , indicando uma ponta da conexão para a recuperação do recurso através da API . - Citado em 4 , 42 , 46 , 47 , 54
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 33 , 46 , 58
front-end	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 2 , 6 , 16 , 18 , 19 , 20 , 21 , 23 , 24 , 25 , 26 , 27 , 33 , 42 , 46 , 47
Git	Sistema de controle de versão de arquivos. - Citado em 16 , 57
GitHub	provedor de hospedagem na internet para desenvolvimento de software e controle de versionamento usando Git . - Citado em 44
GitStats	Ferramenta que gera estatísticas de repositórios Git - Citado em 16
Heroku	Plataforma em nuvem como um serviço que suporta diversas linguagens de programação. - Citado em 33 , 48 , 57
Heroku CI	Instância da Heroku responsável pela integração contínua. - Citado em 44
Jira Software	Ferramenta de gerenciamento que permite o monitoramento de tarefas e acompanhamento de projetos. - Citado em 11 , 12

Let's Encrypt	Gerador de certificado TLS gratuito para <i>websites</i> - Citado em 23
Lighthouse	Ferramenta que verifica o desempenho de um site e aponta melhorias - Citado em 24
Linux	Kernel open-source usado em diversos sistemas operacionais. - Citado em 48
Logback	Logback é uma estrutura de log para aplicações java, criada como sucessora do popular projeto log4j. - Citado em 41
Netlify	Plataforma em nuvem que faz o <i>host</i> de páginas web. - Citado em 23, 33, 48
PostgreSQL	Sistema de Gerenciamento de Banco de Dados Relacional, gratuito e open-source. - Citado em 33
RDS	Amazon Relational Databases - Serviço de banco de dados da AWS . - Citado em 48
React	Biblioteca JavaScript gratuita e open-source para a construção de interfaces baseadas em componentes. - Citado em 33
Scrum	Metodologia ágil de software concebida por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 90. - Citado em 11, 58
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 11
Security Headers	Site que verifica a resposta de um <i>endpoint</i> e atribui uma nota de acordo com a quantidade de <i>headers</i> de segurança que esse <i>endpoint</i> retorna - Citado em 22
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 33, 41
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 4, 11, 12, 13
StatSVN	Ferramenta que gera estatísticas de repositórios SVN - Citado em 14
SUN MICROSYSTEMS	I. Java coding conventions, 1996. - Citado em 41
SVN	Subversion - Sistema de controle de versão de arquivos. - Citado em 14
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 46

- TypeScript Linguagem de programação fortemente tipada sobre JavaScript. -
Citado em [33](#)
- VLibras Conjunto de ferramentas para a tradução de texto em Português para
LIBRAS gratuitas e de código aberto, mais informações disponíveis
no endereço <<https://www.gov.br/governodigital/pt-br/vlibras>>. -
Citado em [40](#)

Apêndices

APÊNDICE A – Publicações do Blog

Figura 44 – URL do blog da equipe LATEX



<<https://wecodeifsp.blogspot.com/>>

Fonte: Os Autores.

Figura 45 – Blog: Apresentação

Apresentação

Olá, nos somos a **Equipe WeCode** do curso **Tecnológico de Análise e Desenvolvimento de Sistemas** do **Instituto Federal de Educação, Ciência e Tecnologia de São Paulo**. Aqui apresentaremos o caminhar do desenvolvimento do projeto final do curso, que será realizado em dois semestres nas disciplinas de PI1A5 e PI1A6.

A Equipe WeCode espera que tenham uma boa experiência em nosso blog e que o conteúdo seja útil em auxiliar os próximos concluintes, informativo aos que buscam saber mais sobre nosso projeto e interessante aos meros curiosos.

Todos são bem-vindos!

Equipe WeCode

Por Bruna Pires - março 21, 2022 Nenhum comentário:



Marcadores: Introdução

Figura 46 – Blog: Semana 1

Semana 1 - 14/03 até 20/03/22

Início, definição de equipes, pesquisa de temas

No dia 14/03/22 (segunda-feira) tivemos a primeira aula da disciplina de PI1A5, onde recebemos as orientações sobre como a disciplina irá funcionar, dicas importantes e materiais de apoio. Nossa equipe já estava formada de antemão, então neste dia acordamos um nome: *WeCode*.

Nos reunimos neste mesmo dia para iniciarmos pesquisas e sugerirmos temas de soluções a serem apresentadas na próxima aula.

Ao longo da semana conseguimos chegar em cinco ideias:

- Um sistema de chamados que supra as necessidades do dia-a-dia de uma empresa de suporte técnico;
- Aplicativo para encontrar postos de gasolina próximos e selecionar o(s) melhor(es) de acordo com o preço da região (área, bairro, cidade);
- Adaptação de um jogo de cartas educacional para uma versão digital;
- Aplicação (similar ao *Get Ninjas*) para a contratação de serviços digitais remotos, como desenvolvimento de sites, edições de mídia, entre outros;
- Portal de empregos, cujo o foco é facilitar o encontro de empregadores em busca de estagiários/trainees e quem está em busca de sua primeira oportunidade com uma vaga coerente com a posição de estágio/trainee, ou seja, que não exija experiência prévia de trabalho.

Por Bruna Pires - [março 28, 2022](#) Nenhum comentário:



Marcadores: [Concepção](#)

Figura 47 – Blog: Semana 2

Semana 2 - 21/03 até 27/03/22

Definições de metodologias e gestão, detalhamento das propostas, eleição do tema preferido

Na aula do dia 21/03/22 (segunda-feira) a equipe se reuniu com o professor para mostrar os rascunhos das ideias levantadas. Nessa mesma aula, a equipe definiu seguir a metodologia Scrum, com sprints de uma semana, usando a ferramenta Jira para nos organizarmos.

Assim, já iniciamos nossa primeira sprint, a princípio, com tarefas relativas aos preparativos para o início do desenvolvimento do projeto logo que o tema fosse aprovado, como a criação deste blog, criação da organização no GitHub, etc.

No dia 26/03/22 (sábado), entramos em reunião com o Professor orientador via plataforma RNP, para que ele nos auxiliasse quanto a preparação da planilha com a elicitação de requisitos para cada ideia que tivemos. Durante o próprio sábado e no domingo, trabalhamos na elaboração dessa planilha com os requisitos essenciais das nossas propostas.

Também desenvolvemos uma apresentação e levamos em conta os seguintes pontos para definir que o Portal de vaga de estágio seria a proposta a ser levada para avaliação:

- Objetivo;
- Melhorias quanto aos sistemas semelhantes;
- Tecnologias a serem utilizadas;
- Valor da solução.

Após a conclusão da montagem da planilha e da apresentação, esperamos o dia seguinte chegar para termos o veredito sobre o projeto.

Por Daniel Roberto - março 30, 2022 Nenhum comentário:



Marcadores: Concepção

Figura 48 – Blog: Semana 3

Semana 3 - 28/03 até 03/04/22

Aprovação da proposta, definição das tecnologias, e inicio do desenho da arquitetura

Na aula do dia 28/03/2022, após levantarmos os requisitos macros de cada tema, apresentamos as nossas cinco propostas para a avaliação do professor, foi apresentado e aprofundado na primeira proposta de tema, o portal de vagas para estágio, logo foi discutido melhorias para os requisitos apresentados e ideias de escopo na qual o projeto irá seguir. Com o tema escolhido, a equipe no mesmo dia se juntou para avaliar as primeiras possibilidades de tecnologias, e maneiras de implementar requisitos como o mapa de calor, dashboard de situação de vagas dos candidatos, questões de como funcionará o papel da empresa, e o relacionamento entre candidato e empresa.

Durante a semana foi discutido mais profundamente as tecnologias a serem utilizadas, bem como a arquitetura inicial do projeto, de inicio optamos pelo padrão cliente-servidor, no qual teria 2 servidores de aplicação, um para alocar os serviços REST em Java com o apoio do Spring MVC, e outro para alocar a parte de Machine Learning e IA em Python, como ainda estamos avaliando a questão do Machine Learning no Python, a decisão do 2º servidor continua prematura. Ainda no backend, foi pensado na possibilidade da utilização do PostgreSQL, o banco de dados estaria disponível em um servidor separado da aplicação a fim de fornecer os dados e possibilitar maior performance. O frontend foi pensado em React com Typescript, e para a estilização será utilizado o framework AntD.

Sobre a arquitetura, pensamos em utilizar os serviços da Amazon AWS para as aplicações Backend, Heroku para o banco de dados, e o Vercel para o frontend.

Ao sábado dia 02/04/2022, continuamos a debater as tecnologias com o professor, e a nos preparar para a realização do documento de proposta inicial.

Por Igor Nathan - abril 04, 2022 Nenhum comentário:



Marcadores: Concepção

Figura 49 – Blog: Semana 4

Semana 4 - 04/04 até 10/04/22

Elaboração da proposta Inicial

Bom dia / tarde / noite, pessoal.

Durante essa semana, após a aprovação do nosso tema, começamos a fazer o documento da nossa proposta inicial, onde descrevemos um pouco sobre o projeto (objetivos, justificativa, etc.), os requisitos (funcionais, não funcionais e regras de negócio), sobre as tecnologias usadas, arquitetura e possíveis integrações que nossa aplicação terá e uma pequena análise de concorrentes, onde comparamos as funcionalidades que cada concorrente oferece aos seus usuários.

No geral, eu (Leonardo) e a Bruna ficamos responsáveis pela elaboração do documento nessa semana, o Daniel e o Marcelo ajudaram em coisas específicas.

Após a elaboração do documento, alteramos nosso slide que tínhamos antes e adicionamos novas informações que não estavam lá, depois disso, upamos esses documentos no SVN.

Estou bastante ansioso para desenvolvermos nosso projeto na prática!

Por Leonardo Marques - abril 11, 2022 Nenhum comentário:



Marcadores: Concepção

Figura 50 – Blog: Semana 5

Semana 5 - 11/04 até 17/04/22

Incremento da proposta inicial e início do desenho da aplicação

Nessa semana foram feitas melhorias no documento da proposta inicial, fazendo modificações nos requisitos, também houve um incremento nas regras de negócios e requisitos não funcionais, fizemos correções nas tabelas e adicionamos as dependências das regras de negócios com os respectivos requisitos relacionados.

O assunto da possibilidade de implementação do machine learning com python veio a tona novamente e ainda estamos avaliando a possibilidade

Após as alterações no documento da proposta inicial começamos a discutir o desenho da aplicação e alguns tópicos foram citados como:

- Modelagem do banco de dados (pensar na modelagem do banco de dados cedo pode ajudar bastante na frente, pois nos ajuda a ter uma ideia do que precisaremos para o projeto futuramente).
- Organização da equipe (definição dos papéis de cada integrante da equipe).

Apesar de muitas coisas para fazer com a entrega do desenho da aplicação, estou ansioso para a etapa de desenvolvimento.

Por Lucas Lima - [abril 19, 2022](#) Nenhum comentário:



Marcadores: [Concepção](#)

Figura 51 – Blog: Semana 6

Semana 6 - 18/04 até 24/04/22

Finalização do desenho da aplicação e pensando na POC

Mais uma semana se passou na disciplina.

Neste meio tempo, estivemos desenvolvendo o desenho da aplicação, e refinando os textos pertinentes e necessários para a sua entrega. Estamos um pouco atrasados com a elaboração deste documento, então precisamos correr com algumas partes, principalmente em relação à modelagem de dados da aplicação.

Tendo isto feito, começaremos o desenvolvimento da POC que entregaremos no mês que vem! Como já decidido no desenho da aplicação, focaremos em desenvolver os seguintes itens do projeto, que são as funcionalidades mais básicas de login e cadastros simples:

- Registro de estudantes;
- Registro de empresas;
- Registro de representante de recursos humanos;
- Visualização de vagas disponíveis

Temos bastante trabalho a ser feito pela frente! Espero que tudo dê certo daqui para frente, que em pouco tempo será o momento em que colocaremos a mão no código.

Até a próxima semana!

Por Marcelo Junior - [abril 26, 2022](#) Nenhum comentário:



Marcadores: [Concepção](#)

Figura 52 – Blog: Semana 7 - 1

Semana 7 - 25/04 até 01/05/22

Ajustes do Desenho da Aplicação e Preparação para a POC

Nesta semana continuamos a elaborar a documentação do desenho da aplicação, a qual está quase terminada, e iniciamos a preparação dos ambientes para a Prova de Conceito (POC, na sigla em inglês).

Dentre alguns itens do desenho da aplicação, temos:

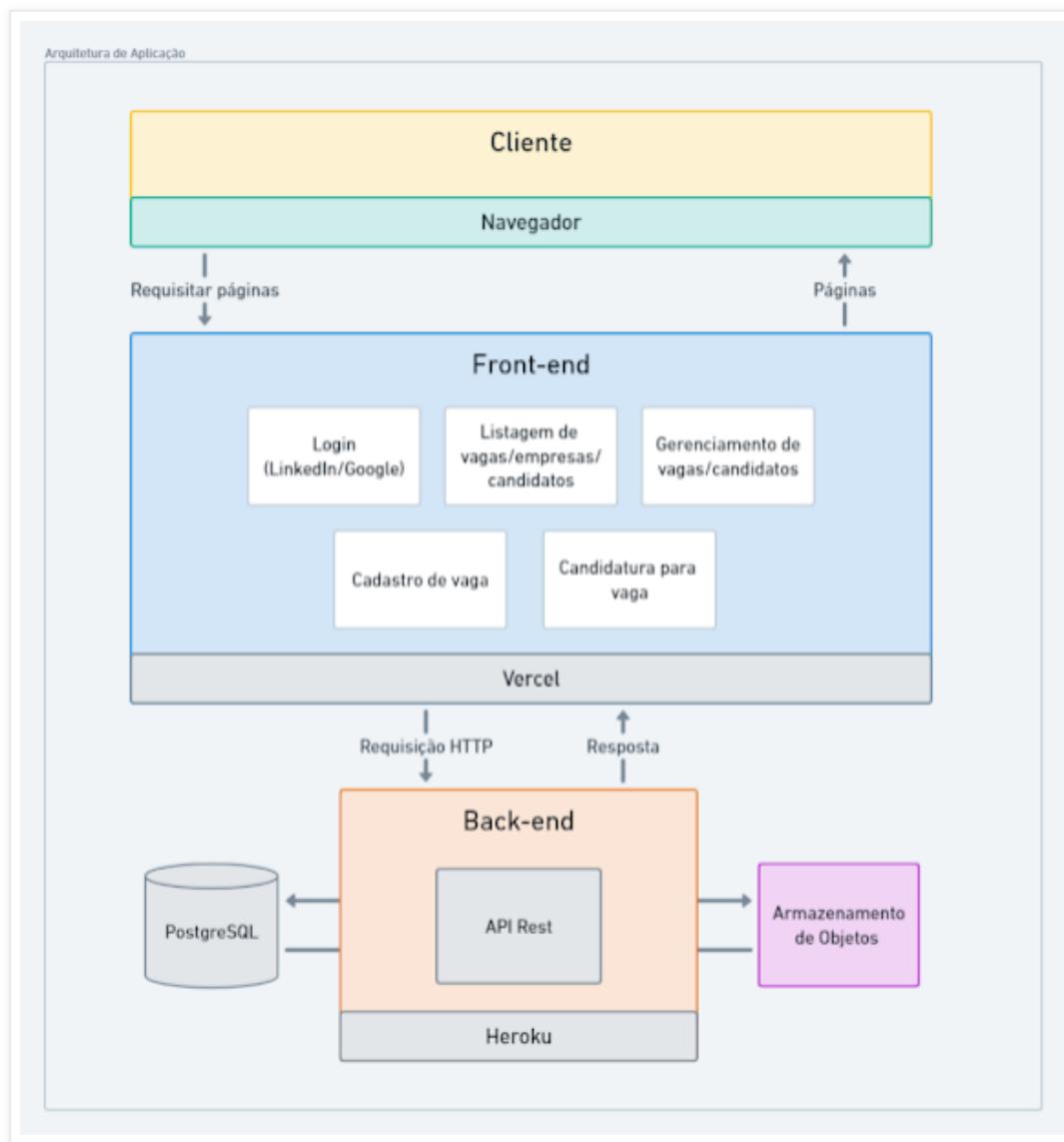
Os papéis dos membros da equipe

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

*Arquitetura geral da aplicação,
exibindo as partes do cliente, font-end e back-end e a comunicação entre elas*

Figura 53 – Blog: Semana 7 - 2



Fonte: Os Autores

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 <i>feedback</i> da empresa pós-intervista, 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

Figura 54 – Blog: Semana 7 - 3

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

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 mantido 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 <i>feedback</i> , ele deve ter pelo menos iniciado o processo seletivo	RF-008
RN-005	O <i>feedback</i> pode ser feito de forma anônima, mas o usuário deve estar logado e ter passado pelo processo seletivo	RF-008

Fonte: Os Autores

Foi a partir dos itens acima, primariamente, que decidimos as implementações que serão feitas na POC, então, essa semana iniciamos o desenvolvimento, primeiro criando contas da equipe na Vercel e no Heroku, além de iniciar as configurações das plataformas.

Por Bruna Pires - maio 03, 2022

Nenhum comentário:



Marcadores: Concepção, POC

Figura 55 – Blog: Semana 8 - 1

Semana 8 - 02/05 até 08/05/22

Apresentação do Desenho da Aplicação e Desenvolvimento da POC

Nesta semana finalizamos e realizamos a apresentação em sala de aula do Desenho da Aplicação desenvolvido ao longo das semanas anteriores. Abordamos tópicos como:

1. Objetivos da aplicação

2. Planejamento e metodologias

Responsabilidade	Bruna	Daniel	Igor	Leonardo	Lucas	Marcelo
Backend			X	X		X
Frontend	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					

3. Cronograma

Sprint	Data Inicial	Data Final	Descrição	Status
Desenho da aplicação 1	18/04/2022	25/04/2022	Elaboração da documentação do Desenho da Aplicação.	Concluída
Desenho da aplicação 2	25/04/2022	02/05/2022	Continuação da elaboração do Desenho da Aplicação. Planejamento para a POC.	Concluída
POC	02/05/2022	09/05/2022	Finalização do Desenho da Aplicação. Início do desenvolvimento dos itens da POC.	Em Progresso
POC 2	09/05/2022	16/05/2022	Continuação do desenvolvimento dos itens da POC.	Não Iniciada
MVP	16/05/2022	23/05/2022	Aproveitamento do que foi desenvolvido para a POC com melhorias e ampliação conforme possível para o MVP.	Não Iniciada
MVP 2	23/05/2022	30/05/2022	Continuação do trabalho no desenvolvimento do MVP.	Não Iniciada
Revisões: código e documentos	30/05/2022	06/06/2022	Finalização e revisão tanto do desenvolvimento quanto da documentação.	Não Iniciada
Preparação para a Apresentação	06/06/2022	13/06/2022	Organização e planejamento da apresentação do projeto e sua documentação.	Não Iniciada
Ajustes finais	13/06/2022	20/06/2022	Ajustes a serem feitos para correção e/ou melhoria do projeto apresentado.	Não Iniciada
Ajustes finais 2	20/06/2022	27/06/2022	Finalização dos ajustes finais para a entrega definitiva do projeto no semestre.	Não Iniciada

4. Arquitetura da aplicação

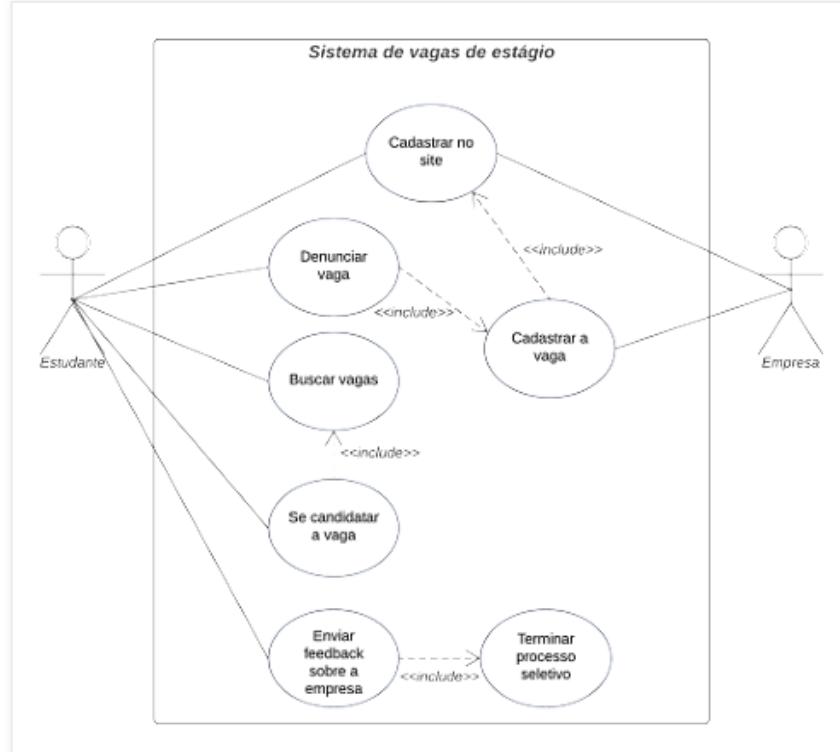
Conforme exemplificada no post da semana 7.

5. Requisitos e regras de negócio da Aplicação.

Figura 56 – Blog: Semana 8 - 2

6. Casos de uso

Apresentamos os casos de uso pertinentes a nossa aplicação conforme exemplo abaixo:



E por fim, apresentamos as funcionalidades que estarão presentes na POC e também no MVP, como exemplificado abaixo:

Funcionalidades da POC:

- Login via conta Google;
- Listagem das vagas;
- Demonstração da comunicação da camada de backend com a camada de frontend;
- Backend hospedado na Heroku;
- Frontend hospedado na Vercel/Netlify*;

Funcionalidades MVP:

- Incrementar o que já foi desenvolvido para a POC;
- Candidatura do estudante à uma vaga;
- Denúncia de vaga por não ser coerente com uma vaga de estágio de fato;
- Login via LinkedIn para os estudantes;
- Recomendações de vagas para os estudantes e de estudantes para as empresas;
- Opção de contato com o estudante via Whatsapp e
- Testes unitários.

Nessa semana também concluímos a preparação do ambiente de desenvolvimento e já começamos a desenvolver as funcionalidades da POC, que será apresentada na próxima semana. Iniciamos com a integração para login via google e informação do usuário logado.

Por Daniel Roberto - maio 09, 2022 Nenhum comentário:



Marcadores: Concepção, POC

Figura 57 – Blog: Semana 9 - 1

Semana 9 - 09/05 até 15/05/22

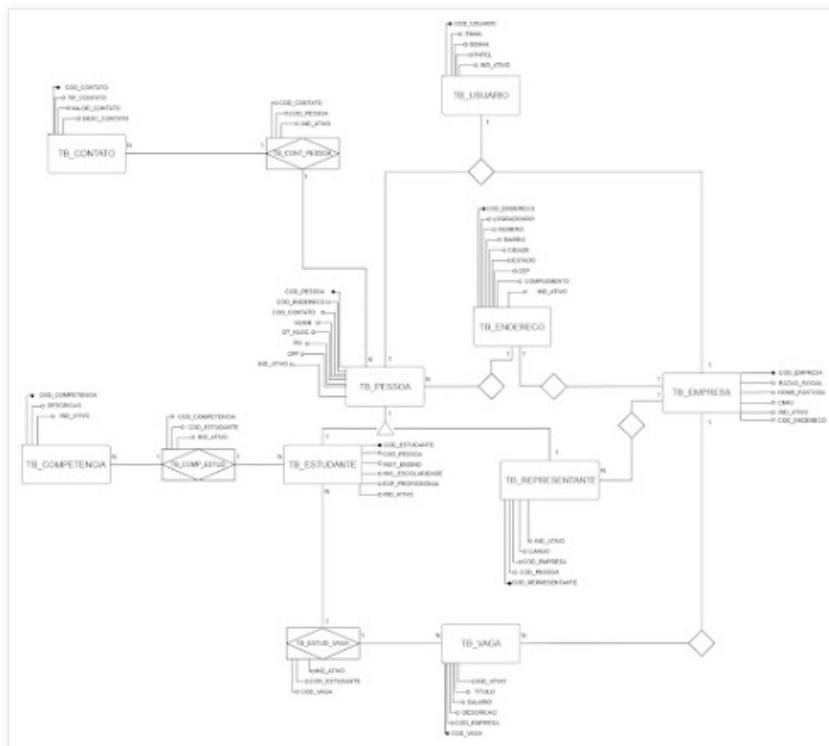
Preparação para POC

Nesta semana realizamos a preparação da apresentação POC, foi dado inicio no desenvolvimento das funcionalidades de login e listagem de vagas, para isso foi iniciado o desenvolvimento da diagramação do banco de dados com o dicionário de dados e a elaboração do MER, bem como o desenvolvimento do primeiro endpoint no back-end, a página de login e a página inicial com a listagem das vagas, bem como a integração do back-end com o front-end por meio de serviços Rest.

Dicionário de dados:

tb_usuario			tb_pessoas			tb_vaga		
Campo	Tipo	Restrição	Campo	Tipo	Restrição	Campo	Tipo	Restrição
Cod_usuario	SERIAL	PK	Cod_Pessoa	SERIAL	PK	Cod_Vaga	SERIAL	PK
Senha	VARCHAR(50)		Cod_Endereco	SERIAL	FK, NN	Cod_Empresa	SERIAL	FK NN
Papel	VARCHAR(25)	DEFAULT	Tip_Contato	VARCHAR(10)		Descricao	TEXT	
Email	VARCHAR(50)		Valor_Contato	VARCHAR(25)		Salario	FLOAT(5)	
Ind_Ativo	BOOLEAN	DEFAULT	Nome	VARCHAR(50)	NN	Titulo	VARCHAR(30)	
			Dt_Nasc	DATE	NN	Ind_Ativo	BOOLEAN	DEFAULT
			RG	VARCHAR(11)	NN			
			CPF	VARCHAR(13)	NN, UQ			
			Ind_Ativo	BOOLEAN	DEFAULT			
tb_estudante			tb_competet_estud			tb_estud_vaga		
Campo	Tipo	Restrição	Campo	Tipo	Restrição	Campo	Tipo	Restrição
Cod_Estudante	SERIAL	PK	Cod_Estudante	SERIAL	FK	Cod_Vaga	SERIAL	FK
Cod_Pessoa	SERIAL	FK, NN, UQ	Cod_Competencia	SERIAL	FK	Cod_Estudante	SERIAL	FK
Inst_Escolaridade	CHAR(2)	NN	Ind_Ativo	BOOLEAN	DEFAULT	Status_Candidatura	CHAR(1)	
Nvl_Escolaridade	VARCHAR(128)	NN						
Exp_Profissional	TEXT							
Ind_Ativo	BOOLEAN	DEFAULT						
tb_representante_rh			tb_competencia			tb_contato		
Campo	Tipo	Restrição	Campo	Tipo	Restrição	Campo	Tipo	Restrição
Cod_Representante	SERIAL	PK	Cod_Competencia	SERIAL	PK	Cod_Endereco	SERIAL	PK
Cod_Pessoa	SERIAL	FK, NN	Descricao	VARCHAR(50)	NN, UQ	Tip_Contato	VARCHAR(25)	NN
Cod_Empresa	SERIAL	FK	Ind_Ativo	BOOLEAN	DEFAULT	Valor_Contato	VARCHAR(50)	NN
Cargo	VARCHAR(50)	NN				Descr_Contato	VARCHAR(50)	DEFAULT
Ind_Ativo	BOOLEAN	DEFAULT						
tb_empresa			tb_endereco			tb_contato_pessoa		
Campo	Tipo	Restrição	Campo	Tipo	Restrição	Campo	Tipo	Restrição
Cod_Empresa	SERIAL	PK	Cod_Endereco	SERIAL	PK	Cod_Pessoa	SERIAL	FK
Razao_Social	VARCHAR(50)	NN	Logradoiro	VARCHAR(50)	NN	Cod_Endereco	SERIAL	FK
Nome_Fantasia	VARCHAR(50)		Numero	INTEGER(6)	NN	Ind_Ativo	BOOLEAN	DEFAULT
CNPJ	VARCHAR(20)	NN, UQ	Bairro	VARCHAR(50)	NN			
Cod_Endereco	SERIAL	FK, NN	Cidade	VARCHAR(50)	NN			
Ind_Ativo	BOOLEAN	DEFAULT	Estado	CHAR(2)	NN			
			CEP	INTEGER(10)	NN			
			Complemento	VARCHAR(50)				
			Ind_Ativo	BOOLEAN	DEFAULT			
LEGENDA								
SIGLA	DESCRICAÇÃO							
PK	Primary Key							
FK	Foreign Key							
NN	Not Null							
UQ	Unique							
CK	Check							
DEFAULT	Default							

Figura 58 – Blog: Semana 9 - 2

MER

Por Igor Nathan - maio 22, 2022 Nenhum comentário:



Marcadores: POC

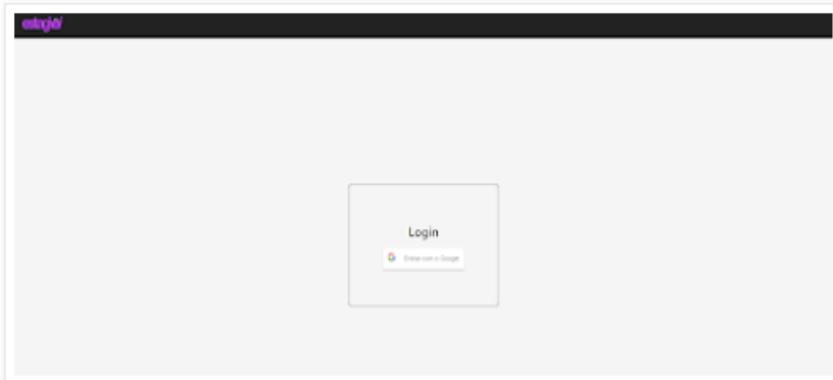
Figura 59 – Blog: Semana 10

Semana 10 - 16/05 até 22/05/22

Desenvolvimento Inicial do MVP

Nesta semana começamos a desenvolver as funcionalidades necessárias para o MVP. Abaixo estão as telas na versão inicial do nosso projeto.

Tela de Login



Listagem de vagas

Informações de cadastro

We

Nome: We Code
Email: wecode@wecode.com.br

Vagas

Desenvolvedor Front End
Vaga disponível para atuar no desenvolvimento front end do nosso sistema feito em código de origem!
R\$ 1.000,00

Desenvolvedor Back End Java
Aqui você irá aprender Java com Spring Boot e Oracle PL/SQL, entre outras novas tecnologias!
R\$ 1.000,00

Monitoração de competências
Se você é dedicado e gosta de aprender, essa é sua sua chance!
R\$ 1.000,00

Por Leonardo Marques - maio 22, 2022 Nenhum comentário:

Marcadores: MVP

Figura 60 – Blog: Semana 11

Semana 11 - 23/05 até 29/05/22

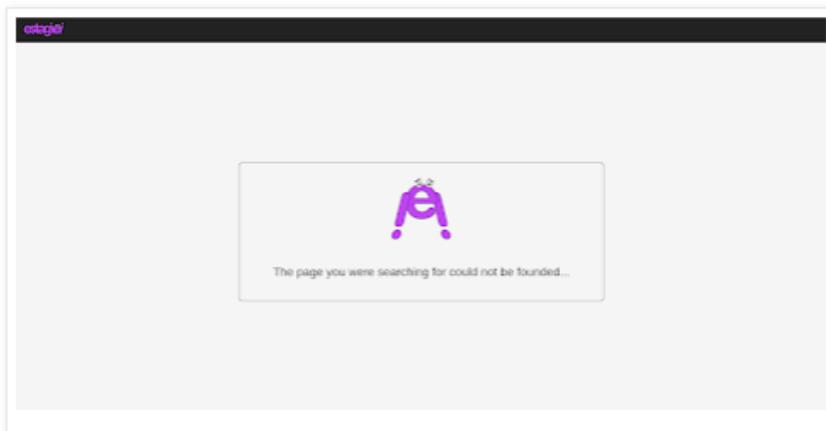
Desenvolvimento MVP Funcionalidades e melhorias no sistema

Nesta semana, foi discutido novamente a modelagem do banco de dados, que por sinal, é a parte do sistema que mais sofre alteração na minha opinião, a cada nova implementação de funcionalidade do sistema, uma nova normalização é feita no banco.

Tenho certeza que a modelagem do nosso banco vai estar uma coisa totalmente diferente do que inicialmente imaginávamos.

Foi incrementado o Layout da página de NotFound da aplicação.

Tela de NotFound



Também demos início no desenvolvimento das outras páginas, que deverão estar com um Layout melhor implementado até o MVP.

Bem como, o planejamento das 'sprints' para as entregas das funcionalidades que deverão ser apresentadas a banca, e o incremento da documentação para a entrega inicial.

Também será utilizado o Swagger para a criação da documentação da API do sistema.

Acredito que a funcionalidade mais difícil de ser implementada no momento para a entrega do MVP ao nível de Backend, seja a recomendação das vagas, cujo objetivo inicial para a entrega, é apresentar uma recomendação baseada em match de 'skills' do candidato com o que a vaga pede.

Mas, creio que conseguiremos implementar tudo, e estar prontos para a entrega, assim como, para a apresentação a banca.

Por Lucas Lima - maio 30, 2022 Nenhum comentário:



Marcadores: MVP

Figura 61 – Blog: Semana 12

Semana 12 - 30/05 até 06/06/22

Continuando o MVP e ajustando a documentação

Para esta semana, revisamos alguns itens pertinentes que a nossa aplicação deveria conter, sobretudo no back-end, como a obtenção das maiores notas no `securityheaders` e `ssllabs`. Também adicionamos a biblioteca que possibilita a documentação da nossa API no back-end através do padrão OpenAPI/Swagger.

Fomos alertados que a documentação final precisará ser entregue com uma semana de antecedência ao dia da apresentação do MVP, portanto também começamos a adiantar algumas questões finais na documentação do nosso projeto, de modo a conseguir entregar tudo a tempo.

Para essa semana, daremos continuidade para o desenvolvimento das funcionalidades da aplicação e finalizaremos a documentação final e os slides para a apresentação.

Por Marcelo Junior - junho 06, 2022 Nenhum comentário:



Marcadores: MVP

APÊNDICE B – Desenho da Aplicação

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

Figura 1 – Roteiro Geral	10
Figura 2 – Roteiro Geral - Detalhe Inicial	10
Figura 3 – Roteiro Geral - Detalhe Final	10
Figura 4 – Ciclo de vida: página web tradicional X SPA	13
Figura 5 – Arquitetura de Aplicação	17
Figura 6 – Arquitetura Tecnológica	18
Figura 7 – Arquitetura de Negócios	18
Figura 8 – Caso de Uso 1 - Funcionalidades do estudante	22
Figura 9 – Caso de Uso 2 - Funcionalidades da empresa	22
Figura 10 – Caso de Uso 3 - Funcionalidades do administrador	23

Lista de quadros

Quadro 1 – Comparação dos aplicativos concorrentes	8
Quadro 2 – Divisão de responsabilidades da equipe.	9
Quadro 3 – Cronograma de Sprints	11
Quadro 4 – Requisitos funcionais	19
Quadro 5 – Requisitos não funcionais	20
Quadro 6 – Regras de negócio	20
Quadro 7 – Histórias de usuário	21

Lista de abreviaturas e siglas

API	<i>Application Programming Interface</i> - 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	<i>Cascading Style Sheets</i> - Folhas de Estilo em Cascata - Citado em 13
HTML	<i>Hypertext Markup Language</i> - Linguagem de Marcação de Hipertexto - Citado em 13 , 25
HTTP	<i>Hypertext Transfer Protocol</i> - Protocolo de transferência de hipertexto - Citado em 14 , 15 , 16
HTTPS	<i>Hypertext Transfer Protocol Secure</i> - Protocolo seguro de transferência de hypertexto - Citado em 15 , 29
JSON	<i>JavaScript Object Notation</i> - 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	<i>Minimum Viable Product</i> - Produto Mínimo Viável - Citado em 11 , 23
POC	<i>Prove of Concept</i> - Prova de Conceito - Citado em 11 , 23
REST	<i>Representational State Transfer</i> - Transferência de Estado Representacional - Citado em 14 , 15 , 33
RH	Recursos Humanos - Citado em 23
SPA	<i>Single Page Application</i> - Aplicação de Página Única - Citado em 2 , 13 , 16
SSD	<i>Solid-State Drive</i> - Unidade de Estado Sólido - Citado em 30
SSO	<i>Single Sign-On</i> - Login único - Citado em 23 , 24
URL	<i>Universal Resource Locator</i> - Localizador universal de recurso - Citado em 33
USD	<i>United States Dollar</i> - Dólares Americanos - Citado em 30 , 31

Sumário

1	INTRODUÇÃO	7
1.1	Justificativa	7
1.2	Proposta de solução	7
1.3	Objetivos	7
1.4	Análise de Concorrentes	8
2	PLANEJAMENTO E GERENCIAMENTO DO PROJETO	9
2.1	Gestão e Desenvolvimento do Projeto	9
2.2	Organização da equipe	9
2.3	Cronograma	10
3	REVISÃO DA LITERATURA	12
3.1	Estágio	12
3.1.1	Definição	12
3.1.2	Tipos de estágio	12
3.1.3	Carga horária	12
3.2	Single-page Application (SPA)	13
3.3	Application Programming Interface (API)	14
3.3.1	API REST	14
4	DESENVOLVIMENTO DA APLICAÇÃO	16
4.1	Arquitetura	16
4.1.1	Diagramas de arquitetura	16
4.2	Escopo	19
4.2.1	Requisitos	19
4.2.1.1	Requisitos Funcionais	19
4.2.1.2	Requisitos Não-funcionais	19
4.2.1.3	Regras de Negócio	20
4.2.2	Histórias de usuário	20
4.2.3	Casos de uso	21
4.2.4	Fases de entrega	23
4.2.4.1	Prova de Conceito (POC)	23
4.2.4.2	Produto Mínimo Viável (MVP)	23
4.2.4.3	Entrega Final	24
4.3	Interações	24
4.3.1	Login com o Google e LinkedIn	24

4.3.2	Entrar em contato via <i>Whatsapp</i>	24
4.3.3	Acessibilidade com VLibras	24
4.4	Manutenibilidade	25
4.4.1	Logs	25
4.4.2	Code Convention	25
4.4.3	Design Patterns	26
4.4.3.1	Clean Code	26
4.4.3.2	SOLID	27
4.4.3.3	12 Factor App	27
4.4.4	Integração continua	28
4.4.5	Testes	28
4.5	Segurança, Privacidade e Legislação	28
4.6	Viabilidade Financeira	29
4.6.1	Gerenciamento de custos	29
4.6.1.1	Desenvolvimento	30
4.6.2	Ambiente de produção	30
4.6.2.1	Frontend	30
4.6.2.2	Backend	30
4.6.2.3	Banco de dados	30
4.6.3	Monetização	31
4.6.4	Conclusão	31
	REFERÊNCIAS	32
	GLOSSÁRIO	32

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, *estagieei* 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, tanto os estudantes quanto as empresas têm papel ativo no processo de encontrar um(a) estudante/vaga ideal, cujas 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.

Capítulo 1. Introdução

8

- 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 <i>WhatsApp</i> .				x
Denúncias de vagas incoerentes com a realidade.				x
<i>Feedback</i> 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.

Capítulo 2. Planejamento e Gerenciamento do Projeto

10

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

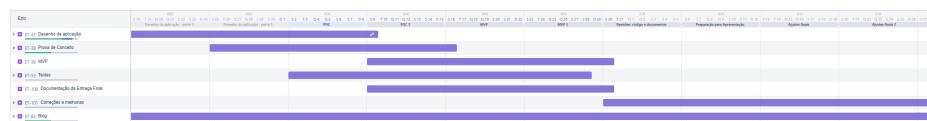
Fonte: Roteiro via ferramenta [Jira Software](#)

Figura 2 – Roteiro Geral - Detalhe Inicial

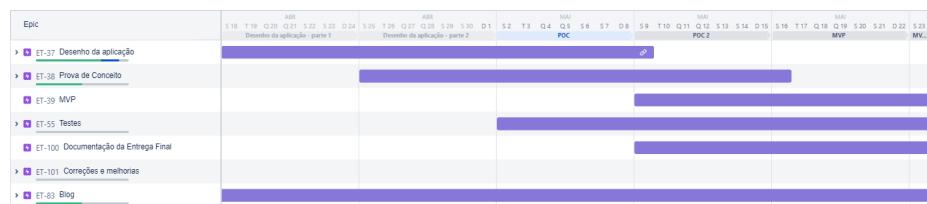
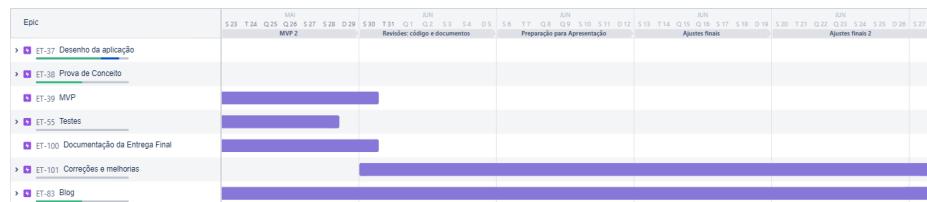
Fonte: Roteiro via ferramenta [Jira Software](#)

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 <i>Prove of Concept</i> (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 <i>Minimum Viable Product</i> (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 *estagieei*, 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.

§ 1º 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.

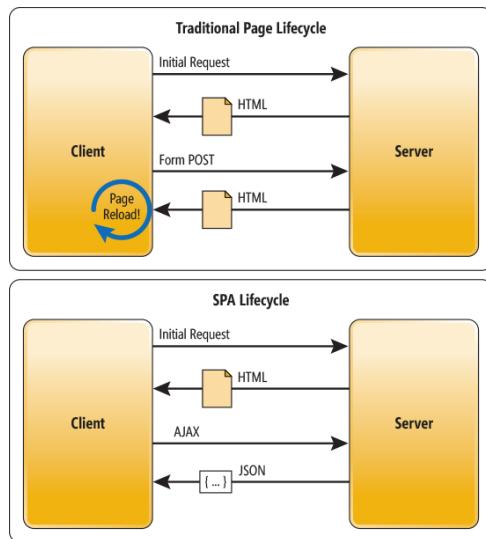
§ 2º Se a instituição de ensino adotar verificações de aprendizagem periódicas ou finais, nos períodos de avaliação, a carga horária do estágio será reduzida pelo menos à metade, segundo estipulado no termo de compromisso, para garantir o bom desempenho do estudante.(BRASIL, 2008)

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平衡amento 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:

Capítulo 3. Revisão da Literatura

15

-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.

-Hipermédia 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.

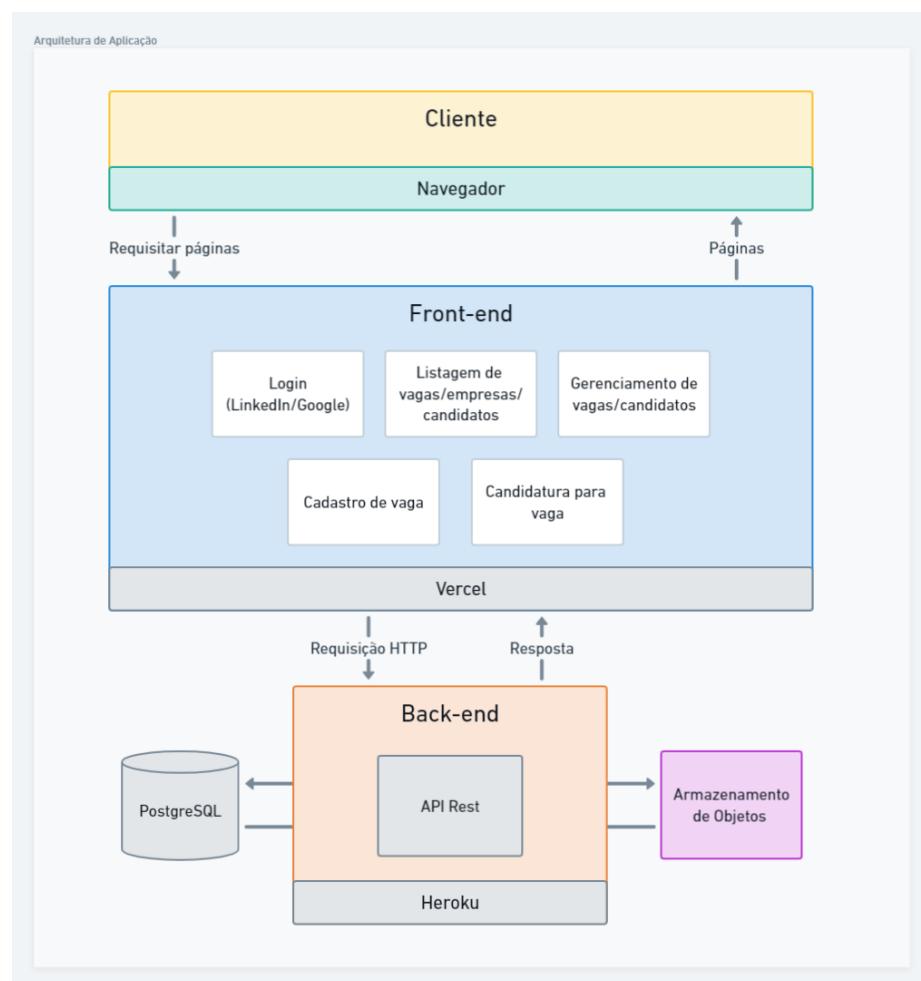
4.1.1 Diagramas de arquitetura

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.

Capítulo 4. Desenvolvimento da Aplicação

17

Figura 5 – Arquitetura de Aplicação

Fonte: Produzido pelos autores utilizando a ferramenta *Whimsical*

Capítulo 4. Desenvolvimento da Aplicação

18

Figura 6 – Arquitetura Tecnológica

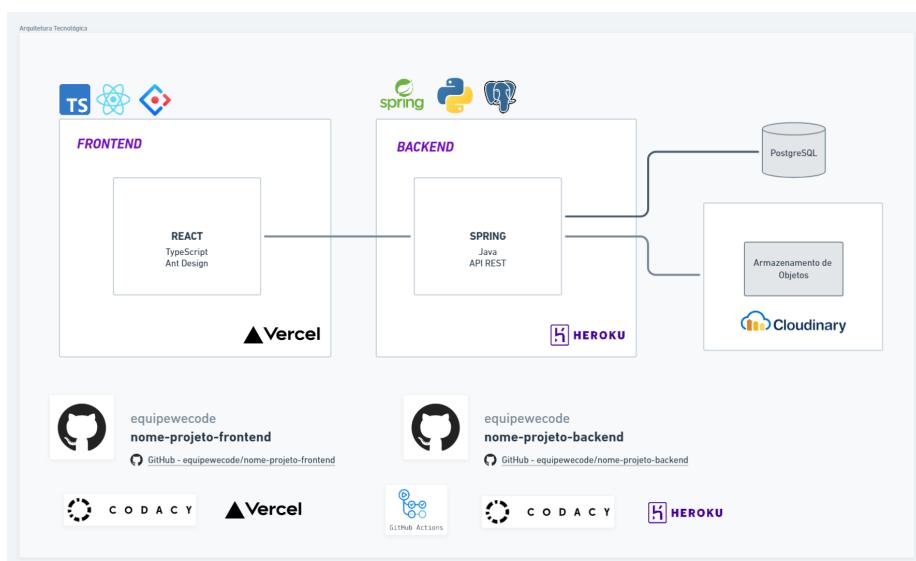
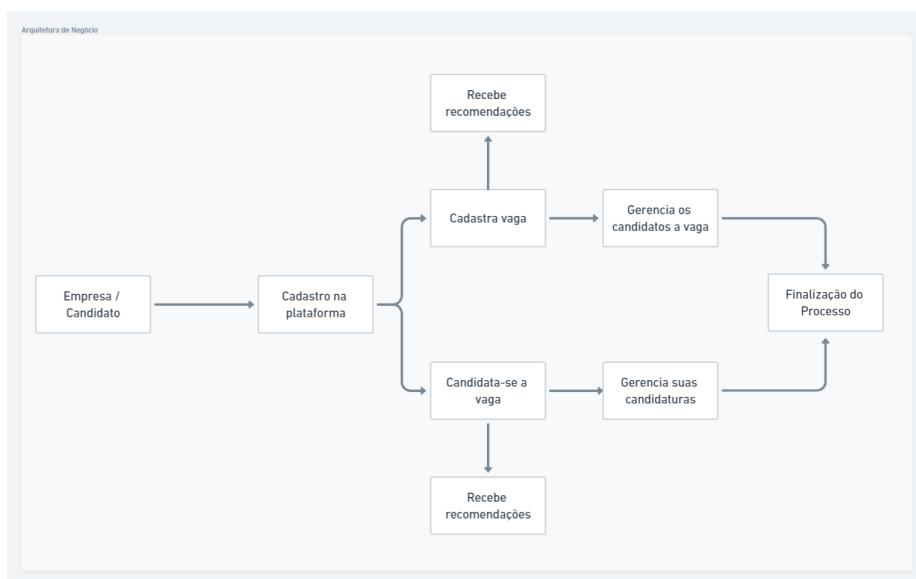
Fonte: Produzido pelos autores utilizando a ferramenta *Whimsical*

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 <i>feedback</i> 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 responsável 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 negócios

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 <i>feedback</i> , ele deve ter pelo menos iniciado o processo seletivo	RF-008
RN-005	O <i>feedback</i> 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

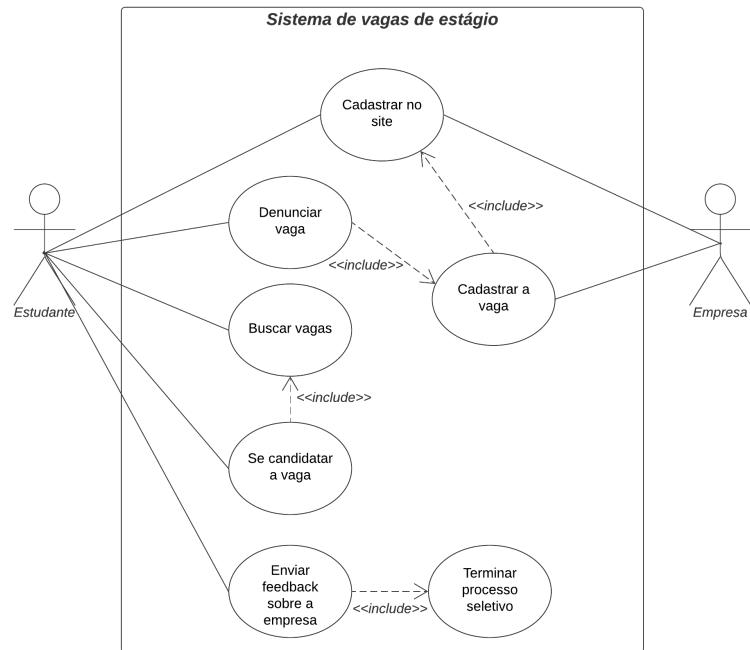
4.2.3 Casos de uso

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

Capítulo 4. Desenvolvimento da Aplicação

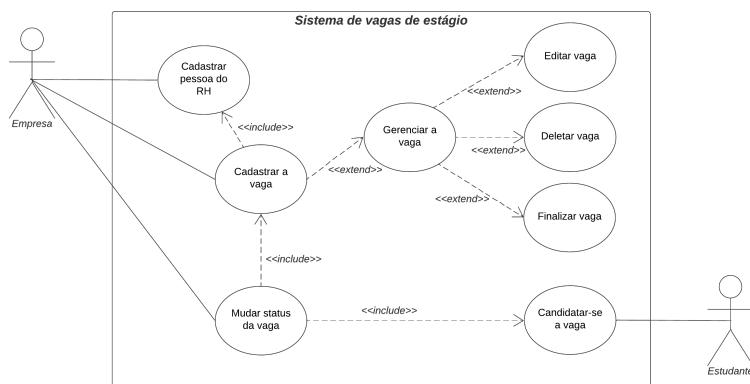
22

Figura 8 – Caso de Uso 1 - Funcionalidades do estudante



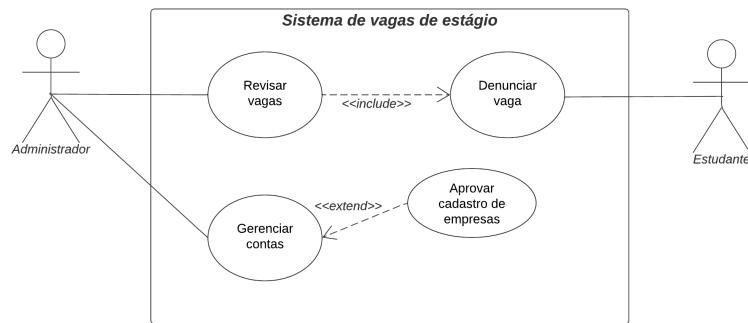
Fonte: Os autores

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 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 Responsibility 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 papéis;
- 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

A fim 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.

33

Glossário

API RESTful	API que segue todas as restrições da arquitetura REST. - Citado em 14, 15
AWS	Amazon Web Services - Plataforma em nuvem <i>on-demand</i> 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 <i>Universal Resource Locator (URL)</i> , 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

Glossário

34

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 <i>host</i> de páginas web. - Citado em 16, 30
VLibras	Conjunto de ferramentas para a tradução de texto em Português para LIBRAS gratuitas e de código aberto, mais informações disponíveis no endereço < https://www.gov.br/governodigital/pt-br/vlibras >. - Citado em 6, 24, 25

APÊNDICE C – POC Overview

Proof Of Concept (POC) Overview (Visão Geral da Prova de Conceito)

Neste documento buscamos relatar de modo mais direto os itens compondo a POC apresentada em aula no dia 16/05/2022.

A equipe se propôs a demonstrar a hospedagem da aplicação seguindo a arquitetura planejada e integrando com as tecnologias e ferramentas escolhidas, possibilitar o Login via uma conta Google e recuperar uma listagem de vagas do banco de dados.

PLANEJAMENTO:

Ferramenta de tarefas: Jira Software;

Metodologia: SCRUM;

Controle de versão: Git com GitHub, sendo os códigos e documentos alocados em uma organização no GitHub;

Comunicação: Commits seguindo um padrão de prefixo que identifica o que foi feito, que tipo de modificações foram realizadas .

BANCO DE DADOS

Tipo: Relacional;

Codificação: SQL;

Hospedagem: Heroku Postgres.

BACKEND

Codificação: Java com framework Spring;

Hospedagem: Heroku.

FRONTEND

Codificação: TypeScript com o framework React;

Hospedagem: Inicialmente na Vercel, posteriormente migrado para a Netlify a fim de podermos prosseguir com o desenvolvimento em uma conta gratuita com uma organização no GitHub.

INTEGRAÇÃO:

Conexão dos repositórios do GitHub com as plataformas de hospedagem, permitindo assim integração contínua, pois a cada atualização no repositório, é feito um deploy nas correspondentes plataformas.

Foram realizadas as configurações de ambientes específicas de cada camada, posteriormente, quando já possuímos clareza o suficiente sobre o projeto, o MER foi desenhado e as tabelas elaboradas.

Para a POC apenas as tabelas essenciais foram adicionadas, assim como apenas os endpoints mínimos foram construídos e mapeados.

Anexos

ANEXO A – Nota dos headers

Security Headers
Sponsored by 

Scan your site now

Scan

Hide results Follow redirects

Security Report Summary

	Site: https://estagiei.herokuapp.com/api/vaga IP Address: 23.22.130.173 Report Time: 06 Jun 2022 16:15:13 UTC Headers: 
---	--

Supported By

Probely Wow, amazing grade! Perform a deeper security analysis of your website and APIs: **Try Now**

Raw Headers

HTTP/1.1	200
Server	Cowboy
Connection	keep-alive
Accept	application/json
Strict-Transport-Security	max-age=63072000; includeSubDomains; preload
Content-Security-Policy	default-src 'self' https://estagiei.herokuapp.com
X-Frame-Options	DENY
X-Content-Type-Options	nosniff
Referrer-Policy	same-origin
Permissions-Policy	microphone=none; geolocation=none; camera=none
Vary	Origin
Vary	Access-Control-Request-Method
Vary	Access-Control-Request-Headers
Content-Type	application/json
Transfer-Encoding	chunked
Date	Mon, 06 Jun 2022 16:15:13 GMT
Via	1.1 vegur

Upcoming Headers

Expect-CT	Expect-CT allows a site to determine if they are ready for the upcoming Chrome requirements and/or enforce their CT policy.
Cross-Origin-Embedder-Policy	Cross-Origin Embedder Policy allows a site to prevent assets being loaded that do not grant permission to load them via CORS or CORP.
Cross-Origin-Opener-Policy	Cross-Origin Opener Policy allows a site to opt-in to Cross-Origin Isolation in the browser.
Cross-Origin-Resource-Policy	Cross-Origin Resource Policy allows a resource owner to specify who can load the resource.

Additional Information

Server	Server value has been changed. Typically you will see values like "Microsoft-IIS/8.0" or "nginx 1.7.2".
Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS.
Content-Security-Policy	Content-Security-Policy is an effective measure to protect your site from XSS attacks. By whitelisting sources of approved content, you can prevent the browser from loading malicious assets. Analyse this policy in more detail. You can sign up for a free account on Report URI to collect reports about problems on your site.
X-Frame-Options	X-Frame-Options tells the browser whether you want to allow your site to be framed or not. By preventing a browser from framing your site you can defend against attacks like clickjacking.
X-Content-Type-Options	X-Content-Type-Options stops a browser from trying to MIME-sniff the content type and forces it to stick with the declared content-type. The only valid value for this header is "X-Content-Type-Options: nosniff".
Referrer-Policy	Referrer Policy is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.
Permissions-Policy	Permissions Policy is a new header that allows a site to control which features and APIs can be used in the browser.

A [scottelme.co.uk](#) project - [CC-BY-SA 4.0](#) Sponsored by 

