Desenvolvimento de Software para Web

UFC - Universidade Federal do Ceará

André Meireles
andre@crateus.ufc.br

Desenvolvimento de Software para Web

MPA - Autenticação e Filtros

UFC - Universidade Federal do Ceará

André Meireles
andre@crateus.ufc.br

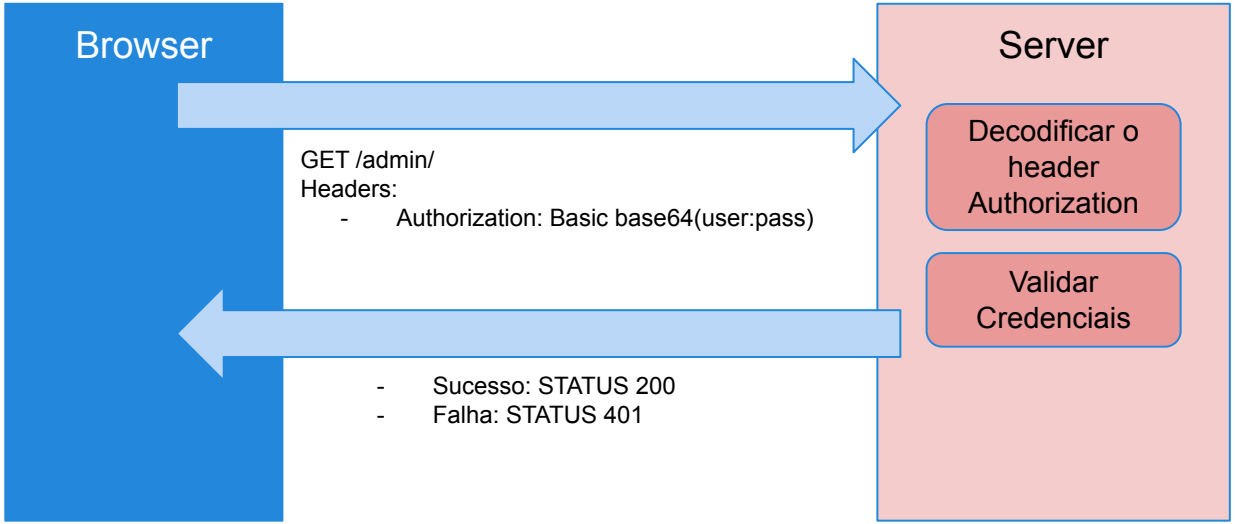
Autenticação HTTP

Os esquemas de autenticação HTTP mais comuns são:

- Basic: veja RFC 7617, credenciais codificadas em base64
- Bearer: veja RFC 6750, tokens bearer (de portador) para acessar recursos protegidos por OAuth 2.0
- Digest: veja RFC 7616, apenas hash md5 é suportado no Firefox
- HOBA: veja RFC 7486 , HTTP Origin-Bound Authentication (Autenticação Vinculada à Origem HTTP), baseado em assinatura digital
- Mutual (veja draft-ietf-httpauth-mutual),
- AWS4-HMAC-SHA256 (veja Documentação AWS)

Autenticação HTTP

A mais comum é a Basic Authentication



Express Middleware

- O **middleware** é uma função invocada antes de uma requisição atingir uma rota.
- Em outras linguagens e frameworks é comumente chamado de filtro (filter)
- No Express pode ser definido usando a função app.use
  - A sequência de inclusão determina a sequência de chamada dos filtros
  - Use next() para seguir para o próximo *midleware* ou rota
  - Você pode usar `res.send()` `res.render()` ou `res.redirect()` normalmente para responder imediatamente a requisição

Express Basic Auth

Instalar como dependência

```
# npm install express-basic-auth
```

Este módulo provê a função `basicAuth` que implementa o Basic Authentication e permite algumas customizações

```
const basicAuth = require('express-basic-auth')
...
app.use(basicAuth({
    users: { 'admin': '12345' },
    challenge: true
}));
```

Express Basic Auth

Com usuários estáticos

```
app.use(basicAuth({
    users: {
        'admin': 'supersecret',
        'adam': 'password1234',
        'eve': 'asdfghjkl',
    }
}));
```

Express Basic Auth

Autorizador customizado

```
app.use(basicAuth( { authorizer: myAuthorizer } ))

function myAuthorizer(username, password) {
    const userMatches = basicAuth.safeCompare(username, 'customuser')
    const passwordMatches = basicAuth.safeCompare(password,
'custompassword')

    return userMatches & passwordMatches
}
```

Express Basic Auth

Autorizador customizado assíncrono

```
app.use(basicAuth({
  authorizer: myAsyncAuthorizer,
  authorizeAsync: true,
}))

function myAsyncAuthorizer(username, password, cb) {
  if (username.startsWith('A') & password.startsWith('secret'))
      return cb(null, true)
  else
      return cb(null, false)
}
```

Acessando um banco de dados

Vamos utilizar o banco de dados NoSQL MongoDB.

Você precisará executar ou utilizar um servidor MongoDB.

Para executar localmente,
- Crie um arquivo docker-compose.yml com o conteúdo ao lado

- Execute na mesma pasta do arquivo:

```
# docker-compose up
```

```yaml
version: '3.1'

services:

  ufc-web-mongo:
    image: mongo
    volumes:
      - ufcwebmongodata:/data/db
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: rootpwd

  ufc-web-mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: admin
      ME_CONFIG_MONGODB_ADMINPASSWORD: adminpwd
      ME_CONFIG_MONGODB_URL: mongodb://root:rootpwd@ufc-web-mongo:27017/

volumes:
  ufcwebmongodata:
```

Acessando o Banco de Dados

Instale o cliente mongo no projeto

npm install --save mongodb

Utilize o código ao lado para criar um módulo de acesso ao banco de dados

```
const { MongoClient } = require('mongodb');

// Connection URL
const url = 'mongodb://root:rootpwd@localhost:27017';
const client = new MongoClient(url);

// Database Name
const dbName = 'ufcwebauth';

var user_collection;
async function main() {
  // Use connect method to connect to the server
  await client.connect();
  console.log('Connected successfully to server');
  const db = client.db(dbName);
  user_collection = db.collection('user');
  // the following code examples can be pasted here...

  return 'done.';
}

main()
  .then(console.log)
  .catch(console.error);
//   .finally(() => client.close());

async function getUsers(username, password) {
    const findResult = await user_collection.find({username: username, password:
password}).toArray();
    // console.log('Found documents =>', findResult);
    return findResult;
}

exports.getUsers = getUsers;
```

Autorização via Banco de Dados

Utilize o autorizador assíncrono para consultar o usuário no banco

```
app.use('/produto/*', basicAuth( { authorizer: myAuthorizerMongo, authorizeAsync: true , challenge:
true} ));

function myAuthorizerMongo(username, password, cb) {
    console.log(database.getUsers(username, password).then(users => {
        return cb(null, users.length > 0);
    }));
}
```

Decodificando as credenciais

Após a autorização, você pode acessar o login do usuário na rota destino:

```
app.get('/', (req, res) => {
...
  basic = Buffer.from(req.headers.authorization.split(' ')[1], 'base64').toString().split(':');
  username = basic[0];
  password = basic[1];
  console.log('Username: ' + username);
  console.log('Password: ' + password);

...
}
```

Links importantes

- https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Authentication
- https://expressjs.com/pt-br/4x/api.html#app.use
- https://www.npmjs.com/package/express-basic-auth
- https://www.mongodb.com
- https://www.npmjs.com/package/mongodb