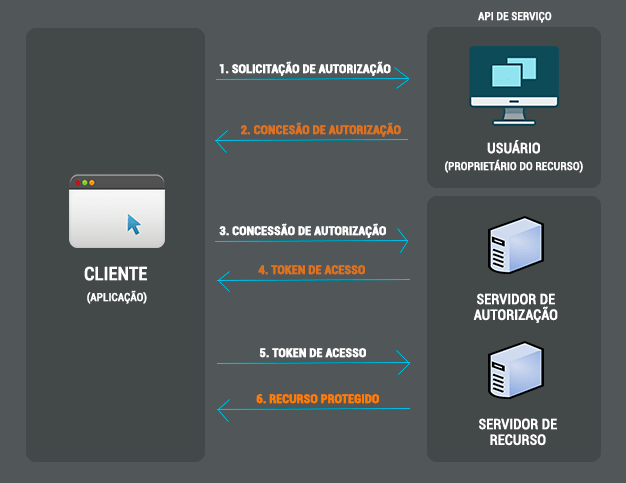
**oauth2.0**



OAuth 2 é um protocolo de autorização que permite que uma aplicação se autentique em outra. Para que isso aconteça, uma aplicação pede permissão de acesso para um usuário, sem que para isso ela tenha acesso a alguma senha dele. O usuário pode conceder ou não o acesso à aplicação. Depois da permissão ser aceita, caso o usuário precise alterar a senha de acesso, a permissão continuará válida para a aplicação e, caso necessário, a permissão dada à aplicação pode ser revogada a qualquer momento também.

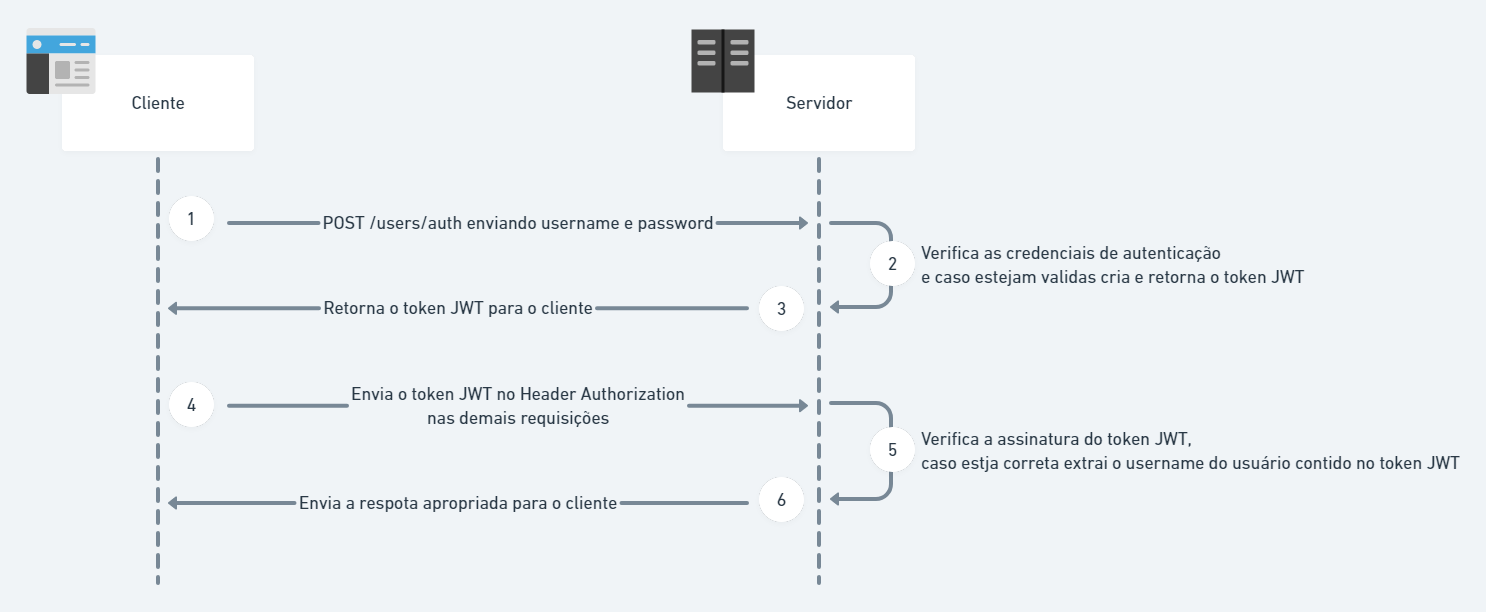
OAuth 2 é utilizado nos mais diversos tipos de autenticação, como em telas de login e na autenticação de [APIs (Application Programming Interface)](https://www.treinaweb.com.br/blog/o-que-e-uma-api/).

**JWT – JSON WEB TOKEN**

Método de autorização de um Token JWT (JSON Web Token) padrão que tem como objetivo transmitir ou armazenar de forma compacta e segura objetos JSON entre diferentes aplicações. O JWT é uma assinatura digital (string de caracteres) que garante a autenticidade do token.

Quando um usuário se autentica no sistema (com usuário e senha), o servidor gera um token com data de expiração pra ele. Durante as requisições seguintes do cliente, o JWT é enviado no cabeçalho da requisição e, caso esteja válido, a API irá permitir acesso aos recursos solicitados, sem a necessidade de se autenticar novamente.

Diagrama abaixo mostra o fluxo de autenticação:



O JWT é um JSON que contém informações que permite mais tarde conceder autorização a determinados recursos de nossa api para um usuário especifico.

Instalação de dependências:

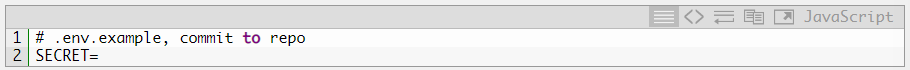
npm i jsonwebtoken (Pacote que implementa o protocolo JSON Web Token)

npm i dotenv-safe (Pacote para gerenciar as variáveis de ambiente)

**Adicionando o JWT**

Para gerenciar as variáveis de ambientes criamos dois arquivos:

1 - .env.example:

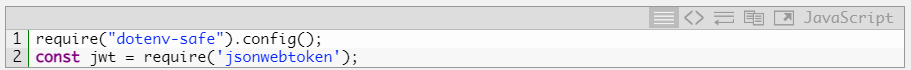


2 - .env, com o valor do segredo definido:



O Segredo será utilizado pela biblioteca jsonwebtoken para assinar o token garantindo que somente o servidor consiga validar.

No index.js carregamos as variáveis de ambiente e criamos um objeto JWT para gerar e verificar tokens.



**Autenticação**

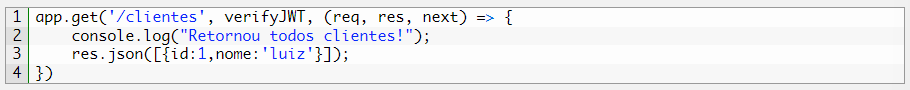
Para que o token JWT seja gerado é necessário que o usuário esteja logado com autenticação normal (usuário e senha), as informações são validadas junto a base de dados.

Criamos duas uma para logar e outra para deslogar o usuário. Na rota ‘/login’ vai receber o usuário e senha, se o usuário estiver ok, retorna um JWT para o cliente, caso contrário retornar erro de autenticação. Já na rota ‘/logout’.

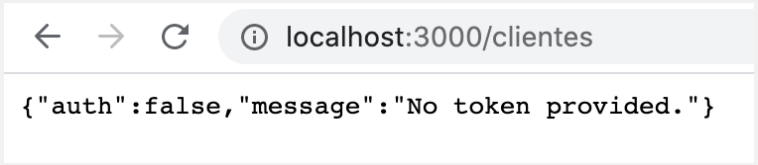
**Autorização**

Criamos uma função em nosso index.js para verificar se existe uma JWT válido, se for válido, caso contrário gera um erro.

Verificamos a autenticidade do token com outra função verifyJWT( ), utilizado a variável de ambiente SECRET, caso não consiga verificar o token, retornará um erro. O acesso a esta aplicação é feito na chamada GET /cliente definido em nossa API:



A API cria uma camada intermediária de autorização baseada em JWT, que bloqueia as requisições que não estejam autenticadas e autorizas. O retorno da rota /clientes sem está autenticado:



Para que seja possível acessar o client da API, obtemos um token válido na rota do POS de login utilizando o POSTMAN para realizar esta requisição.

Nas aplicações da API será preciso inserir token gerado e todas as requisições aos endpoints protegidos.

OBS.: O token gerado expira em 5 minutos após a sua criação, novas requisições podem ser realizadas durante este período sem precisar novo login.