
JWT
AUTENTICAÇÃO
E AUTORIZAÇÃO

O QUE É JWT ?

JSON WEB TOKEN

tecnologia que permite enviar informações de forma segura e compacta entre diferentes partes de um sistema. Ele é muito usado para autenticação e autorização em APIs e sistemas web.

ESTRUTURA JWT

HEADER, PAYLOAD, SIGNATURE

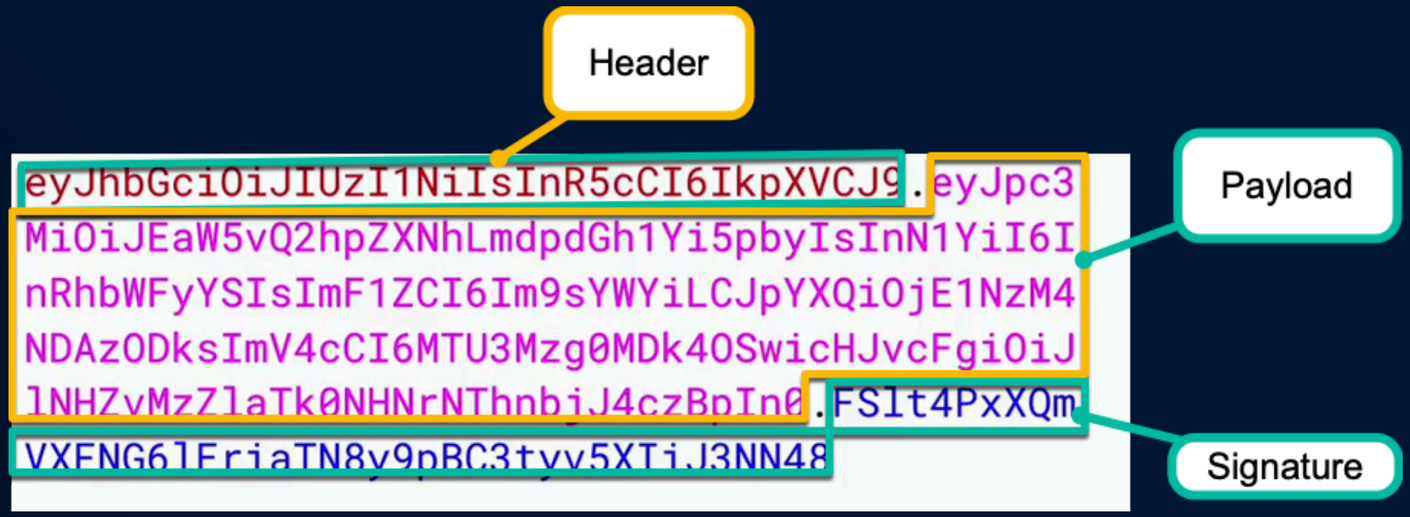
Header: Especifica o algoritmo de assinatura e o tipo de token;

Payload: Carrega os dados do usuário, como nome e permissões, mas não deve conter informações confidenciais, pois é apenas codificado, não criptografado;

Signature: Garante a integridade do token ao combinar o Header, Payload e uma chave secreta conhecida apenas pelo emissor, assegurando que qualquer alteração no conteúdo invalide o token.

DIVISÃO JWT

HEADER, PAYLOAD, SIGNATURE



SECRET KEY

O QUE É A SECRET KEY

• A Secret Key (chave secreta) para JWT (JSON Web Token) é um componente essencial no processo de criação e validação de tokens. Ela é usada principalmente em algoritmos de assinatura simétrica, como o HS256 (HMAC com SHA-256), para assegurar a integridade e autenticidade do token

Use uma chave suficientemente longa (pelo menos 32 caracteres) e evite padrões previsíveis.
Exemplo:
2n&$8!aH^uM@f#qWxzV9*LpR~YjD5

BCRYPT

HASH DE SENHAS

O bcrypt é uma biblioteca amplamente usada para criptografar senhas, garantindo maior segurança em sistemas. Ele utiliza um algoritmo de hashing que incorpora um "salt" (valor aleatório) para tornar cada hash único, mesmo que duas senhas idênticas sejam processadas. Isso dificulta ataques de força bruta e tabelas rainbow. No processo, a senha é convertida em um hash criptografado antes de ser armazenada no banco de dados. Durante a autenticação, o bcrypt compara a senha fornecida pelo usuário com o hash armazenado, verificando se correspondem sem precisar armazenar a senha em texto plano.

AUTORIZAÇÃO E AUTENTICAÇÃO

O QUE É AUTENTICAÇÃO E AUTORIZAÇÃO

Autenticação:
Processo de verificar a identidade do usuário (ex.: login com usuário e senha).

Autorização:
Processo de verificar os privilégios ou permissões do usuário para acessar recursos específicos.

JWT NA PRÁTICA - BACKEND

PASSO 1 - INSTALAR DEPENDÊNCIAS NECESSÁRIAS

Dependências:
NestJS - Jwt - NestJS/Passport - Bcrypt.

Executar: NPM, YARN, PNPM

PASSO 2 - DEFINIR ARQUITETURA NO NESTJS

Nest g module Auth
Nest g service Auth
Nest g guard AuthLocal

PASSO 3 - HASH COM BCRYPT

O bcrypt é uma biblioteca que criptografa senhas com segurança, utilizando um "salt" para gerar hashes únicos. Ele armazena apenas o hash no banco de dados e, na autenticação, compara a senha fornecida com o hash, protegendo contra ataques de força bruta

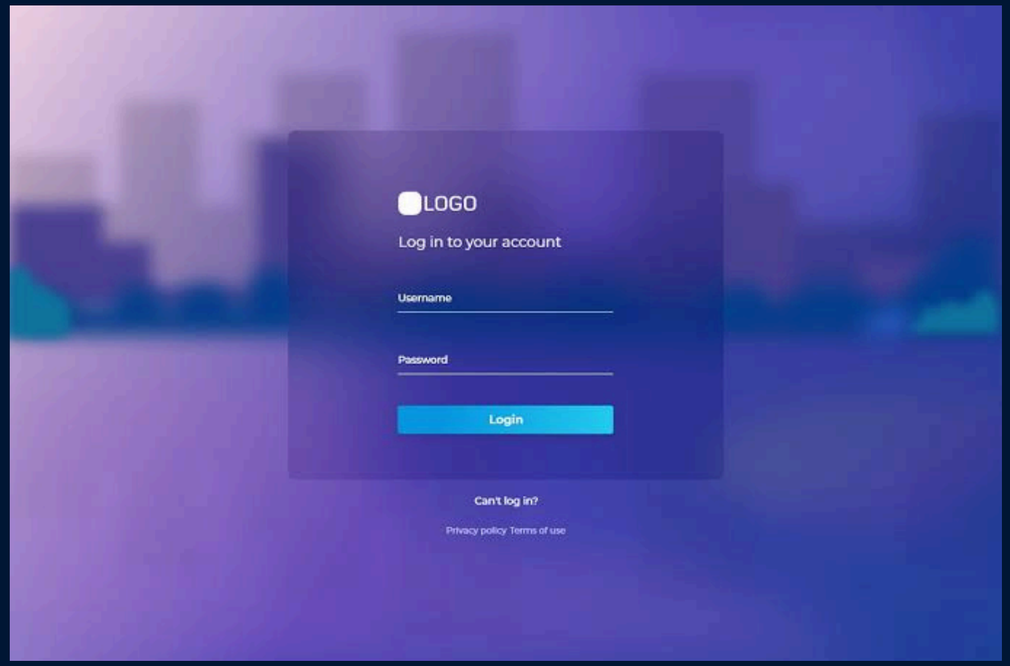
PASSO 4 - AUTORIZAÇÃO

Definir regras de autorização baseado nas regras de negócios para usuários comuns e usuários administradores

JWT NA PRÁTICA - BACKEND

COMO FUNCIONA ?

No fluxo de login, o cliente envia as credenciais (e-mail/usuário e senha) para o servidor por meio de uma requisição. O servidor valida os dados, verifica se o usuário existe no banco de dados e compara a senha enviada com a senha armazenada (geralmente hash). Se as credenciais forem válidas, o servidor retorna um token de autenticação (como JWT), que o cliente usará para acessar recursos protegidos. Caso contrário, uma mensagem de erro é retornada.



JWT NA PRÁTICA - FRONTEND

COMO O JWT É RECEBIDO?

Após o login, o back-end gera um JWT e o retorna ao front-end. Esse token pode ser enviado no corpo da resposta ou no cabeçalho da requisição

ENVIANDO O JWT NAS REQUISIÇÕES

O front-end utiliza o JWT para autenticar requisições feitas a rotas protegidas no back-end. O token é incluído no cabeçalho da requisição HTTP, geralmente no campo Authorization, seguindo o formato Bearer <token>.

ARMAZENAMENTO DO JWT

O token pode ser armazenado no localStorage, sessionStorage ou em HTTP-Only Cookies (opção mais segura). Ele será usado para autenticar futuras requisições.

DECODIFICANDO O JWT

O front-end pode decodificar o token para acessar informações, como permissões ou dados do usuário, utilizando bibliotecas como jwt-decode.