







Histórico de Revisões

| **Data** | **Versão** | **Descrição** | **Autor** |
| --- | --- | --- | --- |
| 01/12/2024 | 1.0 | Complementação e esclarecimento de todas as seções do guia. | Rhamza Mourad Mourad |
|  |  |  |  |
|  |  |  |  |

SUMÁRIO

[1. Introdução 4](#_30j0zll)

[1.1. Referências 4](#_1fob9te)

[2. Planejamento de Implantação 4](#_3znysh7)

[2.1. Responsabilidades 4](#_2et92p0)

[2.2. Cronograma 4](#_17dp8vu)

[3. Recursos NECESSÁRIOS PARA IMPLANTAR O PRODUTO 4](#_3rdcrjn)

[3.1. Ambiente 4](#_tyjcwt)

[3.2. Hardware 4](#_3dy6vkm)

[3.3. Software 4](#_3dy6vkm)

[3.4. Pessoas 5](#_1t3h5sf)

[4. ROTEIRO DE implantação 5](#_4d34og8)

[5. Treinamento e suporte 5](#_2s8eyo1)

# Introdução

Este documento descreve processos necessários para instalar dependências, configurar ambientes, hospedar a aplicação na infraestrutura em nuvem da AWS e colocar a aplicação em operação, com o objetivo de garantir o funcionamento esperado do software. Além disso, cita referências necessárias para execução dos procedimentos descritos.

Em primeira instância, vale destacar que a aplicação é distribuída em duas vias, uma aplicação web gráfica acessada pelo usuário, referida como *frontend*, e uma *API RESTFul*, isto é, aplicação que retorna arquivos *JSON* como resposta a pedidos feitos via *frontend*, a qual se atribui o nome *backend*.

## 1.2. Definições:

Para total compreensão deste documento, define-se os termos, acrônimos e abreviações utilizados no decorrer do mesmo:

* Java é uma linguagem de programação orientada a objetos;
* Spring é um framework para desenvolvimento de aplicações em Java;
* NPM (“Node Package Manager”) é o gerenciador de pacotes do Node.js, utilizado para instalar e gerenciar dependências de aplicações baseadas em JavaScript.
* Node.js é um ambiente de execução JavaScript server-side;
* Um bucket é um contêiner de armazenamento na nuvem, usado para armazenar objetos, como arquivos e metadados. Na AWS, buckets são gerenciados pelo Amazon S3.
* Subnet (“sub-rede”) é uma subdivisão de uma rede maior dentro de uma VPC (Virtual Private Cloud).
* Amazon RDS (“Relational Database Service”) é um serviço da AWS que facilita a configuração, operação e escalabilidade de bancos de dados relacionais na nuvem.
* O Elastic Load Balancer é um serviço da AWS que distribui automaticamente o tráfego de aplicações entre várias instâncias;
* O Gateway NAT (“Network Address Translation”) é usado em redes privadas para permitir que instâncias na sub-rede privada acessem a Internet sem expor seus IPs.
* Amazon SNS (“Simple Notification Service”) é um serviço de notificação que permite o envio de mensagens entre aplicações ou diretamente a usuários via SMS, e-mail ou outros protocolos.
* VPC (“Virtual Private Cloud”) é uma rede virtual isolada dentro da infraestrutura da AWS, usada para hospedar e gerenciar recursos de maneira segura.
* Amazon EC2 (“Elastic Compute Cloud”) fornece instâncias de máquinas virtuais para executar aplicações na nuvem. As instâncias são escaláveis e configuráveis.
* Autoscaling é uma funcionalidade da AWS que ajusta automaticamente a quantidade de instâncias EC2 em resposta à demanda da aplicação;
* Uma branch (“ramo”) é uma versão paralela de um repositório Git. É usada para desenvolver funcionalidades ou corrigir problemas sem interferir no código principal.
* Um script é um conjunto de instruções executáveis que automatizam tarefas ou processos em uma aplicação ou sistema.
* SGBD (“Sistema de Gerenciamento de Banco de Dados”) é um software que permite criar, gerenciar e manipular bancos de dados.
* Lamda

## Referências

**ANGULAR.** *Overview: Angular*. Disponível em:<https://angular.dev/overview>. Acesso em: 29 nov. 2024.

**ORACLE.** *Java Documentation*. Disponível em:<https://docs.oracle.com/en/java/>. Acesso em: 29 nov. 2024.

**SPRING.** *Spring Boot Documentation*. Disponível em:<https://docs.spring.io/spring-boot/documentation.html>. Acesso em: 29 nov. 2024.

**AMAZON WEB SERVICES.** *Amazon EC2 Documentation*. Disponível em:<https://docs.aws.amazon.com/ec2/?nc2=h_ql_doc_ec2>. Acesso em: 29 nov. 2024.

**AMAZON WEB SERVICES.** *Amazon RDS Documentation*. Disponível em:<https://docs.aws.amazon.com/rds/?icmpid=docs_homepage_featuredsvcs>. Acesso em: 29 nov. 2024.

**AMAZON WEB SERVICES.** *AWS Lambda Documentation*. Disponível em:<https://docs.aws.amazon.com/lambda/?icmpid=docs_homepage_featuredsvcs>. Acesso em: 29 nov. 2024.

**AMAZON WEB SERVICES.** *Amazon S3 Documentation*. Disponível em:<https://docs.aws.amazon.com/s3/?icmpid=docs_homepage_featuredsvcs>. Acesso em: 29 nov. 2024.

**AMAZON WEB SERVICES.** *Amazon VPC Documentation*. Disponível em:<https://docs.aws.amazon.com/vpc/?icmpid=docs_homepage_featuredsvcs>. Acesso em: 29 nov. 2024.

**POSTGRESQL.** *Documentation: PostgreSQL*. Disponível em:<https://www.postgresql.org/docs/current/index.html>. Acesso em: 29 nov. 2024.

# Planejamento de Implantação

Para implantar o projeto, é necessário configurar previamente o ambiente de implantação de acordo com o que é estabelecido neste documento, isto inclui, mas não se limita a: a instalação de NPM (versão 9.2.0 ou superior), Node.js (versão 18.19.1 ou superior), Maven, JDK 21, PostgreSQL 17, etc. Todas as bibliotecas de terceiros utilizadas no projeto devem ser instaladas por meio do arquivo package-lock.json, garantindo que as dependências estejam na versão correta.

## Responsabilidades

É de responsabilidade da equipe de desenvolvimento configurar adequadamente o ambiente de implantação, garantindo que todos os pré-requisitos técnicos, como versões específicas de softwares e dependências, estejam corretamente instalados. Isso assegura que a aplicação possa operar sem interrupções ou falhas.

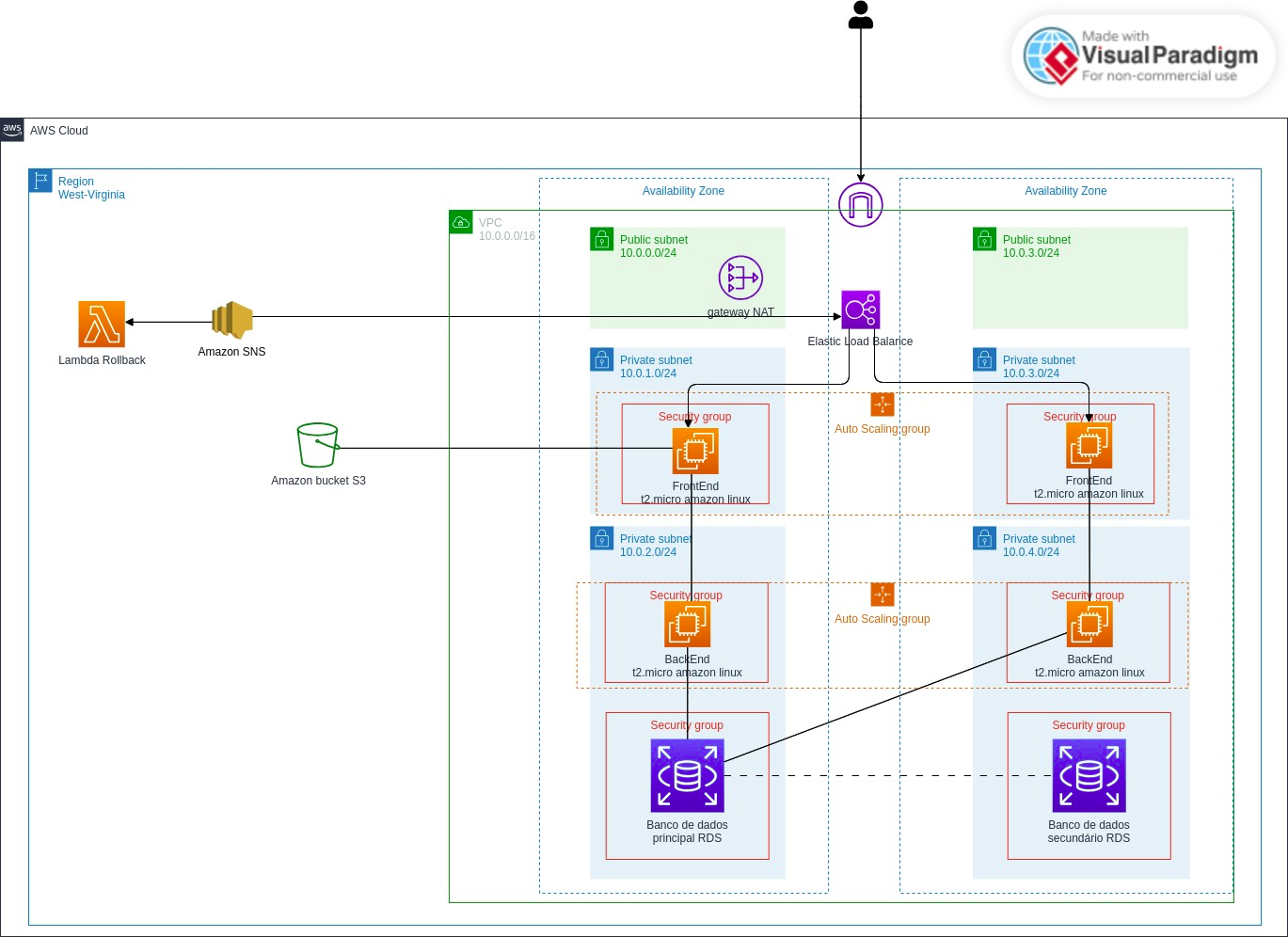
A equipe de desenvolvimento também deve seguir rigorosamente o plano de implantação definido, incluindo a configuração da infraestrutura na *AWS*, como a *VPC*, o *Elastic Load Balancer* e as *subnets*. Dessa forma, a escalabilidade e a resiliência do sistema serão garantidas.

Por fim, cabe ao cliente é responsável por fornecer feedback durante os testes de aceitação, apontando possíveis discrepâncias e validando se a aplicação atende aos requisitos do negócio, para que eventuais ajustes sejam realizados antes da entrada em produção.

# Recursos NECESSÁRIOS PARA IMPLANTAR O PRODUTO

## Ambiente

Quaisquer responsabilidades de hardware são geridas pela própria Amazon, provedora da nuvem. A infraestrutura virtual necessária dentro dos serviços dela pode ser descrita pelo diagrama:



É necessário, no ambiente virtual, uma região, que abrigará um *SNS* conectado a um *lambda* e um *bucket*, além de uma VPC.

A VPC será composta de:

* Uma entrada de internet gateway pelo qual os clientes acessarão via internet;
* Um *Elastic* *Load Balance*, que gerenciara a carga entre os servidores conforme a demanda;
* Duas *subnets* publicas com uma tendo um gateway NAT e 4 subredes privadas, divididas igualmente em duas zonas de disponibilidade diferente;
* Dentro da primeira *subnet* privada da primeira zona de disponibilidade (principal) há uma instância do *EC2* que armazena o front, isolada em um grupo de segurança, mas dividindo o grupo de *autoscaling* com outra instancia duplicada. Ela se conecta ao *bucket s3* na região e ao *EC2* do *backend* em outra subrede privada. O *EC2* está sozinho no seu próprio grupo de segurança, mas compartilha a subrede com um banco de dados *RDS* e um grupo de *autoscaling* com outro *EC2*. Ambas instâncias de *backend* estão conectadas com o *RDS*, e uma cópia deste é armazenado uma instância numa outra zona de disponibilidade, considerando a possibilidade de queda do serviço.

## Software

As duas vias da aplicação são armazenadas em repositórios diferentes, referenciados por um repositório pai que organiza outros documentos pertinentes ao projeto e adequando-se à formatação do Spin-Off.

Para executar o *frontend*, isto é, o conjunto de arquivos que implementam a interface *web*, acessada pelo usuário final, é necessário:

* *NPM*, para instalar as dependências. Versão 9.2.0 ou superior.
* *Node.JS*, para executar o *NPM*. Versão 18.19.1 ou superior.
* Todas as bibliotecas de terceiros usadas na criação do projeto, são instaladas através do *package-lock.json* usando o *NPM*. Este arquivo já possui cada dependência e a versão dessa dependência.

A lógica de negócio é feita através do *backend*, uma aplicação que utiliza *Java* e *Spring* para gerenciar a lógica e as requisições feitas pelo *frontend*. Você precisará de:

* *Maven*, para gerenciar os pacotes.
* *JDK* na versão 21.

Todos os dados são escritos em um banco de dados *Postgre*:

* *PostgreSQL*, versão 17.
* Um SGBD que aceite conexões com esse tipo de banco, sendo altamente recomendado, o *PGAdmin* ou *SQLectron*.

No ambiente de desenvolvimento, para testar as APIs e requisições recomenda-se uma plataforma de testes como *Postman* ou *INSOMNIA*.

Todos os repositórios podem ser trabalhados em qualquer sistema operacional com suporte a *NPM* e *Java*. *Windows*, sistemas baseados em *GNU/Linux* e *MacOS* são suportados para o desenvolvimento.

Contudo, o servidor que hospeda ambas as aplicações é baseado em *Linux*, sendo assim, para total compatibilidade e um ambiente de desenvolvimento similar ao de produção, recomenda-se sistemas operacionais baseados em *Linux*, como *Ubuntu*, *Debian*, *Arch, NixOS* e outros com suporte às ferramentas utilizadas.

## Pessoas

Necessário suporte técnico para implantação e monitoramento dos recursos disponibilizados na nuvem, além de desenvolvedores capacitados para implementação de melhorias e interação com itens da infraestrutura e com o produto, em caso de falhas inesperadas.

# ROTEIRO DE implantação

**BACKEND**

Em caso de atualização, para disponibilizar uma nova versão do *backend* em Java no EC2 via AWS:

1. Suba as alterações dentro do *Github* em uma nova *branch* e solicite um *Pull Request*(PR) para a *branch “main”*.
2. Após revisão, prossiga unindo a *branch* na *main*.
3. Deixe ativa a instância secundária do ec2 enquanto a principal será atualizada.
4. Derrube a instância principal e à inicie novamente.

A instância principal carregará o sistema à partir da *branch main* atualizada e iniciará o projeto.

Caso seja a primeira vez disponibilizando o *backend* em Java no *EC2* via *AWS*:

1. Suba as alterações dentro do Github em uma nova Branch e solicite um *Pull Request*(PR) para a *branch main*.
2. Após revisão, prossiga unindo a *branch* na *main*.
3. Inicie uma instância de EC2, recomendado que siga as seguintes especificações:  
   t2.micro, Amazon Linux, com 8 à 30 GB de armazenamento para a instância.
4. Realize a configuração de grupos de segurança, VPC e sub-rede corretamente para permitir o acesso à instância para atender à chamadas HTTP dentro da porta utilizada pela aplicação e permita o tráfego de saída.
5. Cole um *script\** de inicialização da máquina em *user data* dentro da instância contendo atualização de pacotes, instalação de dependências como a versão do *Java* do projeto e variáveis de ambiente utilizadas, para que assim que a instância subir, seus requisitos já estejam atendidos.

* *\*script* recomendado:

*#!/bin/bash*

*echo "Atualizando pacotes do sistema..."*

*sudo yum update -y*

*echo "Configurando variáveis de ambiente..."*

*export ENV\_VAR\_NAME="valor"*

*echo "Instalando dependências necessárias..."*

*sudo yum install -y git unzip curl wget*

*echo "Instalando SDKMAN..."*

*curl -s "https://get.sdkman.io" | bash*

*source "$HOME/.sdkman/bin/sdkman-init.sh"*

*echo "Instalando JDK 21 via SDKMAN..."*

*sdk install java 21.0.5-amzn*

*echo "Clonando repositório do GitHub..."*

*GITHUB\_REPO\_URL="https://github.com/usuario/repositorio.git"*

*CLONE\_DIR="repo"*

*git clone "$GITHUB\_REPO\_URL" "$CLONE\_DIR"*

*cd "$CLONE\_DIR" || exit*

*echo "Construindo e executando o projeto Maven..."*

*./mvnw clean package*

*java -jar target/\*.jar*

*echo "Script concluído com sucesso!"*

1. Inicialize a instância e verifique se o projeto foi iniciado.

**Frontend**

1. Crie um EC2 com Ubuntu como sistema operacional, certifique-se que é possível acessa-lo da internet, então habilite protocolos e chamadas de ip para todos ou para um grupo de segurança ja configurado.
2. Conecte-se ao servidor
3. Se não existir git na máquina, instale-o atraves do apt seguindo a documentação oficial do git.
4. se ja existir, clone o repositorio com git clone <https://github.com/Gebarito/AgendaAi-APP.git>
5. abra a pasta com cd AgendaAi-APP
6. Nos arquivos enviroment.ts e enviroment.development.ts mude a URL para o ip do servidor ec2 criado.
7. instale o node e então o npm com: sudo apt-get install nodejs && sudo apt install npm
8. na pasta do projeto digite npm install para instalar as dependencias
9. compile o projeto para produção com npm run ng build=production
10. copie a pasta criada em dist/ para a pasta do servidor (/var/www/html/) você pode usar o comando mv ou scp -r dist/<nome-do-app> usuario@<endereço-servidor>:/var/www/html/
11. Ajuste as permissões com sudo chown -R www-data:www-data /var/www/html/<nome-do-app> && sudo chmod -R 755 /var/www/html/<nome-do-app>
12. Instale o nginx para abrir a porta 80 para internet:
    1. sudo apt update
    2. sudo apt install nginx
13. sudo nano /etc/nginx/sites-available/default -> para abrir o arquivo do nginx
14. ajuste server\_name para server\_name <ip\_da\_sua\_ec2>
15. root para root /var/www/hmtl/nome\_da\_pasta\_em\_dist;
16. certifique que location esta buscando por /index.html e que o mesmo é o index.
17. sudo nginx -t para testar a configuração
18. sudo systemctl restart nginx para reiniciar o nginx
19. abra o ip do seu servidor com http:// antes no seu navegador de preferencia.

# Treinamento e suporte

Para a implantação do projeto, é necessário que os desenvolvedores realizem treinamentos específicos, abrangendo as seguintes áreas:

* **AWS:** Para configuração e manutenção da infraestrutura na nuvem, incluindo VPC, Elastic Load Balancer, subnets, instâncias EC2, e RDS.
* **PostgreSQL:** Para criação, manutenção e otimização do banco de dados utilizado no backend do projeto.
* **Node.js:** Para execução, manutenção e deploy do frontend da aplicação, garantindo o correto funcionamento das funcionalidades gráficas.
* **NPM e Gerenciamento de Dependências:** Para instalação e gerenciamento das bibliotecas utilizadas no projeto, assegurando a compatibilidade e as versões corretas.
* **Monitoramento e Logs:** Para uso de ferramentas como CloudWatch e SNS, permitindo identificar e resolver possíveis problemas em tempo real.