¿Cómo elegiste los casos de prueba para validar el correcto funcionamiento

del sistema?

Se eligieron casos con los que se esperaba un resultado exitoso del

funcionamiento del programa, es decir, cadenas que cumplieran con

características que permitan su transformación adecuada. También se

eligieron cadenas con las que se esperaba un error puesto que no cumplian

las precondiciones para que se les aplicará el método correctamente.

¿Después de concluir el reto refactorizar el código? ¿Qué cambios

realizaste?

No, porque nos guiamos del código que ya estaba hecho en los métodos

anteriores, por lo que no hubo fallas significativas en los nuevos

¿Utilizaste un método común o múltiples métodos individuales? ¿Por qué?

Si, se utilizaron métodos individuales para tener el control sobre las

conversiones realizadas con anterioridad

**Codigo StringOperations**

const castPascalCaseToSnakeCase = (pascalCaseString) => {

    return pascalCaseString.split(/(?=[A-Z])/).join('\_').toLowerCase();

}

const castSnakeCaseToPascalCase = (snakeCaseString) => {

    const words = snakeCaseString.split('\_');

    return words.map(word => word[0].toUpperCase() + word.substr(1)).join('');

}

const castPascalCaseToCamelCase = (pascalCaseString) => {

  return pascalCaseString.charAt(0).toLowerCase() + pascalCaseString.slice(1);

}

const castSnakeCaseToCamelCase = (snakeCaseString) => {

const pascalCase = castSnakeCaseToPascalCase(snakeCaseString);

  const camelCase = castPascalCaseToCamelCase(pascalCase);

return camelCase;

}

module.exports = {castPascalCaseToSnakeCase, castSnakeCaseToPascalCase, castPascalCaseToCamelCase, castSnakeCaseToCamelCase };

**Codigo TestStringOperations**

const {castPascalCaseToSnakeCase, castSnakeCaseToPascalCase, castPascalCaseToCamelCase, castSnakeCaseToCamelCase} = require("./StringOperations");

const testCastPascalCaseToSnakeCase = () => {

    executeTest("EstoEsUnEjemplo", "esto\_es\_un\_ejemplo", castPascalCaseToSnakeCase);

    executeTest("otroEjemplo", "otro\_ejemplo", castPascalCaseToSnakeCase);

    executeTest("otroejemplo", "otroejemplo", castPascalCaseToSnakeCase);

}

const testCastSnakeCaseToPascalCase = () => {

    executeTest("esto\_es\_un\_ejemplo", "EstoEsUnEjemplo", castSnakeCaseToPascalCase);

    executeTest("otro\_ejemplo", "OtroEjemplo", castSnakeCaseToPascalCase);

    // Caso de prueba que falla

    executeTest("aotroejemplo", "Otroejemplo", castSnakeCaseToPascalCase);

}

const testCastPascalCaseToCamelCase = () => {

    executeTest("EstoEsUnEjemplo", "estoEsUnEjemplo", castPascalCaseToCamelCase);

    executeTest("OtroEjemplo", "otroEjemplo", castPascalCaseToCamelCase);

    // Caso de prueba que falla

    executeTest("AOtroEjemplo", "otroEjemplo", castPascalCaseToCamelCase);

}

const testCastSnakeCaseToCamelCase = () => {

    executeTest("esto\_es\_un\_ejemplo", "estoEsUnEjemplo", castSnakeCaseToCamelCase);

    executeTest("otro\_ejemplo", "otroEjemplo", castSnakeCaseToCamelCase);

    // Caso de prueba que falla

    executeTest("aotroejemplo", "otroEjemplo", castSnakeCaseToCamelCase);

}

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

    const actualOutput = functionToExecute(input)

    if (actualOutput !== expectedOutput) {

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

    } else {

        console.log("Test pass")

    }

}

testCastPascalCaseToSnakeCase();

testCastSnakeCaseToPascalCase();

testCastPascalCaseToCamelCase();

testCastSnakeCaseToCamelCase();