UT1- Diseño y Generación de Interfaces Gráficas de Usuario. Componentes. Usabilidad y Accesibilidad – RA1, RA3, RA4

Desarrollo de Interfaces (DAD)

Desarrollo de Aplicaciones Multiplataforma (DAM)

PROFESORA: PATRICIA HENRÍQUEZ RODRÍGUEZ

CIFP VILLA DE AGÜIMES

CURSO 2024/2025

17.09.2024

Índice

- Interfaces gráficas de usuario: componentes
- Diseño: Usabilidad y accesibilidad

Interfaces gráficas de usuario: componentes

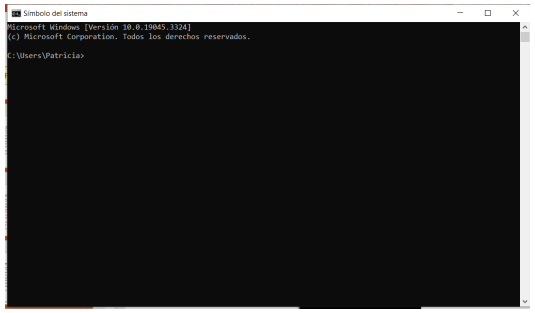
- **Definiciones de CLI, GUI y NUI**
- Librerías para GUI. Entorno de trabajo: React. Mi primera aplicación
- Componentes, eventos y su manejo
- Librería de componentes MUI

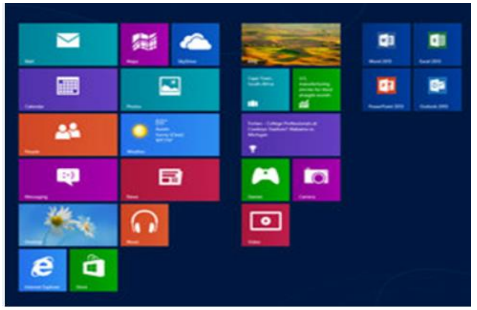
¿Qué es una interfaz de usuario?

- Una interfaz de usuario (UI, *user interface*) es un **punto de interacción** y **comunicación** (ya sea lógica o física) **entre el usuario y un dispositivo** (ya sea ordenador, Tablet, móvil, …). Con esta definición, podemos considerar interfaz a multitud de elementos como son: un teclado, un ratón, pantallas de visualización, **la apariencia de un escritorio, la interacción de un usuario con una aplicación, con una web**, etc.

- En este curso nos interesan las **interfaces gráficas de usuario** y las **interfaces naturales de usuario**.

Evolución interfaz de usuario

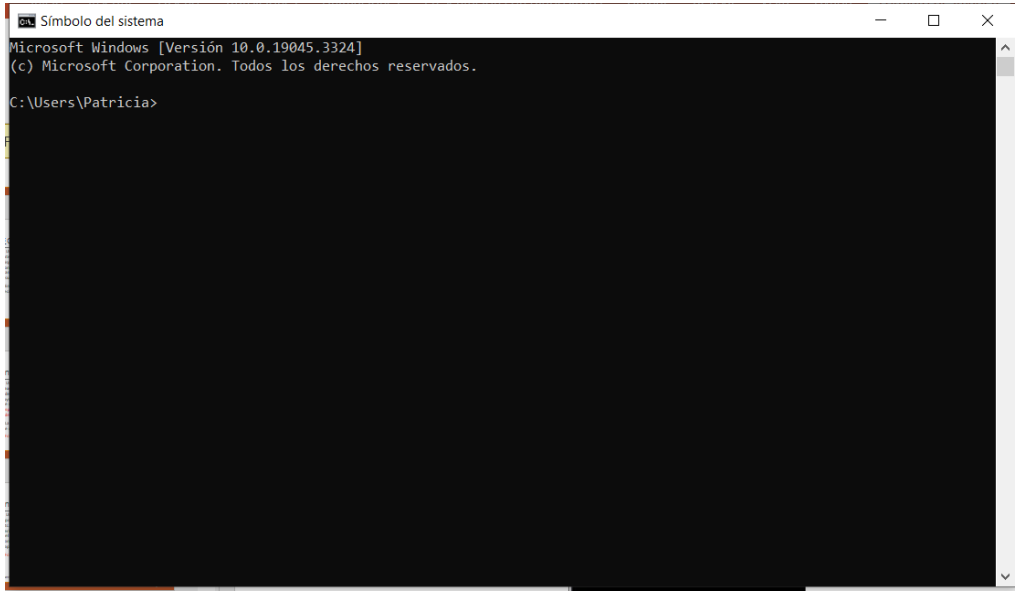




Fuente: https://es.wikipedia.org/wiki/Archivo:CLI-GUI-NUI,_evoluci%C3%B3n_de_interfaces_de_usuario.png

Interfaz por línea de comandos (CLI)

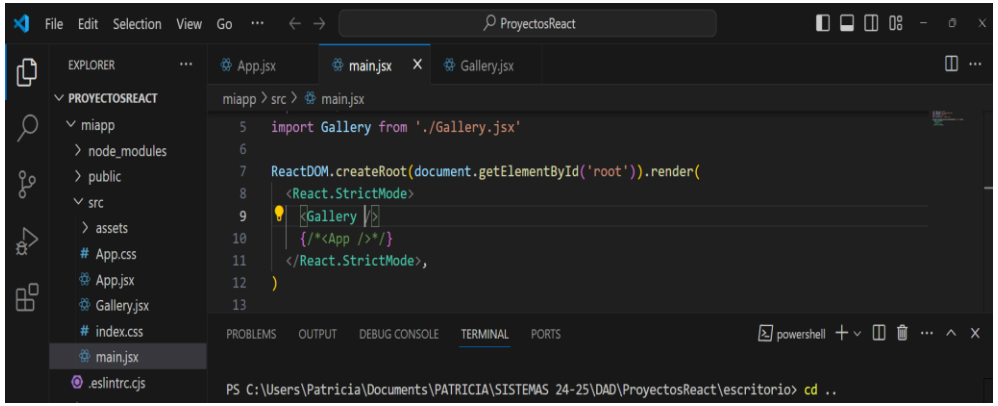
- Una **interfaz por línea de comandos** (CLI**,** del inglés *command line interface*) es un mecanismo software para interactuar con el sistema mediante el teclado. Las CLI fueron las primeras interfaces en aparecer, pero se siguen usando hoy en día.



Interfaz gráfica de usuario (GUI)

- Una **interfaz gráfica** de usuario (GUI, del inglés *graphical user interface*) muestra elementos gráficos como pueden ser menús, ventanas o diálogos para permitir que el usuario interactúe con la interfaz de mantera sencilla e intuitiva. En el diseño de una interfaz gráfica de usuario es importantísimo la usabilidad y la accesibilidad.

GUI del Campus



GUI del Visual Studio Code

Interfaz natural de usuario (NUI)

- Una **interfaz natural** de usuario (NUI, del inglés *natural user interface*) es un tipo de interfaz de usuario en las que se interactúa con un sistema, aplicación, etc., sin usar sistemas de mando o dispositivos de entrada (como en las GUI, sería un ratón, teclado, …), y en su lugar, se hace uso de movimientos gestuales del cuerpo o de alguna de sus partes como las manos, sirviendo de mando de control. También se usa el reconocimiento de voz o la realidad aumentada.




Fuente: https://es.wikipedia.org/wiki/Interfaz_natural_de_usuario

En este tema nos centraremos en las

interfaces gráficas de usuario

Interfaces gráficas de usuario

- Definiciones de CLI, GUI y NUI
- **Librerías para GUI. Entorno de trabajo: React. Mi primera aplicación**
- Componentes, eventos y su manejo
- Librería de componentes MUI

Librerías para programar interfaces gráficas de usuario

En Java:

- AWT
- Swing
- JavaFX

En JavaScript/Typescript:

- React
- Angular*
- Vue*

En C++:

- Qt
- DirectX

En C:

- GTK

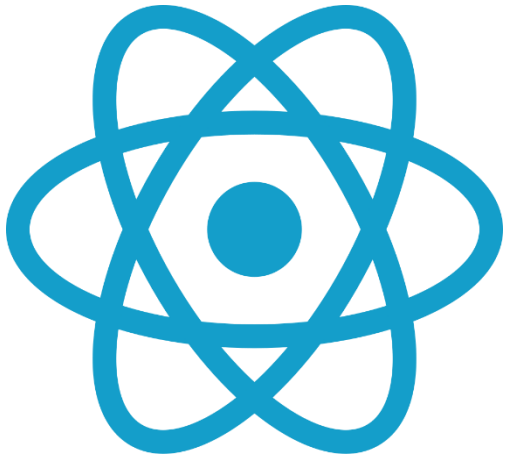
Usaremos **React: librería de Javascript para crear interfaces de usuario usando componentes reutilizables**.

* Son frameworks

Librerías para programar GUI

- AWT (Abstract Windows Toolkit): es la base de la librería Swing. Es la primera generación de interfaces para Java.
- Swing**:** diseñada en Java y creada a partir de la librería AWT. Es la segunda generación de interfaces para Java.
- JavaFX: desarrollada por Java/Oracle, es una plataforma opensource. Es la tercera generación de interfaces para Java.
- Direct X: grupo de APIs desarrolladas para simplificar las distintas tareas de vídeo y juegos. Utiliza la plataforma de Microsoft Windows. Entre ellas:
  - Direct3D: se usa para el procesamiento y programación de gráficos 3D.
  - Direct Graphics: útil para dibujar imágenes 2D en pantalla y representar imágenes 3D.
- GTK: biblioteca de componentes gráficos multiplataforma. Es parte del proyecto GNU, permite creación tanto de software libre como privativo. Se usa para GNOME y GIMP entre otros. Junto con Qt es muy popular para usarlo en sistemas operativos GNU/Linux.
- Qt: basada en las diferentes bibliotecas multiplataforma que permiten el desarrollo de interfaces gráficas. Creada y desarrollada en C++. Usa programación orientada a objetos para hacer uso de otros lenguajes de programación. Se usa por el entorno gráfico KDE. Se usa también en sistemas empotrados como automoción, navegación y aparatos domésticos.
- **React es una librería de Javascript para interfaces de usuario, tanto para web, como para móviles y aplicaciones de escritorio.**
- Angular y Vue se pueden considerar *frameworks* para páginas web que permiten por lo tanto crear interfaces gráficas de usuario. También se usan en móviles y en aplicaciones de escritorio.

Entorno de trabajo:



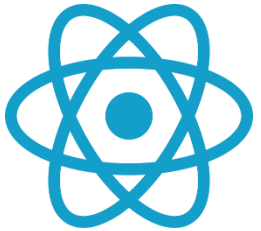
Crearemos aplicaciones con **REACT**:
librería JavaScript para interfaces gráficas de usuario

Para ejecutar una aplicación REACT necesitamos instalar **Node.js**, entorno que trabaja en tiempo de ejecución y que permite ejecutar código JavaScript fuera del navegador.

Para escribir nuestro código: **Visual Studio Code**

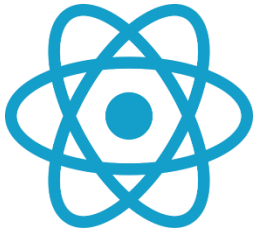
Links descargas:
https://code.visualstudio.com/
https://nodejs.org/en/download/prebuilt-installer

React



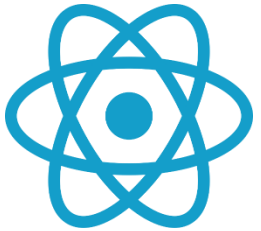
- Librería de JavaScript que permite la creación de elementos de interfaz de usuario reutilizables (componentes). Los componentes son como las piezas de lego que usaremos para la interfaz de usuario. Un componente puede ir desde un botón hasta la aplicación entera.
- Con React trabajamos el *front-end* de nuestra aplicación, creando interfaces de usuario tanto de aplicaciones web, como de aplicaciones móviles y de escritorio.
- React es de Meta y nace en 2011 y es completamente abierto desde 2013.
- Algunas aplicaciones populares que usan React incluyen WhatsApp, Instagram, Facebook, Netflix, Uber, Airbnb, etc.

React



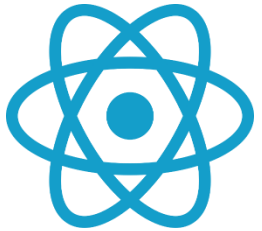
- React tiene diferentes paquetes, el principal es **react**, que es el que se usa para creación de componentes, *hooks*, *props* y estados. Otro paquete importante y que es específico para aplicaciones web es el **react-dom**, que es el encargado de renderizar los componentes React en el navegador. ReactNative haría lo mismo pero para dispositivos móviles.
- Algunas páginas interesantes de React
  - Documentación oficial de React: https://es.react.dev/
  - Wiki de React: https://www.reactjs.wiki/
  - React Native: React específico para aplicaciones móviles: https://reactnative.dev/
  - React en aplicaciones de escritorio: https://microsoft.github.io/react-native-windows/

React: escribir marcado con JSX



- Para crear componentes, React usa **marcado JSX** (JavaScript XML).
- JSX es una extensión de sintaxis para JavaScript que permite escribir marcado similar a HTML dentro de un archivo de JavaScript. JSX es más estricto que HTML y puede mostrar información dinámica.
- Los componente de React se escriben en JSX de forma que el código generado es más **declarativo** puesto que especificamos cómo queremos que sea nuestra interfaz de usuario.
- También podemos usar TSX (TypeScript XML), que es un lenguaje más tipado que JSX.
- Por detrás, tanto si usamos JSX como TSX, el código generado se traspilará (transformará) a JavaScript que es lo que entiende el navegador. Hay diferentes traspiladores, los más conocidos babel y SWC. Vienen incluidos en los empaquetadores de aplicaciones.

React: escribir marcado con JSX



HTML y JavaScript por separado

```
<div>
  <p></p>
  <form>
  </form>
</div>
```

HTML

```
isLoggedIn() {...}
onClick() {...}
onSubmit() {...}
```

JavaScript

JSX

```
Sidebar() {
  if (isLoggedIn()) {
    <p>Welcome</p>
  } else {
    <Form />
  }
}
```

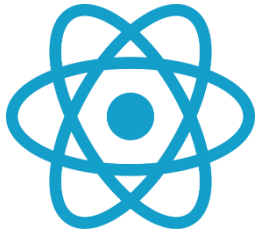
Componente de React `Sidebar.js`

```
Form() {
  onClick() {...}
  onSubmit() {...}
  <form onSubmit>
    <input onClick />
    <input onClick />
  </form>
}
```

Componente de React `Form.js`

https://es.react.dev/learn/writing-markup-with-jsx

React + Vite

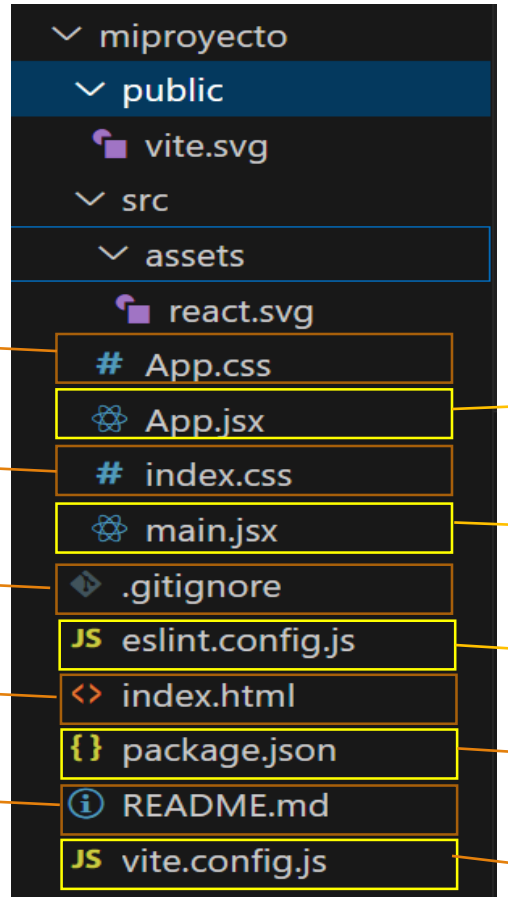


- Vite es un empaquetador de aplicaciones.
- Un empaquetador de aplicaciones se encarga de empaquetar todos los archivos de la aplicación, transformarlos, optimizarlos y minimizarlos para mejorar su funcionamiento tanto en desarrollo como en producción. Existen muchos empaquetadores de aplicaciones en React, como webpack, rollup, Parcel o Vite. Nosotros usaremos Vite (https://vitejs.dev/).

Primera aplicación con React

Actividad guiada (no evaluable): Mi primera aplicación con React

Estructura de un proyecto con React + Vite



Hoja de estilos para App.jsx

Componente raíz de la aplicación: genera los elementos que se renderizan en el DOM Virtual.

Hoja de estilos para index.html

Renderiza al componente raíz de la aplicación (App).

Aquí se añaden los archivos que no queremos subir a git.

Configuración del *linter* EsLint: reglas que debe seguir nuestro código.

Fichero html de nuestra aplicación. Llama al script src/main.jsx.

Metadatos del proyecto, scripts, paquetes necesarios en producción y en desarrollo, …*

Documentación del proyecto.

Archivo de configuración de vite.

*https://alvaroperdiz.com/javascript/node-js/informaci%C3%B3n-basica-package-json/

DOM y DOM Virtual

- **DOM** (Document Object Model) es la estructura del documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina árbol DOM.

- En React se trabaja con el virtual DOM. El **DOM virtual** no es más que una copia en memoria del DOM real en donde React crea una representación virtual de los componentes de la interfaz de usuario. Esto permite que React haga cambios de forma más eficiente minimizando la interacción con el DOM real, puesto que sólo renderiza los elementos necesarios y no todos. El hecho de usar el virtualDOM hace que el código en React sea más declarativo. En vez de escribir instrucciones detalladas de cómo queremos que se actualice la interfaz de usuario, simplemente escribimos cómo queremos que se vea y React es el que se encarga de actualizar lo que se necesita para ello.

Interfaces gráficas de usuario

- Definiciones de CLI, GUI y NUI
- Librerías para GUI. Entorno de trabajo: React. Mi primera aplicación
- **Componentes, eventos y su manejo**
- Librería de componentes MUI

Componentes

- Una interfaz gráfica (GUI) se comporta como un todo para proporcionar un servicio al usuario permitiendo que éste realice peticiones, y mostrando el resultado de las acciones realizadas por la aplicación.

- Las GUI están formadas por una serie de elementos gráficos, los **componentes**. Un **componente es una pieza de la interfaz de usuario que tiene su propia lógica y apariencia**. Pueden ser tan pequeños como un botón o un campo de texto o tan grandes como toda una página.

- Los componentes de React son **funciones de JavaScript que devuelven marcado JSX**. Es decir, un componente se encarga de fabricar los elementos que vemos en nuestra interfaz. Una de las características principales de los **componentes** es que son **reutilizables** y **anidables**.

- Los usuarios interaccionan con la interfaz mediante acciones que realizan sobre ella: picar sobre un botón, escribir en un formulario, seleccionar un elemento de una lista, etc. Los eventos son estas acciones que los usuarios provocan sobre los componentes, así que los **componentes** deben **gestionar** dichos **eventos**.

- Además, los componentes pueden tener **estados** (memoria, persistencia) y se les pueden pasar **propiedades** (**props**). En el caso de React, los componentes padres pueden pasar propiedades a los componentes hijos.

Componentes: características

- Por lo tanto un componente:
  - Debe poder modificarse para adaptarse a la aplicación en la que se integra: ser **reutilizable**.
  - Debe tener **persistencia**, es decir, debe poder guardar su estado cuando ha sido modificado.
  - Debe poder **gestionar eventos**.

Estructura de un componente en React

- Los **componentes en React empiezan por mayúscula** para distinguirlos de las etiquetas HTML.

```
function Componente(aquí van las props si las hubiera) {
→ Aquí se declaran los hooks y las variables
  return (
    → Aquí se escribe el marcado JSX: elementos que aparecerán en la interfaz
  )
}
export default Componente
```

Equivalente

```
export default function Componente(aquí van las props si las hubiera){
→ Aquí se declaran los hooks y las variables
  return (
  → Aquí se escribe el marcado JSX: elementos que aparecerán en la interfaz
  )
}
```

Las palabras clave e*xport default* especifican el componente principal del archivo para luego poderlo importar en otro archivo (*import Componente from 'ruta donde se encuentra el componente'*)
Podemos tener varios componente en un archivo, pero al ponerles delante del nombre *export default* indicamos que ese es el componente principal de dicho archivo.

https://es.react.dev/learn

Reglas marcado JSX **Se aplican a lo que va dentro del return(…)**

- **1era regla: Devolver un solo elemento raíz**. Para devolver múltiples elementos/etiquetas de un componente, hay que envolverlos en una única etiqueta principal. Podemos envolverlos en una etiqueta especial llamada <Fragment></Fragment> (no se suele utilizar mucho), que no deja rastros en el HTML. Para simplificar esta etiqueta podemos usar simplemente <></>. O si queremos, podemos usar un para envolver los elementos. En este último caso sí dejamos rastros en el HTML.

src > App.jsx > ...

```jsx
import { Fragment } from 'react'

function App() {
  return (
    <Fragment>
      <h1>Hola mundo</h1>
      <h1>Adiós mundo</h1>
    </Fragment>
  )
}

export default App
```

Es equivalente

src > App.jsx > ...

```jsx

function App() {
  return (
    <>
      <h1>Hola mundo</h1>
      <h1>Adiós mundo</h1>
    </>
  )
}

export default App
```

https://es.react.dev/learn

Reglas marcado JSX **Se aplican a lo que va dentro del return(…)**

- **2da regla: cerrar todas las etiquetas.** En HTML la etiqueda
 no había que cerrarla. Cuando trabajamos con JSX sí debemos hacerlo.

¡Incorrecto!

```
function App() {
  return (
    <>
      <h1>Hola mundo</h1>
      <p>Adiós<br>mundo</p>
    </>
  )
}

export default App
```

Correcto

```
function App() {
  return (
    <>
      <h1>Hola mundo</h1>
      <p>Adiós<br/>mundo</p>
    </>
  )
}

export default App
```

https://es.react.dev/learn

Reglas marcado JSX **Se aplican a lo que va dentro del return(…)**

- **3ra regla: uso de camelCase**. Los atributos de JSX se van a convertir a JavaScript y JavaScript tiene limitaciones en los nombres de las variables. Por ejemplo, sus nombres no pueden contener guiones ni ser palabras reservadas como *class*. Por eso, en React, muchos atributos HTML están escritos en camelCase. Por ejemplo, en lugar de min-width usa **minWidth**. Dado que *class* es una palabra reservada, en React se escribe ***className*** en su lugar.

src > App.jsx > App

```jsx
import './App.css'
function App() {
  return (
    <>
      <h1>Soy un gato y vivo feliz</h1>
      <img className='avatar' src='/cat-4189697_640.jpg' alt='imagen' />
    </>
  )
}

export default App
```

Fichero App.jsx

src > # App.css > .avatar

```css
.avatar {
  border-radius: 50%;
  width: 150px;
  height: 150px;
}
```

Fichero App.css

Visualización

La imagen del gato está puesta en la carpeta public del proyecto
Enlace para descargar la imagen del gato: https://pixabay.com/es/photos/gato-mascota-felino-animal-pelo-4189697/

Escapar hacia JavaScript: uso de llaves {}

- JSX te permite escribir marcado similar a HTML dentro de un archivo JavaScript. A veces necesitaremos **evaluar una expresión JavaScript** o **referenciar alguna propiedad dinámica dentro del marcado**. En estos casos **se usan llaves en el JSX para poder meter lógica y variables de JavaScript en el código**.
- Funcionan dentro del contenido de la etiqueta JSX o inmediatamente después de los = en los atributos.
- {{}} no es una sintaxis especial: es un objeto JavaScript metido dentro de llaves JSX.

(1) Ejemplo del uso de llaves {}

Fichero App.jsx

src > App.jsx > ...

```jsx
import './App.css'
function App() {
  const datos = {
    animal: 'gato',
    imageUrl: '/cat-4189697_640.jpg',
  }
  return (
    <>
      <h1>Soy un {datos.animal} y vivo feliz</h1>
      <img className='avatar' src={datos.imageUrl} alt='imagen' style={{width: 150, height: 150}} />
    </>
  )
}

export default App
```

Fichero App.css

src > # App.css > .avatar

```css
.avatar {
  border-radius: 50%;
}
```

Visualización

Enlace para descargar la imagen del gato: https://pixabay.com/es/photos/gato-mascota-felino-animal-pelo-4189697/

(2) Ejemplo del uso de llaves{}

- Otro ejemplo:

src > App.jsx > ...

```jsx
import './App.css'
function App() {
  const datos = {
    animal: 'gato',
    imageUrl: '/cat-4189697_640.jpg',
    imageSize: 150,
  }
  return (
    <>
      <h1>Soy un {datos.animal} y vivo feliz</h1>
      <img className='avatar' src={datos.imageUrl} alt='imagen' style={{width: datos.imageSize, height: datos.imageSize}} />
    </>
  )
}

export default App
```

Fichero App.jsx

src > # App.css > .avatar

```css
.avatar {
  border-radius: 50%;
}
```

Fichero App.css

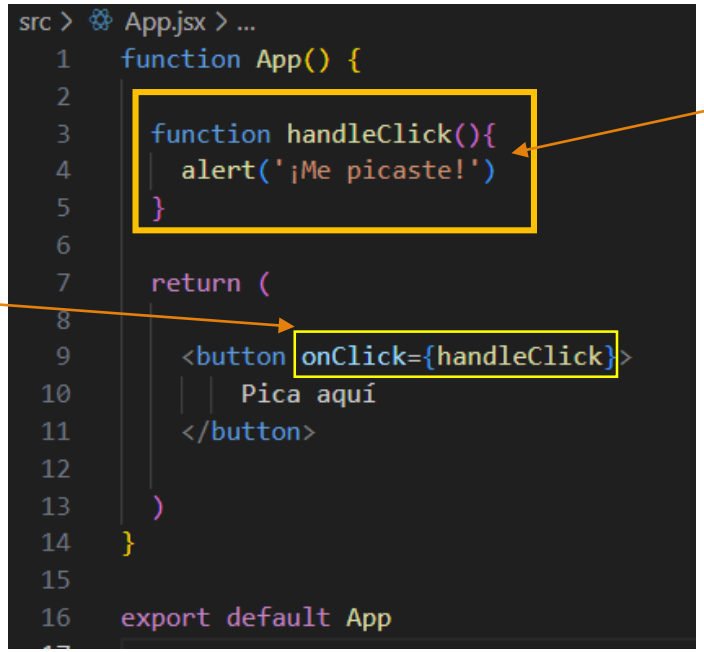
Visualización

Enlace para descargar la imagen del gato: https://pixabay.com/es/photos/gato-mascota-felino-animal-pelo-4189697/

Gestionar eventos

- Cuando un usuario interactúa con la aplicación, esta responde de alguna manera. Ej: si el usuario pica sobre un botón, se envía la información que ha sido previamente rellenada en un formulario. Para gestionar eventos el programador crea una función donde escribirá el código que maneja dichos eventos. Esta función se llama **función controladora de evento**.

2º Pasamos la función controladora como una *prop* al <button>. Debemos hacerlo entre {}.



1º Definimos la función **controladora de evento:** implementa el código que se ejecuta cuando se pica sobre el botón. Tienen nombres que empiezan con *handle* seguido del nombre del evento. En este caso: *handleClick*

Gestionar eventos

- Podemos usar una función flecha para definir la función controladora de evento en línea (dentro del propio JSX). Ojo, usar sólo con funciones cortas.

```
src > App.jsx > ...
1   function App() {
2
3     function handleClick(){
4       alert('¡Me picaste!')
5     }
6
7     return (
8
9       <button onClick={handleClick}>
10          Pica aquí
11      </button>
12
13    )
14  }
15
16  export default App
```

Equivalente

```
src > App.jsx > ...
1   function App() {
2
3     return (
4
5       <button onClick={() => alert('¡Me picaste!')}>
6           Pica aquí
7       </button>
8
9     )
10  }
11
12  export default App
```

Estado de un componente (memoria)

- Hay veces que queremos que el componente “recuerde” alguna información y la muestre como por ejemplo, contar el número de veces que hacemos clic en un botón. Esto lo conseguimos añadiendo **estado** al componente. Para ello usamos un *hook* llamado useState que debemos importar de React. Luego debemos declarar una variable de estado dentro del componente.

→ Primero importa useState de React:

```
import { useState } from 'react‘
```

→ Ahora declaramos una variable de estado en el componente:

```
function App() {
  const [count, setCount] = useState(0)
  …
```

- Con el useState obtenemos el estado actual (count) y la función que permite actualizarlo (setCount). La convención es llamarlos algo como [something, setSomething]

Ejemplo de estado de un componente

```
1  import React from 'react'
2  import { useState } from 'react'
3
4  function App() {
5    const [count, setCount] = useState(0)
6
7    function handleClick(){
8      setCount(count+1)
9    }
10
11   return (
12
13     <button onClick={handleClick}>
14         Me picaste {count} veces
15     </button>
16
17   )
18 }
19
20 export default App
```

Importamos la librería react.

La primera vez que se muestra el botón count será 0 porque pasamos 0 a useState().

Cuando queramos cambiar el estado llamamos a setCount() y pasamos el nuevo valor.

En este caso, al hacer clic en el botón, se llama a una función controladora de evento que lo que hace es incrementar el contador.

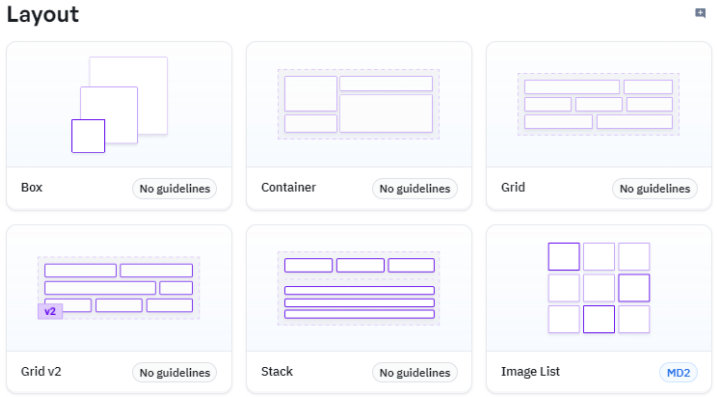
Interfaces gráficas de usuario

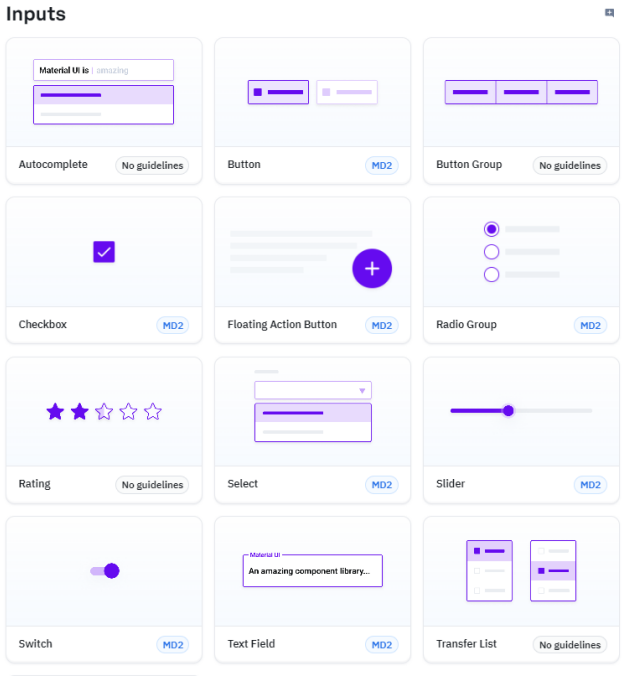
- Definiciones de CLI, GUI y NUI
- Librerías para GUI. Entorno de trabajo: React. Mi primera aplicación
- Componentes, eventos y su manejo
- **Librería de componentes MUI**

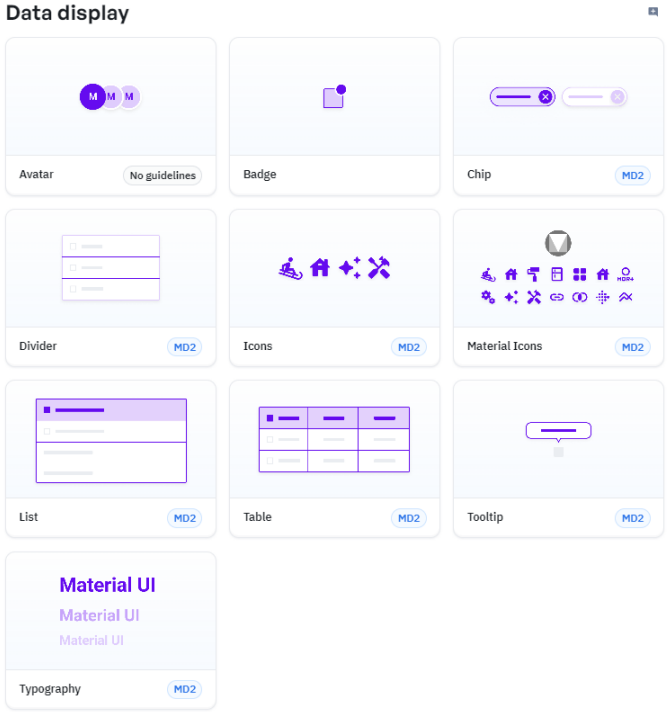
Librerías de componentes GUI para React

- A parte de crear nuestros propios componentes en React, podemos usar librerías de componentes gráficos como son:
  - **Material UI (MUI)**: https://mui.com/material-ui/ ⟵ Trabajaremos con esta
  - Chakra: https://v2.chakra-ui.com/
  - NextUI: https://nextui.org/?ref=horizon-ui.com
  - Ant Design: https://ant.design/?ref=horizon-ui.com

MUI: Material UI





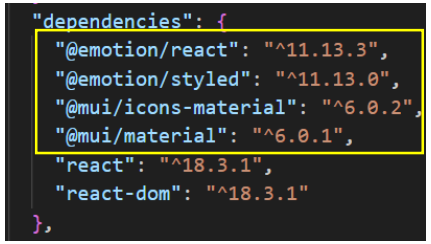


Instalar MUI en nuestro proyecto React

- Siguiendo las indicaciones de la documentación de MUI (https://mui.com/material-ui/getting-started/installation/):
  - En el Visual Studio nos situamos dentro de la carpeta de nuestro proyecto y ejecutamos el siguiente comando:

    npm install @mui/material @emotion/react @emotion/styled
  - Si además, queremos usar los iconos de MUI debemos instalar el siguiente paquete:

    npm install @mui/icons-material
  - Después de la instalación, si abrimos el package.json, veremos que tenemos instaladas las librerías:



Aprendemos a usar la documentación oficial de MUI

Actividad guiada – Inicio a los Componentes MUI

El objetivo de esta actividad **no evaluable** es que aprendan a usar componentes de la librería MUI leyendo la documentación oficial.

Diseño: usabilidad y accesibilidad

- **Usabilidad y accesibilidad. Características. Estándares.**
- Herramientas para el diseño: Esquemas (Wireframes) y Maquetas (Mockups).
- Pautas de diseño de interfaces de usuario usables y accesibles.

"La interfaz del usurio es súper intuitiva "

El usuario:

Diseño de una interfaz

- Pasos para diseñar una interfaz:

  1. Decidir su **estructura y disposición**: qué componentes usar y cómo se relacionan entre sí; decidir la disposición de dichos componentes teniendo en cuenta la **usabilidad y la accesibilidad**.

  2. Decidir el **comportamiento de la interfaz**: algunos controles permiten reaccionar ante eventos del usuario. El comportamiento se especifica programando la respuesta a dichos eventos.

Usabilidad = facilidad de uso

Accesibilidad = acceso universal a los contenidos

Una aplicación accesible aumenta su usabilidad

Usabilidad

- El concepto de usabilidad surgió en los años 80. Según el estándar ISO 25000: *"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso."*
- Se puede definir también como la disciplina que estudia la forma de diseñar aplicaciones para que los usuarios puedan interactuar con ellas de la forma más fácil, cómoda e intuitiva posible.
- Cuanto más sencillo resulte al usuario usar una aplicación, más posibilidades tendrá de realizar de forma eficiente su trabajo y mayor éxito tendrá la aplicación.

Usabilidad

- El diseño de la interfaz de la aplicación (jerarquía de los menús, tipos de formularios, cómo se muestran las imágenes, formato de texto, …) debe adaptarse al modo de pensar y actuar del conjunto de los usuarios y no tratar que ellos se adapten al diseñador.
- En la usabilidad también entra cómo se ve nuestra aplicación (en el caso de que sea una aplicación web) en toda clase de dispositivos. Los diseños deben ser adaptables (*responsive*).

A Jakob Nielsen se le considera el padre de la usabilidad

https://medium.com/@alanmartinez/10-principios-de-usabilidad-para-dise%C3%B1o-de-interfaces-de-usuario-f35d9d01643f

Estándares de usabilidad

- **ISO 25000** – Calidad del software y Métricas de evaluación
  - Definición: La capacidad del componente (software) para ser entendido, comprendido, usado y atractivo para el usuario cuando se usa bajo unas determinadas condiciones
  - Norma que se centra en la calidad del software desarrollado. La usabilidad en esta norma se centra en los requerimientos del producto, los cuales le dan funcionalidad y eficiencia. La usabilidad no sólo depende del producto sino también del usuario, por eso, se entiende dentro de un contexto determinado y con usuarios particulares.
- **ISO 9241** – Requisitos del software en relación a la calidad de usuario
  - Definición: Usabilidad es la eficacia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico
  - Esta es una definición centrada en el concepto de calidad en el uso, es decir, se refiere a cómo el usuario realiza tareas específicas en escenarios específicos con efectividad.
- **ISO 14915** – Ergonomía del software para interfaces de usuarios
  - En general, hace recomendaciones acerca del diseño de controles y navegación (por ejemplo, controles de audio con las funciones "play", "stop"...).

Ejemplos de errores de usabilidad

Errores de usabilidad

CANTINA CHICHILO DE BUENOS AIRES

CAMARONES 1901- CABA-

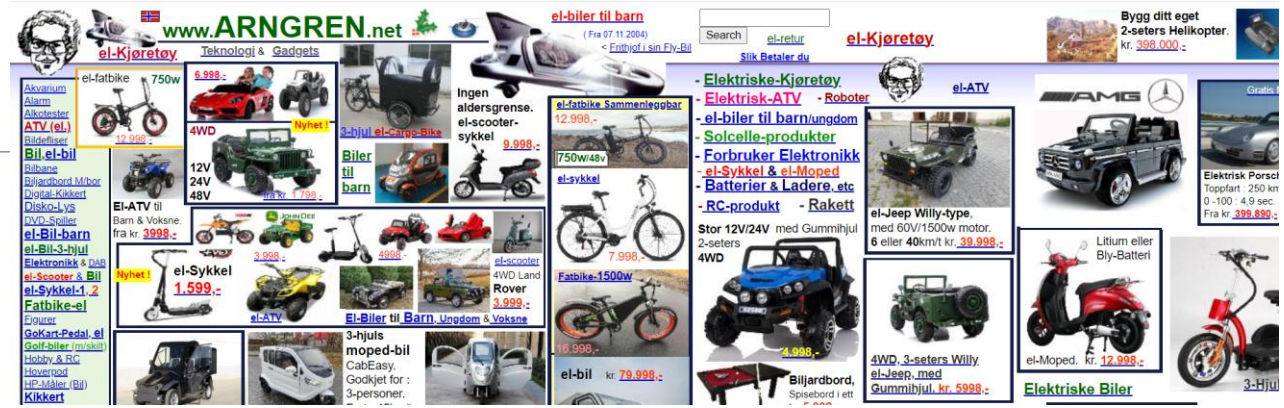
054-011-4584-1263

11-50-200-152

ABRIMOS

http://www.cantinachichilo.com.ar/

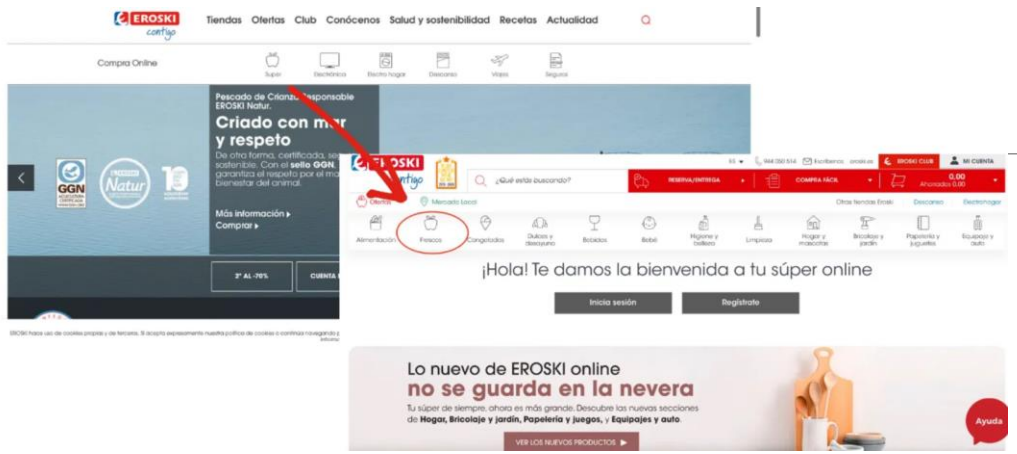
- Todo en una sola página
- Diseñada de forma que no se entiende ni se identifican las secciones
- Las fotos no invitan a reservar mesa …



https://www.arngren.net/

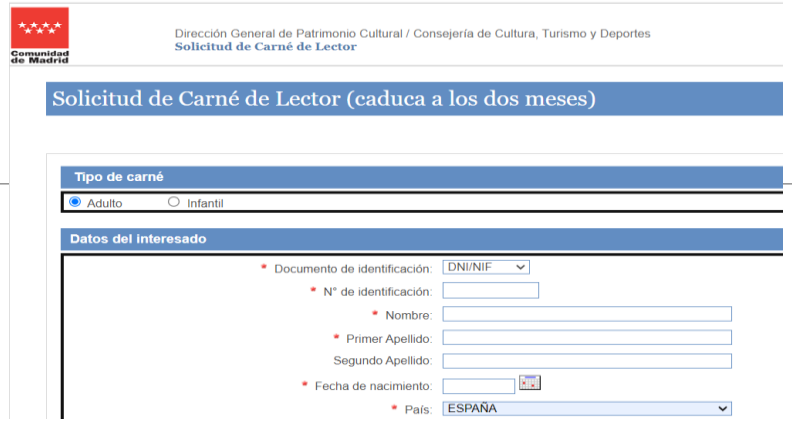
- En una tienda online debería ser fácil que el usuario comprase lo que quisiese y que lo hiciese con el menor número de clics.
  ➔ Esta web desde luego no lo cumple
- Diseño sobrecargado.
- Poca facilidad para encontrar productos.

Errores de usabilidad



https://www.eroski.es/

- No hay consistencia: usa el mismo icono (una manzana) para dirigirnos al súper y una vez allí la manzana representa los productos frescos.



https://gestiona3.madrid.org/i012_opina/run/j/InicialODF.icm?CDFORMULARIO=PLEC002

- En el formulario no sabes si tienes que poner DNI o NIF. Te dan las dos opciones, así que no queda claro.
- Cuando escribes el DNI, te aparece un mensaje de advertencia.
- Debería estar claro desde el principio si lo que tienes que poner es DNI o NIF.

Más ejemplos de errores de usabilidad

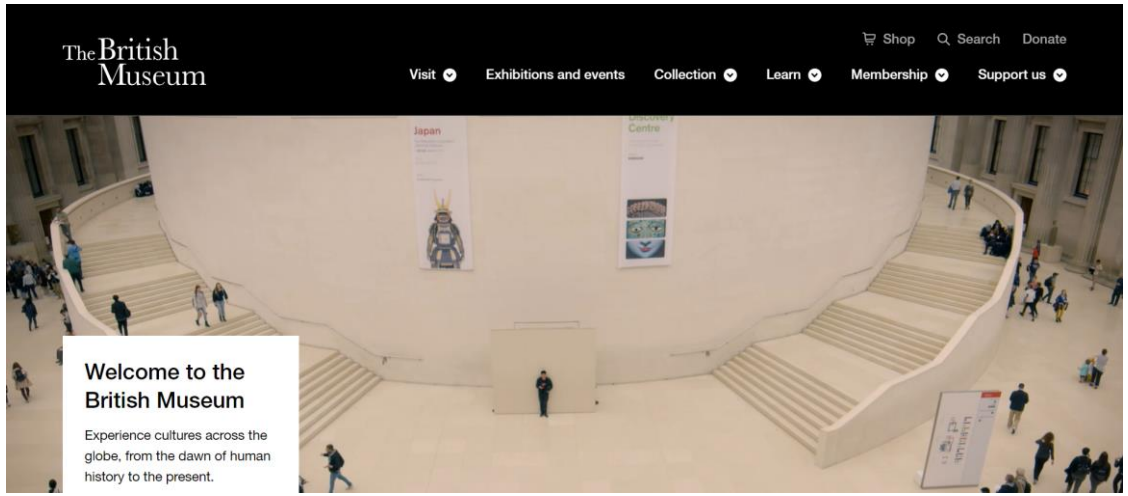
- Más errores de usabilidad:
  - Imposibilidad de ingresar en tu cuenta.
  - Webs no adaptables (responsive) se haga innavegable cuando la veas es un dispositivo móvil.
  - Errores de página no encontrada (404) y no te permitan redirigirte.
  - Tiempo de carga excesivo de la página.

Más ejemplos:

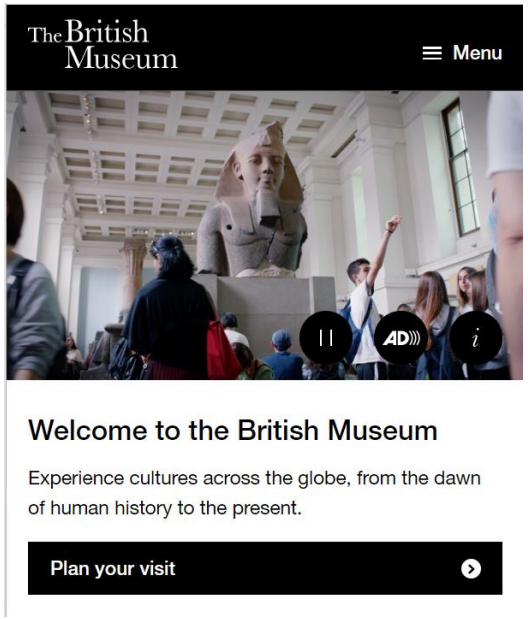
https://henriquegranai.medium.com/diez-ejemplos-b%C3%A1sicos-de-mala-usabilidad-4c5d5aea836b

Ejemplos buena usabilidad

En el ordenador



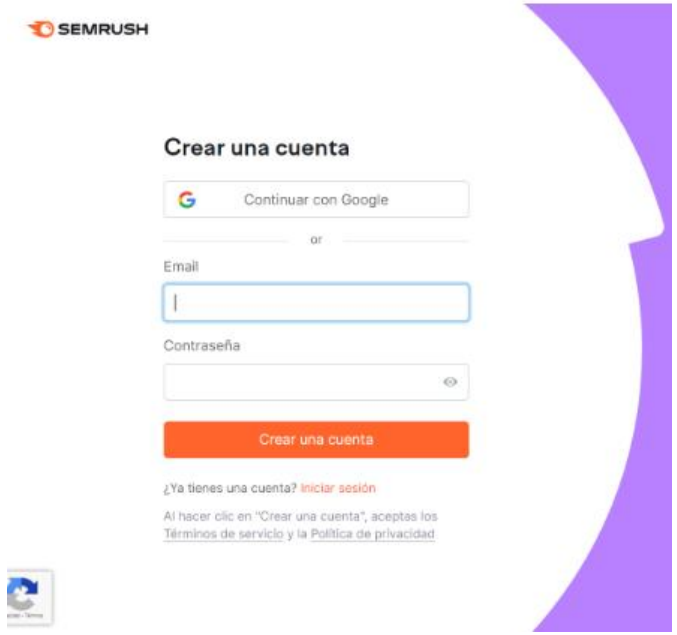
En el móvil



https://www.britishmuseum.org/

- Diseño *responsive* (adaptable): al verlo en el móvil los menús son igual de navegables.
- En la versión del ordenador despliega el menú con las categorías. Mientras que en la versión del móvil compila el menú en una esquina y realiza un acomodamiento vertical, destacando los mismos elementos.

Ejemplos buena usabilidad



https://es.semrush.com/signup/?src=blog&onboarding=off

- Formularios breves y automáticos: Cuando una persona quiere unirse o suscribirse a una página busca que sea un proceso breve. En el ejemplo, la página de registro solo solicita a los usuarios un correo electrónico y una contraseña. Además de permitirles registrarse de forma automática con su correo de Gmail.

Accesibilidad

- El World Wide Web Consortium (W3C) explica que una web es accesible si "***permite que las personas con diversidad funcional puedan percibir, comprender, navegar e interactuar con la web y herramientas***". Fundamentalmente, la accesibilidad garantiza que el diseño de una web pueda compensar, en la medida de lo posible, las discapacidades (de cualquier tipo) de los usuarios.

- Ejemplos de accesibilidad:

  - Cuando hacemos que el texto contraste lo suficiente sobre el fondo de la web, estamos cuidando la accesibilidad de la misma;

  - Uso del atributo 'alt' de la etiqueta HTML 'img': permite sustituir una imagen por una descripción textual de la misma y evitar así (mediante uso de software sintetizador de texto, por ejemplo) que el usuario se pierda información clave de nuestro contenido.

Estándares de Accesibilidad

- La accesibilidad web se basa en varios componentes que trabajan juntos relacionándose entre sí:
  - **Contenido web**: hace referencia a cualquier parte de un sitio web, incluyendo texto, imágenes, formularios y multimedia, así como a cualquier código de marcado, scripts, aplicaciones y demás
  - **Agentes de usuario**: software que las personas utilizan para acceder al contenido web, incluyendo navegadores gráficos de escritorio, navegadores de voz, navegadores de teléfono móvil, reproductores multimedia, plug-ins y algunas tecnologías de apoyo.
  - **Herramientas de autor**: software o servicios que las personas utilizan para producir contenido web, incluyendo editores de código, herramientas de conversión de documentos, sistemas de gestión de contenido, blogs, scripts de base de datos y otras herramientas.
- Ejemplo: el contenido web necesita incluir alternativas textuales para las imágenes. Esta información necesita ser procesada por los navegadores web y luego ser transmitida a las tecnologías de apoyo, como los lectores de pantalla. Para crear dichas alternativas textuales, los autores necesitan herramientas de autor con soporte para dicha función.

https://www.w3.org/WAI/standards-guidelines/es
https://wiki.ead.pucv.cl/Est%C3%A1ndares_de_Accesibilidad_Web

Estándares de Accesibilidad

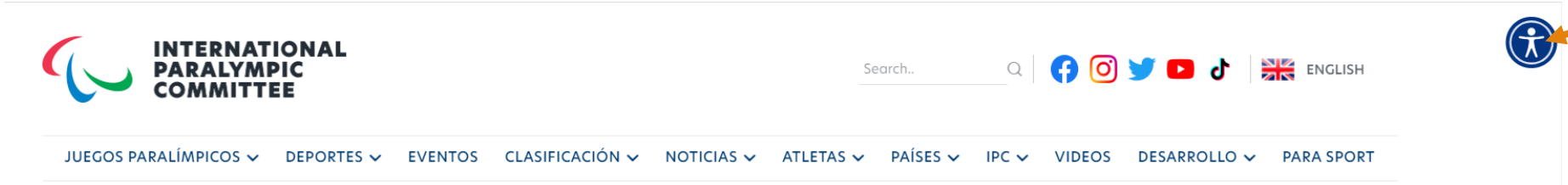
- La iniciativa de Accesibilidad Web del consorcio W3C (WAI) facilita un conjunto de pautas reconocidos como estándares internacionales para la accesibilidad web:
  - Pautas de Accesibilidad al Contenido Web (WCAG)
  - Pautas de Accesibilidad de Agente de Usuario (UAAG)
  - Pautas de Accesibilidad de Herramientas de Autor (ATAG)

- Como ven, se corresponde con lo que vimos en la hoja anterior.

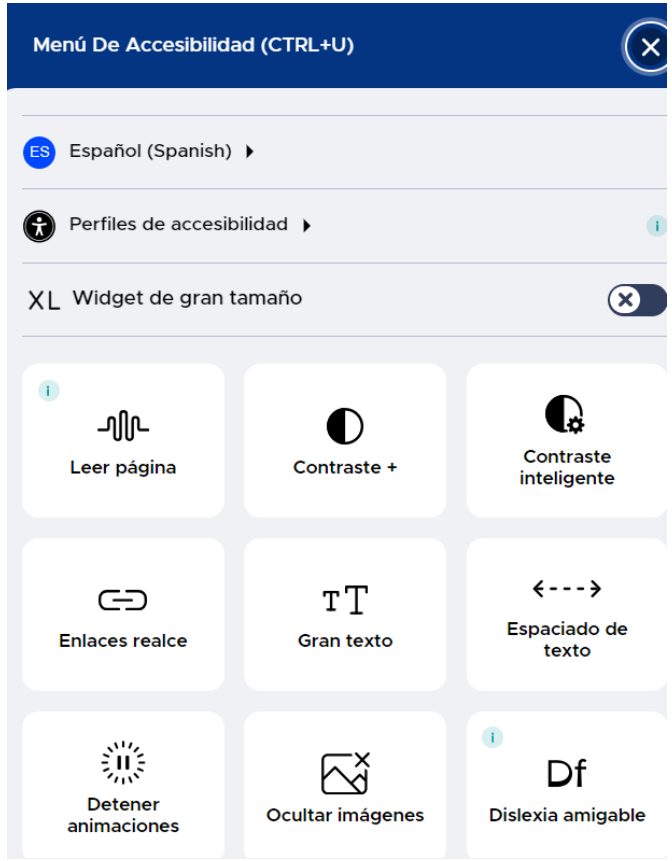
https://www.w3.org/WAI/standards-guidelines/es
https://wiki.ead.pucv.cl/Est%C3%A1ndares_de_Accesibilidad_Web

Ejemplos de webs accesibles

Permite configurar la accesibilidad



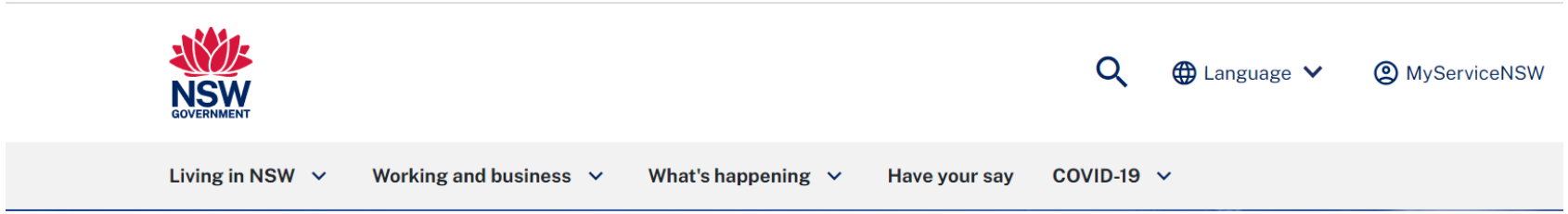
https://www.paralympic.org/es



- Permite configurar la accesibilidad
- Permite navegación por pestañas con el teclado.
- Imágenes y vídeos grandes y altamente visibles.

Más información: https://www.dreamhost.com/blog/es/grandiosos-ejemplos-accesibilidad-web/

Ejemplos de webs accesibles



https://www.nsw.gov.au/

- Permite navegación de pestañas
- Fuentes tipográficas grandes
- Colores que contrastan
- Muy legible

Más información: https://www.dreamhost.com/blog/es/grandiosos-ejemplos-accesibilidad-web/

Diseño: usabilidad y accesibilidad

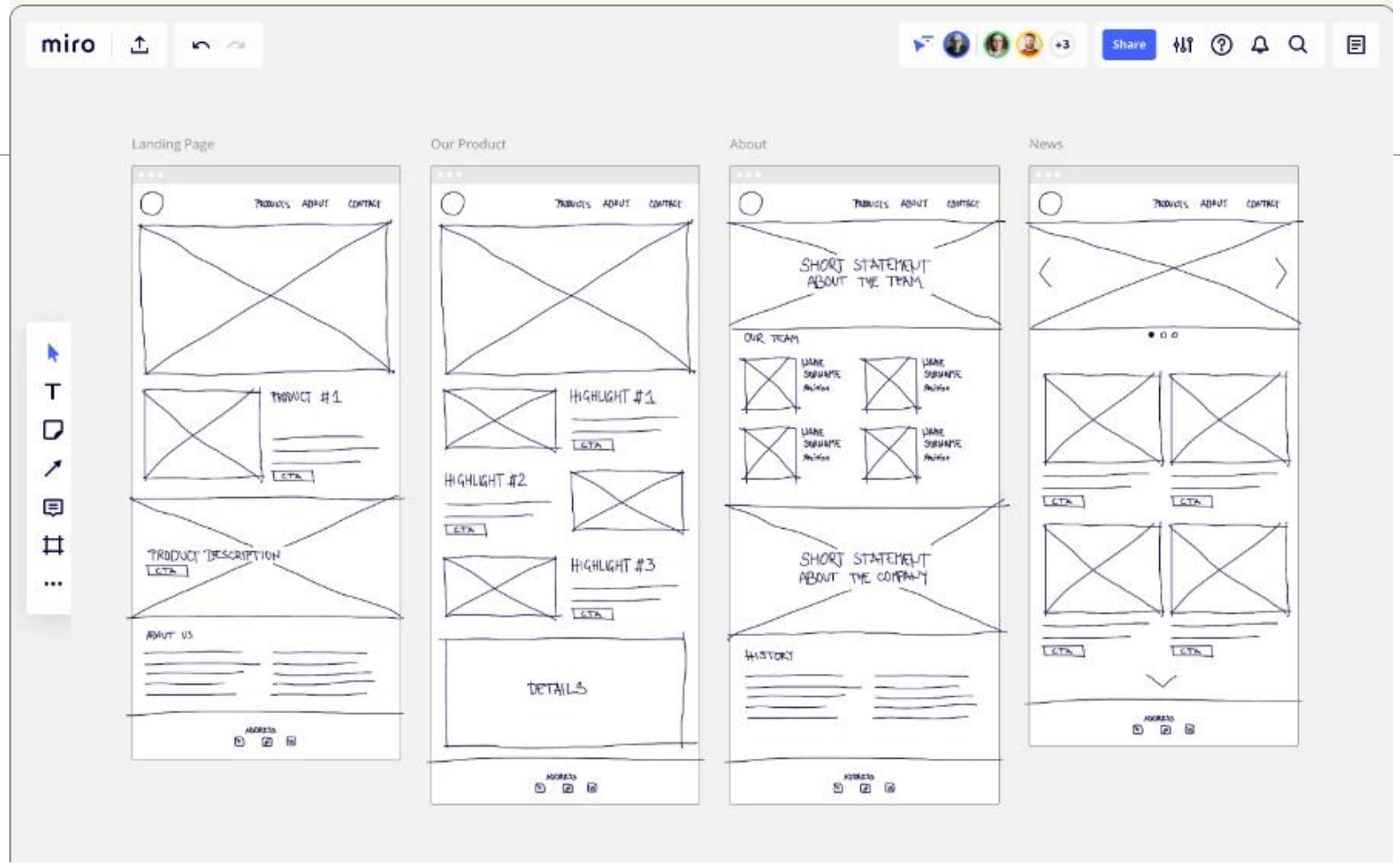
- Usabilidad y accesibilidad. Características. Estándares.
- **Herramientas para el diseño: Esquemas (Wireframes) y Maquetas (Mockups).**
- Pautas de diseño de interfaces de usuario usables y accesibles

Esquemas (wireframes) y maquetas (mockups)

- Son herramientas que podemos usar para el diseño de nuestro producto digital.
- Los **esquemas** (**wireframes**) son **representaciones simplificadas** y de **baja fidelidad** de la **estructura** y el **diseño** de un **producto digital**. Se centran en representar los elementos fundamentales y la funcionalidad sin entrar en la estética visual. Ayudan a comprender cómo los usuarios navegan por el producto y su interactividad con el mismo.
- Las **maquetas** (**mockups**) ofrecen una **representación visual más detallada** y de **alta fidelidad** del **producto** final. Hacen hincapié en la estética visual, incluidos los colores, la tipografía y los elementos gráficos.
- Los wireframes se usan para facilitar la comunicación entre diseñadores y desarrolladores. Los mockups permiten una comunicación más eficaz con los clientes, puesto que es una representación visual del producto final

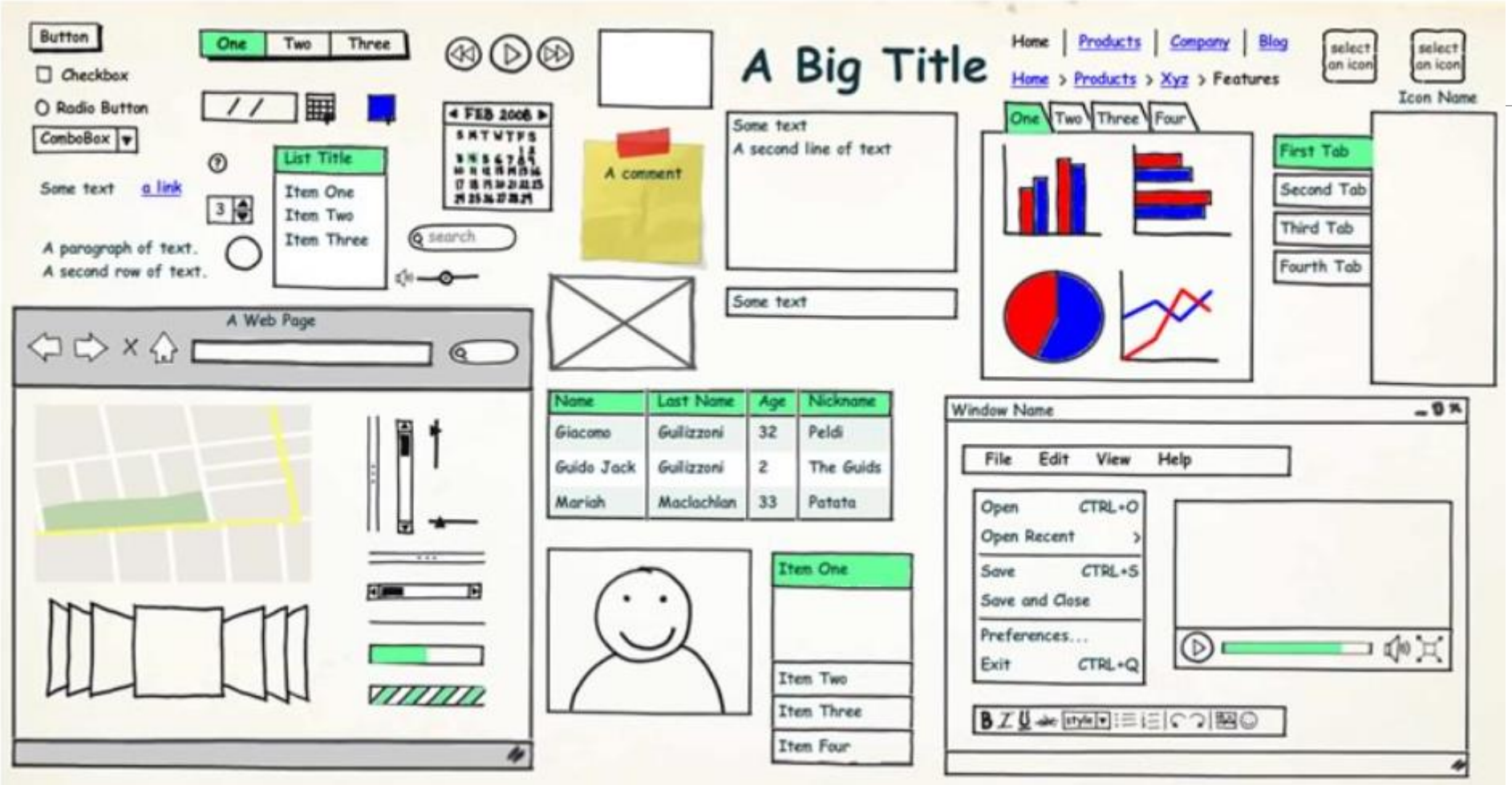
https://miro.com/es/wireframe/wireframe-vs-mockup/

Ejemplo de **WIREFRAME** de una aplicación web



https://miro.com/es/plantillas/wireframes-baja-fidelidad/

Ejemplo de **MOCKUP**



https://blogs.ugr.es/tecweb/mockup-arquitectura-web/

Diseño: usabilidad y accesibilidad

- Usabilidad y accesibilidad. Características. Estándares.
- Herramientas para el diseño: Esquemas (Wireframes) y Maquetas (Mockups).
- **Pautas de diseño de interfaces de usuario usables y accesibles**

Pautas de diseño de una interfaz de usuario

Las pautas de diseño de una interfaz de usuario van dirigidas a que la interfaz sea **usable** y **accesible**, de esta forma el usuario volverá a usar nuestra aplicación.

Algunas pautas para web accesibles

- Usar colores de alto contraste

  Herramienta para comprobar los criterios de contraste de WCAG:

  https://contrastchecker.com/

- Proporcionar texto alternativo para las imágenes (alt)
- Usar jerarquías de encabezado (h1, h2, h3, …): marcado claro.
- Usar los atributos aria que ayudan a hacer la web más accesible:

  https://developer.mozilla.org/es/docs/Web/Accessibility/ARIA

- Añadir leyendas y transcripciones a los vídeos
- Diseñar formularios con los campos etiquetados claramente para que sea fácil de navegar por ellos.
- Permitir navegación por teclado del sitio web.

Más información: https://www.dreamhost.com/blog/es/volver-tu-pagina-web-accesible/

Algunas pautas para interfaz usable

- Interfaz clara y precisa: estructura ordenada que siga una secuencia lógica, el usuario debe reconocer en todo momento lo que está viendo, saber para qué se usa y entender cómo la interfaz le ayudará a interactuar con la aplicación.
- Organización de los elementos de manera adecuada: permite al usuario aprender más rápido la aplicación. Agrupar elementos (en menús por ejemplo) que tengan algo en común, colocarlos en el lugar adecuado. Mostrar en pantalla únicamente lo que sea necesario o al menos poner el énfasis en lo importante. Diseño *responsive*.
- Jerarquía visual: establecer niveles de importancia y aplicarles diferentes estilos. El usuario debe poder establecer, de una pasada, qué es lo más importante.

Pautas de diseño de una interfaz de usuario

- Consistencia: tamaños de tipografía, colores y tamaños de elementos tienen que ser consistentes en toda la aplicación. Los elementos similares deben tener funciones similares.
- Definir las acciones primarias y secundarias: definir cuáles son las acciones principales que el usuario debe realizar en nuestro sitio. Tratar de definir una acción primaria por pantalla, mantener un proceso secuencial. En caso de necesitar varias acciones dentro de una misma pantalla, éstas serán acciones secundarias. Una acción secundaria siempre dependerá de la acción primaria que se eligió previamente, por lo que su diseño no debe resaltar más que la acción principal.
- Dar el control a los usuarios: debemos evitar forzar a los usuarios a realizar interacciones imprevistas. Evitar elementos que salten en pantalla sin aviso, que se muevan o suenen sin la opción de ser suprimidos por el usuario.

Guía de estilos

Todo lo anterior se puede resumir en una guía de estilos, que no es más que un documento en el que se plasma las pautas de estilo que se van a seguir durante el proceso de desarrollo de la aplicación.

- En la guía de estilos se recoge como será:
  - Distribución de los elementos en la interfaz (Layout)
  - La gama o paleta de colores
  - La tipografía
  - Tamaño de la letra
  - Iconos
  - Diseño de botones, menús, formularios y otros componentes

Más información: https://www.eniun.com/guia-estilo-diseno-interfaces-web/

Uso de librerías con patrones de interfaces

- Para generar una interfaz usable y accesible nos podemos ayudar de librerías de interfaces que nos faciliten el trabajo, como por ejemplo, la librería Material UI.

Uso del color

- Según la psicología del color, éstos transmiten emociones y provocan reacciones en nuestro cerebro. Esto es lo que dice la psicología del color de cada uno de ellos:

Fuente de la imagen: https://www.eniun.com/guia-estilo-diseno-interfaces-web/

Uso del color

- Elegir una paleta de colores: muchas veces viene impuesta por el logotipo o la marca de la empresa. Depende del tipo de aplicación: no es lo mismo una web de seguros que una web de juguetes para niños.
- Se pueden usar combinaciones de colores monocromáticas o de colores análogos

- En algunos casos también se pueden poner colores opuestos, pero para crear un alto impacto visual.
- Es preciso **utilizar los mismos colores para cada elemento web** con el fin de **facilitar la interpretación de funcionalidades, mejorar la navegación y transmitir armonía**.

Más información: https://www.lawebera.es/diseno-web/empleo-color-diseno-paginas-web.php

Uso del color. Herramientas de ayuda

- Podemos elegir nuestra paleta de colores ayudándonos de herramientas online

  https://color.adobe.com/es/create/color-wheel

  https://uicolors.app/create

  https://www.colorhunt.co/palettes/winter

  https://paletton.com/#uid=c284X2B5q0kllllaFw0g0qFqFg0w0aF

  https://m2.material.io/inline-tools/color/

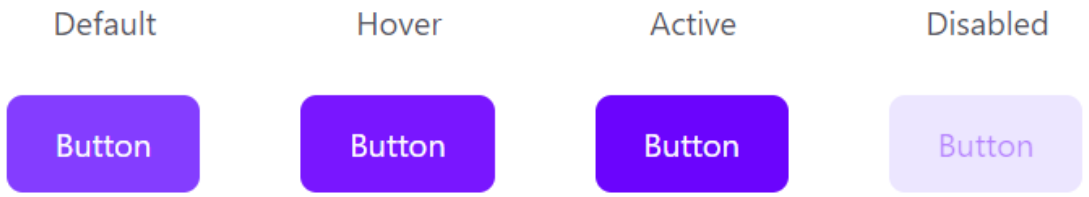
Uso del color: contraste

Contraste

- Podemos usar colores con tonalidades semejantes y con ellos establecer una jerarquía.
- Podemos usar el contraste también para hacer la web más accesible tal y como vimos en la accesibilidad.

Aplicar estilos a elementos enfocados

- Podemos añadir un estilo diferente si el ratón está encima de un elemento o si el elemento está desactivado.



Más información: https://www.lawebera.es/diseno-web/empleo-color-diseno-paginas-web.php

Diseño de botones

- Elegir el estilo del botón: con aristas, más o menos redondeado, ..
- Establecer su jerarquía según el color: colores sólidos botón más importante.
- Contraste entre el contenido del botón y el color del botón.
- Establecer el área del clic, que sea suficiente para cuando lo usemos en un móvil.

Más información: https://dinahosting.com/blog/diseno-de-botones-6-consejos-basicos/

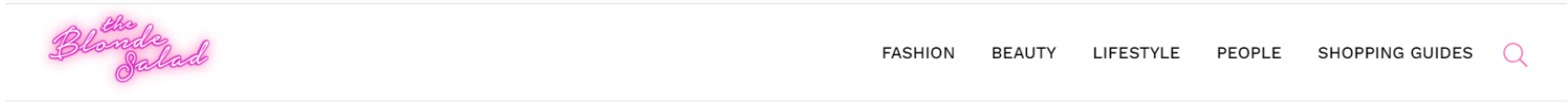
Diseño de menús

- Usar el tipo de menú más adecuado a tu aplicación.
- Algunos tipos de menús son:
  - Estándar
  - One-page
  - Desplegable
  - Hamburguesa

Más información: https://thecookies.agency/ejemplos-disenos-menu/

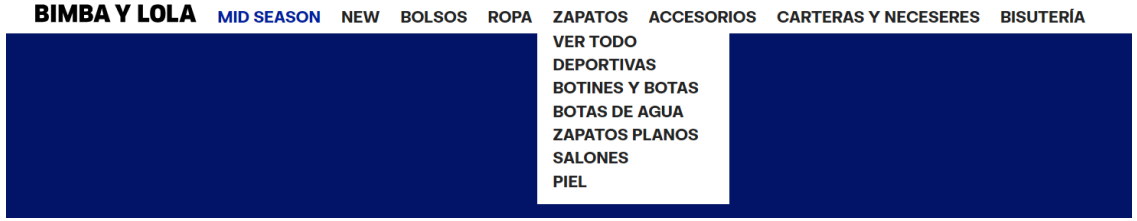
Diseño de menús

- Menú estándar: el menú más simple, son enlaces a las diferentes páginas del sitio. Se suelen colocar en la parte superior de la página, junto al logotipo, colocados en una fila horizontal. Ej: https://www.theblondesalad.com/en/



- Menú one page: es el de las landing pages, que son sitios web que sólo disponen de una página en la que se incluye toda la información. El menú ayuda a navegar dentro de la misma página. Ej: https://thecookies.agency/

- Menú desplegable: es como el menú estándar pero añadiendo un desplegable en cada item del menú. Ej: https://www.bimbaylola.com/es_es/

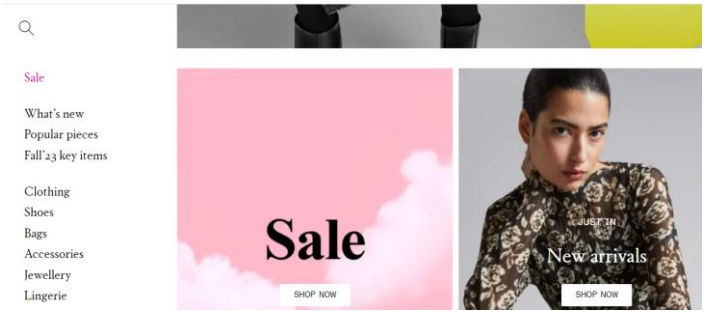


Más información: https://thecookies.agency/ejemplos-disenos-menu/

Diseño de menús

- Menú hamburguesa: recibe el nombre por las tres barras de su diseño. Se emplea en los diseños *responsive*, ya que es un menú sencillo para resoluciones pequeñas donde se puede comprimir la información necesaria. Ej: https://www.britishmuseum.org/ (en el móvil)

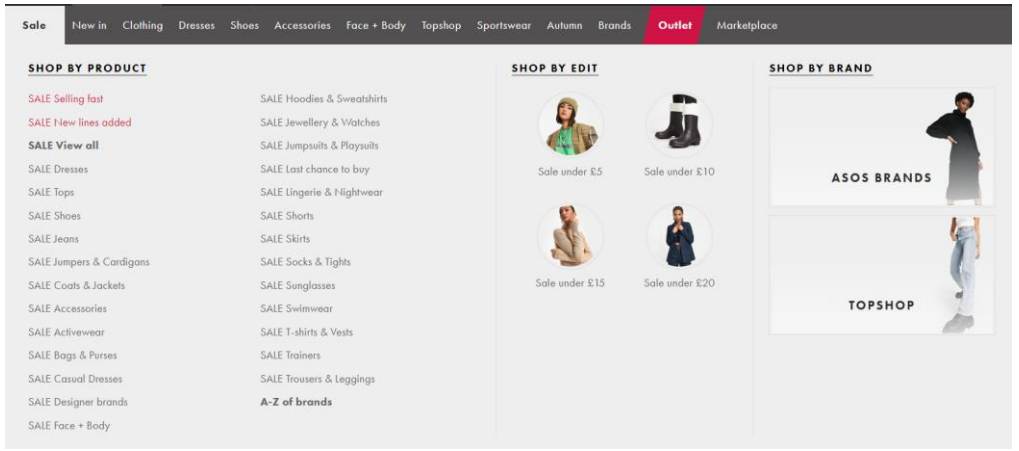
- Menú lateral: no es común, puesto que los menús suelen ir colocados en la parte superior. Se sitúan a la izquierda de la página y se mantienen fijos mientras se navega. Ej: https://www.stories.com/en_eur/index.html



Más información: https://thecookies.agency/ejemplos-disenos-menu/

Diseño de menús

- Mega menú: son menús más complejos que los menús estándar y que pueden contener todo tipo de información; no solo las categorías de la tienda on-line como tal, si no que también pueden incluir fotos, vídeos o incluso un sistema de filtros que ayude a encontrar de forma más rápida el producto o servicio que el usuario. Ej: https://www.decathlon.es/es/; https://www.asos.com/women/



Más información: https://thecookies.agency/ejemplos-disenos-menu/