

# Data Lovers

## Índice

- [1. Preámbulo](#)
  - [2. Resumen del proyecto](#)
  - [3. Objetivos de aprendizaje](#)
  - [4. Consideraciones generales](#)
  - [5. Criterios de aceptación mínimos del proyecto](#)
  - [6. Hacker edition](#)
  - [7. Consideraciones técnicas](#)
  - [8. Pistas, tips y lecturas complementarias](#)
  - [9. Checklist](#)
- 

## 1. Preámbulo

Según [Forbes](#), el 90% de la data que existe hoy ha sido creada durante los últimos dos años. Cada día generamos 2.5 millones de terabytes de datos, una cifra sin precedentes.

No obstante, los datos por sí mismos son de poca utilidad. Para que esas grandes cantidades de datos se conviertan en información fácil de leer para los usuarios, necesitamos entender y procesar estos datos. Una manera simple de hacerlo es creando *interfaces* y *visualizaciones*.

En la siguiente imagen, podrás ver cómo con la data que se ve en la parte izquierda se puede construir una interfaz amigable y entendible por el usuario al lado derecho.

## 2. Resumen del proyecto

En este proyecto construirás una *página web* para visualizar un *conjunto (set) de datos* que se adecúe a lo que descubras que tu usuario necesita.

Como entregable final tendrás una página web que permita visualizar la data, filtrarla, ordenarla y hacer algún cálculo agregado. Con cálculo agregado nos referimos a distintos cálculos que puedes hacer con la data para mostrar información aún más relevante para los usuarios (promedio, el valor máximo o mínimo, etc).

# CSS

Uso de selectores de CSS

Links

- [Intro a CSS](#)
- [CSS Selectors - MDN](#)

Modelo de caja (box model): borde, margen, padding

Links

- [Box Model & Display](#)
- [The box model - MDN](#)
- [Introduction to the CSS box model - MDN](#)
- [CSS display - MDN](#)
- [display - CSS Tricks](#)

Uso de flexbox en CSS

Links

- [A Complete Guide to Flexbox - CSS Tricks](#)
- [Flexbox Froggy](#)
- [Flexbox - MDN](#)

# Web APIs

Uso de selectores del DOM

Links

- [Manipulación del DOM](#)
- [Introducción al DOM - MDN](#)
- [Localizando elementos DOM usando selectores - MDN](#)

Manejo de eventos del DOM (listeners, propagación, delegación)

Links

- [Introducción a eventos - MDN](#)
- [EventTarget.addEventListener\(\) - MDN](#)
- [EventTarget.removeEventListener\(\) - MDN](#)
- [El objeto Event](#)

Manipulación dinámica del DOM

Links

- [Introducción al DOM](#)
- [Node.appendChild\(\) - MDN](#)
- [Document.createElement\(\) - MDN](#)
- [Document.createTextNode\(\)](#)

Pruebas unitarias (unit tests)

Links

- [Empezando con Jest - Documentación oficial](#)

Módulos de ECMAScript (ES Modules)

Links

- [import - MDN](#)
- [export - MDN](#)

Uso de linter (ESLINT)

Uso de identificadores descriptivos (Nomenclatura y Semántica)

Diferenciar entre expresiones (expressions) y sentencias (statements)

## Control de Versiones (Git y GitHub)

Git: Instalación y configuración

Git: Control de versiones con git (init, clone, add, commit, status, push, pull, remote)

Git: Integración de cambios entre ramas (branch, checkout, fetch, merge, reset, rebase, tag)

GitHub: Creación de cuenta y repos, configuración de llaves SSH

GitHub: Despliegue con GitHub Pages

Links

- [Sitio oficial de GitHub Pages](#)

GitHub: Colaboración en Github (branches | forks | pull requests | code review | tags)

## user-centricity

Diseñar un producto o servicio poniendo a la usuaria en el centro

## product-design

Crear prototipos de alta fidelidad que incluyan interacciones

Seguir los principios básicos de diseño visual

## research

Planear y ejecutar testeos de usabilidad de prototipos en distintos niveles de fidelidad

navegador y, además, puedes crear una cuenta gratis. Sin embargo, eres libre de utilizar otros editores gráficos como Illustrator, Photoshop, PowerPoint, Keynote, etc.

El diseño debe representar el *ideal* de tu solución. Digamos que es lo que desearías implementar si tuvieras tiempo ilimitado para trabajar. Además, tu diseño debe seguir los fundamentos de *visual design*.

## Testeos de usabilidad

Durante el reto deberás hacer *tests* de usabilidad con distintos usuarios, y en base a los resultados, deberás iterar tus diseños. Cuéntanos qué problemas de usabilidad detectaste a través de los *tests* y cómo los mejoraste en tu propuesta final.

## Implementación de la Interfaz de Usuario (HTML/CSS/JS)

Luego de diseñar tu interfaz de usuario deberás trabajar en su implementación. No es necesario que construyas la interfaz exactamente como la diseñaste. Tu tiempo de hacking es escaso, así que deberás priorizar

Como mínimo, tu implementación debe:

1. Mostrar la data en una interfaz: puede ser un card, una tabla, una lista, etc.
2. Permitir al usuario interactuar para obtener la información que necesita.
3. Ser *responsive*, es decir, debe visualizarse sin problemas desde distintos tamaños de pantallas: móviles, tablets y desktops.
4. Que la interfaz siga los fundamentos de *visual design*.

## Pruebas unitarias

El *boilerplate* de este proyecto no incluye Pruebas Unitarias (*tests*), así es que tendrás que escribirlas tú para las funciones encargadas de *procesar*, *filtrar* y *ordenar* la data, así como *calcular* estadísticas.

Tus *pruebas unitarias* deben dar una cobertura del 70% de *statements* (*sentencias*), *functions* (*funciones*), *lines* (*líneas*), y *branches* (*ramas*) del archivo `src/data.js` que contenga tus funciones y está detallado en la sección de [Consideraciones técnicas](#).

## 6. Hacker edition

Las secciones llamadas *Hacker Edition* son opcionales. Si terminaste con todo lo anterior y te queda tiempo, intenta completarlas. Así podrás profundizar y/o ejercitar más sobre los objetivos de aprendizaje del proyecto.

Features/características extra sugeridas:

```
└─ test
  └─ data.spec.js
```

directory: 7 file: 20

## src/index.html

Como en el proyecto anterior, existe un archivo `index.html`. Como ya sabes, acá va la página que se mostrará al usuario. También nos sirve para indicar qué scripts se usarán y unir todo lo que hemos hecho.

## src/main.js

Recomendamos usar `src/main.js` para todo tu código que tenga que ver con mostrar los datos en la pantalla. Con esto nos referimos básicamente a la interacción con el DOM. Operaciones como creación de nodos, registro de manejadores de eventos (*event listeners* o *event handlers*), ....

Esta no es la única forma de dividir tu código, puedes usar más archivos y carpetas, siempre y cuando la estructura sea clara para tus compañeras.

En este archivo encontrarás una serie de *imports comentados*. Para *cargar* las diferentes fuentes de datos tendrás que *descomentar* la línea correspondiente.

Por ejemplo, si "descomentamos" la siguiente línea:

```
// import data from '../data/lol/lol.js';
```

La línea quedaría así:

```
import data from '../data/lol/lol.js';
```

Y ahora tendríamos la variable `data` disponible en el script `src/main.js`.

## src/data.js






El corazón de este proyecto es la manipulación de datos a través de arreglos y objetos.

Te recomendamos que este archivo contenga toda la funcionalidad que corresponda a obtener, procesar y manipular datos (tus funciones). Por ejemplo:

- `filterData(data, condition)`: esta función `filter` o filtrar recibiría la `data`, y nos retornaría aquellos datos que sí cumplan con la condición.

- Toda tu investigación previa debe tener como resultado todas las Historias de Usuario de tu proyecto.
- No hagas los prototipos de alta fidelidad de todas tus Historias. Comienza solamente por los que se necesiten para tu Sprint 1 (semana 1 de trabajo). Más pistas en la guía de organización para el proyecto.

Cuando ya estés lista para codear, te sugerimos empezar de esta manera:

1. Una de las integrantes del equipo debe realizar un  [fork](#) del repo de tu cohort, tus *coaches* te compartirán un *link* a un repo y te darán acceso de lectura en ese repo. La otra integrante del equipo deber hacer un fork del repositorio de su compañera y [configurar](#) un `remote` hacia el mismo.
2.  [Clona](#) tu *fork* a tu computadora (copia local).
3.  Instala las dependencias del proyecto con el comando `npm install`. Esto asume que has instalado [Node.js](#) (que incluye [npm](#)).
4. Si todo ha ido bien, deberías poder ejecutar las  pruebas unitarias (unit tests) con el comando `npm test`.
5. Para ver la interfaz de tu programa en el navegador, usa el comando `npm start` para arrancar el servidor web y dirígete a `http://localhost:5000` en tu navegador.
6. ¡A codear se ha dicho! 

## Contenido de referencia

### Diseño de experiencia de usuario (User Experience Design)

- Investigación con usuarios / entrevistas
- Principios de diseño visual

### Desarrollo Front-end

- Unidad de testing en curso de JavaScript en LMS.
- Unidad de arreglos en curso de JavaScript en LMS.
- Unidad de objetos en curso de JavaScript en LMS.
- Unidad de funciones en curso de JavaScript en LMS.
- Unidad de DOM en curso de Browser JavaScript en LMS.
- [Array en MDN](#)
- [Array.sort en MDN](#)
- [Array.map en MDN](#)
- [Array.filter en MDN](#)
- [Array.reduce en MDN](#)
- [Array.forEach en MDN](#)
- [Object.keys en MDN](#)
- [Object.entries en MDN](#)