Casos de Uso

AUTOMATED TEST TYPES

**Equipo** : 2

**Given** un usario

**When** el usuario ingrese sus credenciales correctamente

**Then** el sistema concatenara y codificara su información de usuario.

**Given** un usuario

**When** el usuario intente ingresar 3 veces con una contraseña incorrecta

**Then** el sistema bloqueara al usuario por 5 minutos.

**App.Js**

const {login} = require("./Login");

const {getToken} = require("./Login");

const express = require('express')

const bodyParser = require('body-parser');

const port = 3000

const app = express()

app.use(bodyParser.urlencoded({extended: true}))

app.use(bodyParser.json())

app.use(bodyParser.raw())

app.get('/', (req, res) => {

res.send('Hello World!')

})

app.post('/login', (req, res) => {

const userCredentials = {email: req.body.email, password: req.body.password}

const response = login(userCredentials)

res.send(response)

})

// authentication endpoint

app.post('/authenticate', (req, res) => {

const userCredentials = {email: req.body.email, password: req.body.password}

const token = getToken(userCredentials)

res.send(token)

})

app.listen(port, () => {

console.log(`Example app listening at http://localhost:${port}`)

})

**Login.Js**

const { Console } = require("console");

const users = [{email: "myemail@mail.com", password: "securePassword", attemps: 0, lastAttempt: 0}];

const login = (userCredentials) => {

let isUserRegistered = false;

for (let i = 0; i < users.length; i++) {

if (userCredentials.email === users[0].email) {

isUserRegistered = true;

if (userCredentials.password === users[0].password) {

return "Bienvenido al sistema";

}

}

}

return isUserRegistered ? "Contraseña incorrecta" : "Usuario no registrado";

}

// Get Token function

const getToken = (userCredentials) => {

let isUserRegistered = false;

let authenticated = false;

var elapsed = (new Date().getTime()) - users[0].lastAttempt;

if (Math.floor(elapsed / 60000) >= 5) users[0].attemps = 0;

if (users[0].attemps >= 3) return "Unavailable";

for (let i = 0; i < users.length; i++) {

if (userCredentials.email === users[0].email) {

isUserRegistered = true;

users[0].attemps += 1;

users[0].lastAttempt = new Date().getTime();

if (userCredentials.password === users[0].password) {

users[0].attemps = 0;

authenticated = true;

}

}

}

console.log("Attemps: " + users[0].attemps);

if (authenticated && isUserRegistered){

// Encode user credentials on base64

const encodedCredentials = new Buffer(`${userCredentials.email}:${userCredentials.password}`).toString('base64');

return encodedCredentials;

}

return "";

}

module.exports = {login, getToken};

**TestApp.js**

const axios = require('axios');

const {login} = require("./Login");

const {getToken} = require("./Login");

const executeTest = async (input, expectedOutput, functionToExecute) => {

const actualOutput = await functionToExecute(input)

if (actualOutput !== expectedOutput) {

console.log(`Test fail. Expected output: ${expectedOutput} actual output: ${actualOutput}`)

} else {

console.log("Test pass")

}

}

const makePostApiCall = async (userCredentials) => {

const response = await axios.post('http://localhost:3000/login', userCredentials)

return response.data

}

const makePostApiCallToken = async (userCredentials) => {

const response = await axios.post('http://localhost:3000/authenticate', userCredentials)

return response.data

}

const testReturnsSuccessMessageWhenUserLogsInWithValidEmailAndPassword = () => {

const userCredentials = {email: "myemail@mail.com", password: "securePassword"}

const successMessage = "Bienvenido al sistema"

executeTest(userCredentials, successMessage, makePostApiCall)

}

const testReturnsInvalidPasswordMessageWhenUserLogsInWithValidEmailAndPassword = () => {

const userCredentials = {email: "myemail@mail.com", password: "invalidPassword"}

const successMessage = "Contraseña incorrecta"

executeTest(userCredentials, successMessage, makePostApiCall)

}

const testReturnsUserNotFoundMessageWhenUserLogsInWithValidEmailAndPassword = () => {

const userCredentials = {email: "unregistered\_user@mail.com", password: "securePassword"}

const successMessage = "Usuario no registrado"

executeTest(userCredentials, successMessage, makePostApiCall)

}

const testReturnsSuccesTokenWithValidEmailAndPassword = () => {

const userCredentials = {email: "myemail@mail.com", password: "securePassword"}

const successMessage = new Buffer(`${userCredentials.email}:${userCredentials.password}`).toString('base64')

executeTest(userCredentials, successMessage, makePostApiCallToken)

}

const testReturnsSuccesTokenWithInvalidEmailAndPassword = () => {

const userCredentials = {email: "mybademail@mail.com", password: "nonsecurePassword"}

const successMessage = ""

executeTest(userCredentials, successMessage, makePostApiCallToken)

}

const testReturnsUnavailableMessageWithInvalidPassword = () => {

const userCredentials = {email: "myemail@mail.com", password: "nonsecurePassword"}

// Lock user for 5 minutes

for (let i = 0; i < 4; i++) {

makePostApiCallToken(userCredentials)

}

const successMessage = "Unavailable"

executeTest(userCredentials, successMessage, makePostApiCallToken)

}

testReturnsSuccessMessageWhenUserLogsInWithValidEmailAndPassword()

testReturnsInvalidPasswordMessageWhenUserLogsInWithValidEmailAndPassword()

testReturnsUserNotFoundMessageWhenUserLogsInWithValidEmailAndPassword()

testReturnsSuccesTokenWithValidEmailAndPassword()

testReturnsSuccesTokenWithInvalidEmailAndPassword()

testReturnsUnavailableMessageWithInvalidPassword()