A black background with white text

AI-generated content may be incorrect.

**FATEC OGARI DE CASTRO PACHECO**

**ITAPIRA**

**DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA**

BRUNO GOMES DE OLIVEIRA

CARLOS EDUARDO NICIOLI DE TOLEDO

ESTER DE SOUZA SILVA CARLOS  
LUIS FELIPE SALVARANI  
RAFAEL HENRIQUE DE OLIVEIRA  
TIAGO HENRIQUE DE MORAES

**BLOG ESTÁTICO COM NEXT.JS + S3/CLOUDFRONT**

Itapira-SP

2025

SUMÁRIO

[1 INTRODUÇÃO 3](#_Toc200203846)

[1.1 OBJETIVO DO PROJETO 3](#_Toc200203847)

[1.2 PÚBLICO-ALVO 3](#_Toc200203848)

[1.3 VISÃO GERAL 3](#_Toc200203849)

[2 tecnologias utilizadas 3](#_Toc200203850)

[2.1 FRONTEND 3](#_Toc200203851)

[ Next.js 15.0.2 (com output: 'export' no next.config.js) 3](#_Toc200203852)

[ React 19 RC 3](#_Toc200203853)

[ Tailwind CSS para estilização 3](#_Toc200203854)

[ Markdown com gray-matter e remark para geração de conteúdo 3](#_Toc200203855)

[2.2 INFRAESTRUTURA 3](#_Toc200203856)

[3 ARQUITETURA DA SOLUÇÃO 3](#_Toc200203857)

[3.1 DIAGRAMA DA ARQUITETURA 3](#_Toc200203858)

[3.2 COMPONENTES 4](#_Toc200203859)

[3.3 JUSTIFICATIVAS 4](#_Toc200203860)

[4 DEPLOY DA APLICAÇÃO 4](#_Toc200203861)

[4.1 PRÉ-REQUISITOS 4](#_Toc200203862)

[ Conta AWS com acesso ao S3, CloudFront e Route 53 4](#_Toc200203863)

[ Node.js, npm e Next.js instaladosArtifact Registry: 4](#_Toc200203864)

[4.2 ETAPAS 4](#_Toc200203865)

[5 CONFIGURAÇÃO DOS SERVIÇOS AWS 5](#_Toc200203866)

[5.1 AMAZON S3 5](#_Toc200203867)

[5.2 CLOUNDFRONT 5](#_Toc200203868)

[5.3 ROUT 53 5](#_Toc200203869)

[5.4 IAMPQ 5](#_Toc200203870)

[6 ESTRUTURA DO PROJETO 5](#_Toc200203871)

[7 BOAS PRÁTICAS ADOTADAS 6](#_Toc200203872)

[8 TESTES E VALIDAÇÕES 6](#_Toc200203873)

[9 CUSTOS ESTIMADOS 6](#_Toc200203874)

[10 LIMPEZA DE RECURSOS 6](#_Toc200203875)

[11 CONCLUSÃO 7](#_Toc200203876)

[12 REFERÊNCIA 7](#_Toc200203877)

# INTRODUÇÃO

## OBJETIVO DO PROJETO

O projeto tem como objetivo desenvolver um blog estático utilizando o framework Next.js, com exportação em HTML estático (Static HTML Export), e hospedá-lo de forma eficiente na Amazon Web Services (AWS). Para isso, são utilizados os serviços Amazon S3 para armazenamento de arquivos, CloudFront como CDN para entrega de conteúdo otimizada, e Route 53 para gerenciamento de domínios e DNS.

## PÚBLICO-ALVO

Desenvolvedores web, profissionais de DevOps, estudantes e equipes que desejam hospedar aplicações estáticas escaláveis na AWS.

## VISÃO GERAL

Este projeto é baseado no repositório de exemplo "blog-starter" da Vercel, e foca em mostrar como adaptar esse projeto para ser servido de forma estática na AWS com boas práticas de infraestrutura.

# tecnologias utilizadas

## FRONTEND

* Next.js 15.0.2 (com output: 'export' no next.config.js)
* React 19 RC
* Tailwind CSS para estilização
* Markdown com gray-matter e remark para geração de conteúdo

## INFRAESTRUTURA

* Amazon S3 para hospedagem dos arquivos estáticos
* Amazon CloudFront como CDN
* Amazon Route 53 para gerenciamento de domínios e DNS
* IAM (Identity and Access Management) para permissões seguras

# ARQUITETURA DA SOLUÇÃO

## DIAGRAMA DA ARQUITETURA

Diagrama

O conteúdo gerado por IA pode estar incorreto.

## COMPONENTES

* Next.js: gera arquivos HTML estáticos.
* S3: serve os arquivos .html, .css, .js, imagens e fontes.
* CloudFront: acelera a entrega global de conteúdo.
* Route 53: vincula domínio personalizado ao CloudFront.

## JUSTIFICATIVAS

A utilização da exportação estática com Next.js possibilita uma arquitetura otimizada, com excelente performance e baixos custos de operação. Os arquivos gerados são prontos para serem servidos diretamente por serviços de armazenamento como o Amazon S3, eliminando a necessidade de servidores ou renderização dinâmica.

Embora o uso do Amazon CloudFront e do Amazon Route 53 seja amplamente recomendado para melhorar a performance de distribuição global e o gerenciamento profissional de domínios personalizados, optamos por não implementá-los nesta fase do projeto com o objetivo de manter a solução dentro do limite gratuito da AWS (Free Tier), sem incorrer em custos adicionais.

O Amazon S3, por si só, já fornece entrega de arquivos estáticos confiável e eficiente, suficiente para os propósitos atuais do projeto.

O CloudFront poderá ser adicionado futuramente para otimizar a entrega global, com caching e HTTPS integrados.

O uso de um domínio personalizado com o Route 53 também poderá ser implementado em etapas posteriores, conforme a evolução e visibilidade do projeto justificar a despesa.

Essa abordagem garante uma arquitetura escalável, com infraestrutura mínima, econômica e facilmente extensível conforme as necessidades futuras.

# DEPLOY DA APLICAÇÃO

## PRÉ-REQUISITOS

* Conta AWS com acesso ao S3, CloudFront e Route 53
* Node.js, npm e Next.js instaladosArtifact Registry:

## ETAPAS

1. npm run build com output: 'export' no next.config.js
2. Copiar a pasta out/ para o S3
3. Configurar bucket como estático
4. Criar distribuição no CloudFront
5. (Opcional) Configurar DNS com Route 53

# CONFIGURAÇÃO DOS SERVIÇOS AWS

## AMAZON S3

* Criar bucket com público acesso (ou via OAC com CloudFront)
* Habilitar hospedagem estática
* Upload da pasta outAcionar o Cloud Build:

## CLOUNDFRONT

* Origens: bucket S3
* Comportamento: redirecionar 404 para /index.html
* Habilitar HTTPS

## ROUT 53

* Registrar domínio ou usar um existente
* Criar zona hospedada e vincular ao CloudFront

## IAMPQ

* Criar política para permitir acesso seguro S3 (se for usar OAC)

# ESTRUTURA DO PROJETO

/BLOG-STARTER-APP

├── .GIT/

├── .NEXT/

├── \_POSTS/

├── NODE\_MODULES/

├── OUT/ (GERADO NO EXPORT)

├── PUBLIC/

├── SRC/

├── .GITIGNORE

├── NEXT.CONFIG.JS

├── NEXT-ENV.D.TS

├── PACKAGE.JSON

├── PACKAGE-LOCK.JSON

├── POSTCSS.CONFIG.JS

├── README.MD

├── TAILWIND.CONFIG.TS

└── TSCONFIG.JSON

* src/: Componentes, páginas e estilos customizados
* \_posts/: Arquivos .md para cada post do blog
* public/: Imagens e ativos estáticos

# BOAS PRÁTICAS ADOTADAS

* Otimização de imagens com next/image
* Uso de metadados para SEO
* Layout responsivo com Tailwind CSS
* HTML semântico e acessível

# TESTES E VALIDAÇÕES

* Testes locais com serve -s out
* Validação da entrega via CloudFront
* Validação de links, imagens e 404
* (Opcional) Testes com Lighthouse

# CUSTOS ESTIMADOS

* Gratuito (dentro do Free Tier da AWS):
* S3: 5 GB armazenamento e 20.000 requisições/mês
* CloudFront: 1 TB transferência/mês
* Route 53: Zona hospedada (US$ 0.50/mês) + domínio (opcional)

# LIMPEZA DE RECURSOS

* Deletar arquivos no bucket S3
* Deletar distribuição CloudFront (após desativar)
* Deletar bucket S3
* Deletar zona hospedada no Route 53 (se usada)

# CONCLUSÃO

O projeto demonstra uma forma moderna, eficiente e com baixo custo de hospedar um blog estático usando Next.js com exportação estática na infraestrutura da AWS. Essa solução é escalável, fácil de manter e ideal para conteúdo que não muda com frequência.

# REFERÊNCIA

AMAZON WEB SERVICES. Amazon S3: Static Website Hosting. Disponível em: <https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html>. Acesso em: 7 jun. 2025.

AMAZON WEB SERVICES. Amazon CloudFront: Developer Guide. Disponível em: <https://docs.aws.amazon.com/cloudfront/index.html>. Acesso em: 7 jun. 2025.

AMAZON WEB SERVICES. Amazon Route 53: Developer Guide. Disponível em: <https://docs.aws.amazon.com/route53/>. Acesso em: 7 jun. 2025.

VERCEL. Next.js Documentation. Disponível em: <https://nextjs.org/docs>. Acesso em: 7 jun. 2025.

VERCEL. next.js/examples/blog-starter. Disponível em: <https://github.com/vercel/next.js/tree/canary/examples/blog-starter>. Acesso em: 7 jun. 2025.