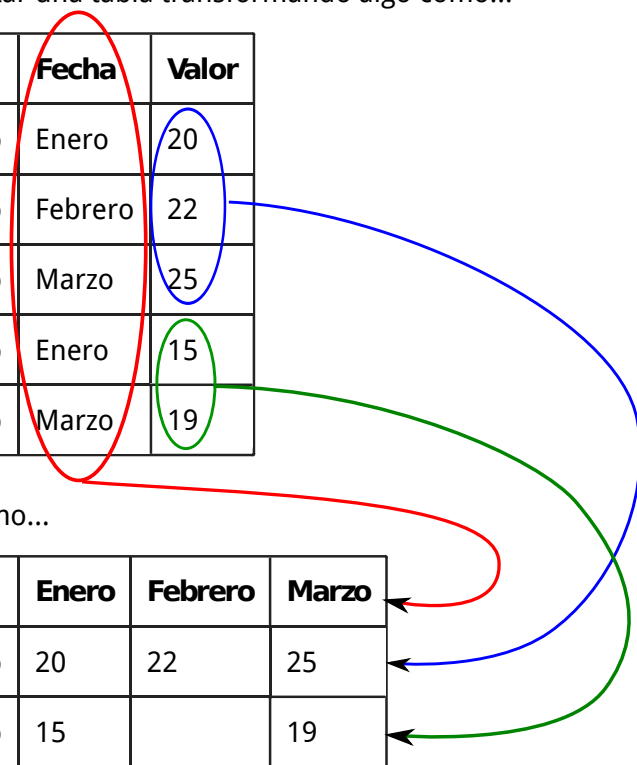


Pivotando tablas.

El proyecto está planteado para seguir de forma aproximada el caso real presentado en Java por Antonio en el que necesitaba pivotar una tabla transformando algo como...

Gestor	Línea	Fecha	Valor
Sam	Segunda Mano	Enero	20
Sam	Segunda Mano	Febrero	22
Sam	Segunda Mano	Marzo	25
Max	Segunda Mano	Enero	15
Max	Segunda Mano	Marzo	19



...en una visualización como...

Gestor	Línea	Enero	Febrero	Marzo
Sam	Segunda Mano	20	22	25
Max	Segunda Mano	15		19

Es decir, se escogen dos columnas en la tabla original:

- "Fecha" es la columna que se convertirá en cabecera
- "Valor" es la columna que se convertirá en valores

Después se agrupa por el resto de columnas y se pivota respecto a las columnas elegidas.

El código de partida

He dejado preparadas las siguientes partes:

- Un *dominio* (`src/domain/`) que contiene 3 objetos: Table, Row, y Cell. Más abajo describo su API. En general los tres producen objetos inmutables, de modo que cualquier operación que suponga modificación resultará en un nuevo objeto del tipo correspondiente.
- Nuestro campo de trabajo es el módulo `src/processing/pivot.js` que debe exportar una función única `pivot` que realiza el pivotaje. De partida la función simplemente devuelve la misma tabla recibida, tal cual.
- Hay un `src/index.js` y un par de utilidades (`src/view/`) por si se quiere tener algo con apariencia de proyecto real. (Se puede ejecutar con `npm run main`.)
- Además hay unos tests básicos (`test/`) que sirven como punto de partida para empezar a trabajar.

Cómo empezar

No es objetivo aquí explicar cómo se instalan pero es requisito tener instalado correctamente NodeJS (y npm, que viene incluido) para poder empezar a hacer algo.

Bajamos el proyecto, descomprimos en un directorio, etc. Abrimos un terminal en ese directorio y antes de nada ejecutamos:

```
> npm install
```

Para instalar las dependencias propias del proyecto.

NOTA: Si queremos trabajar directamente sobre el repositorio, nos lo clonamos, recordamos añadir al gitignore las carpetas `node_modules` y `.c9`, **empezamos una nueva rama** y trabajamos siempre sobre la nueva rama para dejar el ejercicio original disponible para otros.

Una vez hecho esto podemos:

- Ver el proyecto en marcha ejecutando `node src/index.js`. Saldrán por consola dos tablas iguales.
- Ejecutar los tests que nos indicarán por dónde deberíamos empezar a trabajar.
- Podemos usar el editor que más nos guste. Por si alguien quiere, he incluido en el repositorio la carpeta de configuración del proyecto en CloudNine (<http://c9.io/>).

Para ejecutar los tests tenemos configurados algunos comandos:

```
> npm test
```

(O también... `npm run test`)

Ejecutará los tests una vez mostrándonos los errores actuales.

```
> npm run watch
```

Ejecutará un proceso abierto que ejecutará los tests una vez inicial y luego cada vez que modifiquemos algún fichero.

Los tests se encuentran en `test/spec.js` y el código en `src/` (en concreto, `src/processing/pivot.js` es el que deberíamos trabajar, no tocando ningún fichero del modelo).

API del Dominio

Table, Row, Cell. En principio no usaremos Rows porque es sólo un array. Tampoco usaremos builders porque estáis obsesionados con ellos y en JS en el 99.9% de los casos no hacen falta.

Cell

```
var cell = new Cell("Mes", "Enero");
```

```
console.log(cell.name, cell.value); // "Mes" "Enero" - sólo lectura
```

Row

```
var row = new Row([Cell("Mes", "Enero"), Cell("Visitas", 4)]);

console.log(row.cells); // [Cell("Mes", "Enero"), Cell("Visitas", 4)] - copia, sólo lectura
console.log(row.size()); // 2
console.log(row.nameAt(1)); // "Visitas"
console.log(row.valueAt(1)); // 4
console.log(row.valueAtName("Visitas")); // 4
console.log(row.cellNamed("Visitas")); // Cell("Visitas", 4)
console.log(row.values()); // ["Enero", 4]
console.log(row.valuesMap()); // { "Mes": "Enero", "Visitas": 4 }
console.log(row.contains("Visitas")); // true
```

Table

```
var table = new Table(["Mes", "Visitas"], [row, row, row]);

console.log(table.header); // ["Mes", "Visitas"] - copia, sólo lectura
console.log(table.rows); // [row, row, row] - copia, sólo lectura
console.log(table.numRows()); // 3
console.log(table.numCols()); // 2
console.log(table.row(1)); // row
console.log(table.hasContent()); // true
```