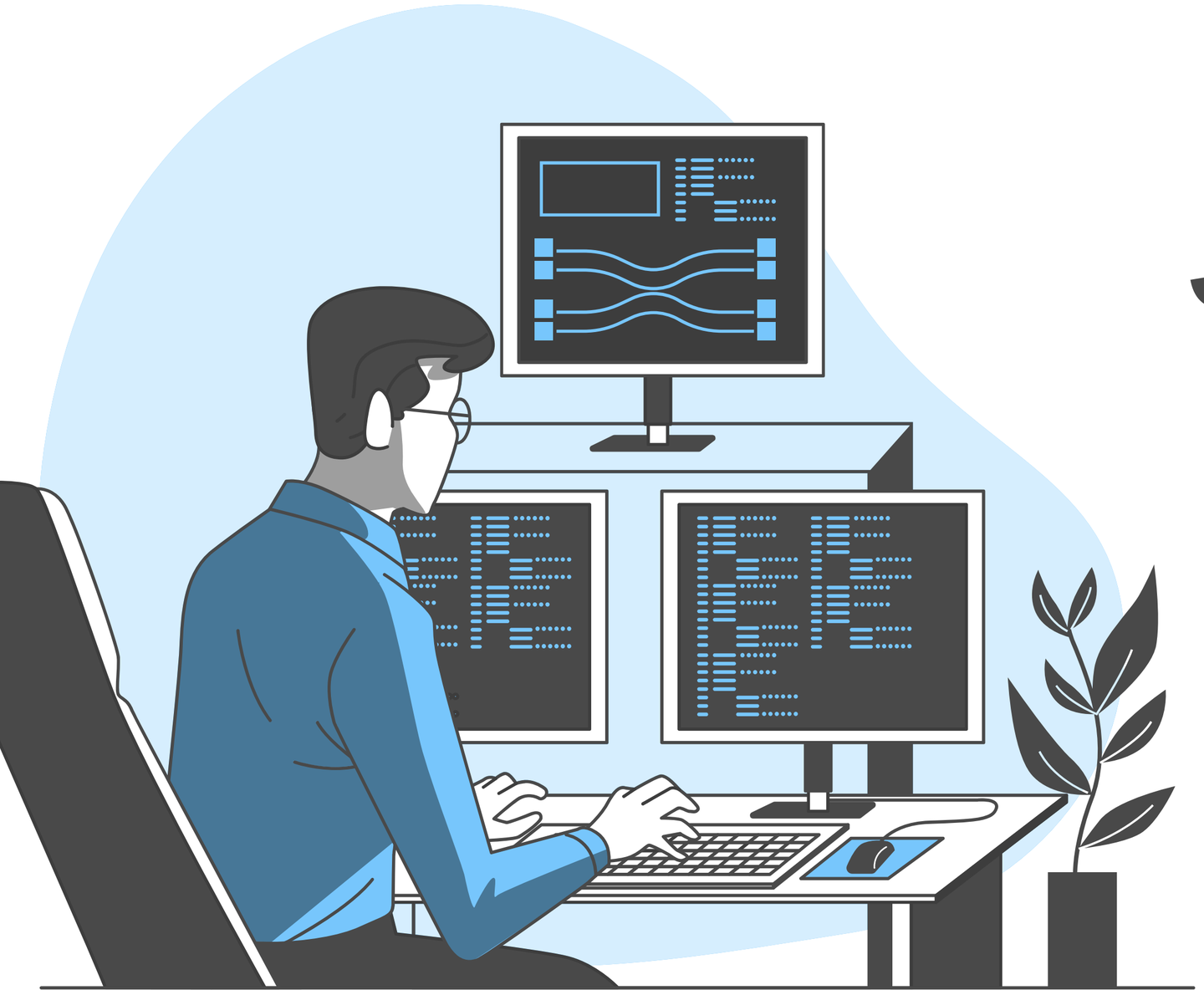


Tarea 1 JavaScript Funcional

Grupo 9

Manejo de I/O con Pipeline

```
function read_csv(file) {
    const data = fs.readFileSync(file, "utf8");
    return Papa.parse(data, { header: false, dynamicTyping: true }).data;
}

function write_csv(data, file="output.csv") {
    const csv = Papa.unparse(data);
    fs.writeFileSync(file, csv, "utf8");
}

function modify_csv(file, func) {
    const pipeline = _.pipe(
        (file) => read_csv(file),
        func,
        write_csv
    );

    return pipeline(file);
};
```

Read_csv

Lee archivo csv, y retorna su contenido procesado como una lista de objetos javascript.

Write_csv

Recibe una lista de objetos javascript y los escribe en un nuevo archivo csv.

Modify_csv

Lee archivo => aplica función de transformación => escribe en un nuevo csv.

Función swap

```
function swap(file, n, m) {
    const func = (data) => {
        return data.map((row) => {
            [row[n], row[m]] = [row[m], row[n]];
            return row;
        });
    };
    modify_csv(file, func)
}
```

Funcion rowstocolumns

```
function rowstocolumns (file) {
    const func = (data) => {
        const transpose = data[0].map((value, col_index) => data.map(row => row[col_index])
        );
        return transpose;
    };
    modify_csv(file, func);
}
```

data[0].map(...)

De la primera fila toma los valores e índices de las columnas

data.map(...)

Toma el valor correspondiente a la columna actual

const transpose = ...

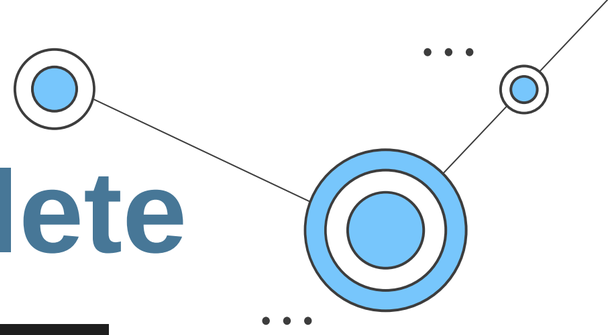
Nueva matriz que almacena la data transpuesta

Funcion columnstorows

- Función reduce recorre la data por filas recibiendo un acumulador
- Si el acumulador no contiene la columna de la iteración se inicializa vacía
- En caso contrario se agrega el valor de la columna al final del acumulador

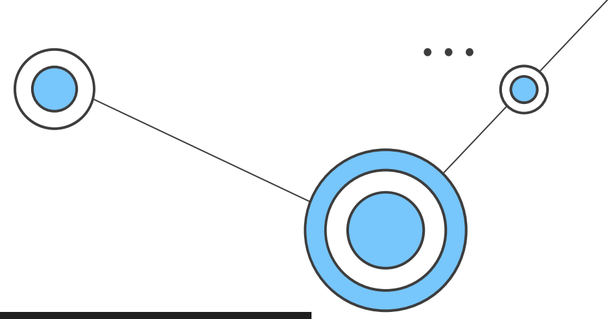
```
function columnstorows (file) {
    const func = (data) => {
        return data.reduce((acc, row) => {
            row.forEach((value, colIndex) => {
                if (!acc[colIndex]) acc[colIndex] = [];
                acc[colIndex].push(value);
            });
            return acc;
        }, []);
    };

    modify_csv(file, func);
}
```

Funciones rowdelete y columndelete



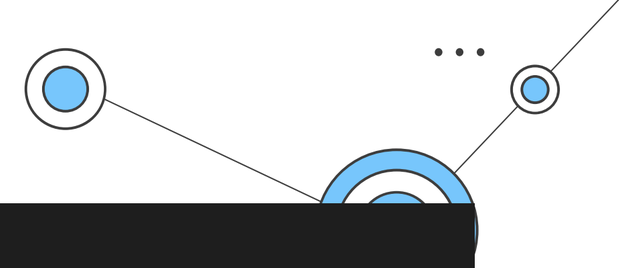
```
function rowdelete (file, n) {
    // elimina la fila n
    const func = (data) => {
        return data.filter((_, index) => index != n);
    }
    modify_csv(file, func)

}

function columndelete (file, n) {
    // elimina la columna n
    const func = (data) => {
        return data.map((row) => row.filter((_, index) => index != n));
    }
    modify_csv(file, func)
}
```

Funcion insertrow



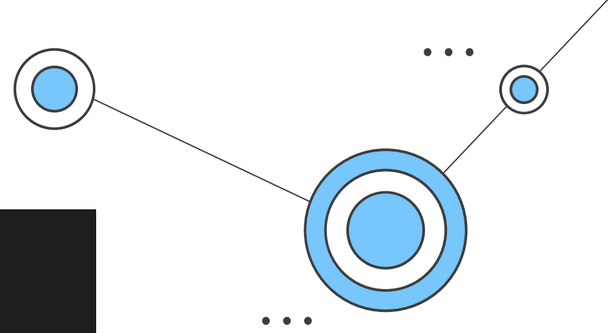
```
function insertrow(file, n, row) {
    const func = (data) => {
        // Verificar que `data` no esté vacío
        if (data.length === 0) {
            throw new Error("El archivo CSV está vacío.");
        }

        // Obtener el número de columnas del archivo CSV
        const numColumns = data[0].length;

        // Validar que la nueva fila tenga el mismo número de columnas
        if (row.length !== numColumns) {
            throw new Error(`Error: La fila insertada tiene ${row.length} columnas, pero se esperaban ${numColumns}.`);
        }

        // Insertar la nueva fila en la posición correcta
        return [...data.slice(0, n + 1), row, ...data.slice(n + 1)];
    };

    modify_csv(file, func);
}
```

Funcion insertcolumn



```
function insertcolumn(file, n, column) {
    const func = (data) => {
        // Verificar que `data` no esté vacío
        if (data.length === 0) {
            throw new Error("El archivo CSV está vacío.");
        }

        // Obtener el número de filas del archivo CSV
        const numRows = data.length;

        // Validar que la nueva columna tenga el mismo número de filas
        if (column.length !== numRows) {
            throw new Error(`Error: La columna insertada tiene ${column.length} filas, pero se esperaban ${numRows}.`);
        }

        // Insertar la nueva columna en la posición `n`
        return data.map((row, i) => [
            ...row.slice(0, n),
            column[i],
            ...row.slice(n)
        ]);
    };

    modify_csv(file, func);
}
```

Funcion tohtmltable



```
function tohtmltable(file) {
    const data = read_csv(file);  // Leer el CSV

    // Crear la tabla HTML
    let html = '<table>\n';

    // Iterar sobre cada fila del CSV
    data.forEach((row) => {
        html += '    <tr>\n'; // Comienza la fila

        // Iterar sobre cada celda de la fila y agregarla al HTML
        row.forEach((cell) => {
            html += `        <td>${cell} </td>\n`;  // Cada celda se coloca dentro de <td>
        });

        html += '    </tr>\n';  // Cerrar la fila
    });

    html += '</table>\n';  // Cerrar la tabla

    return html;
}
const htmlTable = tohtmltable('example.csv');
console.log(htmlTable);
```