UNIVERSIDAD DE GUADALAJARA, CUCEI

Otras herramientas para el manejar errores

ALUMNO:

Ruiz Salcedo Carlos Salvador

Profesor: MICHEL EMANUEL LOPEZ FRANCO

I7036 - Computación Tolerante a Fallas

2023A sección: D06

OBJETIVO:

Conocer herramientas para el control y manejo de errores en el software en un lenguaje de programación.

DESARROLLO DE LA ACTIVIDAD:

La sentencia try consiste en un bloque try que contiene una o más sentencias. Las llaves {} se deben utilizar siempre , incluso para una bloques de una sola sentencia. Al menos un bloque catch o un bloque finally debe estar presente. Esto nos da tres formas posibles para la sentencia try:

try...catch

try...finally

try...catch...finally

Un bloque catch contiene sentencias que especifican que hacer si una excepción es lanzada en el bloque try. Si cualquier sentencia dentro del bloque try (o en una funcion llamada desde dentro del bloque try) lanza una excepción, el control cambia inmediatamente al bloque catch . Si no se lanza ninguna excepcion en el bloque try, el bloque catch se omite.

La bloque finally se ejecuta despues del bloque try y el/los bloque(s) catch hayan finalizado su ejecución. Éste bloque siempre se ejecuta, independientemente de si una excepción fue lanzada o capturada.

Puede anidar una o más sentencias try. Si una sentencia try interna no tiene una bloque catch, se ejecuta el bloque catch de la sentencia try que la encierra.

Bloque catch incondicional

Cuando solo se utiliza un bloque catch, el bloque catch es ejecutado cuando cualquier excepción es lanzada. Por ejemplo, cuando la excepción ocurre en el siguiente código, el control se transfiere a la cláusula catch.

```
try {
  throw "myException"; // genera una excepción
}
catch (e) {
```

```
  // sentencias para manejar cualquier excepción
  logMyErrors(e); // pasa el objeto de la excepción al manejador de errores
}
```

Copy to Clipboard

El bloque catch especifíca un identificador (e en el ejemplo anterior) que contiene el valor de la excepción. Este valor está solo disponible en el scope de el bloque catch.

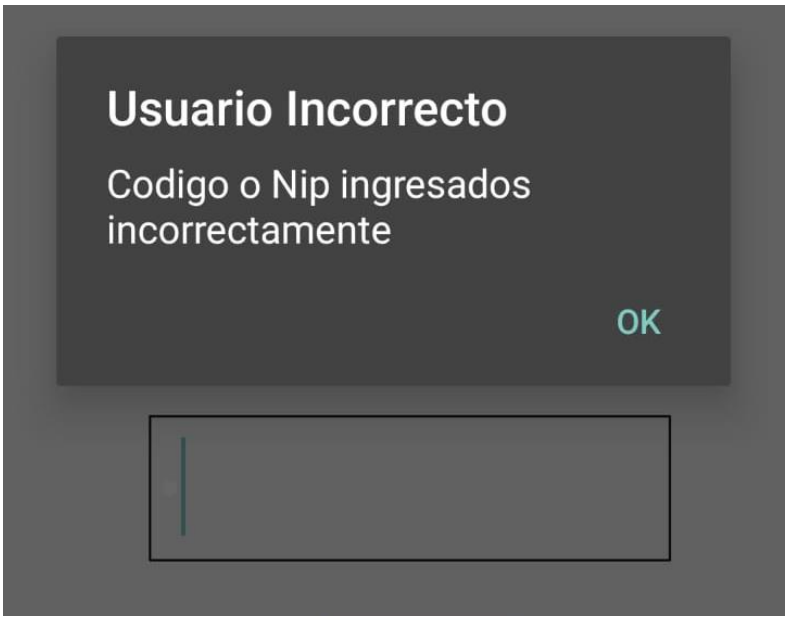
Implementación

```
const [code, setCode] = useState('');
const [password, setPassword] = useState('');

const petition = fetch(`https://ubele.000webhostapp.com/P.php?codigo=${code}&password=${password}`);


const storeData = async (value) => {
    try {
        const jsonValue = JSON.stringify(value)
        await AsyncStorage.setItem('@storage_Key', jsonValue)
    } catch (e) {
        // saving error
        console.log(e)
    }
}

const handleLogIn = () => {

    if(code == '' || password == ''){
        Dialog.show({
            type: ALERT_TYPE.WARNING,
            title: 'Atención',
            textBody: 'Uno de los campos está vacio!',
            button: 'Cerrar',
        })
    }else{
        petition
            .then( resp => resp.json())
            .then( data => {
                if(data == '2'){
                    Dialog.show({
                        type: ALERT_TYPE.DANGER,
                        title: 'Error',
                        textBody: 'Usuario no existe!',
                        button: 'Cerrar',
                    })
                } else if(data == '0'){
                    Dialog.show({
                        type: ALERT_TYPE.DANGER,
                        title: 'Error',
                        textBody: 'Contaseña incorrecta!',
                        button: 'Cerrar',
                    })
```

```
                    } else {
                        setCode('')
                        setPassword('')
                        storeData(data)
                        navigation.navigate('Home');
                    }
                })
                .catch( console.warn )
            }
        }

        const handleSingUp = () => {
            navigation.navigate('SingUp')
        }
```

para la implementación se utilizo java script donde buscando hacer el login de una app en react – native se utilizo un try catch para leer la cadena que ingresa el usuario para poder acceder donde si es correctas las credenciales nos da acceso pero si no nos arroja un mensaje de error y no se podrá acceder como podremos observar en el ejemplo.


```
LOG  Running "HDC" with {"rootTag":31}
LOG  https://prograinter-proyecto.000webhostapp.com/HDCLogin.php?username
LOG  0
```



CONCLUSIONES:

El manejo de errores en la programación me parece una forma útil de ayudar a los usuarios con errores que pueden llegar a realizar y me parece una buena manera de condicionar a los usuarios a ejecutar el programa como es debido y como esta programado es por esta razón que me pareció bastante interesante esta tarea para poder realizar programas que sean tolerantes a fallas.

BIBLIOGRAFIA:

try. . .catch - JavaScript | MDN. (2022, 12 diciembre).

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements/try. .

.catch