OAuth 2.0 e JSON Web Token (JWT)

Mentor: Carlos Júnior

1 - OAuth 2.0

- O que é?
- Como funciona?

2 - JSON Web Token (JWT)

- O que é?
- Como funciona?

3 - JWT com Node.js

- Como funciona?

OAuth 2.0

Open-Authorization (OAuth) é um protocolo de **autorização** que possibilita aplicações Web, Mobile e Desktop acessarem recursos protegidos de um servidor através do protocolo HTTP.

Autenticação:
Processo de confirmar a identidade de um usuário ou dispositivo.

Autorização:
Processo de conceder ou negar acesso a recursos.

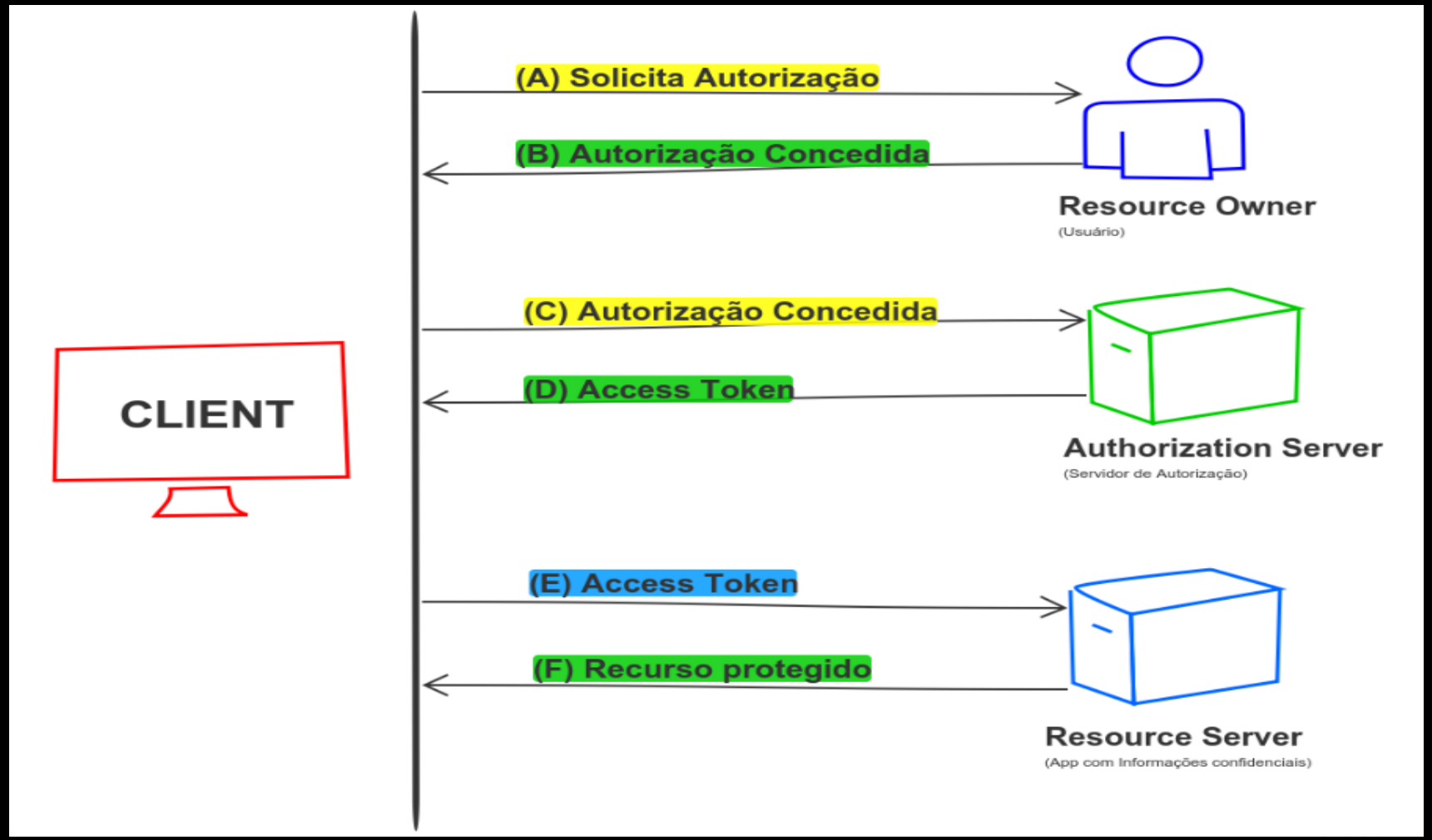
OAuth 2.0

O OAuth 2.0 é composto pelos seguintes componentes:

- **Dono do Recurso (Resource Owner):** uma entidade capaz de conceder acesso à um recurso protegido.
- **Servidor do Recurso (Resource Server):** o servidor que hospeda os recursos protegidos. O acesso a ele é feito através de **tokens**.
- **Servidor de Autorização (Authorization Server):** servidor que emite **tokens** de acesso ao cliente, depois de sua autenticação e obtenção de autorização.
- **Cliente (Client):** uma aplicação requisitando recursos protegidos, através da autorização do dono.

OAuth 2.0

No geral, o fluxo de autorização é o seguinte:



- (A) Para ter acesso ao conteúdo protegido do Servidor do Recurso (Resource Server), o cliente (Client) solicita autorização ao dono do recurso (Resource Owner)
- (B) A **Autorização é Concedida** pelo dono do recurso ao, por exemplo, clicar no botão Login
- (C) O cliente, com a autorização e a autenticação do dono do recurso, solicita um token de acesso ao Servidor de Autorização (Authorization Server)
- (D) Se a autorização e a autenticação passada pelo cliente for válida, um token de acesso (Access Token) será criado e devolvido para o cliente gerenciar
- (E) O cliente, com o token de acesso, solicita ao Servidor de Recurso algum conteúdo protegido
- (F) O Servidor de Recurso faz a validação do token de acesso. Sendo o token válido, o conteúdo protegido é enviado para o cliente

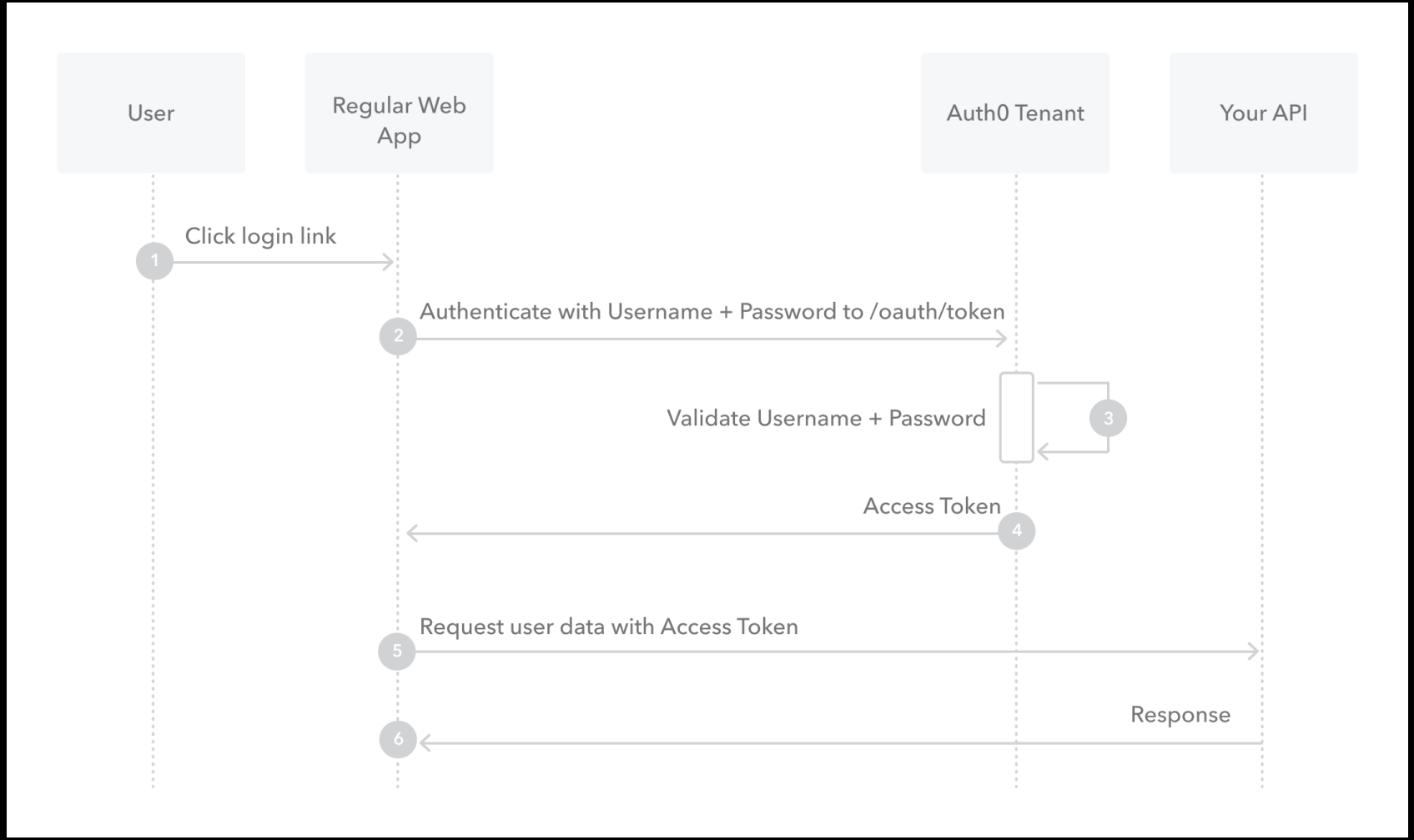
OAuth 2.0

O OAuth 2.0 possui diversos fluxos de autorização de acesso aos recursos. Abaixo listaremos apenas os padrões:

- Resource Owner Password Credentials
- Implicit
- Authorization Code
- Client Credentials

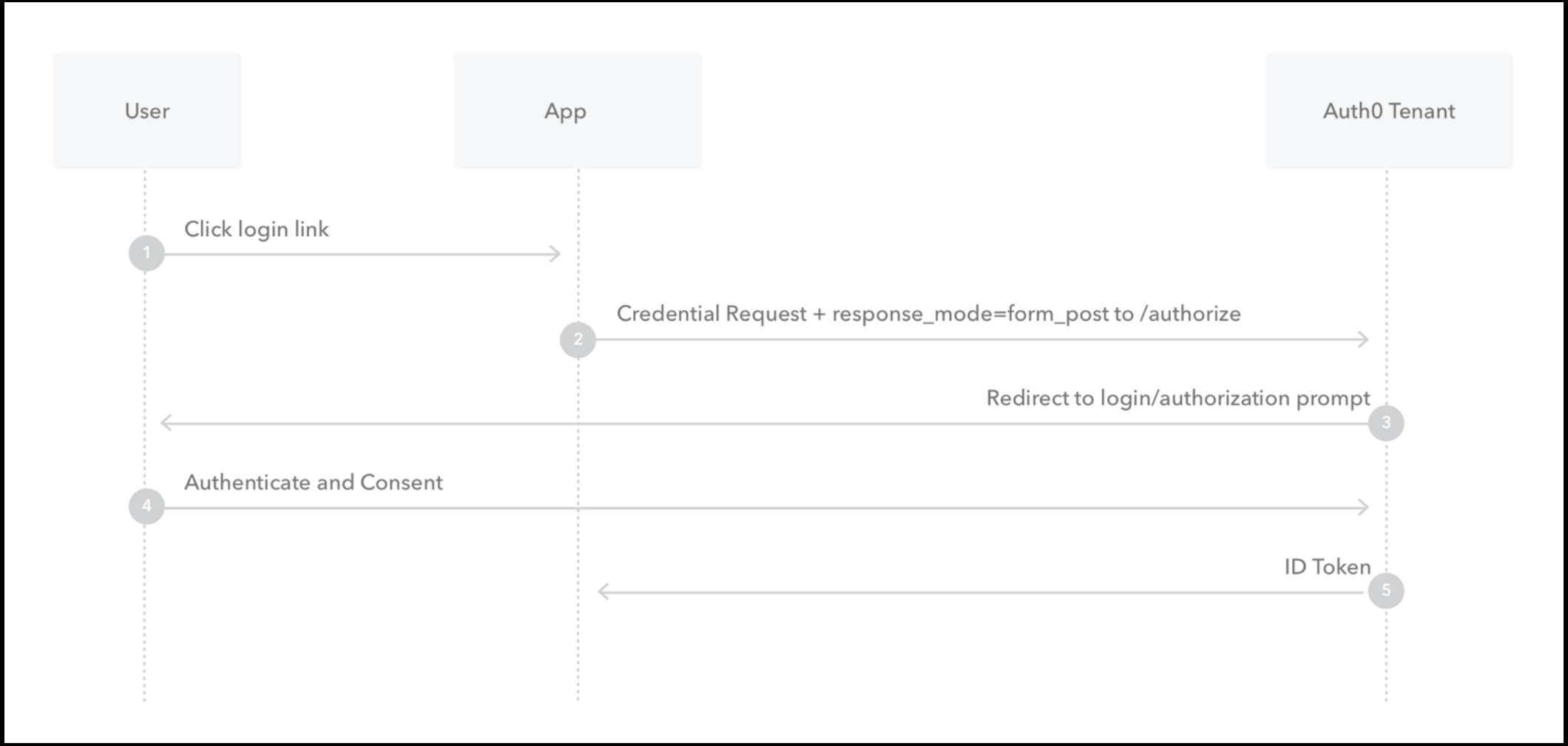
OAuth 2.0

Resource Owner Password Credentials:



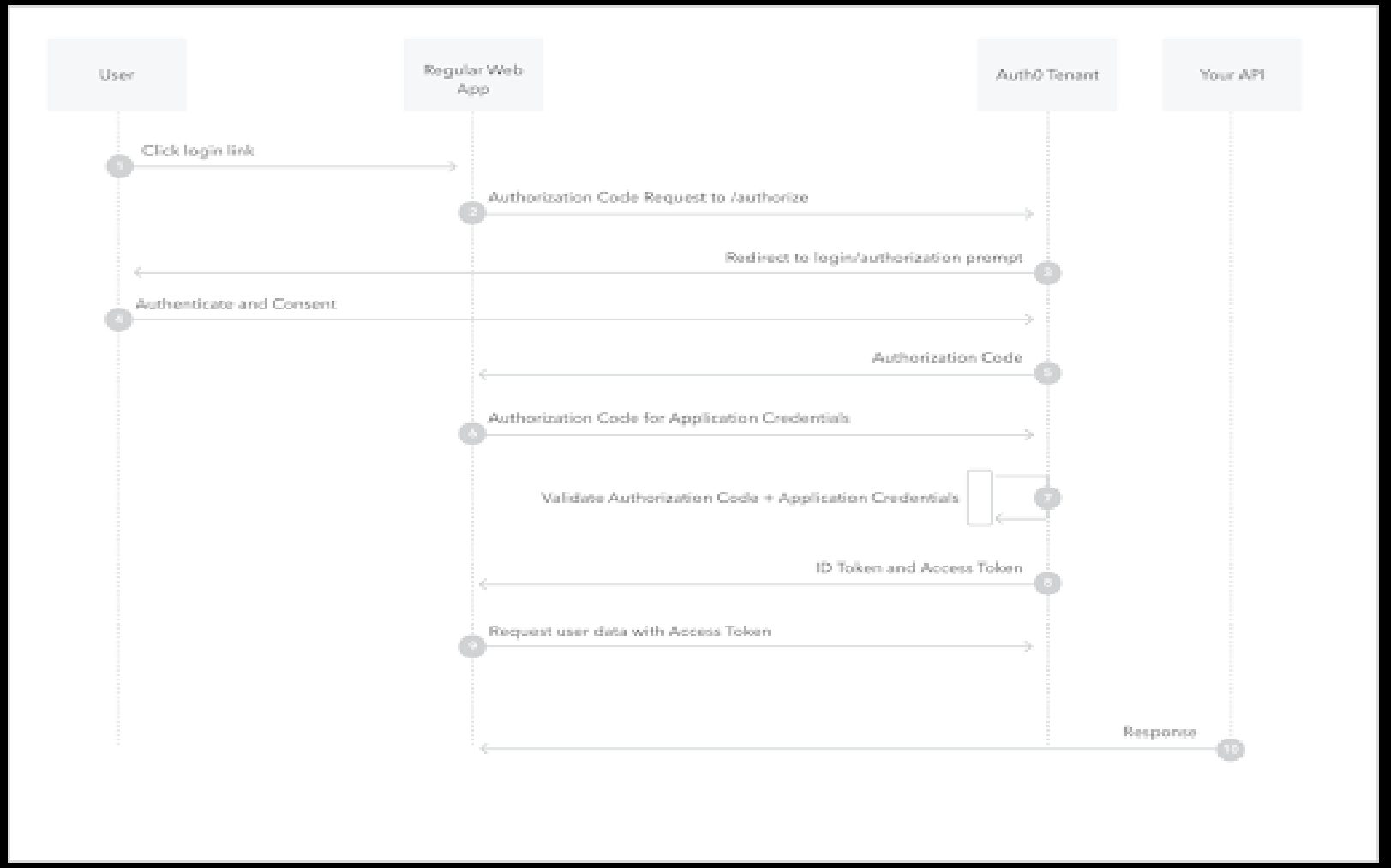
OAuth 2.0

Implicit:



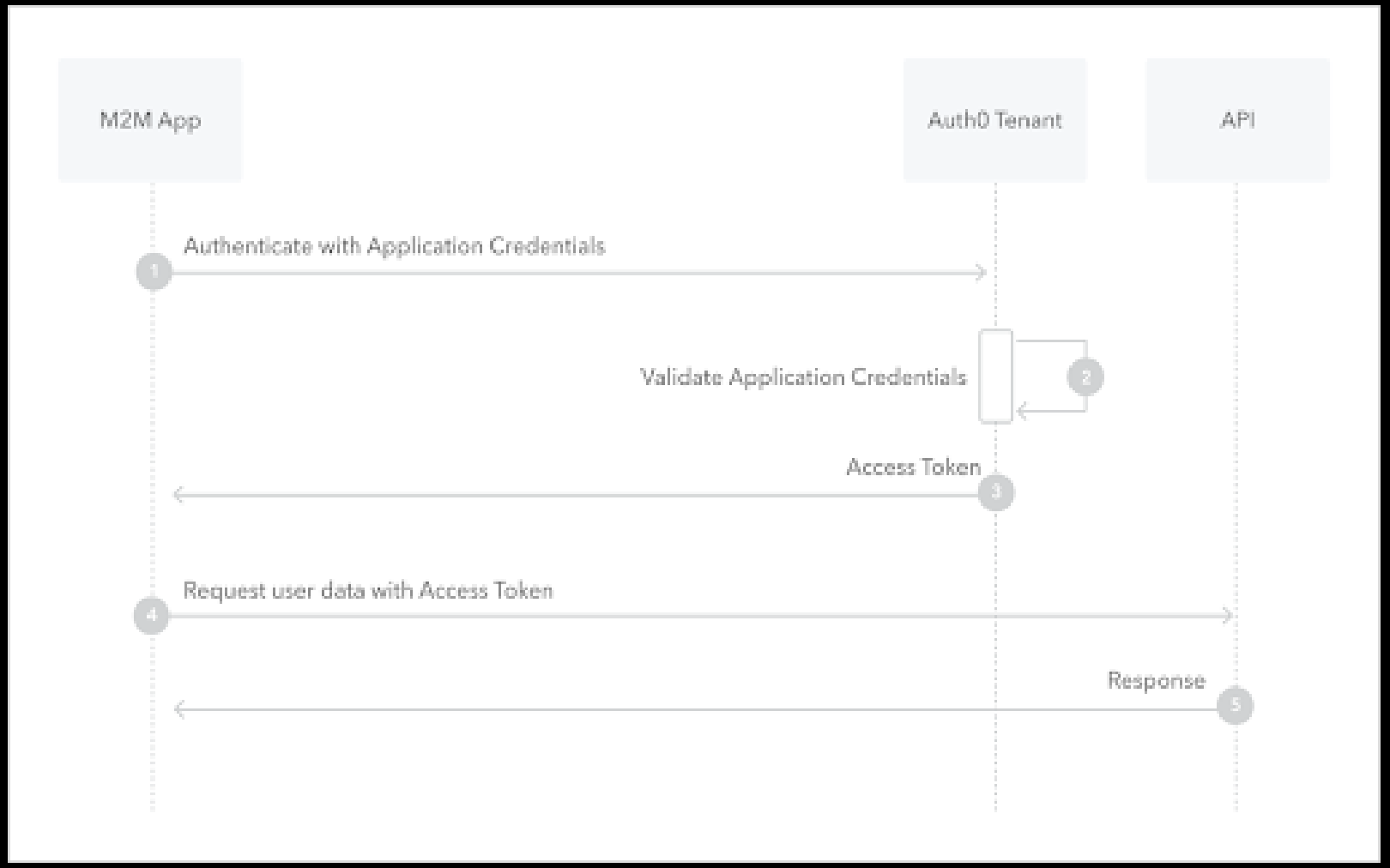
OAuth 2.0

Authorization Code:



OAuth 2.0

Client Credentials:



JSON Web Token (JWT)

Tipo de token que pode ser usado para transmitir informações seguras entre aplicações. Ele é composto por três partes separadas por ponto. Cada parte é um base64 que, decodificado, representa algo.

Exemplo de token JWT:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.*eyJzdWIiOjE1NTQ4Nzc0NSwiaXNzIjoiZWNvbW1lcmNlIGJhY2siLCJhdWQiOiJlY29tbWVyY2UgZnJvbnQiLCJpYXQiOjE3MDYwMDk5OTgsImV4cCI6MTcwNjQ4NjQwMH0*.***oi1jr-o7orYZ5pM3Z0IaSOT8zs9UbSSaRnCm83F_2TY***

JSON Web Token (JWT)

Primeira parte (Header):

```
// eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
{
  "alg": "HS256",   // algoritmo que será usado para criar a assinatura
  "typ": "JWT"
}
```

JSON Web Token (JWT)

Segunda parte (Payload):

```
/* eyJzdWIiOjE1NTQ4Nzc0NSwiaXNzIjoiZWNvbW1lcmNlIGJhY2siLCJhdWQiOiJlY29tb
WVyY2UgZnJvbnQiLCJpYXQiOjE3MDYwMDk5OTgsImV4cCI6MTcwNjQ4NjQwMH0 */
{
    "sub": 155487745, // (subject) entidade à quem o token pertence
    "iss": "ecommerce back", // (issuer) emissor do token
    "aud": "ecommerce front", // (audience) destinatário do token
    "iat": 1706009998, // (issued at) numeric date de quando o token foi criado
    "exp": 1706486400 // (expiration) numeric date de quando o token irá expirar
}
```

JSON Web Token (JWT)

Terceira parte (Signature):

```
// oi1jr-o7orYZ5pM3Z0IaSOT8zs9UbSSaRnCm83F_2TY
base64UrlEncode(
    HMACSHA256(
        base64UrlEncode(header) + "." + base64UrlEncode(payload),
        my_secret
    )
)
```

JWT com Node.js

Existem diversos pacotes que trabalham com tokens JWT, mas precisamos utilizar um e o escolhido será o pacote jsonwebtoken.

Gerando um token JWT com assinatura:

```
const jwt = require('jsonwebtoken')

const payload = {
    sub: 155487745,
    iss: 'ecommerce back',
    aud: 'ecommerce front',
    iat: 1706009998,
    exp: 1706486400
}

const secret = 'ecommerce_secret'

console.log(jwt.sign(payload, secret))
```

JWT com Node.js

Delegando a criação dos atributos iat e exp do Payload para o pacote:

```
const jwt = require('jsonwebtoken')

const payload = {
    sub: 155487745,
    iss: 'ecommerce back',
    aud: 'ecommerce front'
}

const secret = 'ecommerce_secret'

const options = {
    expiresIn: '60s'
}

console.log(jwt.sign(payload, secret, options))
```

JWT com Node.js

Validando a assinatura do token JWT:

```
try {
    const payload = jwt.verify(tokenClient, secret)

    console.log(payload)
} catch(err) {
    console.log(err.message)
}
```

Sua vez!

Agora vamos adicionar autenticação e autorização na API do mini projeto de JavaScript. Apenas as pessoas cadastradas poderão consumir todos os recursos da API. Sendo assim, você precisará adicionar a primeira pessoa manualmente na base de dados.

1. Para autenticar uma pessoa, sua API deve conter uma rota */login*. Escolha qual o método da requisição e o status code das respostas, tendo em mente os motivos. A credencial da pessoa (email) deve ser enviada pelo corpo da requisição no formato JSON. O campo do email deve ser validado em termos de tipo e formato. A resposta deve ser um JSON no seguinte formato

1. ○

caso a pessoa tenha cadastro:

```
{
    "token": "token JWT assinado e com as informações da pessoa, o atributo iat e exp"
}
```

caso a pessoa não tenha cadastro:

```
{
    "message": "mensagem especificando o motivo do erro"
}
```

Sua vez!

2. Adicione autorização em todas as rotas da sua API, exceto na de *login* (kkkk'). Escolha qual o método da requisição e o status code das respostas, tendo em mente os motivos. O token JWT assinado deve ser enviado no header da requisição em uma propriedade chamada *authorization*. Sua API deve validar a assinatura do token e se ele não expirou. A resposta deve ser um JSON no seguinte formato

   caso a pessoa não tenha autorização:

```
{
    "message": "mensagem especificando o motivo do erro"
}
```