Testes Unitários

Função para o login do restaurante:

Na função abaixo, usamos para realizar o login do restaurante. Primeiro, verificamos se o email existe, e caso exista, verificamos se a senha está correta. Se tudo passar, retorna um “true” para o frontend, dizendo que o login foi feito:

```
export async function verificationUser(userData) {
    try {
        const db = await openDB();
        const userTeste = await db.get(`SELECT * FROM restaurantUserData WHERE email='${userData.email}'`);

        if(userTeste != undefined)
        {
            if(comparePasswords(userData.password, userTeste.password))
            {
                return {
                    id: userTeste.ID,
                    status: true,
                }
            }
            else
            {
                return "Esse usuário existe no sistema, porém, sua senha está incorreta!!!"
            }
        }
        else
        {
            return "Esse email não está cadastrado no sistema!!!"
        }
        } catch (err) {
            return err
        }
  }
```

Note que se algo der errado, irá voltar uma mensagem dizendo que o login não foi efetuado informando ou email ou senha incorretos.

Para os testes, criamos uma função que recebe um objeto com email e senha, e volta uma mensagem dizendo se o teste correu bem ou não:

```
async function testVerificationUser(user, esp)
{
  const enc = await verificationUser(user)
  if(enc.status === esp)
  {
      return "Esse usuário existe no sistema, e os dados de acessos estão corretos"
  }
  else
  {
      return enc
  }
}
```

Para a esteira de testes, utilizamos os dados abaixo:

```
console.log(await testVerificationUser({password: "@Jd090722", email: "alphabeta@gmail.com"}, true))
//Retorna um OK, já que existe
console.log(await testVerificationUser({password: "@Jd09072022", email: "alphabeta@gmail.com"}, true))
//Retorna um erro, já que a senha está incorreta
console.log(await testVerificationUser({password: "@Jd090722", email: "alphabetagama@gmail.com"}, true))
//Retorna um erro, já que o email está incorreto
```

E os resultados encontrados são:

```
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src> node verificationUser.js
Esse usuário existe no sistema, e os dados de acessos estão corretos
Esse usuário existe no sistema, porém, sua senha está incorreta!!!
Esse email não está cadastrado no sistema!!!
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src>
```

Função para o cliente entrar na página do restaurante:

Utilizamos essa função para o cliente conseguir encontrar a página do restaurante, e conseguir ver seu menu. Com isso, ele conseguirá ver todos os pratos que aquele restaurante tem, e fazer seus pedidos em sua mesa:

```
export async function getRestaurantId(userID)
{
    try {
        const db = await openDB()
        if(await db.get(`SELECT ID FROM restaurantUserData WHERE ID='${userID}'`))
        {
            await closeDB()
            return true
        }
        else
        {
            await closeDB()
            return false
        }
    } catch (err) {
        return err
    }
}
```

Caso a função ache o ID informado, volta um “true”, e caso não encontre, volta um “false”, informando o usuário que aquele código não existe no sistema.

Para os testes, criamos uma função que recebe um ID, e retorna se o restaurante está cadastrado ou não:

```
async function testGetIDFunction(id, esp)
{
    const idEnc = await getRestaurantId(id)
    if(idEnc === esp)
    {
        return "O Restaurante com esse ID está cadastrado no banco de dados do sistema!"
    }
    else
    {
        return "Não existe nenhum usuário cadastrado com esse ID no banco de dados!"
    }
}
```

Para a esteira de testes, utilizamos os dados abaixo:

```
console.log(await testGetIDFunction(1, true))
//Retorna OK, pois existe um restaurante com esse ID
console.log(await testGetIDFunction(12, true))
//Retorna um erro, dizendo que o restaurante não existe no sistema
```

E os resultados encontrados são:

```
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src> node getData.js
O Restaurante com esse ID está cadastrado no banco de dados do sistema!
Não existe nenhum usuário cadastrado com esse ID no banco de dados!
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src>
```

Função para retornar o nome do restaurante:

Utilizamos essa função para retornar o nome do restaurante, visto que em muitas telas, o nome aparece tanto para clientes, quando para os próprios restaurantes:

```
export async function returnName(userID)
{
    try {
        const db = await openDB()
        const data = await db.get(`SELECT restaurant_name FROM restaurantUserData WHERE ID='${userID.id}'`)
        await closeDB()
        return data
    } catch (err) {
        console.log(err)
        return err
    }
}
```

Ela usa o ID do restaurante para verificar se existe, e caso sim, retorna o nome solicitado pelo frontend.

Para os testes, criamos uma função que vai receber um ID, e um nome para comparar, e diz se o restaurante existe, se o nome informado está correto ou incorreto, ou se ele não existe.

```
async function testReturnNameFunction(id, esp)
{
    const nameEnc = await returnName(id) || ''
    if(nameEnc.restaurant_name != undefined)
    {
        if(nameEnc.restaurant_name === esp)
        {
            return `Nome do Restaurante: ${nameEnc.restaurant_name}`
        }
        else if(nameEnc.restaurant_name != esp)
        {
            return `O ID informado retorna ${nameEnc.restaurant_name}, e não ${esp}!!!`
        }
    }
    else
    {
        return "Não existe nenum restaurante cadastrado com o ID informado!!!"
    }
}
```

Para a esteira de testes, utilizamos os dados abaixo:

```
console.log(await testReturnNameFunction({id: 1}, "Teste Alpha/Beta"))
//Retorna ok, já que o ID é de um restaurante cadastrado, e o nome para comparação está correto
console.log(await testReturnNameFunction({id: 1}, "Teste Alpha"))
//Retorna um erro, já que o ID informado existe no banco de dados, mas o nome comparado está errado
console.log(await testReturnNameFunction({id: 12}, "Teste Alpha/Beta"))
////Retorna um erro, já que o ID informado não existe no banco de dados, mesmo o nome estando correto
```

E os resultados encontrados são:

```
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src> node getData.js
Nome do Restaurante: Teste Alpha/Beta
O ID informado retorna Teste Alpha/Beta, e não Teste Alpha!!!
Não existe nenhum restaurante cadastrado com o ID informado!!!
PS C:\Users\JOÃO\Desktop\WaiterTech\E7-\src\waiter-tech-api\src>
```